A Project report on “SCHOOL MANAGEMENT SYSTEM”

**---------------------------------**

Submitted in partial fulfillment of the requirements

For the award of degree

Of

**Bachelor in Computer Application (BCA)**

Submitted by

Alakh Namdev, Vasu Panwar, Sourav

22TSBCDDD01004, 22TSBCDDN01015, 22TSBCDDD01005



Under the guidance of

**Dr.Bineet Kumar Joshi**

(ICFAI Tech School)



ICFAI TECH SCHOOL

The ICFAI University, Dehradun

Dec 2024



**ICFAI TECH SCHOOL**

**THE ICFAI UNIVERSITY, DEHRADUN**

**Certificate**

This is to certify that the work in the project report entitled **“**SCHOOL MANAGEMENT SYSTEM**”** by (Alakh Namdev, Vasu Panwar, Sourav) bearing enrollment Number **(**22TSBCDDD01004, 22TSBCDDN01015, 22TSBCDDD01005**)** is a Bonafide record of project work carried out by him/her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor in Computer Application in the department of ICFAI Tech School, The ICFAI University, Dehradun. Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Signature of project guide

Date:

Guide Name

(Designation)



**ICFAI TECH SCHOOL**

**THE ICFAI UNIVERSITY, DEHRADUN**

**ACKNOWLEDGMENT**

Name of Student

(Enrollment No.)

**ABSTRACT**

CONTENTS

**Project Report Outline**

1. **Introduction (1-2 pages)**:
   * Purpose of the project.
   * Importance of managing subjects, students, coordinators, attendance, marks, and notices.
   * Overview of the database system used to manage this information.
2. **Database Schema Explanation (3-4 pages)**:
   * Detailed explanation of the database structure, including tables, columns, and relationships.
   * SQL code snippets for table creation (CREATE TABLE).
   * ER diagram showing the relationships between tables (if applicable).
3. **Use Case Scenarios (4-5 pages)**:
   * Examples of how the system would be used in real-world scenarios.
   * Use cases like adding new students, updating attendance, assigning marks, and posting notices.
   * SQL queries used in these processes.
4. **Data Analysis (5-6 pages)**:
   * Analysis of sample data, including attendance tracking, mark distribution, and subject assignment.
   * SQL queries for extracting useful insights, such as attendance percentage, average marks, and subject enrollment.
   * Possible visualizations of data trends (e.g., attendance trends or performance metrics).
5. **Conclusion (2 pages)**:
   * Summary of findings from the analysis.
   * Recommendations for improvements or optimizations in the system.
   * Future enhancements that could be made.
6. **SQL Queries & Sample Data Tables (3-4 pages)**:
   * SQL queries used for database operations.
   * Sample data that demonstrate the system’s functionality.
   * Example outputs for queries like generating reports for attendance, marks, and notices.

**1. Introduction**

**Purpose of the Project**

The purpose of this project is to design and implement a database management system that effectively handles the various aspects of academic administration, including the management of students, subjects, coordinators, attendance, marks, and notices. The system aims to provide a structured and efficient way to store, retrieve, and manage educational data, allowing for improved decision-making, better student performance tracking, and streamlined administrative processes.

**Importance of Managing Subjects, Students, Coordinators, Attendance, Marks, and Notices**

In an academic environment, managing data related to subjects, students, attendance, marks, and notices is a complex and crucial task. A well-organized database system ensures:

* **Accurate Data**: Proper tracking of subjects and their schedules, ensuring students are enrolled correctly.
* **Efficient Administration**: Coordinators and administrators can quickly update attendance records, input marks, and post notices.
* **Performance Monitoring**: Teachers can monitor student progress through marks and attendance, identifying areas where students might need additional support.
* **Transparency and Communication**: Students can access up-to-date notices and marks, improving communication between students and coordinators.

**Overview of the Database System**

The database management system (DBMS) implemented for this project is designed to handle these various requirements efficiently. Using relational database design principles, the database ensures consistency, integrity, and ease of querying data. The main components of the system are:

* **Subjects**: Information about the subjects taught across various classes.
* **Students**: Personal and academic details of students enrolled in the system.
* **Coordinators**: Faculty or staff members responsible for coordinating different classes and subjects.
* **Attendance**: Tracking attendance for students in each class session.
* **Marks**: Recording marks or grades for students in various subjects.
* **Notices**: Managing notices that are shared with students, such as announcements or assignments.

**2. Database Schema Explanation**

The database schema is designed to represent all entities involved in the academic management system and their relationships. Below is an explanation of the main tables involved:

**Tables Overview**

1. **class\_and\_subjects**
   * This table stores information about different classes and the subjects they offer.
   * **Columns**:
     + id: Primary key, uniquely identifies each record.
     + class: Class name (e.g., "Class 1", "Class 2").
     + subject 1 to subject 6: The subjects offered in each class.

