A DBMS Mini-Project Report on

COLLEGE MANAGEMENT SYSTEM

Submitted by

Sarvesh S Gounder

(U25UV23T029092)

V SEM, B. TECH (CSE)

Under the guidance of

Dr. H N Champa
Professor
Dept. of CSE, UVCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

FIRST STATE AUTONOMOUS UNIVERSITY ON IIT MODEL

K.R. Circle, Bengaluru – 560001

UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K.R. Circle, Bengaluru – 560001

CERTIFICATE

This is to certify that **Sarvesh S Gounder** of V Semester, B.Tech, Computer Science and Engineering, bearing the register number **U25UV23T029092** has submitted the DBMS Mini-Project Report on **"COLLEGE MANAGEMENT SYSTEM"**, in partial fulfilment for the DBMS Lab, prescribed by the University of Visvesvaraya College of Engineering for the academic year 2025-26.

……..…………………………. ………………………………

Dr. H N Champa	**Dr. Thriveni J**
Professor	Professor &Chairperson
Department of CSE,	Department of CSE,
UVCE	UVCE

Examiners:

1. ………………………… 2. ...………………………

ACKNOWLEDGEMENT

I take this opportunity to thank our institution **University of Visvesvaraya College of Engineering** for having given me an opportunity to carry out this project.

I would like to thank **Prof. Subhasish Tripathy, Director, UVCE**, for providing us all the facilities to work on this project. I am indebted to him for being my pillar of strength and inspiration.

I wish to place my gratitude to **Dr. Thriveni J, Professor and Chairperson, Department of Computer Science and Engineering, UVCE,** who helped me to make my project a great success.

It gives me great pleasure to express my gratitude to **Dr. H N Champa, Professor, Department of Computer Science Engineering, UVCE** for their valuable guidance and supervision in this course of project.

I express my sincere thanks to all teaching and non-teaching staff, Department of Computer Science and Engineering, UVCE for all the facilities that they have provided me for successfully completing this project.

I also thank my parents and friends for their continuous support and encouragement.

Sarvesh S Gounder

U25UV23T029092

ABSTRACT

This report specifies the various processes and techniques used in gathering requirements ,designing , implementing and testing for the project on College Management System. The problems regarding the current system in the college was analyzed and noted. This project aims to solve some of those problem and thus, add more value to the current system. The requirements were gathered from all the stakeholder and based on that we created a requirements models and designed the software based on the based. The project was implemented in the form of a website using Django(python).

Using the various resources and tools we gathered along the way, we implemented the College Management System using some features that solve the current problems in the system such as a provision to edit the attendance and marks before locking it at the end. The software was also tested using the various testing methods and results were positive.

Thus, the results can be integrated in the current College Management System to improve its working and solve some of the existing problems.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Introduction

The College Management System(College ERP) project is a web-based application developed using the Django framework and MySQL database. The main objective of this project is to simplify and automate the management of institutional data such as student, teacher, and administrative records. The system provides the centralized platform where different users—students, teachers, and administrators can access and manage relevant information efficiently and securely.

This project demonstrates the practical implementation of database concepts such as relational data modelling, CRUD operations, and role-based access control. By integrating Django's powerful backend with a MySQL database, the system ensures data consistency, reliability, and scalability. Each user type is provided with a distinct dashboard and functionalities tailored to their role—for example, teachers can manage student details and marks, students can view their academic information, and administrators can oversee the entire database.

Overall, this project highlights the integration of database management principles with modern web technologies, aiming to replace manual record-keeping with an efficient, user-friendly, and automated solution.

1.2 Objective

● To establish a centralized digital platform for managing student, faculty, and admin data efficiently

● To automate core college processes including student registration, faculty management, and academic record handling

● To reduce administrative errors and improve accuracy in managing college records

● To implement multi-role access (admin/faculty/student) for secure and organized data handling

● To generate real-time reports on student performance, attendance, and faculty workload

● To maintain data consistency and integrity through a structured relational database system

● To improve user experience by providing a simple and unified platform for accessing college information

User Roles in the System

- **Administrator:**
  - Approve/Reject teacher registrations
  - Manage system settings and user accounts
  - Assign courses to teachers and classes
  - Generate reports for students, teachers, and departments

- **Teacher:**
  - Register and maintain personal profile
  - Manage assigned courses and classes
  - Record and update student attendance
  - Enter and update student marks
  - View student academic history

- **Student:**
  - Register and maintain personal profile
  - View assigned courses and class schedules
  - Access attendance and marks
  - Generate personal academic reports

Views Available in the System

The user interface provides the following features and functionalities:

- **Welcome Page –** Displays an introduction to the College ERP System
- **Login Page –** Secure role-based login for Students, Teachers, and Administrators
- **Student Dashboard –** View personal profile, assigned courses, class schedules, attendance, and marks; generate academic reports
- **Teacher Dashboard –** Manage assigned courses and classes, record/update attendance, enter marks, and view student academic history
- **Admin Dashboard –** Approve teacher registrations, monitor system activity, assign courses, and generate reports for students, teachers, and departments
- **Attendance Management –** Interface for recording and updating student attendance per course and class

● **Marks Management –** Interface for entering, updating, and viewing student marks for assignments and exams

● **Course and Class Management –** Searchable list of courses, assign courses to classes and teachers, view class details

● **Report Generation –** Generate attendance summaries, grade reports, and departmental statistics

1.3 Functionality

The ERP System enhances college administrative and academic operations through these key modules:

1. Student Management

● Handles student registration with personal and academic details

● Allows profile updates by students and verification by admin

● Tracks enrollment in classes and courses

● Maintains academic history including attendance and marks

● Generates student-specific reports and performance summaries

2. Attendance Management

● Records student attendance for each class and course

● Allows teachers to update, correct, or modify attendance entries

● Provides real-time attendance summaries and notifications for absentees

● Generates attendance reports for students, teachers, and administrators

3. Marks Management

● Enables teachers to enter, update, and maintain marks for assignments, exams, and projects

● Tracks overall academic performance for each student

● Provides students with access to marks and grade reports

● Supports report generation for classes, courses, and departments

4. Course and Class Management

● Allows admin to add/update courses and assign them to departments and classes

- Enables teachers to view assigned courses and associated classes
- Supports class scheduling and conflict-free course allocation
- Provides searchable course directory for students and staff

5. Admin Controls

- Comprehensive management of all user roles (students, teachers)
- Approve or reject teacher registrations and profile updates
- Monitor system activity and generate analytics reports
- Assign courses to classes and teachers
- Configure system settings for academic year, semesters, and timetable

Key College-Specific Features:

- Centralized academic and administrative data management
- Role-based access for students, teachers, and administrators
- Automated report generation for attendance and performance
- Secure handling of student, teacher, and course information

1.4 Database Management System

A **Database Management System (DBMS)** is essential for managing large datasets efficiently. Our project uses **MySQL** to ensure **data consistency, security, and easy retrieval**.

Why Use a DBMS?

- **Eliminates data redundancy** – Avoids duplicate records.
- **Ensures data integrity** – Prevents inconsistencies.
- **Provides structured queries** – Allows fast data retrieval.

1.4.1. Characteristics of DBMS

- **Self-describing nature** – Metadata is stored within the database.
- **Data redundancy control** – Eliminates duplicate records.
- **Enforces data integrity rules** – Ensures that relationships between tables remain valid.

●**Supports multiple views** – Different users see only relevant data.

●**Handles multiple transactions** – Allows multiple operations without conflicts.

1.4.2. Advantages of DBMS

●**Data Security:** Ensures access control and prevents unauthorized access.

●**Data Consistency:** Provides reliable data storage and retrieval.

●**Efficient Query Processing:** Allows advanced queries to fetch related data.

●**Scalability:** Can be extended as the coaching center grows.

1.5 MySQL

1.5.1 Overview

MySQL is a **relational database management system (RDBMS)** used for structured data storage. It is widely adopted due to its **performance, security, and scalability**.

1.5.2 Why MySQL for ABCC-DBMS?

●**Efficient Data Management** – Handles structured records efficiently.

●**Query Optimization** – Supports complex queries with high-speed performance.

●**Secure Transactions** – Ensures data integrity with ACID compliance.

●**Scalability** – Supports increasing data and users.

SQL Commands Used in ABCC-DBMS

Creating the Student table

```
CREATE TABLE student (
Student_ID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(100), Age INT,
Batch_ID INT, Contact VARCHAR(15), FOREIGN KEY (Batch_ID) REFERENCES
batch(Batch_ID));
```

Inserting a New Student Record

```
INSERT INTO student (Name, Age, Batch_ID, Contact) VALUES ('John Doe', 18, 3, '9876543210');
```

Retrieving Student Details

```
SELECT * FROM student WHERE Batch_ID = 3;
```

Updating Student Information

UPDATE student SET Age = 19 WHERE Student_ID = 1;

Deleting a Student Record

DELETE FROM student WHERE Student_ID = 5;

1.5.3 SQL Statements

SQL provides essential commands to manage and manipulate relational databases. The key statements include:

1. **SELECT** – Retrieves information from a table.

