INSTITUTION OF TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

Course Title: Advanced Database Systems

Course code: SEng2041
Group Project

Project Title: Online Course Registeration and Learning Management System

Phase 1

SECTION 1
GROUP 1

Name	ID
1. YEABTSEGA TESFAYE	WDU161299
2. TESHOME SISAY	WDU161207
3. ETSEGENET DAGNACHEWU	WDU160512
4. GETASIL SETEGN	WDU160614
5. TAMENECH MISSA	WDU161169
6. SAMRAWIT MOLLA	WDU161055

Submitted to instructor:Mr. Seid Mohammed

Submission date: 05/11/2025 G.C

Woldia, Ethiopia

Table of content

Phase 1.1: Requirements Analysis & Scoping

Project Overview

The Online Course Registration and Learning Management System is a sophisticated, enterprise-level platform designed to revolutionize the educational experience in academic institutions. This comprehensive system serves as a centralized hub that seamlessly integrates course registration, content delivery, assessment management, and academic progress tracking. In today's digital education landscape, institutions face significant challenges in managing growing student populations, diverse course offerings, and the increasing demand for flexible learning options. This system addresses these challenges by providing a robust, scalable, and user-friendly platform that bridges the gap between students, instructors, and administrative staff.

The system's primary purpose is to streamline the entire academic lifecycle, from initial course discovery and registration through content delivery, assignment submission, grading, and final performance evaluation. It eliminates traditional bottlenecks associated with manual registration processes, paper-based assignments, and disconnected communication channels between stakeholders. By digitalizing these core educational processes, the platform enhances operational efficiency, improves data accuracy, and provides valuable insights through comprehensive reporting capabilities.

Beyond basic functionality, the system incorporates advanced features such as real-time capacity monitoring, automated prerequisite validation, intelligent conflict detection, and progressive learning path tracking. It supports various learning modalities including traditional classroom settings, hybrid models, and fully online courses, making it adaptable to evolving educational trends. The platform also emphasizes data security and privacy, ensuring that sensitive student information and academic records are protected through robust encryption and access control mechanisms.

Main Entities and Attributes

1. Student

- student_id (Primary Key)
- first_name
- last_name
- email
- phone_number

· date_of_birth
· address
· enrollment_date
· major_department_id (Foreign Key)
· academic_status (Active, Suspended, Graduated, Withdrawn)
· total_credits_earned
· cumulative_gpa
· emergency_contact_name
· emergency_contact_phone

2. Instructor

· instructor_id (Primary Key)
· first_name
· last_name
· email
· office_location
· office_hours
· phone_extension
· hire_date
· department_id (Foreign Key)
· specialization
· academic_rank (Professor, Associate Professor, Assistant Professor, Lecturer)
· employment_status (Full-time, Part-time, Adjunct)

3. Course

· course_id (Primary Key)
· course_code
· course_name
· description
· credits
· department_id (Foreign Key)
· instructor_id (Foreign Key)
· prerequisite_course_id (Foreign Key, self-referencing)
· max_capacity
· current_enrollment
· semester (Fall, Spring, Summer)
· academic_year
· course_level (Undergraduate, Graduate)
· delivery_mode (In-person, Online, Hybrid)
· schedule_info (Days, Time, Location)
· course_status (Active, Archived, Cancelled)

4. Department

· department_id (Primary Key)
· department_name
· department_code
· head_instructor_id (Foreign Key)
· office_location
· contact_emailrollment_date
· enrollment_status (Registered, Waitlisted, Dropped, Completed)
· final_grade
· grade_points
· completion_date
· attendance_percentage

5. Enrollment

· enrollment_id (Primary Key)
· student_id (Foreign Key)
· course_id (Foreign Key)
· enrollment_date
· enrollment_status (Registered, Waitlisted, Dropped, Completed)
· final_grade
· grade_points
· completion_date
· attendance_percentage

6. Assignment

· assignment_id (Primary Key)
· course_id (Foreign Key)
· title
· description
· assignment_type (Homework, Quiz, Project, Exam, Presentation)
· max_points

7. Submission

· submission_id (Primary Key)

· assignment_id (Foreign Key)
· student_id (Foreign Key)
· submission_date
· submission_status (Submitted, Late, Resubmitted)

8. Grade

· grade_id (Primary Key)
· submission_id (Foreign Key)
· points_earned
· grader_id (Foreign Key to Instructor)
· grade_status (Graded, Regrade Requested, Finalized)

9. Announcement

· announcement_id (Primary Key)
· course_id (Foreign Key)
· instructor_id (Foreign Key)
· title
· content
· post_date
· expiry_date

10. LearningMaterial

· material_id (Primary Key)
· course_id (Foreign Key)
· title
· description
· material_type (Lecture Notes, Video, Reading, Slide Deck, Reference)

Business Rules

Enrollment Rules:

I. **Prerequisite Enforcement**: Students cannot enroll in a course without completing all prerequisite courses with a minimum grade of 'D'

II. **Capacity Limits**: Course enrollment cannot exceed the maximum capacity defined for each course
III. **Time Conflict Prevention**: Students cannot enroll in courses with overlapping schedules
IV. **Credit Limit**: Undergraduate students cannot exceed 34 credits per semester without special approval
V. **Academic Standing**: Students on academic probation cannot enroll in more than 12 credits
VI. **Major Restrictions**: Some courses are restricted to students within specific majors until a designated date

Grading Rules:

I. **Grade Validation**: Final grades must be within the defined grading scale (A, B, C, D, F)
II. **Submission Deadline**: Assignments submitted after the due date receive penalties according to description
III. **Grade Finalization**: Grades cannot be modified after the grade submission deadline without department chair approval
IV. **Incomplete Policy**: 'I' (Incomplete) grades must be resolved within one academic year

