Multi-Course Attendance Management Portal

Tejas Lohia
Computer Science and Engineering
Indian Institute of Technology,
Gandhinagar

tejas.lohia@iitgn.ac.in

Umang Shikarvar
Computer Science and Engineering
Indian Institute of Technology,
Gandhinagar

umang.shikarvar@iitgn.ac.in

Siddharth Rajandekar
Computer Science and Engineering
Indian Institute of Technology,
Gandhinagar
siddharth.rajandekar@iitgn.ac.in

Rishabh Jogani
Computer Science and Engineering
Indian Institute of Technology,
Gandhinagar

rishabh.jogani@iitgn.ac.in

Zainab Kapadia
Computer Science and Engineering
Indian Institute of Technology,
Gandhinagar
zainab.kapadia@iitgn.ac.in

GitHub Repository: **https://github.com/zainabkapadia52/Multi-Course-Attendance-Management-Portal**

Abstract - The Multi-Course Attendance Management Portal is a centralized database-driven system designed to efficiently manage attendance across multiple courses, semesters, and instructors within an academic institution. Traditional attendance systems based on paper registers, spreadsheets, or isolated software tools are prone to duplication, delays, and inconsistencies, especially when handling large numbers of students and course offerings. This project addresses these challenges by designing and implementing a normalized relational database that ensures accuracy, scalability, and referential integrity. Many entities were systematically transformed into an ER model reflecting database-level constraints, including one-to-many and many-to-many relationships resolved through associative entities. The final schema satisfies normalization principles and referential integrity requirements while supporting real-world academic workflows. The system provides a structured, secure, and scalable solution for managing attendance across multiple courses, thereby improving transparency, reducing manual errors, and enabling efficient academic administration.

Keywords— Multi-Course Attendance Management, Relational Database Design, Entity-Relationship (ER) Model, Referential Integrity, Normalization (3NF), Primary Key, Foreign Key, Academic Data Management, Database Constraints, Data Consistency, Transaction Integrity, Scalable System Design.

I. Introduction

Attendance management is a fundamental administrative task in academic institutions. In universities offering multiple courses across different semesters and sections, maintaining accurate attendance records becomes increasingly complex. Traditional methods such as paper registers, spreadsheets, or isolated attendance software often lead to inconsistencies, duplicate entries, delayed updates, and difficulty in tracking attendance across multiple course offerings. As the number of students, instructors, and course sections grows, manual systems become inefficient, error-prone, and difficult to scale.

The Multi-Course Attendance Management Portal is designed as a centralized, database-driven solution to address these challenges. The system provides a structured framework for managing students, instructors, courses, semester-wise offerings, enrollments, teaching assignments, attendance sessions, attendance records, and correction requests. By organizing academic data into well-defined relational entities with enforced primary and foreign key constraints, the system ensures consistency, reliability, and referential integrity.

The portal supports key academic workflows such as enrolling students into specific course offerings, assigning instructors to sections, conducting session-wise attendance marking, recording attendance status for each student, and processing correction requests in a controlled and auditable manner. Logical constraints such as valid session timing, unique enrollments, and structured attendance statuses are incorporated to maintain data correctness.

The first stage for designing database systems is to identify the requirements and analyze the data.

II. Requirements Collection and Analysis

A. Data Requirements

This section identifies the data that must be stored in the system to support all functional requirements. Each subsection describes a major data category with its attributes and purpose.

a. User Authentication Data

The system must store authentication credentials and access control information for all users.

Required Attributes:

i. LoginID (Primary Key): Unique integer identifier for each login account
ii. Username (Unique, Not Null): The identifier users enter when logging in
iii. PasswordHash (Not Null): Stores the user's password credentials
iv. Role (Not Null): Categorizes users as STUDENT, INSTRUCTOR, or ADMIN
v. CreatedAt (Not Null): Timestamp of account creation
vi. LastLogin: Timestamp of most recent successful login

Purpose: Centralizes authentication logic while separating it from role-specific information. The Role field determines what interface and permissions each user receives.

b. Student Information

The system must store information about students enrolled in the institution.

Required Attributes:

i. StudentID (Primary Key): Unique integer identifier for database operations
ii. LoginID (Foreign Key, Unique, Not Null): Links to authentication data
iii. RollNo (Unique, Not Null): University roll number (e.g., "21110001")
iv. Program (Not Null): Degree program such as "B'Tech CSE", "M'Tech EE"
v. Batch (Not Null): Year of admission (e.g., 2021)

Purpose: Stores student-specific academic information. StudentID serves as the technical key for relationships, while RollNo is the human-readable identifier used in daily operations.

c. Instructor Information

The system must store information about faculty members who teach courses.

Required Attributes:

i. InstructorID (Primary Key): Unique integer identifier
ii. LoginID (Foreign Key, Unique, Not Null): Links to authentication data
iii. Department (Not Null): Academic department such as "CSE", "EE", "ME"

Purpose: Stores instructor-specific information separate from authentication logic. Department and designation support administrative processes.

d. Course Catalog Data

The system must maintain a catalog of all courses offered by the institution, independent of when they are taught.

