

Student Management System

-By Assignwise

Table of Contents

Part A - Database Logical Model	3
1. Background	3
2. Objective.....	7
3. Problem Statement	8
4. Scope.....	10
5. Hardware & Software Specification	11
6. Entity Relationship Diagram.....	13
7. Relational Schema.....	14
8. Normalization.....	15
Part B (Database Design)	23
Table.....	23
Table Values	26
Output Table	28
Part-C	31
1. Background	31
2. Objective.....	35
3. Problem Statement	36
4. Scope.....	38
5. Hardware & Software Specification	39
6. Entity Relationship Diagram.....	41
7. Relational Schema.....	42
8. Normalization.....	43
Print screen of system user interface	51
Conclusion.....	59

Part A - Database Logical Model

1. Background

i. Introduction to Current System:

In the ever-evolving landscape of education, efficient management of student-related information and administrative processes is crucial for the seamless functioning of educational institutions. Recognizing that the existing Student Management System operates through manual, paper-dependent processes and lacks cohesive integration. Administrative duties, student record maintenance, and communication all rely on manual methods, resulting in inefficiencies and inaccuracies. Administrative personnel are responsible for documenting and maintaining student details through physical forms and records. Educators must assess and evaluate student assignments and tests using hardcopy materials and calculators. Enrolment in courses, fee payments, and transcript inquiries for students entail completing and submitting paper forms. Communication among students, teachers, staff, and parents occurs via telephone conversations, emails, or written correspondence.

ii. Problem with the Existing System:

The existing system is a manual and paper-based system that suffers from various issues that impair the efficiency and quality of student management. Some of the issues are:

- a. Data Redundancy:** Multiple departments (e.g., admissions, academics, administration) maintain their own copies of student records, leading to redundant data entry and increased chances of errors.
- b. Data Inaccuracy:** Manual data entry is prone to errors, resulting in inaccurate student records and potentially affecting academic decisions.
- c. Limited Access:** Students, instructors, and administrators face challenges accessing updated information due to the reliance on physical records and documents.
- d. Inefficient Communication:** Communication between students, instructors, faculty, and administrators is fragmented and slow due to the lack of a centralized platform.

- e. **Limited Reporting:** Generating comprehensive reports, such as student transcripts, enrollment statistics, attendance summaries, academic performance reports, requires manual compilation and is time-consuming.
- f. **Lack of Real-time Information:** The absence of a real-time system prevents timely access to up-to-date information, making decision-making cumbersome.
- g. **Difficulty in Tracking:** Monitoring student progress, course completion, and other milestones becomes challenging due to the absence of automated tracking mechanisms.

These issues result in data entry and processing consuming significant time and being error-prone, data storage and retrieval being difficult and insecure processes, reporting and data analysis being both incomplete and inaccurate, data sharing and communication being characterized by sluggishness and lack of uniformity, and data quality and accessibility being poor and unreliable.

iii. **Significance of Proposed System:**

The proposed student management system is a software application that uses MySQL DBMS or any MySQL environment to manage student data. The system aims to automate administrative tasks, improve data accuracy, enhance communication, and provide efficient reporting capabilities. The system can offer many benefits for the users, such as:

i. **Robotization and efficiency:**

A school management system can help reduce manual work and streamline various activities such as registration, course scheduling, attendance monitoring, and grading. The system can also automatically place students in courses according to their choices and prerequisites and compute their marks in accordance with their performance on assignments and exams.

ii. **Precise data:**

A school management system can store and manage accurate and consolidated data that is accessible to all departments. The system can also integrate security and validation measures to stop mistakes, losses, harm, or

unauthorized access to student data. For instance, the system may regularly encrypt and backup student data and demand identification and authorization before granting access to important information.

iii. Enhanced communication:

A school management system can provide a platform for seamless communication among students, instructors, and administrators. The system can also send and receive messages, notifications, reminders, and announcements through various channels such as email, SMS, or web. For example, the system can notify students about their course deadlines or changes and allow them to communicate with their instructors or peers through chat or video calls.

iv. Real-time Access:

A school management system can establish a centralized database that authorized users can access to up-to-date student information, enabling informed decisions and timely responses.

v. Data analysis and reporting:

A school management system can enable data analysis and reporting for academic decision making and improvement. The system can generate and display various reports, such as student transcripts, enrollment statistics, attendance summaries, academic performance reports using real-time data and graphical tools. For example, the system can show administrators the trends and patterns of student enrollment and retention over time and provide them with insights and recommendations for improving academic programs.

vi. User-friendly interface:

A school management system can offer an intuitive user interface that makes it easy for all stakeholders to interact with and navigate the platform. The system can also customize or personalize the user interface or experience according to the user's preferences or needs. For example, the system can allow users to choose their preferred language or theme for the interface and adjust the font size or contrast for better readability.

- vii. Scalability:** The system can accommodate growing numbers of students, courses, and faculty, ensuring long-term viability.

These are some of the main benefits that the proposed student management system can offer. By implementing this system, the users can expect to improve the efficiency and quality of student management.

Assignmentwise

2. Objective

Here are some more objectives for the current system:

- To automate administrative processes and reduce manual effort by the advanced system.
- To provide a secure and reliable platform for storing and managing student data.
- To maintain accurate and up-to-date student records.
- To improve the communication and collaboration among students, instructors, staff, and parents
- To simplify course management and scheduling.

AssignmentWise

3. Problem Statement

The proposed system is a software application that addresses the limitations of the existing manual system by automating tasks, reducing errors, and providing a centralized platform for efficient student management by using MySQL DBMS or any MySQL environment to manage student data, whereas the existing system is a manual and paper-based system that is prone to errors, losses, damages, and unauthorized access because it relies on human intervention and physical storage.

The several novel features and improvements over the existing system to address its shortcomings and enhance overall efficiency. These advancements can be summarized as follows:

- a. The new system automates tasks that used to take a lot of time and were prone to errors. It helps with student registration, course enrollment, attendance, and grading, making everything faster and more accurate.
- b. With the proposed system, data validation and verification processes ensure that the information entered is accurate and reliable, avoiding mistakes and discrepancies.
- c. Unlike the current system, which spreads student data across different files, the new system centralizes all the information in a well-organized database. This makes it easier to retrieve data and reduces the chances of duplicating information.
- d. The new system will be easy to use with a simple interface accessible through web browsers. It will make it easier for staff, employees, and students to navigate and use.
- e. The proposed system allows for instant communication between departments, faculty, and students. It enables notifications for course changes, assignments, and announcements, promoting collaboration and engagement among students.
- f. The proposed system automates report generation, saving time and effort. Administrators can customize reports and access detailed statistics on student enrollment, attendance, and grades.

- g.** The new system ensures data security and privacy through strong user authentication and access control measures, limiting access to sensitive student information.
- h.** The system is designed to be scalable, accommodating increases in student enrollment and changing academic requirements. Its modular structure allows for future updates and the incorporation of new features and technologies.
- i.** With the proposed system, teachers and administrators can analyze student performance data to gain valuable insights and improve classroom practices.
- j.** The new system will be mobile-friendly, allowing instructors and students to access it from smartphones and tablets for increased convenience and flexibility.
- k.** The proposed system includes regular data backup and recovery processes to protect against data loss and ensure system reliability.

In summary, the proposed Student Management System provides a thorough redesign of the current manual procedures. It makes use of contemporary technology to streamline processes, increase data accuracy, improve communication, and offer insightful information, thus creating a more effective and efficient academic management environment.

4. Scope

The proposed Student Management System aims to transform the way student-related processes are managed within the institution. The scope of the system covers a wide range of functionalities and features to enhance administrative efficiency, communication, and data accuracy. The scope includes, but is not limited to, the following:

- Student registration and enrollment.
- Storing and managing student data such as personal information, courses, grades, fees, attendance, etc.
- Attendance tracking.
- Facilitating online learning and teaching such as creating and delivering courses, assignments, exams, feedback, etc.
- Improving communication and collaboration such as sending and receiving messages, notifications, reminders, etc.
- Enabling data analysis and reporting such as generating and displaying reports, charts, graphs, etc.
- Faculty and administrator access for data management.
- Supporting accreditation and recognition such as providing and verifying certificates, transcripts, etc.
- Student access for course registration and viewing academic information.