   SELECT * FROM student WHERE Batch_ID = 3;

2. **INSERT** – Adds new records to a table.

   INSERT INTO student (Name, Age, Batch_ID, Contact) VALUES ('John Doe',

   18, 3, '9876543210');

3. **DELETE** – Removes one or more existing records from a table. DELETE FROM student

   WHERE Student_ID = 5;

4. **UPDATE** – Modifies existing values in a table.

   UPDATE student SET Age = 19 WHERE Student_ID = 1;

SQL Aggregate Functions

Aggregate functions perform calculations on a set of values and return a single result.

1. **COUNT** – Returns the number of records.

   SELECT COUNT(*) FROM student;

2. **SUM** – Computes the total sum of values in a column.

   SELECT SUM(Amount) FROM payment;

3. **MAX** – Returns the maximum value in

a column. SELECT

MAX(Age) FROM student;

4. **MIN** – Returns the minimum value in a
   column. SELECT

MIN(Age) FROM student;

5. **AVG** – Calculates the average value in
   a column. SELECT

AVG(Age) FROM student;

1.5.4 SQL Constraints

Constraints ensure data integrity and enforce rules in a database.

1. **NOT NULL** – Ensures a column cannot have NULL values. CREATE TABLE student (Student_ID INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(100) NOT NULL);

2. **PRIMARY KEY** – Ensures a column has unique, non-null values. CREATE TABLE student (Student_ID INT PRIMARY KEY, Name VARCHAR(100));

3. **UNIQUE** – Ensures all values in a column are **distinct** (no duplicates allowed). CREATE TABLE student (Email VARCHAR(100) UNIQUE,Name VARCHAR(100));

4. **FOREIGN KEY -** Creates a link between two tables by referencing the primary key of another table.
   CREATE TABLE enrollment (Enrollment_ID INT PRIMARY KEY, Student_ID INT, FOREIGN KEY (Student_ID) REFERENCES student(Student_ID));

5. **CHECK -** Ensures that values in a column meet a specific condition.

   CREATE TABLE student (Age INT CHECK (Age >= 18), Name VARCHAR(100));

6. **DEFAULT -** Assigns a default value to a column when no value is provided.

CREATE TABLE student (Name VARCHAR(100), Country VARCHAR(100)
DEFAULT 'India');

7. **AUTO_INCREMENT -** Automatically generates a unique number when a new record is inserted (used mainly for primary keys).

CREATE TABLE student (Student_ID INT AUTO_INCREMENT PRIMARY KEY,
Name VARCHAR(100))**;**

CHAPTER 2

LITERATURE REVIEW

This chapter focuses on the already existing systems for college management systems and establishes the software requirements for the project.

2.1 Survey of Existing System

Traditionally, colleges relied on **manual record-keeping, spreadsheets, and standalone registers** for managing students, teachers, courses, and academic records. While functional, these methods suffer from significant inefficiencies:

1. **Data Fragmentation**
   ○ Student records, teacher details, course assignments, attendance, and marks were maintained separately, leading to inconsistencies.
   ○ Example: A student's attendance recorded in one register might not match the marks entry in another spreadsheet.
2. **Academic and Administrative Challenges**
   ○ Manual enrollment and course allocation often caused conflicts or missing data.
   ○ No real-time updates for attendance or performance tracking.
   ○ Difficulty in generating reports quickly for individual students, classes, or departments.
3. **Security Risks**
   ○ Paper-based or spreadsheet records were vulnerable to unauthorized access, loss, or accidental deletion.
   ○ No proper audit trail for changes in attendance, marks, or course assignments.
4. **Operational Delays**
   ○ Staff spent excessive time searching for records or verifying student/teacher data.
   ○ Manual report preparation for attendance, marks, or departmental summaries was time-consuming and error-prone.

Limitations of Current Solutions

Even existing digital systems in many colleges have significant limitations due to lack of full integration:

● **Isolated Modules:** For example, attendance tracking, marks entry, and course management often operate independently, making it difficult to retrieve consolidated academic records.

● **Poor Scalability:** Many legacy systems cannot efficiently handle growing numbers of students, teachers, courses, or classes across semesters.

● **Limited Student/Teacher Access:** Few systems provide self-service portals where students can check attendance, marks, or class schedules, and teachers can manage courses or record attendance online.

2.2 Developed System

The new College ERP System addresses the gaps in existing college management methods by offering:

1. **Unified Database**
   ○ Integrates student records, teacher details, courses, attendance, and marks in a single MySQL database.
2. **Real-Time Updates**
   ○ Live tracking of student attendance, marks entry, and course allocations.
3. **Role-Based Access**
   ○ Secure logins for administrators, teachers, and students with activity logs for accountability.
4. **Automation**
   ○ Auto-generated attendance summaries, grade reports, and academic performance notifications.

Example: A teacher updates marks for an assignment → Student can instantly view the updated grade → Administrator can generate a report for the class without manual consolidation.

Feature	**Traditional College Systems**	**Proposed College ERP System**
Data Storage	Paper/registers or separate spreadsheets	Centralized RDBMS (MySQL)
Attendance Tracking	Manual registers or Excel sheets	Online entry with real-time updates
Marks Management	Manual calculation and record-keeping	Automated marks entry and performance reports
Security	Physical files or unprotected spreadsheets	Password-protected logins + audit trails

Feature	Traditional College Systems	Proposed College ERP System
Reporting	Manual report generation	Automated analytics for classes, courses, and departments

2.3 Software Requirements

Operating System: Windows / Linux / MacOS

Frontend: Django (Python), HTML, CSS, JavaScript

Backend: Django (Python)

Database Management System: MySQL

Web Server: Localhost (XAMPP/WAMP)

2.4 Frontend Technologies

The frontend acts as the primary interface through which users interact with the College ERP System. It ensures a smooth, responsive, and intuitive user experience across different devices and browsers. The system's frontend is built using a combination of modern web technologies.

1. HTML

Hypertext Markup Language (HTML) is used to structure the content displayed on the web pages. It defines the layout of elements like headings, forms, tables, buttons, and input fields that are essential for interacting with modules such as student profiles, attendance, and marks entry.

2. CSS

Cascading Style Sheets (CSS) are used to style and visually organize the HTML elements. CSS enhances the user experience by controlling the look and feel of the web application, including fonts, colors, spacing, and responsive layouts for dashboards and reports.

3. JavaScript

JavaScript adds interactivity to the web application. In the College ERP System, it enables real-time updates, form validation, input checking, and dynamic rendering of components based on user roles such as Student, Teacher, or Administrator.

4. Django Templates (Python)

Django templates act as the interface between the frontend and backend. They generate dynamic HTML pages by fetching data from the MySQL database. Template tags and filters allow the display of personalized content such as attendance summaries, grade reports, and course assignments.

2.5 Backend Technologies

The backend of the College ERP System handles the core logic of the application, including processing client requests, managing authentication, performing database operations, and sending appropriate responses to the frontend. It ensures that all academic and administrative transactions are securely and accurately executed.

1. Django (Python)

Django is a high-level Python web framework powering the College ERP backend. It handles business logic, database operations (MySQL), and dynamic HTML generation through templates. Django’s session and authentication management enable secure role-based access for Administrators, Teachers, and Students. Its MVC architecture ensures smooth data flow between frontend forms and the database while maintaining code modularity and scalability.

2. MySQL

MySQL is an open-source relational database management system used to store structured academic and administrative data. It organizes data in the form of tables and supports complex queries for efficient data retrieval and processing. MySQL is reliable, scalable, and supports transactions, which is crucial for handling sensitive student and teacher information.

MySQL offers:

● Structured storage of student records, teacher details, courses, attendance, and marks
● Foreign key constraints to ensure referential integrity across tables

● Secure login and access control features
● Optimized performance for concurrent database operations
● Scalability to accommodate growing student populations and course offerings

2.6 Entity-Relationship (ER) Model

The Entity-Relationship (ER) model is a high-level conceptual data model that provides a systematic approach to database design by defining the data elements, their attributes, and the interrelationships between them. It serves as a blueprint for constructing relational databases and is particularly useful in capturing the requirements of complex systems such as the College ERP System

Purpose of the ER Model

The main goals of the ER model in the College ERP System are to:

● Represent the data structure in a way that is easily understood by both technical and non-technical stakeholders.
● Identify the entities that need to be represented in the database.
● Define the relationships between these entities.
● Specify the attributes of each entity and relationship.
● Establish integrity constraints such as keys and cardinalities.