Administrative Rules:

I. **Course Creation:** New courses require department approval and curriculum committee review
II. **Instructor Assignment:** Courses must have assigned instructors before the registration period begins
III. **Department Hierarchy:** Each department must have one head instructor who oversees course offerings
IV. **Data Retention:** Student records must be maintained for 7 years after graduation or last enrollment

System Rules:

I. **Unique Constraints:** Email addresses must be unique across all user types
II. **Data Validation:** Phone numbers must follow standard formatting rules
III. **Access Control:** Users can only access data relevant to their roles and permissions
IV. **Audit Trail:** All grade modifications and enrollment changes must be logged

Functional Requirements

Student Management Module:

1. Student Registration
- · Capture comprehensive student demographic information
- · Validate email format and uniqueness
- · Generate unique student identification numbers
- · Set initial academic status and track changes

2. Profile Management
- · Allow students to update personal contact information
- · Enable password changes and security preferences
- · Display academic progress and standing
- · Provide transcript and grade history access

Course Management Module:

1. Course Catalog
- · Display available courses with search and filter capabilities
- · Show real-time enrollment numbers and capacity
- · Provide detailed course descriptions and requirements
- · Display instructor profiles and ratings

2. Course Creation and Maintenance
- · Allow authorized staff to create new course offerings
- · Enable course information updates and modifications
- · Manage course prerequisites and corequisites
- · Handle course cancellation and archival processes

Enrollment Module:

1. Registration System
- · Provide intuitive course search and selection interface
- · Implement shopping cart functionality for course planning
- · Validate enrollment against business rules in real-time
- · Handle waitlist management and automatic promotion

2. Schedule Management
- · Generate conflict-free student schedules
- · Display timetable with visual calendar representation

· Allow schedule adjustments during add/drop periods
· Provide calendar export functionality

Learning Management Module:

1. Content Delivery
· Support multiple content types (documents, videos, links)
· Organize materials by week or topic
· Track student access and engagement with materials
· Provide mobile-friendly content viewing
2. Assignment Management
· Facilitate assignment creation with detailed rubrics
· Support multiple submission types (file upload, text entry)
· Implement submission deadline enforcement
· Provide plagiarism detection integration
3. Gradebook
· Calculate running course grades based on weighted assignments
· Allow instructors to enter and modify grades
· Provide students with real-time grade visibility
· Generate grade distribution analytics

Communication Module:

1. Announcement System
· Enable targeted announcements to specific courses or user groups
· Support rich text formatting and file attachments
· Provide read receipt tracking
· Implement announcement scheduling
2. Discussion Forums
· Facilitate course-specific discussion threads
· Support moderated and unmoderated discussions
· Enable file sharing within discussions
· Provide notification of new posts

Use Case Diagrams and Descriptions

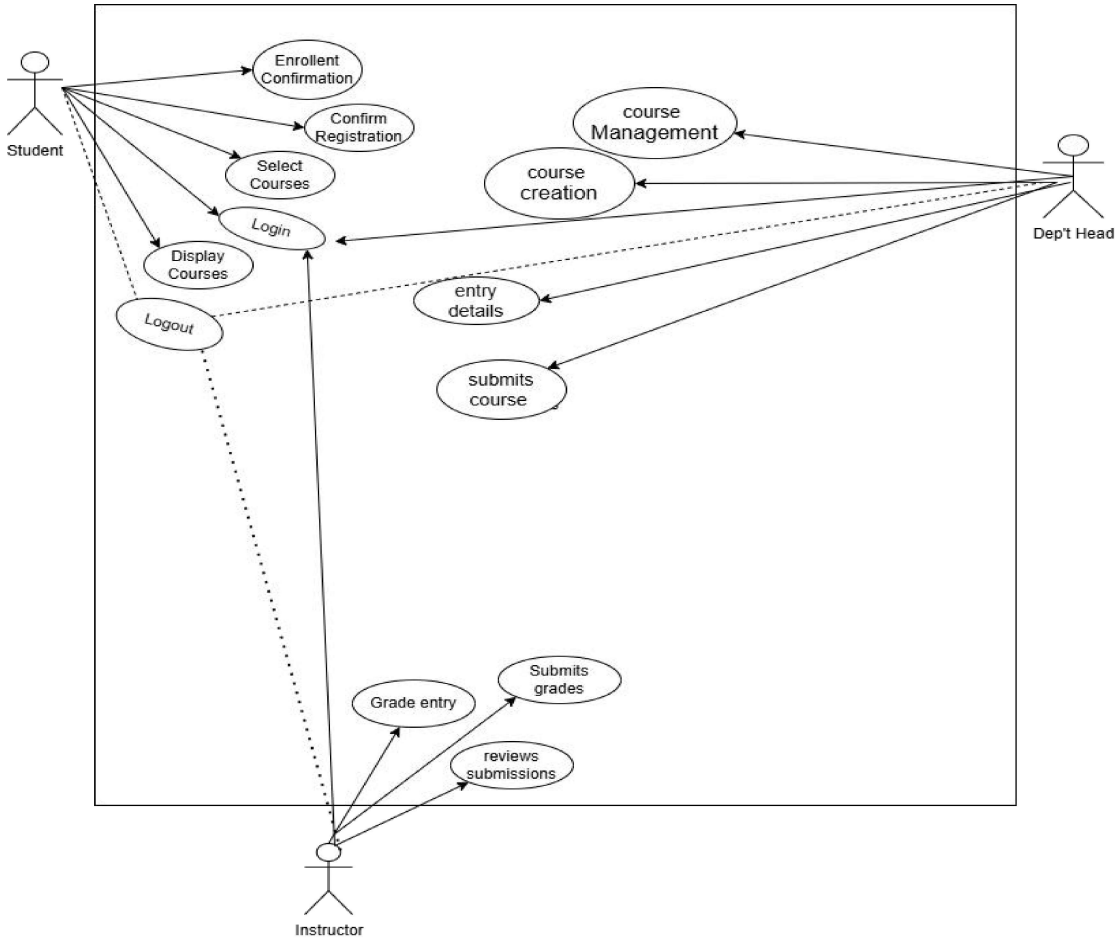


Fig. Use Case

Use cases are typically represented using UML Use Case Diagrams, which visually depict the interactions between actors (users) and the system. Below are the detailed use cases:

Primary Use Case Diagram Components:

Actors:

· Student
· Instructor
· Administrator
· Department Head
· System (automated processes)

Key Use Cases:

1. Student Registration Use Case

· Primary Actor: Student
· Preconditions: Student has valid institutional credentials
· Main Flow:
  1. Student accesses registration system
  2. System displays available courses
  3. Student selects desired courses
  4. System validates selections against business rules
  5. System confirms successful registration
  6. System generates enrollment confirmation
· Alternative Flows: Waitlist handling, prerequisite errors, time conflicts
· Postconditions: Student is registered for selected courses

2. Grade Submission Use Case

· Primary Actor: Instructor
· Secondary Actor: System
· Preconditions: Assignment submission period has ended
· Main Flow:
  1. Instructor accesses grade entry interface
  2. System displays student submissions
  3. Instructor reviews submissions and enters grades
  4. System validates grade format and ranges
  5. Instructor submits grades
  6. System updates student records and calculates overall grades
· Alternative Flows: Late submissions, regrade requests
· Postconditions: Grades are recorded in student transcripts

3. Course Creation Use Case

· Primary Actor: Department Head
· Preconditions: Department head is authenticated and authorized
· Main Flow:
  1. Department head accesses course management
  2. System displays course creation form
  3. Department head enters course details and requirements
  4. System validates course information
  5. Department head submits course for approval
  6. System routes course for curriculum committee review
· Alternative Flows: Course modification, course cancellation
· Postconditions: New course is available in catalog (pending approval)

User Roles

1. Student

Responsibilities:

· Register for courses and manage schedule
· Access course materials and learning resources
· Submit assignments and view grades
· Participate in discussions and communication
· Track academic progress and performance
· Update personal information and preferences

Permissions:

· Read access to own records and courses
· Write access to own submissions and profile
· Limited access to peer information (directory only)
· No access to administrative functions or other student records

2. Instructor

Responsibilities:

· Create and manage course content
· Develop assignments and assessments
· Grade student submissions and provide feedback
· Communicate with students via announcements
· Monitor student progress and engagement
· Submit final grades

Permissions:

· Full read/write access to assigned courses
· Access to enrolled student information and performance
· Ability to generate course analytics and reports
· No access to student records outside assigned courses

3. Department Head

Responsibilities:

· Oversee course offerings and curriculum
· Approve new courses and modifications
· Assign instructors to courses
· Monitor department enrollment and performance
· Handle student appeals and exceptions
· Manage department resources

Permissions:

· Access to all courses within department
· Ability to override certain system restrictions
· Access to comprehensive department reports
· Approval authority for special requests

4. Administrator

Responsibilities:

· System configuration and maintenance
· User account management and security
· Data backup and recovery operations
· System performance monitoring
· Generate institutional reports
· Handle technical support issues

Permissions:

· Full system access with oversight capabilities
· Ability to modify all data (with audit trail)
· System configuration and parameter settings
· User role and permission management

Non-Functional Requirements

Performance Requirements:

1. Response Time
- Page load times: ≤ 2 seconds for 95% of requests
- Search operations: ≤ 3 seconds for complex queries
- Grade calculation: ≤ 5 seconds for classes of 100 students
- Report generation: ≤ 30 seconds for standard reports

2. Throughput
- Support 10,000 concurrent users during peak registration
- Handle 1,000 simultaneous course enrollments per minute
- Process 500 assignment submissions simultaneously
- Support 200 concurrent video streams for course content

3. Scalability
- Horizontal scaling to support 50% user growth without architecture changes
- Database capable of storing 100,000 student records
- Support for 1000 active courses annually
- Ability to handle 1TB of course materials and submissions

Reliability Requirements:

1. Availability
- 99.5% uptime during academic terms
- Maximum of 4 hours scheduled downtime per month
- Zero data loss for committed transactions
- 15-minute recovery time for critical failures

2. Error Handling
- Graceful degradation during partial system failures
- Comprehensive error logging and monitoring
- Automated alerting for system errors
- User-friendly error messages without technical details

Security Requirements:

1. Data Protection
- Encryption of all sensitive data at rest (AES-256)
- SSL/TLS encryption for all data in transit
- Secure storage of passwords with salted hashing
- Regular security patches and vulnerability assessments

2. Access Control
- Role-based access control (RBAC) with minimum privileges
- Multi-factor authentication for administrative access
- Session timeout after 30 minutes of inactivity
- Comprehensive audit trails for sensitive operations

3. Compliance

· FERPA compliance for student record protection
· GDPR compliance for international students
· WCAG 2.1 AA accessibility standards
· Institutional data governance policies

Usability Requirements:

1. User Experience
· Intuitive navigation with maximum 3 clicks to key functions
· Consistent interface across all modules
· Responsive design for mobile and tablet devices
· Accessibility for users with disabilities
2. Learning Curve
· New students able to register for courses within 10 minutes
· Instructors able to create basic course within 30 minutes
· Comprehensive online help and documentation
· Context-sensitive guidance and tooltips