5. Hardware & Software Specification

Hardware Specifications:

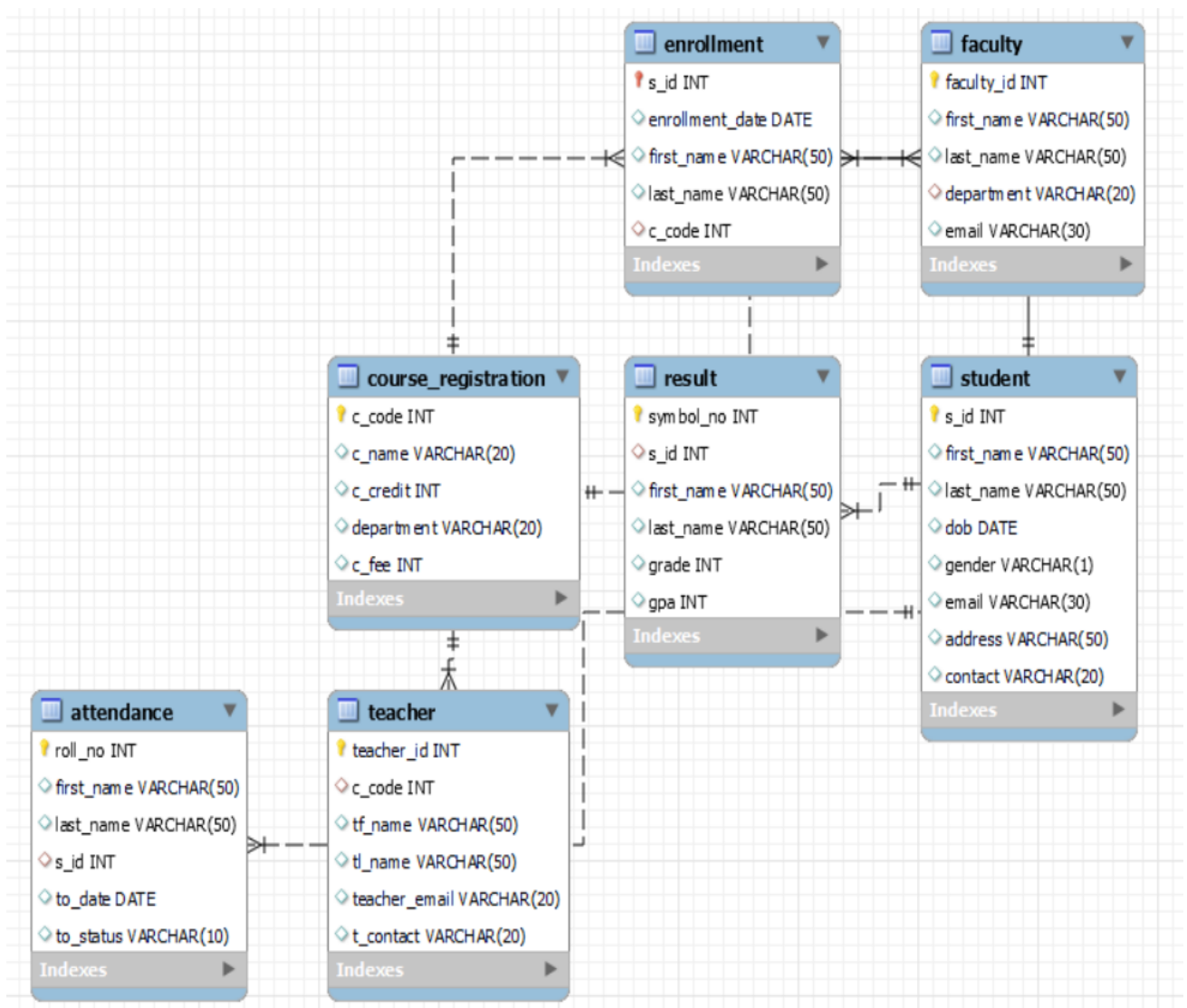
Name of Component		Specification
Server	Processor	Dual-core or higher
	RAM	8GB or more
	Storage	Minimum 500GB HDD or SSD
Client Workstations	Processor	Core i3 or equivalent
	RAM	4GB or more
	Storage	256GB HDD or SSD
	Monitor	16" color monitor
	Keyboard	122 Keys

Software Requirements:

Name of Component	Specification
Operating System	Linux (Ubuntu Server 20.04 LTS, CentOS 8) or Windows Server for hosting Client Workstations: Windows 10 or macOS, or Linux for user devices.
Web Server	Apache 2.4 or Nginx web server
Database Management System	MySQL Workbench 8.0 CE
Programming Language	HTML, CSS, JavaScript, PHP 37.4
Framework	Laravel 8 \ (PHP framework) for building the application
Front-end Framework	Bootstrap 5
Security Measures	Git
Development Environment	SSL/ TLS, Firewall and intrusion
IDE(Integrated Development Environment)	Visual Studio Code, Php Storm, sublime Text or any preferred IDE for coding
Browser Compatibility	Google Chrome, Mozilla Firefox, Safari, Microsoft Edge
Data Backup and Recovery	Regular automated data backups and recovery strategy
Network Infrastructure	LAN
Communication Tools	Email
Deployment Platform	AWS, Azure, shared hosting

The above hardware and software requirements provide a foundation for developing and deploying the Student Management System. It's essential to ensure that the chosen hardware and software components are capable of meeting the system's performance, security, and scalability needs.

6. Entity Relationship Diagram



7. Relational Schema

A relational schema is a textual representation of the tables, columns, keys, and constraints in database. A relational schema can help to implement database design and show the physical structure of the database.

- 1. student** (s_id, first_name, last_name, dob, gender, email, address, contact)
 - s_id is the primary key
- 2. course_registration** (c_code, c_name, c_credit, department, c_fee)
 - c_code is the primary key
- 3. faculty** (faculty_id, first_name, last_name, department, email)
 - faculty_id is Primary Key
 - department is the foreign key referencing course_registration
- 4. teacher** (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
 - teacher_id is the primary key
 - c_code is the foreign key referencing course_registration
- 5. result** (symbol_no, s_id, first_name, last_name, grade, gpa)
 - symbol_no is the primary key
 - s_id is the foreign key referencing student
- 6. attendance** (roll_no, first_name, last_name, s_id, to_date, to_status)
 - roll_no is primary key
 - s_id is the foreign key referencing student
- 7. enrollment** (s_id, enrollment_date, first_name, last_name, c_code)
 - s_id and c_code are the primary key
 - s_id is a foreign key referencing student
 - c_code is a foreign key referencing course_registration

8. Normalization

1. student(s_id, first_name, last_name, dob, gender, email, address, contact)

- s_id is the primary key

s_id	first_name	last_name	dob	gender	email	address	contact
1	Manisha	Khatri	2000-01-21	F	m@gmail.com	KTM	9842849306
2	Abin	Rai	2001-02-18	M	a@gmail.com	BKT	9746994366
3	Mandip	Rai	2003-05-06	M	mr@gmail.com	KTNG	9819794812
4	Dikshya	Khanal	2003-12-56	O	d@gmail.com	BRT	9746994365

First-Normal Form (1NF):

The original schema is already in 1NF since it contains atomic values.

Second-Normal Form (2NF):

The original schema satisfies the 2NF condition because the primary key (s_id) uniquely identifies all the other attributes (all non-key attributes depend entirely on the whole primary key (s_id)). There is no need to modify the schema.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this student schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

2. course_registration (c_code, c_name, c_credit, department, c_fee)

- c_code is the primary key

c_code	c_name	c_credit	department	c_fee
50	Science	4	SD	12300
51	Math	5	MD	13400
52	Computer	4	CD	12300
53	Nepali	7	ND	12600

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

We can check for partial dependency by seeing if any non-key attribute depends on only part of the key. There is no partial dependency in this schema because all the other attributes (c_name, c_credit, department, c_fee) depend on the whole primary key (c_code). So, the schema is already in 2NF. We don't need to do anything else to create new tables.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this course_registration schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

3. faculty (faculty_id, first_name, last_name, department, email)
- faculty_id is Primary Key
 - department is the foreign key referencing course_registration

faculty_id	first_name	last_name	department	email
100	Manisha	Khatri	MD	m@gmail.com
101	Abin	Rai	CD	a@gmail.com
102	Mandip	Rai	ND	mr@gmail.com
103	Dikshya	Khanal	SD	d@gmail.com

First-Normal Form (1NF):

The original schema seems to be in 1NF, as all attributes appear to hold atomic values.