Core Components of the ER Model

The ER model consists of three fundamental components: **entities, attributes, and relationships**.

1. Entities

Entities are objects or concepts that are distinguishable from other objects. Each entity represents a set of real-world objects that share the same properties.

● **Strong Entity:** Has a primary key that uniquely identifies each instance (e.g., Student, Teacher, Course).
● **Weak Entity:** Cannot be uniquely identified without a related strong entity (e.g., Attendance depends on Student and Course).

Entities are represented as rectangles in an ER diagram.

2. Attributes

Attributes are the properties or characteristics of an entity or relationship. For example, a Student entity may have attributes such as **Student_ID, Name, Email, Contact, Class, and Section**.

Types of attributes include:

● **Simple (Atomic):** Cannot be divided further (e.g., Student_ID).
● **Composite:** Can be broken down into smaller sub-parts (e.g., Name → First_Name, Last_Name).
● **Derived:** Can be calculated from other attributes (e.g., Age from Date_of_Birth).
● **Multivalued:** May have more than one value for an entity instance (e.g., multiple contact numbers).

Attributes are typically represented as ellipses in an ER diagram, connected to their respective entity or relationship.

3. Relationships

Relationships describe how two or more entities are associated with each other. Examples in a college domain include:

● A Student enrolls in one or more Courses.
● A Teacher teaches one or more Courses.
● Attendance is recorded for each Student in a Course.
● Marks are assigned to a Student for a particular Course.

Relationships are depicted as diamonds in an ER diagram, connected to the participating entities.

Each relationship can have:

● **Cardinality constraints:** Define the number of instances of one entity that can or must be associated with instances of another entity.
 o **One-to-One (1:1):** Each entity instance in the relationship will have exactly one related instance.
 o **One-to-Many (1:N):** One instance of an entity is associated with multiple instances of another.
 o **Many-to-Many (M:N):** Multiple instances of one entity are related to multiple instances of another.

● **Participation constraints:** Specify whether all instances of an entity must participate in the relationship.

o **Total participation:** Every instance of the entity must be involved in the relationship.

o **Partial participation:** Some instances may not participate.

Keys and Integrity Constraints

● **Primary Key (PK):** An attribute or set of attributes that uniquely identifies an entity instance (e.g., Student_ID for Student).

● **Foreign Key (FK):** An attribute that creates a link between entities by referring to the primary key of another entity (e.g., Course_ID in Attendance).

● **Referential Integrity:** Ensures that relationships between tables remain consistent (e.g., a foreign key must match an existing primary key or be null).

Advanced Concepts in the ER Model

● **Generalization:** Extracts shared characteristics from two or more entities and creates a generalized entity (e.g., combining Undergraduate and Postgraduate into Student).

● **Specialization:** Divides a higher-level entity into sub-entities based on distinguishing characteristics (e.g., Employee → Teacher, Administrator).

● **Aggregation:** Treats a relationship set as an entity set for participation in other relationships (e.g., Assigning Teacher to Course for a Class).

● **Weak Entity and Identifying Relationships:** Used when an entity cannot be uniquely identified by its own attributes and relies on a related entity and a partial key (e.g., Attendance depends on Student and Course).

2.7 Relational Schema Design

The **Relational Schema** is a representation of the database structure in terms of **relations (tables)**, where data is organized in rows and columns. It is derived from the conceptual design represented in the ER model and forms the **logical structure** that is eventually implemented in a relational database system. Each entity and relationship in the ER model is translated into one or more relations, and constraints such as primary keys, foreign keys, and cardinality are enforced through appropriate schema definitions. The goal is to create a **normalized, non-redundant**, and **integrity-preserving** relational schema that accurately models the domain.

7-Step Approach to Convert ER Model to Relational Schema

The process of converting the Entity-Relationship (ER) Model of the **College ERP System** into a Relational Schema ensures that every entity, attribute, and relationship identified during the conceptual design phase is accurately represented in a logical, implementable structure. This step-by-step transformation produces a set of normalized relations with enforced keys, referential integrity, and minimal redundancy, forming the foundation of the database implemented in **MySQL**.

Step 1: Mapping Regular (Strong) Entity Sets

Each strong entity in the ER model becomes an individual table. All attributes, including primary keys, are directly mapped to columns.

Example Mappings:

- info_user(id, password, last_login)
- info_dept(id, name)
- info_class(id, section, sem, dept_id)
- info_course(id, name, shortname, dept_id)
- info_teacher(id, name, sex, DOB, dept_id, user_id)
- info_student(USN, name, sex, DOB, class_id_id, user_id)

Each of these entities contains a **primary key** (id or USN) to uniquely identify every record.

Step 2: Mapping Weak Entity Sets

Weak entities depend on strong entities for identification. In this project, tables such as info_studentcourse, info_assign, and info_attendance rely on foreign keys from parent entities.

Example:
info_studentcourse(id, course_id, student_id)

• Weak entity connecting **Student** and **Course**.
• Composite uniqueness on (student_id, course_id) ensures one student cannot enroll in the same course twice.

Step 3: Mapping Binary 1:1 Relationships

One-to-one relationships are represented using unique foreign keys.

Example:
info_teacher ↔ info_user

- Each teacher is linked to exactly one user account through a **unique foreign key** user_id.
- Ensures one login corresponds to one faculty profile.

Step 4: Mapping Binary 1:N Relationships

One-to-many relationships are implemented by placing the foreign key in the "many" side table.

Examples:

- A **department** has many **classes** → info_class(dept_id) references info_dept(id)
- A **department** has many **courses** → info_course(dept_id) references info_dept(id)
- A **department** has many **teachers** → info_teacher(dept_id) references info_dept(id)
- A **class** has many **students** → info_student(class_id_id) references info_class(id)

These links maintain referential integrity and define academic hierarchies within the college

Step 5: Mapping Binary M:N Relationships

Many-to-many relationships are broken down into intermediate relations.

Examples:

- **Teacher – Class – Course** mapping → info_assign(class_id_id, course_id, teacher_id)
  - Connects which teacher handles which course for which class.
- **Student – Course** mapping → info_studentcourse(student_id, course_id)
  - Tracks course enrollments of each student.

Both tables contain **foreign keys** referencing their respective parent entities, forming composite uniqueness constraints to prevent duplicate assignments or enrollments.

Step 6: Mapping Multivalued Attributes

When an entity can have multiple values for a particular attribute, separate tables handle them.
In this design, repeated entries like attendance and marks are stored in distinct tables instead of being duplicated inside student or course tables.

Examples:

• info_marks(studentcourse_id, name, marks1) — stores multiple marks records per student-course pair.
• info_attendance(attendanceclass_id, date, status, student_id, course_id) — stores multiple attendance entries per student across dates.

Step 7: Mapping Ternary (and Higher) Relationships

For relationships involving three or more entities, composite tables are created.

Examples:

• info_assign(class_id_id, course_id, teacher_id) — links three entities (Class, Course, Teacher).
• info_assigntime(period, day, assign_id) — adds time-table data linked to an assignment.
• info_attendance(date, student_id, course_id, attendanceclass_id) — represents attendance as a ternary relation among Student, Course, and AttendanceClass.

These ensure accurate many-way associations between academic entities.

Outcome of the Mapping Process

Following this systematic conversion, the **College ERP relational schema** provides:

• **Data integrity** through properly defined primary and foreign keys.
• **Logical consistency** preserving real-world relationships between students, teachers, courses, and departments.
• **Scalability and normalization**, avoiding data redundancy.
• **Efficient data access** for modules such as attendance, marks, and timetable management.

This finalized schema, implemented in MySQL under the database college_erp, serves as the backbone for all functional modules in the system, ensuring reliability and consistency across every operation.

2.8 Normalization

Normalization is a systematic database design technique used to minimize data redundancy and eliminate undesirable anomalies such as **insertion**, **update**, and **deletion anomalies**.
It involves decomposing larger relations (tables) into smaller, well-structured relations while maintaining the necessary relationships between them.

Normalization ensures that the database structure is logical, consistent, and optimized for data integrity and efficient access.

Objectives of Normalization

Normalization aims to:

- Minimize data redundancy.
- Eliminate insertion, deletion, and update anomalies.
- Maintain data integrity through well-defined relationships.

Functional Dependency

A **Functional Dependency (FD)**, denoted as $\mathbf{X \rightarrow Y}$, expresses a relationship between two sets of attributes **X** and **Y** within a relation **R**.