Maintainability Requirements:

1. System Architecture
· Modular design with clear separation of concerns
· Comprehensive API documentation
· Database schema version control
· Automated deployment pipelines
2. Supportability
· Detailed logging for troubleshooting
· Performance monitoring and alerting
· Backup and restore procedures
· Disaster recovery documentation

Data Management Requirements:

1. Data Integrity
· Referential integrity enforced at database level
· Validation rules for all user inputs
· Atomic transactions for critical operations
· Data consistency across distributed components
2. Backup and Recovery
· Daily automated full backups
· Hourly transaction log backups

· 7-year retention for student records
· Tested restore procedures with 4-hour RTO

Phase 1.2: Conceptual Design – ER Diagram

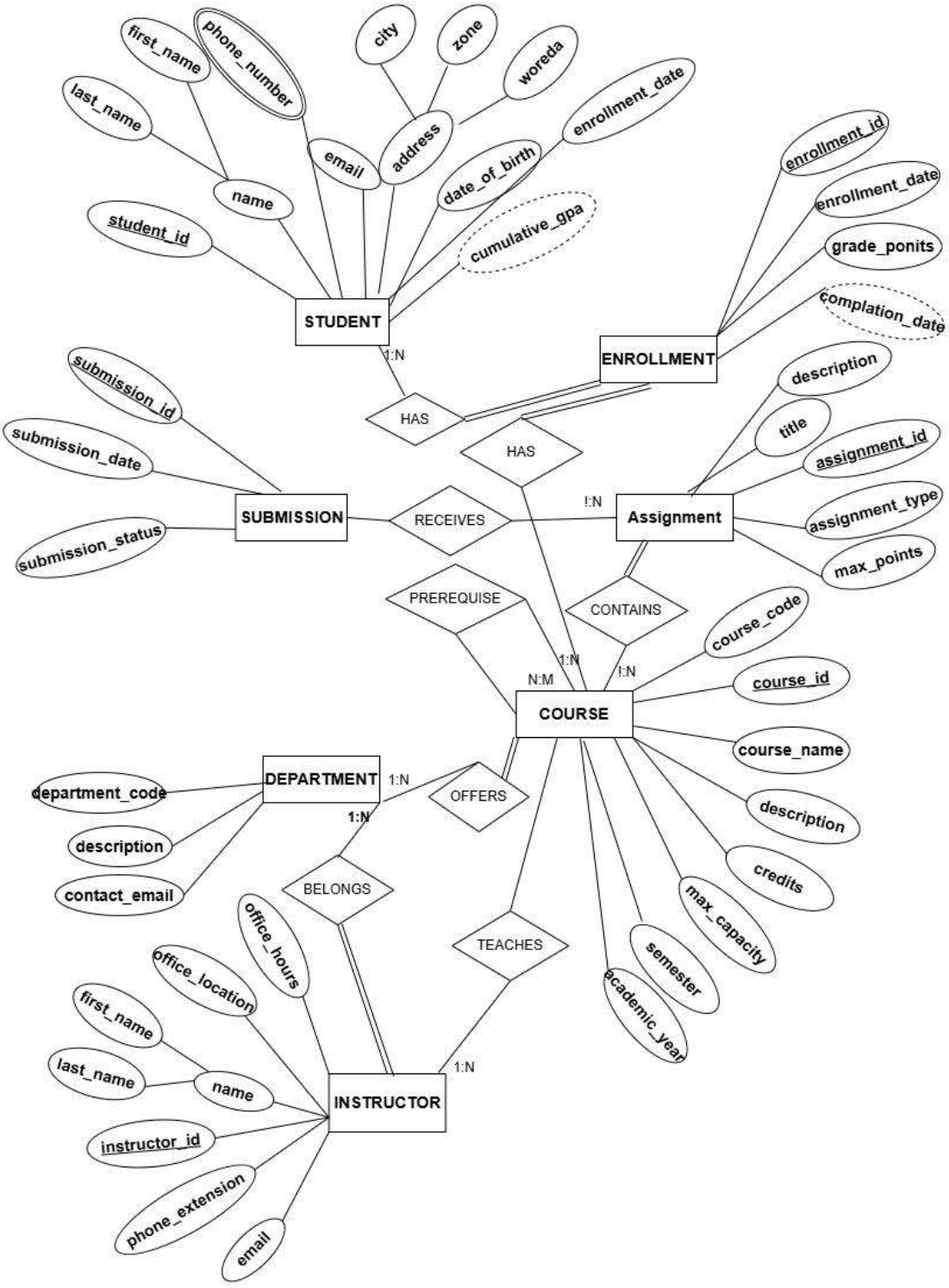


Fig. ER-Diagram

Total Participation:

- ✓ Every Enrollment must be associated with exactly one Student and one Course
- ✓ Every Instructor must belong to exactly one Department
- ✓ Every Course must belong to exactly one Department

Partial Participation:

- ✓ A Student may have zero or more Enrollments
- ✓ An Instructor may teach zero or more Courses
- ✓ · A Course may have zero or more Assignments

Key Constraints

Primary Keys:

- ❖ All entities use surrogate keys (ID fields) as primary keys
- ❖ Composite keys used in associative entities where appropriate

Foreign Keys:

· Department → Instructor (head_instructor_id)
· Student → Department (major_department_id)
· Instructor → Department (department_id)
· Course → Department (department_id)
· Course → Instructor (instructor_id)
· Enrollment → Student (student_id)
· Enrollment → Course (course_id)
· Assignment → Course (course_id)
· Submission → Assignment (assignment_id)
· Submission → Student (student_id)

Business Rules Validation through ERD

The ER diagram ensures:

1. Data Integrity: Foreign key constraints maintain referential integrity
2. Cardinality Enforcement: Relationships reflect real-world constraints
3. Normalization Foundation: Proper entity separation reduces redundancy

4. Scalability: Structure supports future enhancements
5. Query Efficiency: Clear relationships enable optimized database design

This comprehensive ER diagram serves as the blueprint for the relational database schema and will guide the implementation of tables, constraints, and relationships in the subsequent phases.