Required Attributes:

i. CourseID (Primary Key): Unique integer identifier
ii. CourseCode (Unique, Not Null): Short identifier like "CS432", "MA101"
iii. CourseName (Not Null): Full descriptive name like "Databases"
iv. Credits (Not Null): Integer representing credits (typically 2, 3, or 4)

Purpose: Serves as the permanent course catalog. Courses exist here even when not currently offered.

e. Semester Data

The system must define academic time periods during which courses are taught.

Required Attributes:

i. SemesterID (Primary Key): Unique integer identifier
ii. Term (Not Null): Designation like "Semester 1 2025-26", "Semester 2 2025-26"
iii. Year (Not Null): Calendar year as integer
iv. StartDate (Not Null): Semester beginning date
v. EndDate (Not Null): Semester ending date

Constraint: EndDate must be after StartDate. These dates define the valid window for course offerings and attendance sessions.

f. Course Offering Data

The system must track specific instances of courses being taught in particular semesters and sections.

Required Attributes:

- OfferingID (Primary Key): Unique integer identifier
- CourseID (Foreign Key, Not Null): Which course is being offered
- SemesterID (Foreign Key, Not Null): When the offering occurs
- Section (Not Null): Section identifier like "A", "B", "C"

Purpose: Bridges courses and semesters with section information. One course in one semester can have multiple sections, each being a distinct offering.

g. Enrollment Data

The system must track which students are enrolled in which course offerings.

Required Attributes:

i. EnrollmentID (Primary Key): Unique integer identifier

ii. StudentID (Foreign Key, Not Null): Which student is enrolling
iii. OfferingID (Foreign Key, Not Null): Which course offering they're enrolling in
iv. EnrolledOn (Not Null): Date of enrollment

Constraint: Unique (StudentID, OfferingID) prevents duplicate enrollments in the same offering. Students can enroll in the same course in different semesters (different OfferingIDs).

h. Teaching Assignment Data

The system must track which instructors are assigned to teach which course offerings.

Required Attributes:

i. AssignmentID (Primary Key): Unique integer identifier
ii. InstructorID (Foreign Key, Not Null): Which instructor is assigned
iii. OfferingID (Foreign Key, Not Null): Which course offering they're teaching
iv. AssignedOn (Not Null): Date of assignment

Constraint: Unique (InstructorID, OfferingID) prevents duplicate assignments. One instructor can teach multiple offerings; one offering can have multiple instructors.

i. Attendance Session Data

The system must record individual class meetings for which attendance is taken.

Required Attributes:

i. SessionID (Primary Key): Unique integer identifier
ii. OfferingID (Foreign Key, Not Null): Which course offering this session belongs to
iii. SessionDate (Not Null): Date when class occurred
iv. StartTime (Not Null): Class start time
v. EndTime (Not Null): Class end time
vi. CreatedBy (Foreign Key, Not Null): Which instructor created the session

Constraint: EndTime must be greater than StartTime.

j. Attendance Record Data

The system must store individual attendance marks for students in specific sessions.

Required Attributes:

i. RecordID (Primary Key): Unique integer identifier
ii. SessionID (Foreign Key, Not Null): Which session this mark applies to
iii. StudentID (Foreign Key, Not Null): Which student this mark is for
iv. Status (Not Null): Either "PRESENT" or "ABSENT"
v. MarkedAt (Not Null): Timestamp when attendance was recorded

Constraint: Unique (SessionID, StudentID) ensures each student has exactly one record per session.

k. Correction Request Data

The system must manage the workflow for disputing and correcting attendance records.

Required Attributes:

i. RequestID (Primary Key): Unique integer identifier
ii. RecordID (Foreign Key, Not Null): Which attendance record is disputed
iii. StudentID (Foreign Key, Not Null): Which student is submitting the request
iv. Reason (Not Null): Text explanation of why the mark is believed incorrect
v. Status (Not Null): "PENDING", "APPROVED", or "REJECTED"
vi. RequestedAt (Not Null): Timestamp of request submission
vii. ReviewedBy (Foreign Key, Nullable): Which instructor reviewed it (NULL when pending)

Purpose: Supports the correction workflow from submission through review to resolution.

B. Funtional Requirements

The system supports five core functionalities.

a. User Authentication and Authorization

The system provides a secure login-based access mechanism with role-based authorization control.

1) The system shall authenticate users using unique login credentials.
2) The system shall allow registered users to securely log in and log out of the system.
3) The system shall maintain distinct user roles, including Student, Instructor, and Administrator.
4) The system shall enforce role-based access control, allowing users to access only those modules and operations permitted for their role.
5) The system shall manage user sessions and automatically terminate inactive sessions after a predefined timeout period.
6) The system shall record login activity, including the most recent login timestamp for each user.

b. Course Management

The system enables comprehensive management of academic courses and their structured offerings across semesters.

1) Maintains a centralized catalog of all university courses, each identified by a unique course code and associated metadata.
2) Support multiple semesters (Spring (Sem2), Fall (Sem1), Summer (additional sem.) with date boundaries.
3) Facilitates the creation of course offerings by linking specific courses to designated semesters.
4) Allows multiple sections (such as A, B, C) for the same course within a semester to accommodate parallel teaching.
5) Tracks enrollment capacity and monitors the current enrollment count for each course offering
6) Stores essential course metadata, including credits, department, course type, and descriptive information.

c. Enrollment and Teaching Assignment.