It states that for any two tuples $\mathbf{t_1}$ and $\mathbf{t_2}$ in a relation **r(R)**,
if $\mathbf{t_1[X] = t_2[X]}$, then $\mathbf{t_1[Y] = t_2[Y]}$ must also hold true.

In simpler terms, the value of attribute(s) **Y** depends on the value of attribute(s) **X**, meaning **X uniquely determines Y**.

Normal Forms in SQL

Normalization is performed through a series of stages known as **Normal Forms (NF)**.
Each normal form has specific rules and focuses on a particular type of dependency or redundancy.

The major normal forms are:

1. **First Normal Form (1NF)**
2. **Second Normal Form (2NF)**
3. **Third Normal Form (3NF)**
4. **Boyce–Codd Normal Form (BCNF)**
5. **Fourth Normal Form (4NF)**
6. **Fifth Normal Form (5NF)**

a) First Normal Form (1NF)

A relation is in **1NF** if:

• All attribute values are **atomic** (indivisible).
• Each cell of a table contains a single value.
• There are no repeating groups or arrays.

Example:
A column with multiple phone numbers must be split into separate rows or moved to a different table.

b) Second Normal Form (2NF)

A relation is in **2NF** if:

• It is already in 1NF.
• Every **non-prime attribute** (non-key attribute) is **fully functionally dependent** on the **entire primary key**.

This eliminates **partial dependencies** (where a non-key attribute depends only on part of a composite key).

Example:
If a table's primary key is (Student_ID, Course_ID), then attributes like Course_Name should depend only on Course_ID, not on both.

c) Third Normal Form (3NF)

A relation is in **3NF** if:

- It is in 2NF.
- There are **no transitive dependencies**, meaning a non-prime attribute should not depend on another non-prime attribute.

Example:
If Student_ID → Department_ID and Department_ID → Department_Name, then Department_Name should not be in the same table as Student_ID.

d) Boyce–Codd Normal Form (BCNF)

A relation is in **BCNF** if:

- For every functional dependency **X → Y**, **X** must be a **super key**.

BCNF is a stricter version of 3NF.
It removes all remaining anomalies caused by overlapping candidate keys or complex dependencies.

e) Fourth Normal Form (4NF)

A relation is in **4NF** if:

- It is in BCNF.
- It contains **no multi-valued dependencies (MVDs)**.

A multi-valued dependency occurs when one attribute determines multiple independent values of another attribute.

f) Fifth Normal Form (5NF)

A relation is in **5NF** if:

- It is in 4NF.
- It contains no **join dependencies** that are not implied by candidate keys.

This form eliminates redundancy arising from complex relationships and ensures that data cannot be recombined in unexpected ways.

2.9 Advantages of the Developed System

The **College ERP System** provides substantial improvements over conventional manual processes used

in college administration. By utilizing database management concepts with technologies such as **MySQL** and **Django Framework**, the system ensures efficient, secure, and centralized management of academic and administrative activities.

Below are the key benefits of the developed system:

• **Automation of Core Academic Operations**
The system automates vital college functions such as attendance tracking, marks entry, timetable generation, and faculty allocation. This reduces manual work and improves accuracy in record-keeping.

• **Improved Accuracy and Efficiency**
Maintaining digital records for students, teachers, courses, and classes eliminates data duplication and human errors. Academic information can be accessed or updated instantly by authorized users.

• **Enhanced Data Security**
With role-based access and authentication mechanisms, data is securely managed. Only authorized users such as administrators, faculty, and students can access specific modules relevant to their roles.

• **Centralized Database Management**
All information related to departments, courses, teachers, students, attendance, and marks is stored within a single structured database, ensuring consistency and easy data retrieval.

• **Real-Time Information Access**
Faculty and students can access real-time updates related to attendance, schedules, and marks, improving transparency and communication within the institution.

• **Scalable and Modular Architecture**
The database design supports expansion — new modules like placement records, fee management, or student feedback can be integrated without modifying the existing schema.

• **Comprehensive Reporting and Analysis**
The system can generate analytical reports such as student performance summaries, attendance statistics, and course-wise results, enabling data-driven academic decision-making.

CHAPTER 3

PROPOSED WORK

This chapter presents the design and architecture of the proposed **College ERP System (CERP)**, detailing the interaction between various entities, data modelling, and normalization principles. The system is structured to efficiently manage essential college operations, including **student enrolment, faculty assignment, course management, attendance tracking, timetable scheduling, and marks entry**.

3.1 Entity Relationship (ER) Model

The Entity–Relationship Diagram (ERD) serves as the core blueprint of the College ERP System's database design. It provides a structured visual representation of how different types of academic and administrative data are stored, organized, and connected. By outlining entities, attributes, and relationships clearly, the ERD ensures that the system remains consistent, scalable, and easy to maintain.

The ERD includes all crucial entities required for college operations, such as **Department**, **Class**, **Student**, **Teacher**, **Course**, **Assignment**, **Attendance**, and **Marks**. Each entity has its own attributes—for instance, student personal details, department identifiers, class information, and course metadata. Separating these elements into distinct entities prevents data duplication and helps maintain normalization across the database.

Primary key and foreign key constraints define how these entities interact. The **Student** entity links to **Class** and **Department**, representing a student's academic grouping and department enrollment. The **Teacher** entity connects with **Department** and **Course**, showing the subjects handled by faculty and their departmental affiliation. These connections reflect real-world college workflows and establish clear relational pathways.

Certain academic processes involve many-to-many relationships—students can take multiple courses, and courses can have many students. Similarly, teachers may teach multiple classes. To handle these complex mappings, the ERD uses **junction tables** such as Student_Course. These tables ensure accurate relational mapping while keeping the database normalized and flexible for future expansion.

Operational components like **Assignment**, **Attendance**, and **Marks** depend directly on students and courses. Each assignment record corresponds to a course and student, each attendance entry links to a class session and student, and each marks record ties to a student's evaluation in a particular course. These relationships guarantee meaningful, traceable academic data.

Figure 3.1.1 displays the complete ER Diagram, showing all entities and their interconnections. It provides a clear overview of how the entire system's data structure operates.

Overall, the ERD establishes a logical, efficient, and well-organized foundation for the College ERP System, ensuring accurate data handling and smooth academic management.