Phase 1.3: Relational Schema

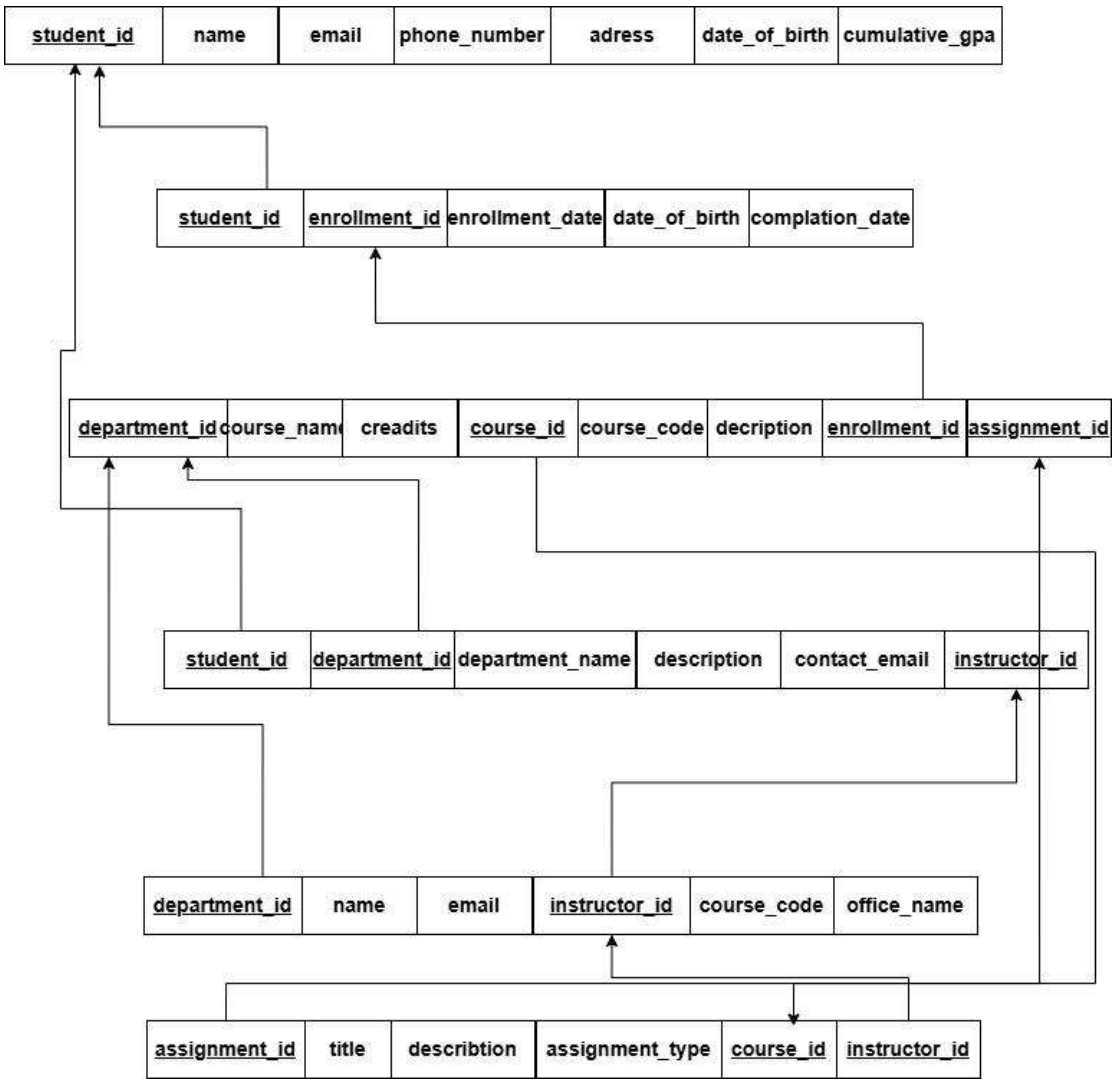


Fig. Relational Schema

Phase 1.4: Normalization

Initial Unnormalized Form (UNF)

student_id	first_name	Last_name	Phone_number	email	Date_of_birth

```

| department_id | department_name | department_code | office_location | contact_email | description |
|---|---|---|---|---|---|

```

instructor_id	first_name	Last_name	office_location	email

```

| address | Enrollment_date | Cumlative_gpa |
|---|---|---|

```

course_id	course_name	Course_code	description	credits

```

| max_capacity | academic_year | semester |
|---|---|---|

```

enrollment_id	enrollment_date	Compllletion_date

```

| description | title | assignment_type | assignment_id | max_points |
|---|---|---|---|---|

```

submission_status	submission_date	submission_id

Problems with UNF:

· Repeated student information for each course
· Repeated course information for each student
· Mixed entity attributes
· Update anomalies
· Insertion anomalies
· Deletion anomalies

First Normal Form (1NF)

Rule: Eliminate repeating groups and ensure atomic values

We separate into distinct tables with atomic values:

1. DEPARTMENT Table

department_id	department_name	department_code	office_location	contact_email	description