Second-Normal Form (2NF):

In this case, it seems that all attributes fully depend on the primary key (faculty_id). So, there doesn't seem to be any partial dependency, and the schema seems to be in 2NF.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

4. teacher (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
 - teacher_id is the primary key
 - c_code is the foreign key referencing course_registration

teacher_id	c_code	tf_name	tl_name	teacher_email	t_contact
300	51	Prabhat	Shrestha	ps@gmail.com	9837463728
301	52	Meghraj	Adhikari	mi@gmail.com	9283746501
302	53	Roshan	Khatri	rk@gmail.com	9182635486
303	50	Ram	Rai	rr@gmail.com	9785634251

First-Normal Form (1NF):

The original schema seems to be in 1NF, as all attributes hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "c_code" is a foreign key referencing a course, and it seems to depend on the whole primary key (teacher_id). So, there doesn't seem to be partial dependency, and the schema seems to be in 2NF.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, "c_code" seems to depend directly on the primary key "teacher_id," and there are no transitive dependencies among the attributes. So, based on the information provided, the schema is already in 3NF.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

5. result (symbol_no, s_id , first_name, last_name, grade, gpa)

- symbol_no is the primary key
- s_id is the foreign key referencing student

symbol_no	s_id	first_name	last_name	grade	gpa
1230	1	Manisha	Khatri	8	3.44
1231	2	Abin	Rai	9	2.19
1232	3	Mandip	Rai	10	3.67
1233	4	Dikshya	Khanal	8	1.23

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (symbol_no), and it seems to depend on "s_id." This shows a partial dependency, as the student name depends only on some of the primary key. To fix this, we need to split the schema into more tables:

Student

s_id	first_name	last_name
1	Manisha	Khatri
2	Abin	Rai
3	Mandip	Rai
4	Dikshya	Khanal

Result

symbol_no	s_id	grade	gpa
1230	1	8	3.44
1231	2	9	2.19
1232	3	10	3.67
1233	4	8	1.23

By doing this, we eliminate the partial dependency by making sure that "first_name" and "last_name" are dependent on the whole primary key of the "Student" table.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

6. attendance (roll_no, first_name, last_name, s_id, to_date, to_status)

- roll_no is primary key
- s_id is the foreign key referencing student

roll_no	first_name	last_name	s_id	to_date	to_status
25	Abin	Rai	2	2023-08-21	P
26	Dikshya	Khanal	4	2023-08-21	A
27	Mandip	Rai	3	2023-08-21	P
28	Manisha	Khatri	1	2023-08-21	P

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (roll_no), and it seems to depend on "roll_no." This shows a partial dependency, as the student name depends only on some of the primary key. To fix this, we should make a different table for students:

Student

s_id	first_name	last_name
2	Abin	Rai
4	Dikshya	Khanal
3	Mandip	Rai
1	Manisha	Khatri

Attendance

roll_no	s_id	to_date	to_status
25	2	2023-08-21	P
26	4	2023-08-21	A
27	3	2023-08-21	P
28	1	2023-08-21	P

By doing this, we remove the partial dependency by ensuring that "first_name" and "last_name" are dependent on the whole primary key of the "student" table.

Third-Normal Form (3NF):

To evaluate for potential transitive dependencies, we need to consider whether any non-key attribute depends on another non-key attribute. In this schema, there doesn't appear to be any transitive dependencies among the attributes. The tables created based on the 2NF process seem to satisfy 3NF as well.

Assignmentwise

7. enrollment (s_id, enrollment_date, first_name, last_name, c_code)

- s_id and c_code are the primary key
- s_id is a foreign key referencing student
- c_code is a foreign key referencing course_registration

s_id	enrollment_date	first_name	last_name	c_code
2	2023-02-12	Abin	Rai	52
4	2023-03-02	Dikshya	Khanal	50
3	2023-02-20	Mandip	Rai	53
1	2023-01-25	Manisha	Khatrri	51

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (s_id, c_code), and it seems to depend only on "s_id." This shows a partial dependency, as the first_name and last_name depends only on part of the primary key. To fix this, we should make a different table for students:

Student

s_id	first_name	last_name
2	Abin	Rai
4	Dikshya	Khanal
3	Mandip	Rai
1	Manisha	Khatrri

Enrollment

s_id	enrollment_date	c_code
2	2023-02-12	62
4	2023-03-02	60
3	2023-02-20	63
1	2023-01-25	61

By doing this, we remove the partial dependency by ensuring that "first_name" and "last_name" is dependent on the whole primary key of the "Student" table.

Third-Normal Form (3NF):

To evaluate for potential transitive dependencies, we need to consider whether any non-key attribute depends on another non-key attribute. In this schema, there doesn't seem to be any transitive dependencies among the attributes. The tables created based on the 2NF process seem to satisfy 3NF as well.

Part B (Database Design)

Table

```
1 • CREATE DATABASE student_management_system;
2 • USE student_management_system;
3
4 • CREATE TABLE student(
5     s_id INT AUTO_INCREMENT PRIMARY KEY,
6     first_name VARCHAR(50),
7     last_name VARCHAR(50),
8     dob DATE,
9     gender VARCHAR(1),
10    email VARCHAR(30),
11    address VARCHAR(50),
12    contact VARCHAR(20)
13 );
14
15 • SELECT * FROM student;

```

```
18 • CREATE TABLE course_registration (
19     c_code INT PRIMARY KEY,
20     c_name VARCHAR(20),
21     c_credit INT,
22     department VARCHAR(20),
23     c_fee INT
24 );
25
26 • CREATE INDEX idx_department ON course_registration(department);

```

```
31 • CREATE TABLE faculty (
32     faculty_id INT PRIMARY KEY,
33     first_name VARCHAR(50),
34     last_name VARCHAR(50),
35     department VARCHAR(20),
36     email VARCHAR(30),
37     FOREIGN KEY (department) REFERENCES course_registration(department)
38 );
39
40 • SELECT * FROM faculty;

```

```
40 • CREATE TABLE teacher (  
41     teacher_id INT PRIMARY KEY,  
42     c_code INT,  
43     tf_name VARCHAR(50),  
44     tl_name VARCHAR(50),  
45     teacher_email VARCHAR(20),  
46     t_contact VARCHAR(20),  
47     FOREIGN KEY (c_code) REFERENCES course_registration(c_code)  
48 );  
49  
50 • SELECT * FROM teacher;
```

```
53 • CREATE TABLE result (  
54     symbol_no INT PRIMARY KEY,  
55     s_id INT,  
56     first_name VARCHAR(50),  
57     last_name VARCHAR(50),  
58     grade INT,  
59     gpa INT,  
60     FOREIGN KEY (s_id) REFERENCES student(s_id)  
61 );  
62  
63 • SELECT * FROM result;  
64
```

```
66 • CREATE TABLE attendance (  
67     roll_no INT PRIMARY KEY,  
68     first_name VARCHAR(50),  
69     last_name VARCHAR(50),  
70     s_id INT,  
71     to_date DATE,  
72     to_status VARCHAR(10),  
73     FOREIGN KEY (s_id) REFERENCES student(s_id)  
74 );  
75  
76 • SELECT * FROM attendance;  
77
```

```
79 • CREATE TABLE enrollment (  
80     s_id INT PRIMARY KEY,  
81     enrollment_date DATE,  
82     first_name VARCHAR(50),  
83     last_name VARCHAR(50),  
84     c_code INT,  
85     FOREIGN KEY (s_id) REFERENCES student(s_id),  
86     FOREIGN KEY (c_code) REFERENCES course_registration(c_code)  
87 );  
88  
89 • SELECT * FROM enrollment;  
90
```