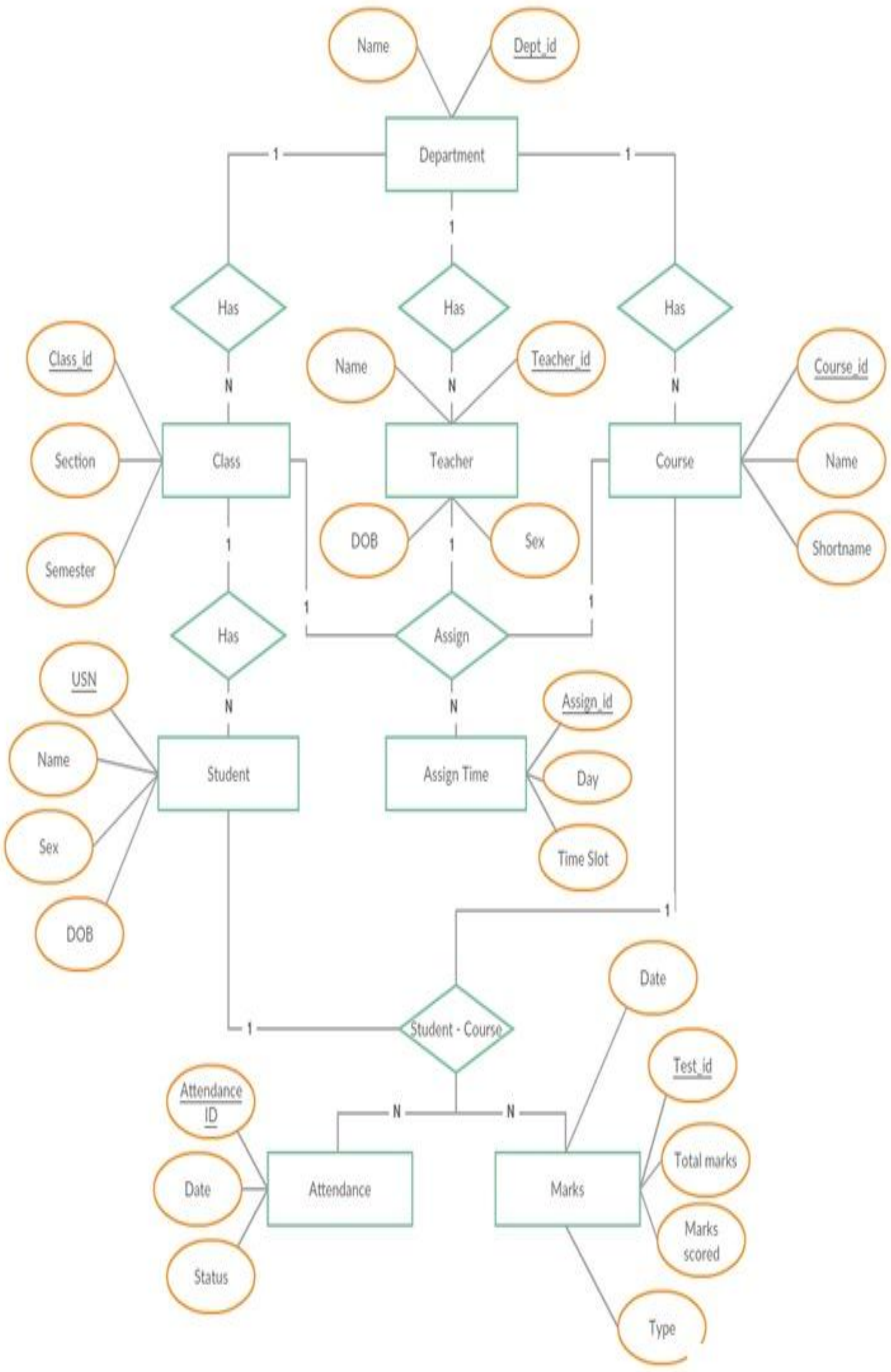


Fig 3.1.1. ER-Diagram of College Management System

Entities and their Attributes

The proposed **College ERP System** consists of several core entities that collectively manage academic and administrative operations. Each entity represents a key component of the college structure and is connected through well-defined relationships ensuring data consistency and integrity.

1. **Department**
   Represents the various academic departments in the college. Each department manages its own teachers, classes, and courses.
   Attributes: Dept_id, Name.

2. **Class**
   Represents a specific section and semester under a department. Each class contains multiple students.
   Attributes: Class_id, Section, Semester, Dept_id (FK to Department).

3. **Teacher**
   Contains details of faculty members, their personal information, and departmental association. Teachers are assigned to classes and courses.
   Attributes: Teacher_id, Name, Sex, DOB, Dept_id (FK to Department).

4. **Course**
   Stores details of the subjects offered by departments and taught by teachers to different classes.
   Attributes: Course_id, Name, Shortname, Dept_id (FK to Department).

5. **Student**
   Represents students enrolled in different classes, storing their personal and academic details. Each student is associated with a class and linked to attendance and marks records.
   Attributes: USN, Name, Sex, DOB, Class_id (FK to Class).

6. **Attendance and Marks**
   These entities record student academic performance and participation. Attendance tracks student presence, while Marks stores test details and scores.
   Attendance Attributes: Attendance_id, Date, Status.
   Marks Attributes: Test_id, Date, Total Marks, Marks Scored, Type.

All these entities are interlinked using **primary and foreign key constraints**, enabling seamless data flow between departments, classes, teachers, students, and courses.

3.2 Relational Model

The relational model defines the structure of the database in terms of tables, their attributes, and the relationships between them. The **College ERP System (CERP)** is developed using a set of relational tables that efficiently store, retrieve, and manage various types of academic and administrative data. Each table represents a distinct entity such as Department, Class, Student, Teacher, Course, Attendance, and Marks. These tables are normalized and interconnected through **primary and foreign key constraints**, ensuring data integrity, consistency, and minimal redundancy throughout the system.

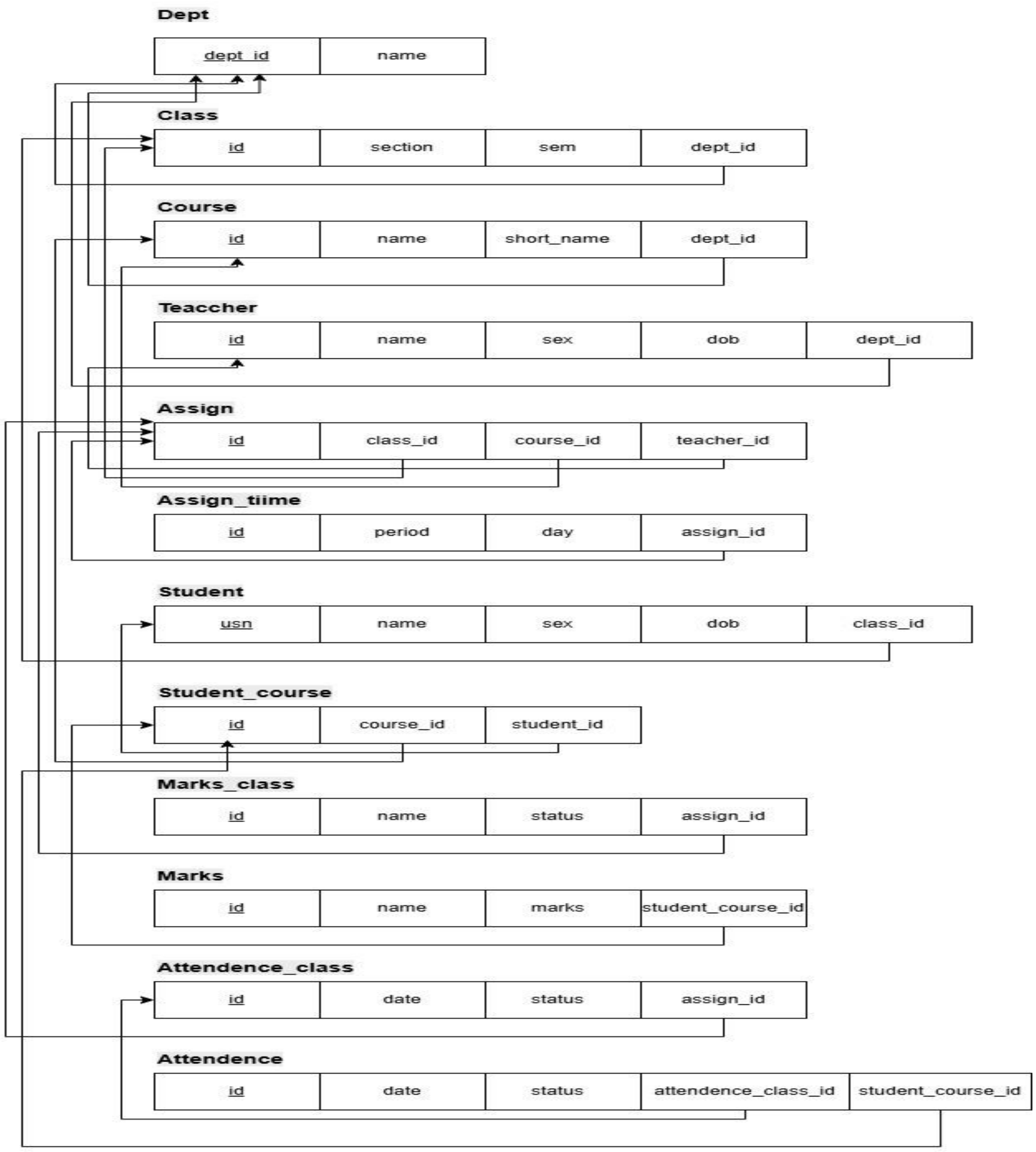


Fig 3.2.1. Relational Model

Relationships and Cardinalities

1. **A Department manages multiple Classes** (DEPARTMENT → CLASS) **(1:M)**
   a. One department can have many classes, but each class belongs to only one department.

2. **A Department offers multiple Courses** (DEPARTMENT → COURSE) **(1:M)**
   a. One department can offer many courses, but each course is associated with only one department.

3. **A Department employs multiple Teachers** (DEPARTMENT → TEACHER) **(1:M)**
   a. One department can have many teachers, but each teacher is assigned to only one department.

4. **A Class consists of multiple Students** (CLASS → STUDENT) **(1:M)**
   a. One class can have many students, but each student belongs to only one class.

5. **A Teacher teaches multiple Courses** (TEACHER → ASSIGN) (**1:M)**
   a. One teacher can handle multiple course assignments, but each assignment is linked to one teacher.

6. **A Class includes multiple Course Assignments** (CLASS → ASSIGN) **(1:M)**
   a. One class can have several course-teacher combinations, but each assignment belongs to one class.

7. **A Course can have multiple Assignments** (COURSE → ASSIGN) **(1:M)**
   a. One course can appear in multiple assignments, but each assignment refers to one specific course.

8. **A Student can enrol in multiple Courses** (STUDENT → STUDENTCOURSE) **(1:M)**
   a. One student can register for several courses, but each record in the student-course table to a student.

9. **A Course can have multiple Students** (COURSE → STUDENTCOURSE) **(1:M)**
   a. One course can include many students, but each student-course record refers to one course.