2. INSTRUCTOR Table

instructor_id	first_name	Last_name	office_location	email	Department_id

3. STUDENT Table

student_id	first_name	Last_name	Phone_number	email	Date_of_birth

```

| address | Department_id | Enrollment_date | Cumlative_gpa | demartment_name |
|---|---|---|---|---|
| | | | | |

4. COURSE Table

| course_id | course_name | Course_code | description | credits | Department_id |
|---|---|---|---|---|---|
| | | | | | |

```

max_capacity	instructor_id	academic_year	semester

5. ENROLLMENT Table

enrollment_id	Course_id	student_id	enrollment_date	Compllletion_date

6. ASSIGNMENT Table

description	title	assignment_type	assignment_id	max_points	course_id

7. SUBMISSION Table

course_id	submission_status	submission_date	submission_id

1NF Achieved:

- ✓ All attributes are atomic
- ✓ No repeating groups
- ✓ Each table has a primary key

Second Normal Form (2NF)

Rule: Remove partial dependencies (all non-key attributes must depend on the entire primary key)

Checking Partial Dependencies:

ENROLLMENT Table Analysis:

Primary Key: enrollment_id
student_id + course_id → enrollment_date, enrollment_status, etc.

- ✓ All attributes depend on the entire primary key

COURSE Table Analysis:

Primary Key: course_id

course_id → course_name, credits, department_id, instructor_id, etc.

- ✓ All attributes depend on the entire primary key

STUDENT Table Analysis:

Primary Key: student_id
student_id → first_name, last_name, email, major_department_id, etc.

- ✓ All attributes depend on the entire primary key

No partial dependencies found. Our schema is already in 2NF because:

- ✓ All tables have single-column primary keys
- ✓ No composite primary keys that could cause partial dependencies

Third Normal Form (3NF)

Rule: Remove transitive dependencies (no non-key attribute should depend on another non-key attribute)

Checking Transitive Dependencies:

One Transitive Dependency Found

STUDENT Table:

student_id → major_department_id → department_name
Problem: department_name in STUDENT table, it transitively depend on department_id
Solution: Remove department_name and get it by joining with Department table.

DEPARTMENT Table

department_id	department_name	department_code	office_location	contact_email	description

STUDENT Table

student_id	first_name	Last_name	Phone_number	email	Date_of_birth

```

| address | Department_id | Enrollment_date | Cumlative_gpa |
|---|---|---|---|

INSTRUCTOR Table:

· instructor_id → department_id → department_name?
· Problem: If we had department_name in INSTRUCTOR table, it would be transitive
  ✓ Solution: We correctly have only department_id as FK

COURSE Table:

· course_id → department_id → department_name?
· course_id → instructor_id → instructor_name?
  ✓ Solution: We correctly have only foreign keys, not the actual names

**Final 3NF Schema**

1. DEPARTMENT Table

| department_id | department_name | department_code | office_location | contact_email | description |
|---|---|---|---|---|---|

3. INSTRUCTOR Table

| __ instructor_id | first_name | Last_name | office_location | email | Department_id |
|---|---|---|---|---|---|

3. STUDENT Table

| __student_id | first_name | Last_name | Phone_number | email | Date_of_birth |
|---|---|---|---|---|---|

```

address	Department_id	Enrollment_date	Cumlative_gpa

4. COURSE Table

__course_id	course_name	Course_code	description	credits	Department_id

```

| max_capacity | instructor_id | academic_year | semester |
|---|---|---|---|

8. ENROLLMENT Table

| enrollment_id | Course_id | student_id | enrollment_date | Compllletion_date |
|---|---|---|---|---|

9. ASSIGNMENT Table

| description | title | assignment_type | assignment_id | max_points | course_id |
|---|---|---|---|---|---|

10. SUBMISSION Table

| course_id | submission_status | submission_date | submission_id |
|---|---|---|---|

## Normalization Verification

**All Normal Forms Achieved:**

✓ 1NF

· All tables have primary keys
· All attributes are atomic
· No repeating groups

✓ 2NF

· All tables have single-column primary keys
· No partial dependencies possible

✓ 3NF

· No transitive dependencies
· All non-key attributes depend only on the primary key
· Non-key attributes are independent of each other
```

Advantages of Normalization

1. Reduced Data Redundancy
   · Student information stored once, not repeated for each course
   · Course information stored once, not repeated for each student
2. Improved Data Integrity
   · Updates to student email affect only one record
   · Changes to course details propagate automatically
3. Easier Maintenance
   · Modify department information in one place
   · Add new courses without duplicating instructor data
4. Better Query Performance
   · Smaller tables for specific queries
   · Efficient joins when needed
5. Prevention of Anomalies
   · Update Anomaly: Changing a student's phone number updates all their enrollments
   · Insertion Anomaly: Can add a new department without needing courses or students
   · Deletion Anomaly: Deleting a course doesn't delete student information

Disadvantages of Normalization

1. Increased Complexity
   · More tables to manage
   · More joins required for comprehensive queries
2. Performance Overhead
   · Multiple table joins for student course history
   · Additional foreign key constraints
3. Design Complexity
   · Need to understand relationships between tables
   · More complex application logic

For an educational system where data accuracy is critical and read operations outnumber writes, the benefits of normalization outweigh the disadvantages. The simplified 7-table structure maintains efficiency while ensuring data integrity.

Normalized Schema Ready for Implementation

The database design is now fully normalized to 3NF with:

· 7 well-structured tables
· Proper primary and foreign keys
· No redundancy or anomalies
· Efficient relationship management

This normalized design provides a solid foundation for the physical implementation in the next phase while maintaining all business rules and requirements from Phase 1.1.