Assignment 2

Table Values

```
1 • INSERT INTO student(first_name, last_name, dob, gender, email, address, contact)
2 VALUES
3 ('Manisha','Khatrri', '2000-01-21', 'F', 'm@gmail.com', 'KTM', '9842849306' ),
4 ('Abin', 'Rai', '2001-02-18', 'M', 'a@gmail.com', 'BKT', '9746994366' ),
5 ('Mandip', 'Rai', '2003-05-06', 'M', 'mr@gmail.com', 'KTNG', '9819794812' ),
6 ('Dikshya', 'Khanal', '2003-12-26', 'O', 'd@gmail.com', 'BRT', '9746994365' );
7
```

```
8 • INSERT INTO course_registration (c_code, c_name, c_credit, department, c_fee)
9 VALUES
10 (50, 'Science', 4, 'SD', 12300),
11 (51, 'Math', 5, 'MD', 13400),
12 (52, 'Computer', 4, 'CD', 12300),
13 (53, 'Nepali', 7, 'ND', 12600);
14
```

```
15 • INSERT INTO faculty (faculty_id , first_name , last_name , department , email)
16 VALUES
17 (100, 'Manisha', 'Khatrri', 'MD', 'm@gmail.com'),
18 (101, 'Abin', 'Rai', 'CD', 'a@gmail.com'),
19 (102, 'Mandip', 'Rai', 'ND', 'mr@gmail.com'),
20 (103, 'Dikshya', 'Khanal', 'SD', 'd@gmail.com');
21
```

```
22 • INSERT INTO teacher (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
23 VALUES
24 (300, 51, 'Prabhat', 'Shrestha', 'ps@gmail.com', '9837463728'),
25 (301, 52, 'Meghraj', 'Adhikari', 'mi@gmail.com', '9283746501'),
26 (302, 53, 'Roshan', 'Khatrri', 'rk@gmail.com', '9182635486'),
27 (303, 50, 'Ram', 'Rai', 'rr@gmail.com', '9785634251');
28
```

```
29 • INSERT INTO result (symbol_no, s_id , first_name, last_name, grade, gpa)
30 VALUES
31 (1230, 1, 'Manisha', 'Khatrri', 8, 3.44),
32 (1231, 2, 'Abin', 'Rai', 9, 2.19),
33 (1232, 3, 'Mandip', 'Rai', 10, 3.67),
34 (1233, 4, 'Dikshya', 'Khanal', 8, 1.23);
35
```

```
36 • INSERT INTO attendance (roll_no, first_name, last_name, s_id, to_date, to_status)
37 VALUES
38 (25, 'Abin', 'Rai', 2, '2023-08-21', 'P'),
39 (26, 'Dikshya', 'Khanal', 4, '2023-08-21', 'A'),
40 (27, 'Mandip', 'Rai', 3, '2023-08-21', 'P'),
41 (28, 'Manisha', 'Khatrri', 1, '2023-08-21', 'P');
42
```

```
43 • INSERT INTO enrollment (s_id, enrollment_date, first_name, last_name, c_code)
44 VALUES
45 (2, '2023-02-12', 'Abin', 'Rai', 52),
46 (4, '2023-03-02', 'Dikshya', 'Khanal', 50),
47 (3, '2023-02-20', 'Mandip', 'Rai', 53),
48 (1, '2023-01-25', 'Manisha', 'Khatrri', 51);
```

Output Table

student

	s_id	first_name	last_name	dob	gender	email	address	contact
▶	1	Manisha	Khatri	2000-01-21	F	m@gmail.com	KTM	9842849306
	2	Abin	Rai	2001-02-18	M	a@gmail.com	BKT	9746994366
	3	Mandip	Rai	2003-05-06	M	mr@gmail.com	KTNG	9819794812
	4	Dikshya	Khanal	2003-12-26	O	d@gmail.com	BRT	9746994365
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

course_registration

	c_code	c_name	c_credit	department	c_fee
▶	50	Science	4	SD	12300
	51	Math	5	MD	13400
	52	Computer	4	CD	12300
	53	Nepali	7	ND	12600
✱	NULL	NULL	NULL	NULL	NULL

faculty

	faculty_id	first_name	last_name	department	email
▶	100	Manisha	Khatri	MD	m@gmail.com
	101	Abin	Rai	CD	a@gmail.com
	102	Mandip	Rai	ND	mr@gmail.com
	103	Dikshya	Khanal	SD	d@gmail.com
✱	NULL	NULL	NULL	NULL	NULL

teacher

Result Grid						
	teacher_id	c_code	tf_name	tl_name	teacher_email	t_contact
▶	300	51	Prabhat	Shrestha	ps@gmail.com	9837463728
	301	52	Meghraj	Adhikari	mi@gmail.com	9283746501
	302	53	Roshan	Khatri	rk@gmail.com	9182635486
	303	50	Ram	Rai	rr@gmail.com	9785634251
✱	NULL	NULL	NULL	NULL	NULL	NULL

result

Result Grid						
	symbol_no	s_id	first_name	last_name	grade	gpa
▶	1230	1	Manisha	Khatri	8	3
	1231	2	Abin	Rai	9	2
	1232	3	Mandip	Rai	10	4
	1233	4	Dikshya	Khanal	8	1
✱	NULL	NULL	NULL	NULL	NULL	NULL

attendance

Result Grid						
	roll_no	first_name	last_name	s_id	to_date	to_status
▶	25	Abin	Rai	2	2023-08-21	P
	26	Dikshya	Khanal	4	2023-08-21	A
	27	Mandip	Rai	3	2023-08-21	P
	28	Manisha	Khatri	1	2023-08-21	P
✱	NULL	NULL	NULL	NULL	NULL	NULL

enrollment

Result Grid					
		Filter Rows:		Edit:	
	s_id	enrollment_date	first_name	last_name	c_code
▶	1	2023-01-25	Manisha	Khatri	51
	2	2023-02-12	Abin	Rai	52
	3	2023-02-20	Mandip	Rai	53
	4	2023-03-02	Dikshya	Khanal	50
★	NULL	NULL	NULL	NULL	NULL

Assignmentwise

Part-C

1. Background

i. Introduction to Current System:

In the ever-evolving landscape of education, efficient management of student-related information and administrative processes is crucial for the seamless functioning of educational institutions. Recognizing that the existing Student Management System operates through manual, paper-dependent processes and lacks cohesive integration. Administrative duties, student record maintenance, and communication all rely on manual methods, resulting in inefficiencies and inaccuracies. Administrative personnel are responsible for documenting and maintaining student details through physical forms and records. Educators must assess and evaluate student assignments and tests using hardcopy materials and calculators. Enrolment in courses, fee payments, and transcript inquiries for students entail completing and submitting paper forms. Communication among students, teachers, staff, and parents occurs via telephone conversations, emails, or written correspondence.

ii. Problem with the Existing System:

The existing system is a manual and paper-based system that suffers from various issues that impair the efficiency and quality of student management. Some of the issues are:

- a. Data Redundancy:** Multiple departments (e.g., admissions, academics, administration) maintain their own copies of student records, leading to redundant data entry and increased chances of errors.
- b. Data Inaccuracy:** Manual data entry is prone to errors, resulting in inaccurate student records and potentially affecting academic decisions.
- c. Limited Access:** Students, instructors, and administrators face challenges accessing updated information due to the reliance on physical records and documents.

- d. Inefficient Communication:** Communication between students, instructors, faculty, and administrators is fragmented and slow due to the lack of a centralized platform.
- e. Limited Reporting:** Generating comprehensive reports, such as student transcripts, enrollment statistics, attendance summaries, academic performance reports, requires manual compilation and is time-consuming.
- f. Lack of Real-time Information:** The absence of a real-time system prevents timely access to up-to-date information, making decision-making cumbersome.
- g. Difficulty in Tracking:** Monitoring student progress, course completion, and other milestones becomes challenging due to the absence of automated tracking mechanisms.