Key Relationship Tables

• **CLASS_COURSE_ASSIGN**:
Represents the relationship between classes, teachers, and courses, defining which teacher handles

which subject for a specific class.
(Fields: assign_id, class_id, course_id, teacher_id)

- **STUDENT_COURSE_ENROLL**:
Tracks the enrollment of students in different courses. It links each student to the subjects they have registered for.
(Fields: id, student_id, course_id)
- **ATTENDANCE_RECORD**:
Stores attendance information for students in each class and course session. It records whether a student was present or absent on a given date.
(Fields: attendance_id, student_id, course_id, date, status)
- **MARKS_ENTRY**:
Maintains details of marks obtained by students in various tests or exams. Each record corresponds to a specific student and course.
(Fields: marks_id, student_id, course_id, test_name, marks_scored, total_marks)

Implementation Notes

- Foreign keys enforce all relationships (e.g., info_class.dept_id references info_dept.id)
- **1:M** relationships implemented directly via foreign keys
- Status fields (attendanceStatus, marksStatus) manage academic workflows

These relationships define the logical connections between entities in the College Database Management System. They form the foundation for implementing foreign key constraints and relationship tables in MySQL schema.

3.3 Normalization

Normalization is a key principle in database design that eliminates redundancy and ensures data consistency. By organizing data into structured tables and enforcing relationships through keys, normalization minimizes anomalies during insertions, deletions, and updates. The **College ERP System (CERP)** adheres to normalization rules up to **Fifth Normal Form (5NF)** to ensure a clean, reliable, and scalable database structure.

1. First Normal Form (1NF)

A table is in 1NF if:

- It has no repeating groups or arrays.
- All attribute values are atomic (indivisible).
- Each row is unique and identified by a primary key.

CERP Compliance:

- **Normalization Applied To:**
All tables (info_user, info_dept, info_class, info_teacher, info_student, info_course, info_assign, info_studentcourse, info_attendance, info_marks) satisfy 1NF as they contain only atomic values.
- Each table has a properly defined primary key ensuring row uniqueness.
- No repeating groups or arrays exist in any table structure.

2. Second Normal Form (2NF)

A table is in 2NF if:

- It is already in 1NF.
- All non-key attributes are fully dependent on the entire primary key (especially for composite key tables).

CERP Compliance:

- **Normalization Applied To:**
info_studentcourse and info_assign contain composite dependencies where multiple foreign keys form a unique combination. These dependencies have been structured to ensure each non-key attribute fully depends on the primary key.
- **Normalization Not Required For:**
Tables with single-column primary keys such as info_dept, info_class, info_course, info_teacher, and info_student already satisfy 2NF.

3. Third Normal Form (3NF)

A table is in 3NF if:

- It is already in 2NF.
- No non-key attribute depends on another non-key attribute (no transitive dependencies).

CERP Compliance:

- **Normalization Applied To:**
In info_teacher and info_student, attributes such as dept_id and class_id_id depend on external keys rather than internal non-key attributes. This ensures transitive dependencies are eliminated.

• **Normalization Not Required For:**
Tables like info_dept, info_class, and info_course are already in 3NF since each attribute directly depends on its primary key.

4. Boyce–Codd Normal Form (BCNF)

A table is in BCNF if:

• It is already in 3NF.

• For every functional dependency X → Y, X is a superkey (X uniquely determines all attributes in the table).

CERP Compliance:

• **Normalization Applied To:**
The info_studentcourse and info_assign tables have been structured such that each composite key (combining multiple foreign keys) acts as a superkey.

• **Normalization Not Required For:**
Other tables like info_dept, info_class, and info_course already satisfy BCNF as they follow clear one-to-many relationships without overlapping dependencies.

5. Fourth Normal Form (4NF)

A table is in 4NF if:

• It is already in BCNF.

• It has no multi-valued dependencies except those on superkeys.

CERP Compliance:

• **Normalization Applied To:**
info_attendance and info_marks tables ensure that each record contains a single value for attendance or marks per student per course, eliminating multi-valued dependencies.

• **Normalization Not Required For:**
Other tables do not exhibit multi-valued dependencies based on the given schema.

6. Fifth Normal Form (5NF)

A table is in 5NF if:

• It is already in 4NF.

• It cannot be decomposed into smaller tables without loss of information (join dependency preservation).

CERP Compliance:

- **Normalization Applied To:**
No complex join dependencies exist in the current schema.

- **Normalization Not Required For:**
All tables, once normalized to 4NF, satisfy 5NF automatically as all dependencies are maintained through primary and foreign keys.

The **College ERP System** benefits from this structured normalization approach by reducing data redundancy, improving data accuracy, and maintaining integrity across all entities. This design ensures smooth scalability, efficient query processing, and reliable data management for all academic operations.

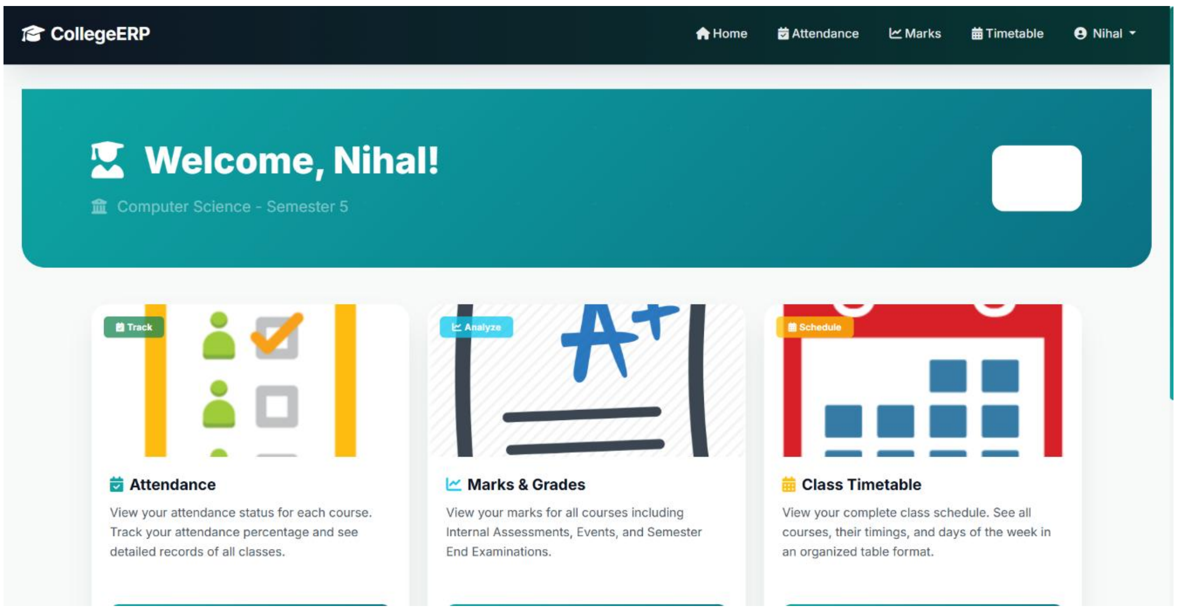
CHAPTER 4

RESULT

The **College Management System (College ERP)** was successfully implemented and tested using sample institutional data. The system demonstrated its ability to manage student records, faculty information, course registration, attendance tracking, and academic performance with efficiency and accuracy. By automating these operations, ERP significantly reduces administrative workload and minimizes the chances of errors in data handling. Test users found the system interface intuitive and user-friendly, enabling smooth navigation through modules such as student registration, faculty management, course allocation, and result generation. Real-time feedback and well-structured workflows allowed both faculty and administrators to access and manage college data effortlessly. The system efficiently generates reports on student performance, faculty workload, and departmental activities, providing valuable insights into academic operations and institutional performance. These features support quick decision-making and ensure that administrators have complete visibility over all academic and administrative processes.

4.1 Screenshots

The following is a series of screenshots of the developed application.



Welcome, Nihal!

Computer Science - Semester 5

Attendance

View your attendance status for each course. Track your attendance percentage and see detailed records of all classes.