Phase 1.5: Physical Design & Indexing

Data Types and Constraints

1. ENROLLMENT Table
Primary key: enrollment_ id
Foreign key: student_ id→ STUDENT. student_ id, course_ id→ COURSE. course_ id

column	Data type	Constraints	Description
Enrollment_ id	INT	PRIMARY KEY, IDENTITY (1,1)	Unique enrollment ID
Student_ id	INT	NOT NULL , FORELGN KEY	Student reference
Course_ id	INT	NOT NULL, FORELGN KEY	Course reference
Enrollment_ date	DATETIME2	DEFAULT GETDATE ()	Enrollment date
Enrollment_ status	VARCHER (20)	DEFAULT 'registered' , CHECK (valid Statuses)	Enrollment status
Final_ grade	VARCHER (2)	CHEKED (valid grades)	Final course grade
Grade_ point	DECIMAL (4,2)	CHEKED (0.00-4.00)	Grade point
Completion_ date	DATE	NULL , CHEKED (>= enrollment_ date)	Completion date
Attendance_ persentage	DECIMAL (5,2)	CHEKED (0.00-100.00)	Attendance percentage

2. ASSLGNMENT TABLE

Primary key: assignment _ id

Foreign keys : course _ id→ COURSE. course _ id

Column	Date Type	Constraints	description
Assigniment _ id	INT	PRIMARY KEY, IDENTITY (1,1)	Unique assignment ID
Course _id	INT	NOT NULL , FORELGN KEY	Course reference
Title	VARCHER (200)	NOT NULL	Assignment title
Description	VARCHER (1000)	NULL	Assignment description
Assigniment _type	VARCHER (50)	CHECK(valid types)	Type of assignment
Due _date	DATETIME2	NOT NULL	Due date
Max _points	DECIMAL (5,2)	NOT NULL CHEKD (>0)	Maximum points
Submission _ format	VARCHER (20)	CHEKED (file/Text/both)	Submission format
Allowed _ film _ types	VARCHER (100)	NULL	Allowed file types
Late _ submission _ policy	VARCHER (200)	NULL	Late policy
instruction	VARCHER (MAX)	NULL	Detailed instruction
Weight _ in_ course	DECIMAL (5,2)	CHEKED 0.00-100.00)	Weight in course %

3. INSTRUCTION Table

Primary key : instruction _ id

Foreign keys : department _ id→ department _ id

column	Data type	Constraints	description
Instruction_id	INT	PRAMARY KEY,IDENTITY(1,1)	Unique instruction identifier
First_name	VARCHAR(50)	NOT NULL	First name
Last_ name	VARCHAR(50)	NOT NULL	Last name
Email	VARCHAR(255)	UNLQE,NOT NULL CHECK(email format)	Email address
Office_location	VARCHAR(200)	NULL	Office location

Offic _hours	VARCHAR(100)	NULL	Office hours
Phone_extesion	VARCHAR(10)	NULL	Phone extension
Hire_date	DATE	NOT NULL,CHECK(past date)	Hire date
Department_id	INT	NOT NULL,FOREIGE KEY	Department affiliation
Academic_rank	VARCHAR(50)	CHECK(valid ranks)	Academic rank
Employment_status	VARCHAR(20)	CHECK(employment type)	Employment status

4. DEPARTMENT TABLE

Primary key : department_id

Foreign keys :head_instructor_ id→ INSTRUCTOR.instroctor_ id

column	Data Type	Constraints	description
Department_id	INT	PRAMARY KEY,IDENTITY(1,1)	Unique department identifier
Department_name	VARCHER(100)	NOT NULL	Department name
Department_code	VARCHAR(10)	NOT NULL UNIQUE	Short department code
Head_instructor_id	INT	NULL,FOREGN KEY	Department head
Office_location	VARCHAR(200)	NULL	Physical location
Contact_email	VARCHAR(255)	CHECK(email format)	Contact email
Contact_phone	VARCHAR(20)	NULL	Contact phone
description	VARCHAR(500)	NULL	Department description

5. COURSE TABLE

Primary key:course_id

Foreign keys: department_id→DEPARTMENT_id, instractor _id→INSTRACTOR_.instractor_id

Column	Date type	constraints	Description
Course_id	INT	PRIMARY KEY,IDENTITY(1,1)	Unique course identifier
Course_code	VARCHAR(20)	NOT NULL	Course code
Course_name	VARCHAR(100)	NOT NULL	Course name
Description	VARCHAR(1000)	NULL	Course description
Credits	INT	NOT NULL,CHECK(1-6)	Credit hours
Department_id	INT	NOT NULL,FOREGN KEY	Offering department
instructor_id	INT	NULL,FORELGN KEY	Primary instructor
Max_capacity	INT	DEFALT 0 CHECK(10.500)	Maximum student
Current_enrollment	INT	DEFALT 0 CHECK(0-	Current enrollment

		max_capacity)	
Semester	VARCHAR(20)	CHECK(valid semesters)	Semester offered
Academic_year	INT	CHECK(2000-2030)	Academic year
Course_level	VARCHAR(20)	CHECK(UG/Grad/doctoral)	Course level
Delivery_mode	VARCHAR(20)	CHECK(in-person/online/hybrid)	Delivery method
Schedule_info	VARCHAR(200)	NULL	Schedule details
Course_status	VARCHAR(20)	DEFAULT'active'CHEKD(valid statuses)	Course status