These issues result in data entry and processing consuming significant time and being error-prone, data storage and retrieval being difficult and insecure processes, reporting and data analysis being both incomplete and inaccurate, data sharing and communication being characterized by sluggishness and lack of uniformity, and data quality and accessibility being poor and unreliable.

iii. Significance of Proposed System:

The proposed student management system is a software application that uses MySQL DBMS or any MySQL environment to manage student data. The system aims to automate administrative tasks, improve data accuracy, enhance communication, and provide efficient reporting capabilities. The system can offer many benefits for the users, such as:

a. Robotization and efficiency:

A school management system can help reduce manual work and streamline various activities such as registration, course scheduling, attendance monitoring, and grading. The system can also automatically place students in courses according to their choices and prerequisites and compute their marks in accordance with their performance on assignments and exams.

b. Precise data:

A school management system can store and manage accurate and consolidated data that is accessible to all departments. The system can also integrate security and validation measures to stop mistakes, losses, harm, or unauthorized access to student data. For instance, the system may regularly encrypt and backup student data and demand identification and authorization before granting access to important information.

c. Enhanced communication:

A school management system can provide a platform for seamless communication among students, instructors, and administrators. The system can also send and receive messages, notifications, reminders, and announcements through various channels such as email, SMS, or web. For example, the system can notify students about their course deadlines or changes and allow them to communicate with their instructors or peers through chat or video calls.

d. Real-time Access:

A school management system can establish a centralized database that authorized users can access to up-to-date student information, enabling informed decisions and timely responses.

e. Data analysis and reporting:

A school management system can enable data analysis and reporting for academic decision making and improvement. The system can generate and display various reports, such as student transcripts, enrollment statistics, attendance summaries, academic performance reports using real-time data and graphical tools. For example, the system can show administrators the trends and patterns of student enrollment and retention over time and provide them with insights and recommendations for improving academic programs.

f. User-friendly interface:

A school management system can offer an intuitive user interface that makes it easy for all stakeholders to interact with and navigate the platform. The system can also customize or personalize the user interface or experience according to the user's preferences or needs. For example, the system can

allow users to choose their preferred language or theme for the interface and adjust the font size or contrast for better readability.

- g. Scalability:** The system can accommodate growing numbers of students, courses, and faculty, ensuring long-term viability.

These are some of the main benefits that the proposed student management system can offer. By implementing this system, the users can expect to improve the efficiency and quality of student management.

Assignmentwise

2. Objective

Here are some more objectives for the current system:

- To automate administrative processes and reduce manual effort by the advanced system.
- To provide a secure and reliable platform for storing and managing student data.
- To maintain accurate and up-to-date student records.
- To improve the communication and collaboration among students, instructors, staff, and parents
- To simplify course management and scheduling.

AssignmentWise

3. Problem Statement

The proposed system is a software application that addresses the limitations of the existing manual system by automating tasks, reducing errors, and providing a centralized platform for efficient student management by using MySQL DBMS or any MySQL environment to manage student data, whereas the existing system is a manual and paper-based system that is prone to errors, losses, damages, and unauthorized access because it relies on human intervention and physical storage.

The several novel features and improvements over the existing system to address its shortcomings and enhance overall efficiency. These advancements can be summarized as follows:

- a. The new system automates tasks that used to take a lot of time and were prone to errors. It helps with student registration, course enrollment, attendance, and grading, making everything faster and more accurate.
- b. With the proposed system, data validation and verification processes ensure that the information entered is accurate and reliable, avoiding mistakes and discrepancies.
- c. Unlike the current system, which spreads student data across different files, the new system centralizes all the information in a well-organized database. This makes it easier to retrieve data and reduces the chances of duplicating information.
- d. The new system will be easy to use with a simple interface accessible through web browsers. It will make it easier for staff, employees, and students to navigate and use.
- e. The proposed system allows for instant communication between departments, faculty, and students. It enables notifications for course changes, assignments, and announcements, promoting collaboration and engagement among students.
- f. The proposed system automates report generation, saving time and effort. Administrators can customize reports and access detailed statistics on student enrollment, attendance, and grades.

- g. The new system ensures data security and privacy through strong user authentication and access control measures, limiting access to sensitive student information.
- h. The system is designed to be scalable, accommodating increases in student enrollment and changing academic requirements. Its modular structure allows for future updates and the incorporation of new features and technologies.
- i. With the proposed system, teachers and administrators can analyze student performance data to gain valuable insights and improve classroom practices.
- j. The new system will be mobile-friendly, allowing instructors and students to access it from smartphones and tablets for increased convenience and flexibility.
- k. The proposed system includes regular data backup and recovery processes to protect against data loss and ensure system reliability.

In summary, the proposed Student Management System provides a thorough redesign of the current manual procedures. It makes use of contemporary technology to streamline processes, increase data accuracy, improve communication, and offer insightful information, thus creating a more effective and efficient academic management environment.

4. Scope

The proposed Student Management System aims to transform the way student-related processes are managed within the institution. The scope of the system covers a wide range of functionalities and features to enhance administrative efficiency, communication, and data accuracy. The scope includes, but is not limited to, the following:

- Student registration and enrollment.
- Storing and managing student data such as personal information, courses, grades, fees, attendance, etc.
- Attendance tracking.
- Facilitating online learning and teaching such as creating and delivering courses, assignments, exams, feedback, etc.
- Improving communication and collaboration such as sending and receiving messages, notifications, reminders, etc.
- Enabling data analysis and reporting such as generating and displaying reports, charts, graphs, etc.
- Faculty and administrator access for data management.
- Supporting accreditation and recognition such as providing and verifying certificates, transcripts, etc.
- Student access for course registration and viewing academic information.

5. Hardware & Software Specification

Hardware Specifications:

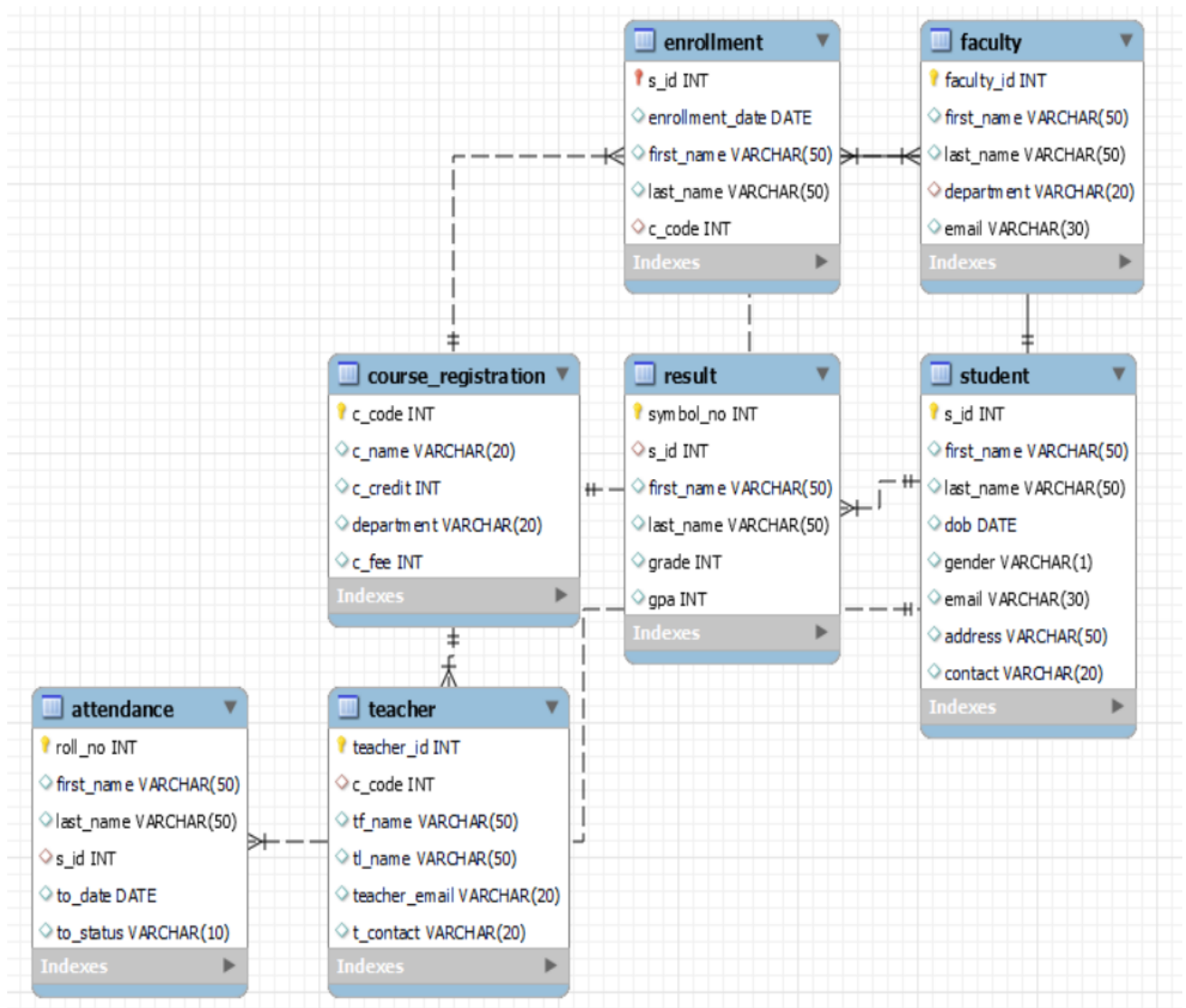
Name of Component		Specification
Server	Processor	Dual-core or higher
	RAM	8GB or more
	Storage	Minimum 500GB HDD or SSD
Client Workstations	Processor	Core i3 or equivalent
	RAM	4GB or more
	Storage	256GB HDD or SSD
	Monitor	16" color monitor
	Keyboard	122 Keys