This module manages student enrollments and instructor allocations for specific course offerings.

1. Enables students to enroll in multiple course offerings across semesters.
2. Ensures that duplicate enrollments are prevented, avoiding multiple registrations for the same offering.
3. Records enrollment details, including the date of enrollment and the current enrollment status (active or dropped).
4. Supports assignment of one or more instructors to a course offering, allowing co-teaching scenarios where applicable.
5. Maintains clear instructor–offering associations to ensure accountability and structured teaching allocation.

d. Attendance Tracking

This module delivers a structured and intelligent mechanism for session-based attendance management across multiple course offerings.

1) Enables instructors to schedule attendance sessions tied to specific course offerings, defining the exact date and time of each academic interaction.
2) Capture essential session details, including session date, start time, and end time, ensuring precise lecture-level tracking.
3) Ensures temporal consistency by validating that each session's end time occurs after its start time.
4) Associates for every attendance session with the corresponding academic semester, maintaining alignment within defined term boundaries.
5) Automatically prepares attendance entries for all actively enrolled students in a course offering, streamlining the marking process.
6) Allows instructors to record attendance status for each student in a session, supporting classifications such as Present, Absent, or Late.
7) Maintains session-level uniqueness by ensuring that each student receives only one attendance record per session.
8) Supports multiple instructional formats—including Lectures, Labs, and Tutorials—allowing flexible academic structure.

e. Attendance Correction Workflow

Structured process for handling attendance disputes

1) Allows students to raise correction requests for specific attendance records when discrepancies are identified.
2) Requires each request to include a clear textual justification, ensuring that submissions are meaningful and traceable.
3) Enables instructors to review submitted requests and take an informed decision—either approving or rejecting the request—with optional remarks.
4) Automatically updates the associated attendance record upon approval, ensuring consistency without manual intervention.
5) Preserves the original record when a request is rejected, maintaining data integrity.
6) Captures reviewer identity and decision timestamp, creating accountability within the workflow.
7) Supports multiple correction attempts, allowing students to resubmit requests if previous ones are rejected.
8) Maintains a complete audit trail of all correction activities, ensuring transparency, traceability, and historical record preservation.

f. Attendance Reporting and Analytics

This module transforms raw attendance records into meaningful academic insights, enabling data-driven monitoring and decision-making.

1. Generates comprehensive attendance statistics, including total sessions conducted, sessions attended, and sessions missed for each student.
2. Computes attendance percentages for every student–course combination, providing a clear measure of academic participation.
3. Presents detailed session-wise attendance histories, allowing transparent tracking over time.
4. Enables dynamic filtering of attendance data based on date range, course, semester, or section for focused analysis.
5. Identifies students whose attendance falls below predefined academic thresholds, facilitating timely intervention.
6. Produces structured attendance reports in standardized formats such as PDF and CSV for documentation and administrative use.

III. ER Diagram Notation and Visual Element

The ER diagram follows the following visual conventions:-

A) Entities

i. Strong Entities: Represented as single rectangles. The following entities are shown as single rectangles, indicating they can exist independently:
- STUDENT
- INSTRUCTOR
- COURSE
- SEMESTER
- ATTENDANCE_SESSION

ii. Weak Entity: Represented as double rectangle.

- COURSE_OFFERING is shown with a double rectangle border, indicating it has existence dependency on its owner entities (COURSE and SEMESTER)

B) Relationships

i. Regular Relationships: Represented as single diamonds. The following relationships are shown as single diamonds:
- ENROLLS_IN (Student enrolls in course offerings)
- TEACHES (Instructor teaches course offerings)
- HAS (Course offering has attendance sessions)
- CREATES (Instructor creates attendance sessions)
- ATTENDS (Student attends sessions)
- REQUESTS_CORRECTION (Student requests correction on attendance)
- REVIEWS (Instructor reviews correction requests)

ii. Identifying Relationships: Represented as double diamonds. The following relationships are shown with double diamond borders, connecting owner entities to the weak entity:
- OFFERED_AS (Course to Course Offering)
- OCCURS_IN (Semester to Course Offering)

C) Attributes

i. Key Attributes: Ovals with underlined text. Primary key attributes are displayed as ovals containing underlined text:
- StudentID (for STUDENT)
- InstructorID (for INSTRUCTOR)
- CourseID (for COURSE)
- SemesterID (for SEMESTER)
- SessionID (for ATTENDANCE_SESSION)
- OfferingID (for COURSE_OFFERING)

ii. Partial Key Attribute: Dashed oval with underlined text.
- Section attribute is shown as a dashed oval with underlined text, identifying it as the partial key (discriminator) for the weak entity COURSE_OFFERING

iii. Regular Attributes: Normal ovals with plain text. Descriptive attributes are shown as regular ovals connected to their parent entities:

- For STUDENT: RollNo, Program, Batch
- For INSTRUCTOR: Department, Designation
- For COURSE: CourseCode, CourseName, Credits
- For SEMESTER: Term, Year, StartDate, EndDate
- For ATTENDANCE_SESSION: SessionDate, StartTime, EndTime

iv. Relationship Attributes: Ovals connected to relationship diamonds. Attributes that belong to relationships rather than entities:
- EnrolledOn (connected to ENROLLS_IN)
- AssignedOn (connected to TEACHES)
- Status, MarkedAt (connected to ATTENDS)
- Reason, Status, RequestedAt (connected to REQUESTS_CORRECTION)