6. STUDENT TABLE

Primary key: student_id

Foreign keys:major_department_id →DEPARTMENT.department_id

Column	Data Type	constraints	description
Student_id	INT	PRAMARY KEY,IDENTITY(1,1)	Unique student identifier
First_name	VARCHAR(50)	NOT NULL	First name
Lasr_name	VARCHAR(50)	NOT NULL	Last name
Email	VARCHAR(255)	UNIQUE,NOT NULL CHECK(email format)	Email address
Phone_number	VARCHAR(20)	NULL	Phone number
Date_of_birth	DATE	NOT NULL CHECK(AGE>=16)	Date of birth
Address	VARCHAR(500)	NULL	Physical address
Enrollment_date	DATE	NOT NULL,DEFAULT GETDATE()	Enrollment date
Major_department_id	INT	NULL,FOREIGN KEY	Major department
Academic_status	VARCHAR	DEFAULT'active',CHECK(valid statuses)	Academic standing
Total_credits_earned	INT	DEFAULT 0,CHECK(>=0)	Total credits
Cumulative_gpa	DECIMAL(3,2)	CHECK(0.00-4.00)	Cumulative GPA

7. SUBMISSION TABLE

Primary key: submission_id

Foreign keys:assignment_id →ASSIGNMENT.assignment_id, STUDENT.student_ID

Column	Data type	constraints	description
Submission_id	INT	PRAMARY KEY,IDENTITY(1,1)	Unique submission
Assignment_id	INT	NOT NULL,FOREIGN KEY	Assignment reference
Student_id	INT	NOT NULL,FOREIGN KEY	Student reference
Submission_DATE	DATETIME2	DEFAULT GETDATE()	Submission timestamp
Submission_conte	VARCHAR(MAX	NULL	Text submission

nt	0		
File_path	VARCHAR(500)	NULL	File path for uplodes
File_name	VARCHAR(255)	NULL	Original file name
File_size	BIGINT	NULL CHECK(>=0)	File size in bytes
Submission_status	VARCHAR(20)	CHECK(sublimitted/late/re subitted)	Submission status
Version_number	INT	DEFAULT 1,CHECK(>=1)	Submission version

Indexing Plan

Primary Indexes (Clustered):

- DEPARTMENT: department_id
- INSTRUCTOR: instructor_id
- STUDENT: student_id
- COURSE: course_id
- ENROLLMENT: enrollment_id
- ASSIGNMENT: assignment_id
- SUBMISSION: submission_id

Secondary Indexes (Non-clustered):

High-Priority Indexes:

1. IX_Enrollment_Student (student_id) - Student course queries
2. IX_Enrollment_Course (course_id) - Course roster queries
3. IX_Student_LastName (last_name, first_name) - Name searches
4. IX_Course_SemesterYear (semester, academic_year, course_status) - Catalog queries
5. IX_Assignment_Course (course_id, due_date) - Course assignment lists
6. IX_Submission_AssignmentStudent (assignment_id, student_id) - Gradebook queries

Medium-Priority Indexes:

1. IX_Instructor_Department (department_id) - Department staff
2. IX_Student_MajorDepartment (major_department_id) - Department majors
3. IX_Submission_Student (student_id, submission_date) - Student submissions
4. IX_Assignment_DueDate (due_date) - Upcoming assignments

Low-Priority Indexes:

1. IX_Student_AcademicStatus (academic_status) - At-risk students
2. IX_Enrollment_Status (enrollment_status) - Registration stats

3. IX_Course_Instructor (instructor_id) - Teaching load

Storage Estimation

Table Size Estimates:

Table Estimated Records Size
DEPARTMENT = 50 100 KB
INSTRUCTOR = 1,000 2 MB
STUDENT 50,000 = 100 MB
COURSE 5,000 = 20 MB
ENROLLMENT 500,000 = 500 MB
ASSIGNMENT 50,000 = 150 MB
SUBMISSION 1,000,000 = 2 GB

Total Storage Requirements:

· Data: ~2.77 GB
· Indexes: ~1.38 GB (50% overhead)
· Total: ~4.15 GB

Performance Expectations

Query Performance Targets:

· Student dashboard loading: < 100ms
· Course registration: < 200ms
· Grade submission: < 150ms
· Assignment submission: < 300ms
· Department reports: < 2 seconds
· Student transcript: < 1 second

Concurrent User Support:

· Peak registration: 10,000 concurrent users
· Normal operations: 2,000 concurrent users
· Assignment submissions: 500 concurrent submissions

Physical Design Decisions

Data Type Rationale:

· VARCHAR for text fields supporting Unicode
· INT for identifiers and counts
· DECIMAL for grades and percentages requiring precision
· DATETIME2 for timestamps with high precision
· DATE for simple dates without time

Constraint Strategy:

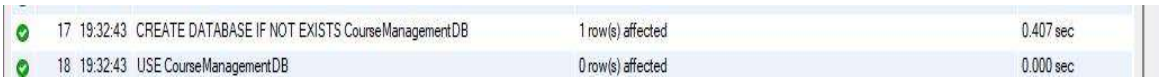
· NOT NULL for essential business data
· CHECK constraints for domain validation
· UNIQUE constraints for business key integrity
· FOREIGN KEY with appropriate cascade rules

Indexing Strategy:

· Covering indexes for frequent query patterns
· Composite indexes for multi-column searches
· Filtered indexes for partial data access
· Balance between read performance and write overhead

Phase 1.6: SQL Server Database Creation (DDL Implementation)

```
-- Create Database
CREATE DATABASE IF NOT EXISTS CourseManagementDB;
USE CourseManagementDB;
```




```
-- ============================================
-- Table: DEPARTMENT
-- Description: Stores academic department information
-- ============================================
CREATE TABLE DEPARTMENT (
    department_id INT AUTO_INCREMENT PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL,
    department_code VARCHAR(10) NOT NULL UNIQUE,
    head_instructor_id INT NULL,
    office_location VARCHAR(200),
    contact_email VARCHAR(255),
    contact_phone VARCHAR(20),
```