Software Requirements:

Name of Component	Specification
Operating System	Linux (Ubuntu Server 20.04 LTS, CentOS 8) or Windows Server for hosting Client Workstations: Windows 10 or macOS, or Linux for user devices.
Web Server	Apache 2.4 or Nginx web server
Database Management System	MySQL Workbench 8.0 CE
Programming Language	HTML, CSS, JavaScript, PHP 37.4
Framework	Laravel 8 \ (PHP framework) for building the application
Front-end Framework	Bootstrap 5
Security Measures	Git
Development Environment	SSL/ TLS, Firewall and intrusion
IDE(Integrated Development Environment)	Visual Studio Code, Php Storm, sublime Text or any preferred IDE for coding
Browser Compatibility	Google Chrome, Mozilla Firefox, Safari, Microsoft Edge
Data Backup and Recovery	Regular automated data backups and recovery strategy
Network Infrastructure	LAN
Communication Tools	Email
Deployment Platform	AWS, Azure, shared hosting

The above hardware and software requirements provide a foundation for developing and deploying the Student Management System. It's essential to ensure that the chosen hardware and software components are capable of meeting the system's performance, security, and scalability needs.

6. Entity Relationship Diagram



7. Relational Schema

A relational schema is a textual representation of the tables, columns, keys, and constraints in database. A relational schema can help to implement database design and show the physical structure of the database.

1. student (s_id, first_name, last_name, dob, gender, email, address, contact)
 - s_id is the primary key
2. course_registration (c_code, c_name, c_credit, department, c_fee)
 - c_code is the primary key
 - department is the foreign key referencing course_registration
3. faculty (faculty_id, first_name, last_name, department, email)
 - faculty_id is Primary Key
4. teacher (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
 - teacher_id is the primary key
 - c_code is the foreign key referencing course_registration
5. result (symbol_no, s_id, first_name, last_name, grade, gpa)
 - symbol_no is the primary key
 - s_id is the foreign key referencing student
6. attendance (roll_no, first_name, last_name, s_id, to_date, to_status)
 - roll_no is primary key
 - s_id is the foreign key referencing student
7. enrollment (s_id, enrollment_date, first_name, last_name, c_code)
 - s_id and c_code are the primary key
 - s_id is a foreign key referencing student
 - c_code is a foreign key referencing course_registration

8. Normalization

1. student(s_id, first_name, last_name, dob, gender, email, address, contact)

- s_id is the primary key

<u>s_id</u>	first_name	last_name	dob	gender	email	address	contact
1	Manisha	Khatri	2000-01-21	F	m@gmail.com	KTM	9842849306
2	Abin	Rai	2001-02-18	M	a@gmail.com	BKT	9746994366
3	Mandip	Rai	2003-05-06	M	mr@gmail.com	KTNG	9819794812
4	Dikshya	Khanal	2003-12-56	O	d@gmail.com	BRT	9746994365

First-Normal Form (1NF):

The original schema is already in 1NF since it contains atomic values.

Second-Normal Form (2NF):

The original schema satisfies the 2NF condition because the primary key (s_id) uniquely identifies all the other attributes (all non-key attributes depend entirely on the whole primary key (s_id)). There is no need to modify the schema.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this student schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

2. course_registration (c_code, c_name, c_credit, department, c_fee)

- c_code is the primary key

<u>c_code</u>	c_name	c_credit	department	c_fee
50	Science	4	SD	12300
51	Math	5	MD	13400
52	Computer	4	CD	12300
53	Nepali	7	ND	12600

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

We can check for partial dependency by seeing if any non-key attribute depends on only part of the key. There is no partial dependency in this schema because all the other attributes (c_name, c_credit, department, c_fee) depend on the whole primary key (c_code). So, the schema is already in 2NF. We don't need to do anything else to create new tables.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this course_registration schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

3. faculty (faculty_id, first_name, last_name, department, email)

- faculty_id is Primary Key
- department is the foreign key referencing course_registration

faculty_id	first_name	last_name	department	email
100	Manisha	Khatr	MD	m@gmail.com
101	Abin	Rai	CD	a@gmail.com
102	Mandip	Rai	ND	mr@gmail.com
103	Dikshya	Khanal	SD	d@gmail.com

First-Normal Form (1NF):

The original schema seems to be in 1NF, as all attributes appear to hold atomic values.

Second-Normal Form (2NF):

In this case, it seems that all attributes fully depend on the primary key (faculty_id). So, there doesn't seem to be any partial dependency, and the schema seems to be in 2NF.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

4. teacher (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
- teacher_id is the primary key
 - c_code is the foreign key referencing course_registration

teacher_id	c_code	tf_name	tl_name	teacher_email	t_contact
300	51	Prabhat	Shrestha	ps@gmail.com	9837463728
301	52	Meghraj	Adhikari	mi@gmail.com	9283746501
302	53	Roshan	Khatri	rk@gmail.com	9182635486
303	50	Ram	Rai	rr@gmail.com	9785634251

First-Normal Form (1NF):

The original schema seems to be in 1NF, as all attributes hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "c_code" is a foreign key referencing a course, and it seems to depend on the whole primary key (teacher_id). So, there doesn't seem to be partial dependency, and the schema seems to be in 2NF.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, "c_code" seems to depend directly on the primary key "teacher_id," and there are no transitive dependencies among the attributes. So, based on the information provided, the schema is already in 3NF.

In this case, there's no need to decompose the schema further, as it appears to satisfy 1NF, 2NF, and 3NF.

5. result (symbol_no, s_id , first_name, last_name, grade, gpa)

- symbol_no is the primary key
- s_id is the foreign key referencing student

symbol_no	s_id	first_name	last_name	grade	gpa
1230	1	Manisha	Khatri	8	3.44
1231	2	Abin	Rai	9	2.19
1232	3	Mandip	Rai	10	3.67
1233	4	Dikshya	Khanal	8	1.23

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (symbol_no), and it seems to depend on "s_id." This shows a partial dependency, as the student name depends only on some of the primary key. To fix this, we need to split the schema into more tables:

Student

s_id	first_name	last_name
1	Manisha	Khatri
2	Abin	Rai
3	Mandip	Rai
4	Dikshya	Khanal

Result

symbol_no	s_id	grade	gpa
1230	1	8	3.44
1231	2	9	2.19
1232	3	10	3.67
1233	4	8	1.23

By doing this, we eliminate the partial dependency by making sure that "first_name" and "last_name" are dependent on the whole primary key of the "Student" table.