D) Structural Constraints

i. Cardinality Ratios: Represented as labels on edges. Cardinality labels (1, M, or N) are placed on connection lines near relationship diamonds to indicate the maximum number of relationship instances an entity can participate in:
- 1 indicates "one" (maximum of one relationship instance)
- M or N indicates "many" (zero or more relationship instances)

ii. Participation Constraints: Represented by line thickness.
- Double lines indicate total participation (mandatory participation - entity must participate in the relationship)
- Single lines indicate partial participation (optional participation - entity may or may not participate)

IV. Examples of total participation in the diagram:

- COURSE_OFFERING must participate in both OFFERED_AS and OCCURS_IN (existence dependency)
- COURSE_OFFERING must have at least one instructor (TEACHES relationship)
- ATTENDANCE_SESSION must belong to a course offering (HAS relationship)
- ATTENDANCE_SESSION must be created by an instructor (CREATES relationship)

V. Examples of partial participation in the diagram:

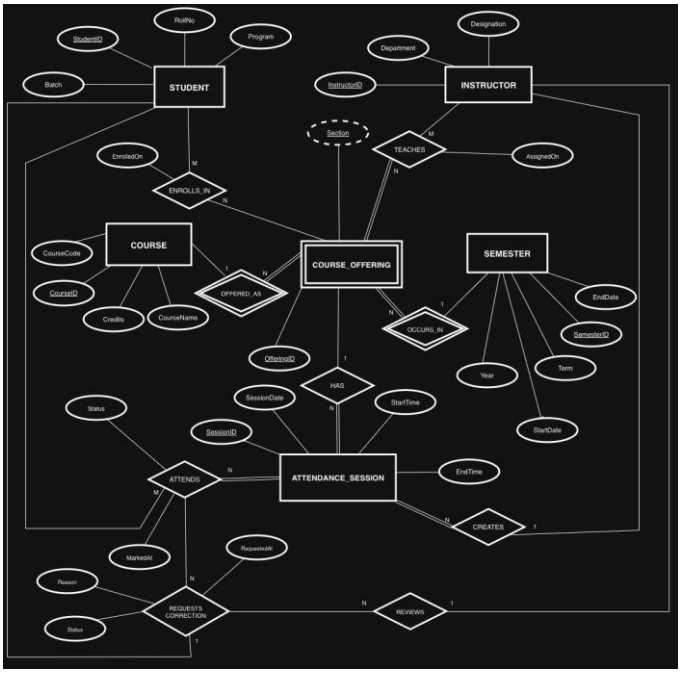
- Not all students are currently enrolled in courses (ENROLLS_IN)
- Not all instructors are currently teaching (TEACHES)
- Not all courses are currently being offered (OFFERED_AS)
- Not all attendance records have correction requests (REQUESTS_CORRECTION)

E) Connection Line Types

- Entity-to-Relationship connections: Lines from entity rectangles to relationship diamonds
- Attribute-to-Entity connections: Lines from attribute ovals to entity rectangles
- Attribute-to-Relationship connections: Lines from attribute ovals to relationship diamonds

VI. ER Diagram

The Entity-Relationship diagram has been shown below. It has been uploaded to https://drive.google.com/file/d/1QXJaciOZ9Qxd1lrN68uU_VDlTHtM6ZLQ/view?usp=sharing for a better image quality.



VII. Initial Entity Type Identification

Based on the requirements described above, we identified the fundamental entity types for the Attendance Management Portal. Following the principle that "nouns appearing in the narrative tend to give rise to entity type names", we extracted candidate entities from the requirements of specification as follows:-

1) *Students* who are enrolled in courses and attend classes.
2) *Instructors* who teach courses and mark attendance.
3) *Courses* offered by the institution.
4) *Semesters* during which courses are taught.
5) *Course offerings* (specific instances of courses in particular semesters).
6) *Attendance* sessions representing individual class meetings.

VIII. Refined Entity Type Selection

As entity identification is an iterative process, we now refine our initial entity candidates by determining which concepts should be modeled as entities versus relationships or attributes.

A. Entity 1: STUDENT

It represents a real-world person with physical existence, an individual enrolled in the institution. Each student has an independent existence, regardless of whether they are currently enrolled in courses. A student entity persists across multiple semesters and multiple enrollments.

Key Attributes:

- StudentID (key attribute): Unique identifier for each student
- RollNo: University-assigned roll number (also unique)
- Program: Degree program (e.g., "BTech CSE")
- Batch: Year of admission

Attributes analysis:

The Name attribute was initially considered but determined to be insufficient as a sole identifier because multiple students may share the same name. Following the principle that "an entity type usually has one or more attributes whose values are distinct for each individual entity", we designated StudentID as the key attribute. RollNo serves as an alternative key, which is a human-readable identifier that uniquely identifies students in day-to-day operations.