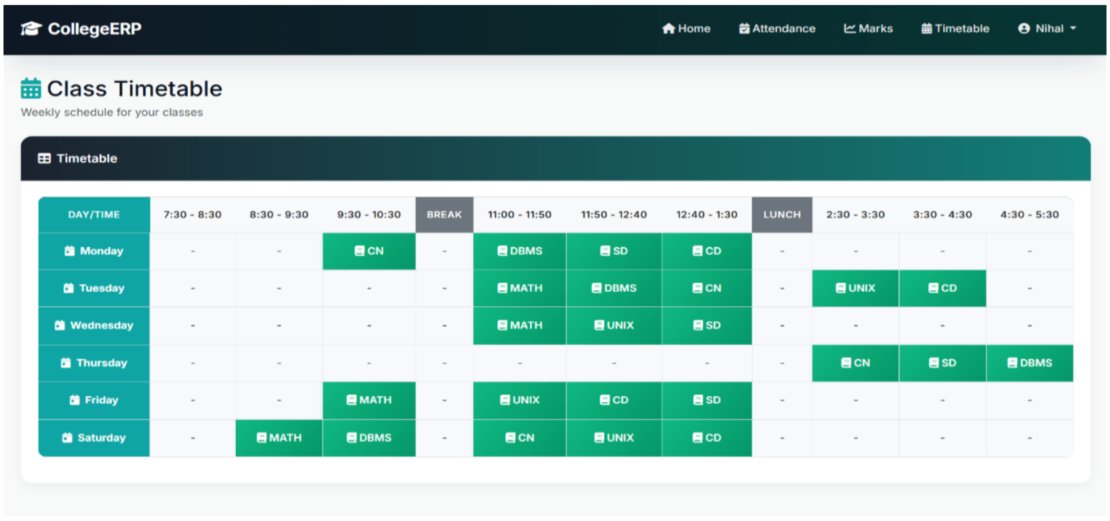
Marks & Grades

View your marks for all courses including Internal Assessments, Events, and Semester End Examinations.

Class Timetable

View your complete class schedule. See all courses, their timings, and days of the week in an organized table format.

Fig 4.1.1. Student login



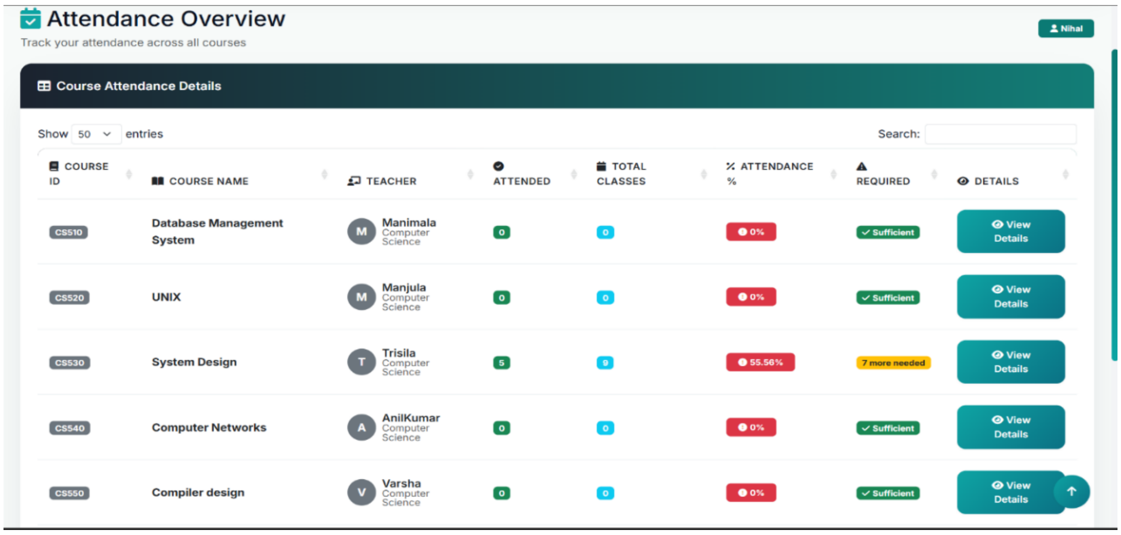
Class Timetable

Weekly schedule for your classes

Timetable

DAY/TIME	7:30 - 8:30	8:30 - 9:30	9:30 - 10:30	BREAK	11:00 - 11:50	11:50 - 12:40	12:40 - 1:30	LUNCH	2:30 - 3:30	3:30 - 4:30	4:30 - 5:30
Monday	-	-	CN	-	DBMS	SD	CD	-	-	-	-
Tuesday	-	-	-	-	MATH	DBMS	CN	-	UNIX	CD	-
Wednesday	-	-	-	-	MATH	UNIX	SD	-	-	-	-
Thursday	-	-	-	-	-	-	-	-	CN	SD	DBMS
Friday	-	-	MATH	-	UNIX	CD	SD	-	-	-	-
Saturday	-	MATH	DBMS	-	CN	UNIX	CD	-	-	-	-

Fig 4.2.2.Class Timetable



Attendance Overview

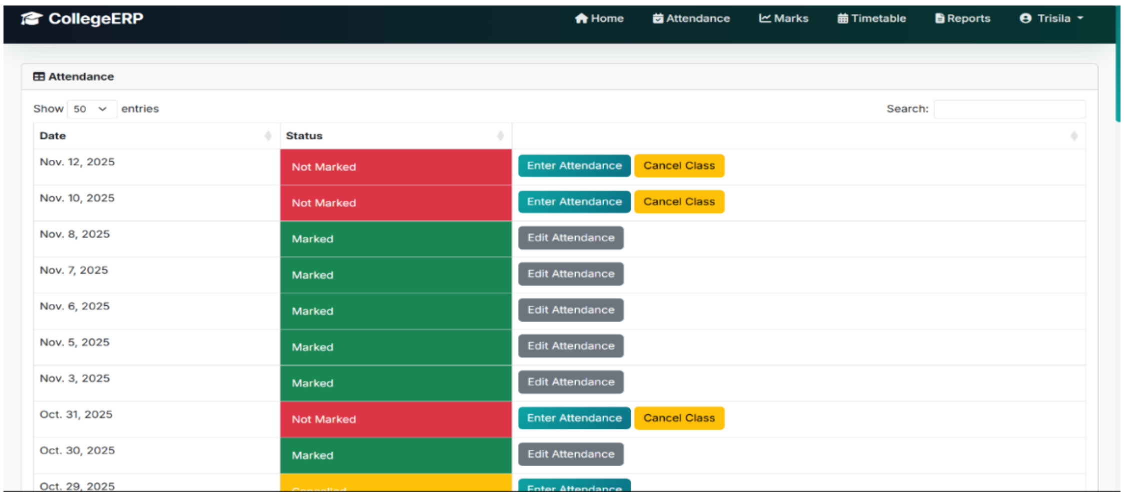
Track your attendance across all courses

Course Attendance Details

Show 50 entries Search:

COURSE ID	COURSE NAME	TEACHER	ATTENDED	TOTAL CLASSES	ATTENDANCE %	REQUIRED	DETAILS
CS510	Database Management System	Manimala Computer Science	0	0	0%	Sufficient	View Details
CS520	UNIX	Manjula Computer Science	0	0	0%	Sufficient	View Details
CS530	System Design	Trisila Computer Science	5	9	55.56%	7 more needed	View Details
CS540	Computer Networks	AnilKumar Computer Science	0	0	0%	Sufficient	View Details
CS550	Compiler design	Varsha Computer Science	0	0	0%	Sufficient	View Details

Fig 4.2.3. Attendance Overview

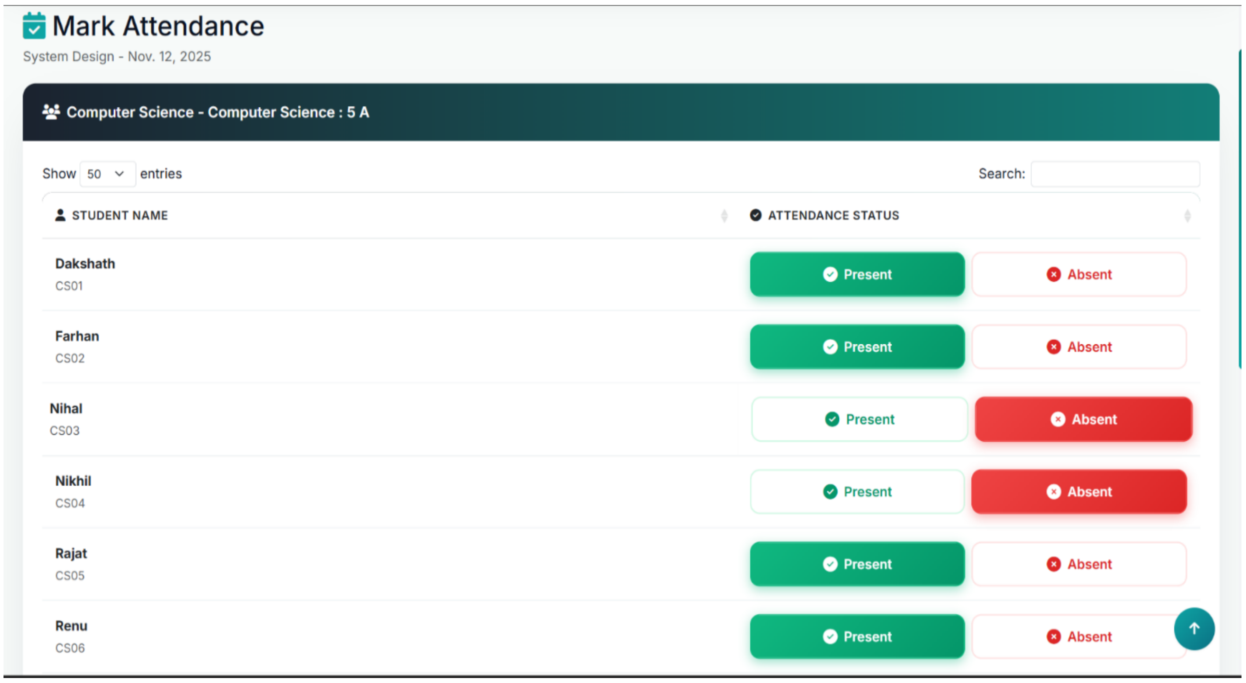


Attendance

Show 50 entries Search:

Date	Status	
Nov. 12, 2025	Not Marked	Enter Attendance Cancel Class
Nov. 10, 2025	Not Marked	Enter Attendance Cancel Class
Nov. 8, 2025	Marked	Edit Attendance
Nov. 7, 2025	Marked	Edit Attendance
Nov. 6, 2025	Marked	Edit Attendance
Nov. 5, 2025	Marked	Edit Attendance
Nov. 3, 2025	Marked	Edit Attendance
Oct. 31, 2025	Not Marked	Enter Attendance Cancel Class
Oct. 30, 2025	Marked	Edit Attendance
Oct. 29, 2025	[illegible]	Enter Attendance

Fig 4.2.4. Status Of Attendance



Mark Attendance

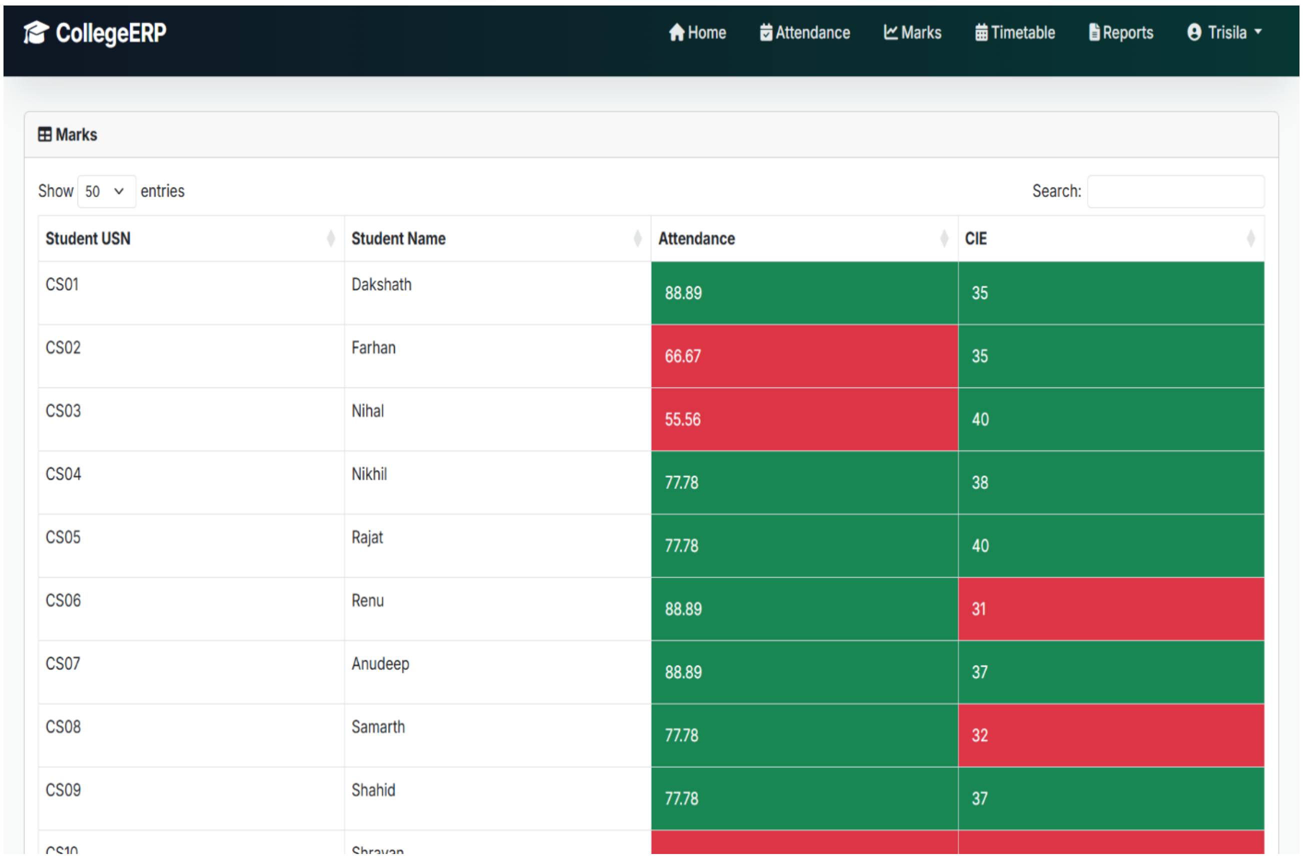
System Design - Nov. 12, 2025

Computer Science - Computer Science : 5 A

Show 50 entries Search:

STUDENT NAME	ATTENDANCE STATUS
Dakshath CS01	Present ✓ / Absent
Farhan CS02	Present ✓ / Absent
Nihal CS03	Present / Absent ✓
Nikhil CS04	Present / Absent ✓
Rajat CS05	Present ✓ / Absent
Renu CS06	Present ✓ / Absent

Fig 4.2.5. Mark Attendance



Marks

Show 50 entries Search:

Student USN	Student Name	Attendance	CIE
CS01	Dakshath	🟩 88.89	🟩 35
CS02	Farhan	🟥 66.67	🟩 35
CS03	Nihal	🟥 55.56	🟩 40
CS04	Nikhil	🟩 77.78	🟩 38
CS05	Rajat	🟩 77.78	🟩 40
CS06	Renu	🟩 88.89	🟥 31
CS07	Anudeep	🟩 88.89	🟩 37
CS08	Samarth	🟩 77.78	🟥 32
CS09	Shahid	🟩 77.78	🟩 37
CS10	Shravan	🟥	🟥

Fig 4.2.6. Student Report

CONCLUSION

The College ERP System has been developed to overcome the limitations of the existing manual system, where accessing and managing information about students and staff was time-consuming and inefficient. The proposed system automates most administrative and academic processes, thereby reducing workload, saving staff time, and improving overall efficiency.

Through this centralized and web-based solution, administrators can easily manage class schedules, timetables, attendance, and reports without manual intervention. The system provides secure and organized data storage, ensures quick access to accurate information, and supports the generation of reports whenever required. By eliminating paperwork and manual calculations, the College ERP System improves productivity and ensures data reliability.

The implementation of this system simplifies the administrator's tasks, enhances data accuracy, and allows proper utilization of institutional resources. It provides an efficient platform that benefits both faculty and students by delivering relevant information based on their roles in the system.

Further Improvements

Although the current version of the College ERP System successfully automates core college operations such as student management, attendance, and marks processing, there is significant potential for future enhancement.

Planned improvements include adding modules for online fee payment, placement management, library automation, and student feedback systems to provide a complete digital ecosystem for college operations. Integration with mobile applications and cloud storage will further enhance accessibility, scalability, and remote management.

Additionally, analytical dashboards can be introduced for data-driven decision-making, enabling administrators to visualize student performance trends, faculty workload, and institutional efficiency. By implementing these future enhancements, the system can evolve into a comprehensive, smart college management solution supporting digital transformation across all departments.

Bibliography

[1] Elmasri and Navathe: Fundamentals of Database Systems, 7th Edition, Pearson Education, 2016.

[2] Ian Sommerville: Software Engineering, 10th edition, Person Education Ltd, 2015.

[3] https://en.wikipedia.org/wiki/Requirements-engineering

[4] https://web.cs.dal.ca/ hawkey/3130/srs-template-ieee.doc

[5] http://www.ntu.edu.sg/home/cfcavallaro/Reports/Report%20writing.htmTop

[6] https://en.wikipedia.org/wiki/Class diagram

[7] https://www.djangoproject.com/

[8] https://getbootstrap.com/

[9] https://www.tutorialspoint.com/

[10] https://creately.com/

[11] https://www.overleaf.com/project