Third-Normal Form (3NF):

To find out if there's a transitive dependency, we need to see if any non-key attribute depends on another non-key attribute. In this schema, there are no obvious transitive dependencies among the attributes. So, the tables made based on the 2NF process seem to meet 3NF as well.

6. attendance (roll_no, first_name, last_name, s_id, to_date, to_status)

- roll_no is primary key
- s_id is the foreign key referencing student

roll_no	first_name	last_name	s_id	to_date	to_status
25	Abin	Rai	2	2023-08-21	P
26	Dikshya	Khanal	4	2023-08-21	A
27	Mandip	Rai	3	2023-08-21	P
28	Manisha	Khatri	1	2023-08-21	P

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes seem to hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (roll_no), and it seems to depend on "roll_no." This shows a partial dependency, as the student name depends only on some of the primary key. To fix this, we should make a different table for students:

Student

s_id	first_name	last_name
2	Abin	Rai
4	Dikshya	Khanal
3	Mandip	Rai
1	Manisha	Khatri

Attendance

roll_no	s_id	to_date	to_status
25	2	2023-08-21	P
26	4	2023-08-21	A
27	3	2023-08-21	P
28	1	2023-08-21	P

By doing this, we remove the partial dependency by ensuring that "first_name" and "last_name" are dependent on the whole primary key of the "student" table.

Third-Normal Form (3NF):

To evaluate for potential transitive dependencies, we need to consider whether any non-key attribute depends on another non-key attribute. In this schema, there doesn't appear to be any transitive dependencies among the attributes. The tables created based on the 2NF process seem to satisfy 3NF as well.

Assignmentwise

7. enrollment (s_id, enrollment_date, first_name, last_name, c_code)

- s_id and c_code are the primary key
- s_id is a foreign key referencing student
- c_code is a foreign key referencing course_registration

s_id	enrollment_date	first_name	last_name	c_code
2	2023-02-12	Abin	Rai	52
4	2023-03-02	Dikshya	Khanal	50
3	2023-02-20	Mandip	Rai	53
1	2023-01-25	Manisha	Khatrri	51

First-Normal Form (1NF):

The original schema appears to be in 1NF, as all attributes hold atomic values.

Second-Normal Form (2NF):

To check for partial dependency, we need to see if any non-key attribute depends on only some of the primary key. In this case, "first_name" and "last_name" is not part of the primary key (s_id, c_code), and it seems to depend only on "s_id." This shows a partial dependency, as the first_name and last_name depends only on part of the primary key. To fix this, we should make a different table for students:

Student

s_id	first_name	last_name
2	Abin	Rai
4	Dikshya	Khanal
3	Mandip	Rai
1	Manisha	Khatrri

Enrollment

s_id	enrollment_date	c_code
2	2023-02-12	62
4	2023-03-02	60
3	2023-02-20	63
1	2023-01-25	61

By doing this, we remove the partial dependency by ensuring that "first_name" and "last_name" is dependent on the whole primary key of the "Student" table.

Third-Normal Form (3NF):

To evaluate for potential transitive dependencies, we need to consider whether any non-key attribute depends on another non-key attribute. In this schema, there doesn't seem to be any transitive dependencies among the attributes. The tables created based on the 2NF process seem to satisfy 3NF as well.

Print screen of system user interface

Table

```
1 • CREATE DATABASE student_management_system;
2 • USE student_management_system;
3
4 • CREATE TABLE student(
5     s_id INT AUTO_INCREMENT PRIMARY KEY,
6     first_name VARCHAR(50),
7     last_name VARCHAR(50),
8     dob DATE,
9     gender VARCHAR(1),
10    email VARCHAR(30),
11    address VARCHAR(50),
12    contact VARCHAR(20)
13 );
14
15 • SELECT * FROM student;

18 • CREATE TABLE course_registration (
19     c_code INT PRIMARY KEY,
20     c_name VARCHAR(20),
21     c_credit INT,
22     department VARCHAR(20),
23     c_fee INT
24 );
25
26 • CREATE INDEX idx_department ON course_registration(department);

31 • CREATE TABLE faculty (
32     faculty_id INT PRIMARY KEY,
33     first_name VARCHAR(50),
34     last_name VARCHAR(50),
35     department VARCHAR(20),
36     email VARCHAR(30),
37     FOREIGN KEY (department) REFERENCES course_registration(department)
38 );
39
40 • SELECT * FROM faculty;
```

```
40 • CREATE TABLE teacher (  
41     teacher_id INT PRIMARY KEY,  
42     c_code INT,  
43     tf_name VARCHAR(50),  
44     tl_name VARCHAR(50),  
45     teacher_email VARCHAR(20),  
46     t_contact VARCHAR(20),  
47     FOREIGN KEY (c_code) REFERENCES course_registration(c_code)  
48 );  
49  
50 • SELECT * FROM teacher;
```

```
53 • CREATE TABLE result (  
54     symbol_no INT PRIMARY KEY,  
55     s_id INT,  
56     first_name VARCHAR(50),  
57     last_name VARCHAR(50),  
58     grade INT,  
59     gpa INT,  
60     FOREIGN KEY (s_id) REFERENCES student(s_id)  
61 );  
62  
63 • SELECT * FROM result;  
64
```

```
66 • CREATE TABLE attendance (  
67     roll_no INT PRIMARY KEY,  
68     first_name VARCHAR(50),  
69     last_name VARCHAR(50),  
70     s_id INT,  
71     to_date DATE,  
72     to_status VARCHAR(10),  
73     FOREIGN KEY (s_id) REFERENCES student(s_id)  
74 );  
75  
76 • SELECT * FROM attendance;  
77
```

```
79 • CREATE TABLE enrollment (  
80     s_id INT PRIMARY KEY,  
81     enrollment_date DATE,  
82     first_name VARCHAR(50),  
83     last_name VARCHAR(50),  
84     c_code INT,  
85     FOREIGN KEY (s_id) REFERENCES student(s_id),  
86     FOREIGN KEY (c_code) REFERENCES course_registration(c_code)  
87 );  
88  
89 • SELECT * FROM enrollment;  
90
```

Assignment 2

Table Values

```
1 • INSERT INTO student(first_name, last_name, dob, gender, email, address, contact)
2 VALUES
3 ('Manisha','Khatrri', '2000-01-21', 'F', 'm@gmail.com', 'KTM', '9842849306' ),
4 ('Abin', 'Rai', '2001-02-18', 'M', 'a@gmail.com', 'BKT', '9746994366' ),
5 ('Mandip', 'Rai', '2003-05-06', 'M', 'mr@gmail.com', 'KTNG', '9819794812' ),
6 ('Dikshya', 'Khanal', '2003-12-26', 'O', 'd@gmail.com', 'BRT', '9746994365' );
7
8 • INSERT INTO course_registration (c_code, c_name, c_credit, department, c_fee)
9 VALUES
10 (50, 'Science', 4, 'SD', 12300),
11 (51, 'Math', 5, 'MD', 13400),
12 (52, 'Computer', 4, 'CD', 12300),
13 (53, 'Nepali', 7, 'ND', 12600);
14
15 • INSERT INTO faculty (faculty_id , first_name , last_name , department , email)
16 VALUES
17 (100, 'Manisha', 'Khatrri', 'MD', 'm@gmail.com'),
18 (101, 'Abin', 'Rai', 'CD', 'a@gmail.com'),
19 (102, 'Mandip', 'Rai', 'ND', 'mr@gmail.com'),
20 (103, 'Dikshya', 'Khanal', 'SD', 'd@gmail.com');
21
22 • INSERT INTO teacher (teacher_id, c_code, tf_name, tl_name, teacher_email, t_contact)
23 VALUES
24 (300, 51, 'Prabhat', 'Shrestha', 'ps@gmail.com', '9837463728'),
25 (301, 52, 'Meghraj', 'Adhikari', 'mi@gmail.com', '9283746501'),
26 (302, 53, 'Roshan', 'Khatrri', 'rk@gmail.com', '9182635486'),
27 (303, 50, 'Ram', 'Rai', 'rr@gmail.com', '9785634251');
28
```

```
29 • INSERT INTO result (symbol_no, s_id , first_name, last_name, grade, gpa)
30 VALUES
31 (1230, 1, 'Manisha', 'Khatrri', 8, 3.44),
32 (1231, 2, 'Abin', 'Rai', 9, 2.19),
33 (1232, 3, 'Mandip', 'Rai', 10, 3.67),
34 (1233, 4, 'Dikshya', 'Khanal', 8, 1.23);
35
```

```
36 • INSERT INTO attendance (roll_no, first_name, last_name, s_id, to_date, to_status)
37 VALUES
38 (25, 'Abin', 'Rai', 2, '2023-08-21', 'P'),
39 (26, 'Dikshya', 'Khanal', 4, '2023-08-21', 'A'),
40 (27, 'Mandip', 'Rai', 3, '2023-08-21', 'P'),
41 (28, 'Manisha', 'Khatrri', 1, '2023-08-21', 'P');
42
```

```
43 • INSERT INTO enrollment (s_id, enrollment_date, first_name, last_name, c_code)
44 VALUES
45 (2, '2023-02-12', 'Abin', 'Rai', 52),
46 (4, '2023-03-02', 'Dikshya', 'Khanal', 50),
47 (3, '2023-02-20', 'Mandip', 'Rai', 53),
48 (1, '2023-01-25', 'Manisha', 'Khatrri', 51);
```