Design Decision:

We initially considered including login credentials (username, password) as attributes of STUDENT. However, following the refinement principle that "an attribute that exists in several entity types may be elevated to an independent entity type", and recognizing that authentication logic should be separated from academic information, we decided that login information should not be part of the STUDENT entity. Instead, authentication will be handled through attributes directly associated with the student (stored separately in the implementation phase).

B. Entity 2: INSTRUCTOR

It represents a real-world person with physical existence which is a faculty member employed by the institution. Like students, instructors have independent existence and persist across multiple semesters regardless of their current teaching assignments.

Key Attribute:

a. InstructorID (key attribute): Unique identifier.
b. Department: Academic department affiliation.
c. Designation: Academic rank (Professor, Associate Professor, etc.)

Design Decision:

The choice of attributes for INSTRUCTOR follows the same principles as STUDENT. We focus on attributes that are intrinsic to the instructor as an individual, avoiding attributes that represent relationships (such as "courses taught" or "sessions created"), which will be modeled as explicit relationships.

Attributes analysis:

InstructorID (key): Unique identifier for each faculty member. Makes it easy to reference instructors without worrying about name changes. Department: Which academic department they belong to (CSE, EE, etc.). Important for assigning courses and administrative tasks. Designation: Their positions are like "Professor", "Associate Professor", or "Assistant Professor". Might be used later for access control or approval of workflows.

C. Entity 3: COURSE

COURSE represents an abstract concept with conceptual existence, a defined academic course in the institution's catalog. A course exists independently of whether it is currently being taught. For example, "CS432 - Databases" exists as a permanent catalog of entry even during semesters when it is not offered.

Key Attributes:

a. CourseID (key attribute): Unique identifier.
b. CourseCode: Short code (e.g., "CS432") (also unique).
c. CourseName: Full descriptive name (e.g., "Databases").
d. Credits: Credit hours assigned to the course.

This was chosen to be an entity and not as an attribute because one might consider representing courses simply as attributes of other entities. However, courses have multiple properties (code, name, credits) and participate in relationships with multiple other entities (students, instructors, semesters). Following the principle that concepts with multiple descriptive properties and multiple relationships should be modeled as entities, COURSE qualifies as an entity type.

Attribute analysis:

CourseID (key): System-generated unique ID. Never changes even if course codes are updated. CourseCode: The short code everyone uses, like "CS432" or "MA101". This is what appears on timetables and transcripts. CourseName: Full name "Database Management Systems". Helps when the code alone isn't clear. Credits: Number of credits (usually 1-4). Can be used to calculate total course load and graduation requirements.

D. Entity 4: SEMESTER ID

SEMESTER represents a distinct time period with conceptual existence. Each semester is a defined period during which academic activities occur. Semesters exist as organizational units independent of which courses are offered during them.

Key Attribute:

a. SemesterID (key attribute): Unique identifier

b. Term: Semester designation (e.g., "Semester 1 2025-26")
c. Year: Calendar year
d. StartDate: Beginning date of the semester
e. EndDate: Ending date of the semester

Attribute analysis:

SemesterID (key): Unique identifier for each semester. Simpler than using a combination of term and year. Term: Which semester it is, like "Semester 1 2025-26" or "Summer 2025". Year: Calendar year (2025, 2026, etc.). Makes it easier to sort and filter semesters. StartDate and EndDate: When the semester begins and ends. Essential for validating attendance dates and enrollment periods.

E. Entity 5: COURSE_OFFERING

COURSE_OFFERING represents a specific instance of a course being taught during a particular semester in a designated section.

Initially, one might question whether COURSE_OFFERING should be a relationship between COURSE and SEMESTER rather than an entity.

Multiple Relationships: A course offering doesn't just connect a course to a semester but it also connects to students (enrollment), instructors (teaching), and attendance sessions. If we modeled it as a relationship, we would need a ternary or quaternary relationship, which becomes unwieldy when we try to attach additional relationships to it.

Descriptive Attributes: Course offerings have their own property, the Section identifier (A, B, C, etc.), that distinguishes multiple sections of the same course in the same semester. This attribute belongs to the offering itself, not to the course or semester individually.

COURSE_OFFERING qualifies as a weak entity type because it cannot exist independently.

Key Attributes:

OfferingID (key): A surrogate key we added for convenience. Makes it simple for other tables to reference a specific offering. Section (partial key): Section labels like A, B or C. Distinguishes different sections of the same course in the same semester.

F. Entity 6: ATTENDANCE_SESSION

ATTENDANCE_SESSION represents a specific class meeting event with conceptual existence. Each session corresponds to a real-world occurrence, a particular class that took place on a specific date and time.

Key Attributes:

a. SessionID (key attribute): Unique identifier
b. OfferingID (foreign key): Which course offering the session belongs to
c. SessionDate: Date of the class
d. StartTime: Class start time
e. EndTime: Class end time
f. CreatedBy (foreign key): Which instructor created the session record

Attribute Analysis:

SessionID (key): Unique ID for each class session. Prevents confusion when multiple classes happen on the same day. SessionDate: Date of the class. Used to display attendance records chronologically. StartTime and EndTime: When the class begins and ends. Helps instructors and students verify they're marking attendance for the right session.

Design Decision:

We initially considered creating an ADMIN entity alongside STUDENT and INSTRUCTOR. However, following the principle that "entities should have multiple descriptive attributes and

participate in relationships," we determined that ADMIN should be implemented as a role in the authentication system rather than a separate entity.

Rationale:

- Admin users have no unique attributes beyond authentication credentials
- Admin users do not participate in academic relationships (enrollments, teaching assignments, attendance tracking)
- Admin functionality is purely authorization-based (controlling who can access certain operations)
- Implementing Admin as a role in the LOGIN table (Role = 'ADMIN') provides the necessary access control without creating an empty entity

This design separates authentication/authorization concerns from domain modeling, following best practices in database design. Future extensions that require admin-specific attributes (e.g., permission levels, managed departments) could elevate Admin to a full entity.

IX. Relationship Types

After identifying entity types, we now define the relationships that connect with them. We convert these references into explicit relationships.

A. OFFERED_AS(COURSE – COURSE_OFFERING)

1) *Type:* One-to-Many
2) *Cardinality Rationale:* One course can have multiple offerings across different semesters and sections, but each offering belongs to exactly one course.
3) *Example:* The course "CS432 - Databases" (CourseID=101) might have:
   - Offering 1: CS432 Section A in Semester 1 2025-26
   - Offering 2: CS432 Section B in Semester 1 2025-26
   - Offering 3: CS432 Section A in Semester 2 2025-26
4) *Participation:*
   - COURSE: Partial participation (single line) - Not all courses are currently offered
   - COURSE_OFFERING: Total participation (double line) - Every offering must be associated with a course

This is an identifying relationship for the weak entity COURSE_OFFERING, shown with a double diamond in the ER diagram.

B. Relationship 2: OCCURS_IN (SEMESTER - COURSE_OFFERING)

1) Type: One-to-Many
2) *Cardinality Rationale:* One semester contains multiple course offerings (all courses taught that semester), but each offering occurs in exactly one semester.
3) *Example:* Semester 1 2025-26 (SemesterID=5) contains:
   - CS432 Section A
   - CS101 Section A
   - MA201 Section B
   - And 50+ other course offerings
4) *Participation:*
   - SEMESTER: Partial participation - Some semesters (like planned future semesters) might have no courses yet
   - COURSE_OFFERING: Total participation - Every offering must occur in a semester

This is the second identifying relationship for COURSE_OFFERING (double diamond).

C. ENROLLS_IN (STUDENT - COURSE_OFFERING)

1) *Type:* Many-to-Many
2) *Cardinality Rationale:* A student enrolls in multiple course offerings each semester, and each course offering has multiple students enrolled.
3) *Example:*
   - Student Zainab (StudentID=2021101) is enrolled in: CS432-A, MA201-B, PH101-A (3 offerings)
   - CS432-A has 45 students enrolled: {2021101, 2021102, 2021103, ..., 2021145}
4) Initially, one might model "Enrollment" as a separate entity with attributes {EnrollmentID, StudentID, OfferingID, EnrolledOn}. However, enrollment is fundamentally the association between a student and an offering. The textbook states: "In the initial design of entity types, relationships are typically captured in the form of attributes. As the design is refined, these attributes get converted into relationships." Hence, it is not an entity.

5) *Relationship Attribute:*
   - EnrolledOn (Date): When the student enrolled. This belongs to the relationship because it's determined by the combination of student + offering, not by either entity alone.
6) *Participation:*
   - STUDENT: Partial participation - A student might be registered but not enrolled in courses during a gap semester
   - COURSE_OFFERING: Partial participation - An offering can exist (be scheduled) before any students enroll.

D. TEACHES (INSTRUCTOR - COURSE_OFFERING)

1) *Type:* Many-to-Many
2) *Cardinality Rationale:* An instructor can teach multiple offerings (different courses or multiple sections), and an offering can be taught by multiple instructors (team teaching).
3) *Example:*
   - Prof. Meena (InstructorID=301) teaches: CS432-A, CS432-B, CS534-A (3 offerings)
   - CS432-A is taught by: Prof. Meena and Prof. Mondal(2 instructors - team teaching)
4) It is many to many as although many courses have a single instructor, the system must support:
   - Instructors teaching multiple courses/sections

- Team-taught courses (two instructors co-teaching)
- Lab sections with separate instructors

5) *Relationship Attribute:*
- AssignedOn (Date): When the teaching assignment was made.

6) *Participation:*
- INSTRUCTOR: Partial participation - An instructor might be on vacation or only doing research
- COURSE_OFFERING: Total participation - Every offering must have at least one instructor assigned

E. HAS (COURSE_OFFERING - ATTENDANCE_SESSION)

1) *Type*: One-to-Many
2) *Cardinality Rationale:* One offering has many class sessions throughout the semester, but each session belongs to exactly one offering.
3) *Example:*
- CS432-A (OfferingID=501) has sessions:
- Session 1: Jan 15, 2026, 10:00-11:30
- Session 2: Jan 17, 2026, 10:00-11:30
- Session 3: Jan 20, 2026, 10:00-11:30
- Continues for approximately 40 sessions throughout the semester

4) *Participation:*
- COURSE_OFFERING: Partial participation - An offering exists when scheduled, before any sessions are created
- ATTENDANCE_SESSION: Total participation - Every session must belong to an offering