Output Table

student

Result Grid

Filter Rows:



Edit:


Export/Import:

	s_id	first_name	last_name	dob	gender	email	address	contact
▶	1	Manisha	Khatri	2000-01-21	F	m@gmail.com	KTM	9842849306
	2	Abin	Rai	2001-02-18	M	a@gmail.com	BKT	9746994366
	3	Mandip	Rai	2003-05-06	M	mr@gmail.com	KTNG	9819794812
	4	Dikshya	Khanal	2003-12-26	O	d@gmail.com	BRT	9746994365
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

course_registration

Result Grid










Filter Rows:

Edit: 

	c_code	c_name	c_credit	department	c_fee
▶	50	Science	4	SD	12300
	51	Math	5	MD	13400
	52	Computer	4	CD	12300
	53	Nepali	7	ND	12600
●	NULL	NULL	NULL	NULL	NULL

faculty

faculty

Result Grid		 Filter Rows: <input type="text"/>	Edit:   		
	faculty_id	first_name	last_name	department	email
	100	Manisha	Khatri	MD	m@gmail.com
	101	Abin	Rai	CD	a@gmail.com
	102	Mandip	Rai	ND	mr@gmail.com
	103	Dikshya	Khanal	SD	d@gmail.com
	NULL	NULL	NULL	NULL	NULL

teacher

Result Grid						
		Filter Rows:		Edit:		
	teacher_id	c_code	tf_name	tl_name	teacher_email	t_contact
▶	300	51	Prabhat	Shrestha	ps@gmail.com	9837463728
	301	52	Meghraj	Adhikari	mi@gmail.com	9283746501
	302	53	Roshan	Khatri	rk@gmail.com	9182635486
	303	50	Ram	Rai	rr@gmail.com	9785634251
✱	NULL	NULL	NULL	NULL	NULL	NULL

result

Result Grid						
		Filter Rows:		Edit:		
	symbol_no	s_id	first_name	last_name	grade	gpa
▶	1230	1	Manisha	Khatri	8	3
	1231	2	Abin	Rai	9	2
	1232	3	Mandip	Rai	10	4
	1233	4	Dikshya	Khanal	8	1
✱	NULL	NULL	NULL	NULL	NULL	NULL

attendance

Result Grid						
		Filter Rows:		Edit:		
	roll_no	first_name	last_name	s_id	to_date	to_status
▶	25	Abin	Rai	2	2023-08-21	P
	26	Dikshya	Khanal	4	2023-08-21	A
	27	Mandip	Rai	3	2023-08-21	P
	28	Manisha	Khatri	1	2023-08-21	P
✱	NULL	NULL	NULL	NULL	NULL	NULL

enrollment

Result Grid					
		Filter Rows:		Edit:	
	s_id	enrollment_date	first_name	last_name	c_code
▶	1	2023-01-25	Manisha	Khatri	51
	2	2023-02-12	Abin	Rai	52
	3	2023-02-20	Mandip	Rai	53
	4	2023-03-02	Dikshya	Khanal	50
★	NULL	NULL	NULL	NULL	NULL

Assignmentwise

Conclusion

Our student management system offers a centralized and organized platform for schools to handle student data, communication, and scheduling. It provides easy access and management of student information, allowing teachers to track progress, administrators to generate reports, and parents to monitor their children's development. Storing data in the cloud ensures it's up-to-date and protected.

However, it's important to address some limitations. The system's availability and usability may be affected by internet connection and device compatibility. Natural language processing and machine learning can introduce errors due to the complexity of language. Ethical and privacy concerns should be considered when collecting and using student data. Therefore, we need to improve our system's offline functionality, device compatibility, natural language processing, machine learning, ethical and legal compliance, and data security.

To further enhance our student management system, we can focus on optimizing the system's performance and accuracy by improving the code, updating the data, and conducting thorough testing. Additionally, we should prioritize maintaining ethical and legal compliance by adhering to relevant laws and regulations, obtaining user consent, and conducting ethical reviews. Integrating the system with online learning materials and data analytics tools would be beneficial, as it would connect the system with learning management systems, online courses, and data analysis tools to improve the learning experience for students and teachers. Lastly, gathering user feedback through surveys, interviews, and focus groups and implementing their suggestions and preferences would ensure that the system meets the changing needs of users, increasing their satisfaction and engagement.

In the end, the student management system serves as a practical and user-friendly solution for addressing educational challenges. Its benefits outweigh any drawbacks it may have. By promoting effective communication, data-driven decision-making, and enhanced learning outcomes, it has the potential to propel education forward. To unlock its full potential and ensure a comprehensive and enriching experience for students, faculty, and administrators, it is crucial to implement the system carefully and continuously improve it.

**EC3119 DATABASE SYSTEM
ASSIGNMENT II MARKING SCHEME**

PART A – DATABASE LOGICAL DESIGN		
NO	TASK	MARKS
1.	Background Background should include the following: <ul style="list-style-type: none"> • Introduction to current system • Problem with the existing system • How the problem could be solve using the proposed system 	[/5]
2.	Objective The objective of current system SHOULD NOT BE MORE THAN FIVE AND LESS THAN THREE and it should be listed in bullet form.	[/5]
3.	Problem Statement -State what the problems with current process / system are	[/5]
4.	Scope -Define scope for your proposed system	[/5]
5.	Hardware & Software specification -Identify the hardware and software requirements to develop the system	[/5]
6.	Entity Relationship Diagram -Draw entity relationship diagram with its attributes	[/15]
7.	Database Schema -Provide the data schema for the proposed system. Data schema should be documented using the entity relationship diagram. - Identify the primary and foreign key	[/10]
8.	Normalization - Normalize the table into 3NF	[/10]
PART A TOTAL		[/50]

Part B	System Development	Mark
1.	Achieve the scope mentioned in Part A	[/3]
2.	Able to run with free of errors	[/3]
3.	Has data validation	[/3]
4.	Enable Entity integrity and referential integrity	[/3]
5.	Overall design of the system / creativity	[/3]
PART B TOTAL		[/15]

Part C	Documentation	Mark
1.	All components listed in Part A	[/5]
2.	Print screen of result	[/5]
3.	Conclusions -Must consist of brief introduction of your system -Advantage of your developed system -Limitations of your system -Future work / Suggestion	[/5]
PART C TOTAL		[/15]
Part D	Description (Presentation)	Mark
1.	The presentation contained a brief and effective introduction	[/2]
2.	The explanation of the System design was clear and logical	[/2]
3.	Additional recommendations for expansion plan were helpful	[/1]
4.	The conclusions was satisfying and gave a sense of closure	[/1]
5.	Demonstration of the system	[/2]
6.	Able to answer question	[/1]
7.	Attire and cooperation of the group	[/1]
PART D TOTAL		[/10]
TOTAL A+B+C+D		[/100]