F. CREATES (INSTRUCTOR - ATTENDANCE_SESSION)

1) *Type:* One-to-Many
2) *Cardinality Rationale:* One instructor creates many attendance sessions, but each session is created by exactly one instructor.
3) *Example:* Prof. Meena creates all sessions for his courses:
- Session for CS432-A on Jan 15
- Session for CS432-B on Jan 16
- Session for CS534-A on Jan 17

4) This relationship exists as it accounts for who opened the attendance marking for each session. It is important for audit trails and resolving disputes about when attendance was opened.
5) *Participation:*
- INSTRUCTOR: Partial participation - Not all instructors create sessions (some might only review corrections)
- ATTENDANCE_SESSION: Total participation - Every session must be created by an instructor

G. Relationship 7: ATTENDS (STUDENT - ATTENDANCE_SESSION)
1) *Type:* Many-to-Many
2) *Cardinality Rationale:* A student attends many sessions across all enrolled courses, and each session is attended by many students.
3) *Example:* Student Zainab attends:
- CS432-A Session on Jan 15: PRESENT
- MA201-B Session on Jan 15: PRESENT
- CS432-A Session on Jan 17: ABSENT
- PH101-A Session on Jan 18: PRESENT

CS432-A Session on Jan 15 has:
- StudentID 2021101: PRESENT
- StudentID 2021102: PRESENT
- StudentID 2021103: ABSENT

All 45 enrolled students have records

4) We could have modeled AttendanceRecord" as an entity with {RecordID, SessionID, StudentID, Status, MarkedAt}. However, this is fundamentally an association - the relationship instance is the attendance record. As for M:N relationship types, some attributes may be determined by the combination of participating entities in a relationship instance, not by any single entity. Such attributes must be specified as relationship attributes.
5) *Relationship Attributes:*
- Status: Values are 'PRESENT' or 'ABSENT' - Determined by the specific student-session combination
- MarkedAt (Timestamp): When attendance was marked for this student in this session

6) *Participation:*
- STUDENT: Partial participation - Students enrolled in a course don't necessarily attend all sessions
- ATTENDANCE_SESSION: Total participation - Every session must have attendance records generated for all enrolled students

H. REQUESTS_CORRECTION (STUDENT - ATTENDS)

1) Type: Many-to-One (N:1)

2) Special Note: This is a relationship on a relationship—a student requests correction for their specific attendance instance.

3) Cardinality Rationale: A student can submit multiple correction requests (for different sessions or resubmissions), but each correction request is for one specific attendance record.

4) Example: Student Zainab (StudentID=2021101) was marked ABSENT for CS432-A on Jan 17. She submits correction request CR001:

For attendance instance: Student 2021101 + Session 215
Reason: "I was present but forgot to mark attendance"
Status: PENDING
RequestedAt: Jan 18, 2026 14:30

5) Relationship Attributes:

- *Reason* (Text): Student's explanation for the correction request
- *Status*: Values are 'PENDING', 'APPROVED', or 'REJECTED'
- *RequestedAt* (Timestamp): When the request was submitted

6) Participation:

- STUDENT: Partial participation - Most students don't request corrections
- ATTENDS: Partial participation - Most attendance records don't have correction requests

Since this is a relationship on a relationship, in the relational schema this will be implemented by referencing the combination of StudentID and SessionID that uniquely identifies an ATTENDS relationship instance.

I. REVIEWS (INSTRUCTOR - REQUESTS_CORRECTION)

1) *Type:* One-to-Many
2) *Cardinality Rationale:* One instructor reviews many correction requests, but each request is reviewed by at most one instructor.
3) *Example:* Prof. Meena reviews correction request CR001:
- Checks session attendance records and timestamps
- Verifies student's claim
- Updates status to APPROVED
- System automatically updates the original ATTENDS record from ABSENT to PRESENT

4) *Participation:*
- INSTRUCTOR: Partial participation - Not all instructors review corrections (only instructors teaching that course)
- REQUESTS_CORRECTION: Partial participation - Some requests are PENDING (not yet reviewed)

X. Additional Constraints

Along with the functionalities which will result as a function of additional constraints and beyond what can be shown in the ER diagram, our system has several rules and constraints that ensure data integrity and match real-world requirements.

A. Enrollment Constraints:- No duplicate enrollments are allowed i.e., a student cannot enroll in multiple sections of the same course in the same semester. Example: If Zainab is already enrolled in CS432 Section A, she cannot also enroll in CS432 Section B in the same semester.

B. Teaching Assignment Constraints

1) Instructor Qualification:- An instructor can only be assigned to teach courses within their department. Example: A CSE department instructor cannot be assigned to teach a Mechanical Engineering course.
2) At Least One Instructor:- Every course offering must have at least one instructor assigned before the semester starts. Before SemesterStartDate, we will verify that each COURSE_OFFERING has at least one TEACHES relationship.

C. Session and Attendance Constraints

1) Session Date Validity:- All attendance sessions for a course offering must have SessionDate between the semester's StartDate and EndDate. Example: If Semester II runs from Jan 1 to April 30, 2026, all sessions must fall within this period.

2) Session Time Validity:- EndTime must be greater than StartTime for every session. Example: A session cannot end at 10:00 AM if it starts at 11:00 AM. Sessions usually last 1-3 hours. We can enforce: (EndTime - StartTime) >= 30 minutes and <= 4 hours.

3) One Session Per Offering Per Time Slot:- A course offering cannot have two sessions scheduled at the exact same date and time. Example: CS432-A cannot have two different sessions both on Jan 15, 2026 at 10:00 AM.

4) Instructor Must Teach the Course:- An instructor can only create attendance sessions for course offerings they are assigned to teach. Example: Prof. Meena (who teaches CS432-A) cannot create attendance sessions for MA201-B.

5) Attendance Records for Enrolled Students Only:- Attendance can only be marked for students who are actually enrolled in that course offering. When creating attendance records for a session, we will only generate records for students with ENROLLS_IN relationship to that offering.
6) One Attendance Record Per Student Per Session
Each student can have only one attendance record (PRESENT or ABSENT) for each session.
Example: Student 2021101 cannot have two different Status values for the same session.

D. Correction Request Constraints

1) Time Window for Requests:- Students can request attendance corrections only within 7 days of the session date. Example: For a session on Jan 15, correction requests must be submitted by Jan 22.

2) One Active Request Per Attendance Record:- A student cannot submit a new correction request for the same session while a previous request is still PENDING. Example: If Zainab has a PENDING request for CS432-A session on Jan 15, she cannot submit another request for the same session until the first is APPROVED or REJECTED.

3) Request After Marking:- Correction requests can only be created after attendance has been marked (MarkedAt timestamp exists).

4) Only Request for ABSENT Records:- Students can only request corrections for sessions where they were marked ABSENT.

E. Review Constraints

1) Instructor Must Teach That Course: - An instructor can only review correction requests for course offerings they teach. Example: Prof. Meena (who teaches CS432-A) cannot review correction requests for MA201-B sessions.

2) Finality of Decision: - Once a correction request is APPROVED or REJECTED, the status cannot be changed. It helps maintain audit trails and prevents endless back-and-forth changes.

F. Data Value Constraints

1) Student Batch:- Batch year must be between (current year - 10) and (current year + 1). Example in 2026: Valid batch years are 2016-2027.

2) Section Labels:- Section identifiers must be single uppercase letters (A-Z) or combinations like "L1" (Lab 1), "T1" (Tutorial 1). Example: Valid: A, B, C, L1, T1. Invalid: section1, abc, 123.

3) Status Values:- Attendance Status can only be 'PRESENT' or 'ABSENT' (no other values allowed). Correction request Status can only be 'PENDING', 'APPROVED', or 'REJECTED'.

4) Semester Term Format:- Term should follow pattern: "Semester [1|2] YYYY-YY" or "Summer YYYY". Example: "Semester 1 2025-26", "Semester 2 2025-26", "Summer 2025".

G. Derived Data Rules

1) Attendance Percentage:- For reporting, attendance percentage = (Number of PRESENT records / Total sessions) × 100. This is calculated, not stored. It changes as more sessions are added.

2) Course Offering Capacity:- While not enforced strictly, course offerings typically have maximum enrollment capacity (e.g., 60 students for a classroom). Hence, system warns when enrollments exceed recommended capacity but doesn't block.

H. Cascading Rules

1) Delete Course Offering:- If a course offering is cancelled (deleted), all related data must be handled which include enrollments are deleted, teaching assignments are deleted, attendance sessions are deleted, all attendance records for those sessions are deleted and correction requests for those records are deleted

2) Delete Semester:- If a semester is deleted (e.g., planned semester cancelled), all course offerings in that semester must be deleted first (along with all their related data).

I. Concurrency Constraints

1) Attendance Marking Window:- Only one instructor can mark/modify attendance for a session at a time to prevent conflicts. Example: If Prof. Meena is marking attendance for CS432-A Session 1, Prof. Mondal (co-instructor) must wait until marking is complete.

2) Enrollment Conflicts:- A student cannot enroll in two course offerings that have overlapping scheduled session times. Example: If CS432-A meets Mon-Wed 10-11 AM and MA201-B also meets Mon-Wed 10-11 AM, a student cannot enroll in both.

J. Category Key Constraints

1) Sessions Must fall within semester dates; EndTime > StartTime

2) Attendance Only for enrolled students; one record per student per session

3) Corrections Within 7 days; only for ABSENT records; one pending request at a time

4) Reviews Only by course instructor; final once decided

These constraints ensure data quality and match real-world academic rules that cannot be fully represented in the ER diagram structure alone.

XI. Contribution

Name	Roll No.	Contribution
Rishabh Jogani	23110276	SQL DUMP (Module A), SQL Designing
Tejas Lohia	23110335	Report, functional req., Constraints, SQL Designing
Zainab Kapadia	23110373	Report, data requirements, ER Diagram, SQL Designing
Siddharth Rajandekar	23110310	Report, SQL Schema Designing
Umang Shikarvar	23110301	SQL Schema Designing

XII. References

[1] Ramez, Elmasri, Shamkant, and Navathe, *Fundamentals of Database System*. Pearson Education India, 2010.
[2] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*. McGraw-Hill Science, Engineering & Mathematics, 2006.