**SQL Code**:

sql

Copy code

CREATE TABLE `class\_and\_subjects` (

`id` INT(11) NOT NULL,

`class` VARCHAR(12) NOT NULL,

`subject\_1` VARCHAR(57) DEFAULT NULL,

`subject\_2` VARCHAR(57) DEFAULT NULL,

`subject\_3` VARCHAR(57) DEFAULT NULL,

`subject\_4` VARCHAR(57) DEFAULT NULL,

`subject\_5` VARCHAR(57) DEFAULT NULL,

`subject\_6` VARCHAR(57) DEFAULT NULL,

PRIMARY KEY (`id`)

);

1. **students**
   * This table stores student details including their personal information and enrolled classes.
   * **Columns**:
     + id: Primary key, uniquely identifies each student.
     + student\_name: Full name of the student.
     + dob: Date of birth.
     + class\_id: Foreign key referencing class\_and\_subjects to associate the student with a specific class.

**SQL Code**:

sql

Copy code

CREATE TABLE `students` (

`id` INT(11) NOT NULL,

`student\_name` VARCHAR(100) NOT NULL,

`dob` DATE NOT NULL,

`class\_id` INT(11),

PRIMARY KEY (`id`),

FOREIGN KEY (`class\_id`) REFERENCES `class\_and\_subjects`(`id`)

);

1. **attendance**
   * This table records the attendance of students for each class session.
   * **Columns**:
     + attendance\_id: Primary key, uniquely identifies each attendance record.
     + student\_id: Foreign key referencing students to identify the student.
     + date: Date of the class session.
     + status: Attendance status (e.g., "Present", "Absent").

**SQL Code**:

sql

Copy code

CREATE TABLE `attendance` (

`attendance\_id` INT(11) NOT NULL,

`student\_id` INT(11),

`date` DATE NOT NULL,

`status` VARCHAR(10) NOT NULL,

PRIMARY KEY (`attendance\_id`),

FOREIGN KEY (`student\_id`) REFERENCES `students`(`id`)

);

1. **marks**
   * This table records the marks of students for each subject.
   * **Columns**:
     + mark\_id: Primary key, uniquely identifies each marks record.
     + student\_id: Foreign key referencing students.
     + subject: Subject name (e.g., "Math", "Science").
     + marks: The marks obtained by the student.

**SQL Code**:

sql

Copy code

CREATE TABLE `marks` (

`mark\_id` INT(11) NOT NULL,

`student\_id` INT(11),

`subject` VARCHAR(50) NOT NULL,

`marks` INT(5) NOT NULL,

PRIMARY KEY (`mark\_id`),

FOREIGN KEY (`student\_id`) REFERENCES `students`(`id`)

);

1. **notices**
   * This table stores notices that can be shared with students.
   * **Columns**:
     + notice\_id: Primary key, uniquely identifies each notice.
     + notice\_content: Content of the notice.
     + post\_date: Date when the notice was posted.

**SQL Code**:

sql

Copy code

CREATE TABLE `notices` (

`notice\_id` INT(11) NOT NULL,

`notice\_content` TEXT NOT NULL,

`post\_date` DATE NOT NULL,

PRIMARY KEY (`notice\_id`)

);

**ER Diagram (Sample)**

The following is a conceptual diagram of the relationships between the tables:

* **Students** are linked to **Class and Subjects** through class\_id.
* **Attendance** records are linked to **Students** via student\_id.
* **Marks** are linked to **Students** via student\_id and can be categorized by subject.
* **Notices** are independent but serve to communicate information to all students.

**3. Use Case Scenarios**

This section explores real-world examples of how the system manages data for subjects, students, coordinators, attendance, marks, and notices. Each scenario is accompanied by relevant SQL queries.

**Use Case 1: Adding a New Student**

A new student named "John Doe" is enrolling in "Class 10". His date of birth is "2007-05-15".

**SQL Query:**

sql

Copy code

INSERT INTO `students` (`id`, `student\_name`, `dob`, `class\_id`)

VALUES (101, 'John Doe', '2007-05-15', 3);

*Explanation:*

* The id is 101 (a unique identifier).
* The class\_id is 3, assuming it refers to "Class 10" in the class\_and\_subjects table.

**Use Case 2: Recording Attendance**

Mark "John Doe" as "Present" for the date "2024-11-29".

**SQL Query:**

sql

Copy code

INSERT INTO `attendance` (`attendance\_id`, `student\_id`, `date`, `status`)

VALUES (201, 101, '2024-11-29', 'Present');

*Explanation:*

* student\_id is 101 (linked to "John Doe").
* status is set to "Present".

**Use Case 3: Assigning Marks**

Record "John Doe's" score in "Mathematics" as 85.

**SQL Query:**

sql

Copy code

INSERT INTO `marks` (`mark\_id`, `student\_id`, `subject`, `marks`)

VALUES (301, 101, 'Mathematics', 85);

*Explanation:*

* This query links the mark to "John Doe" and the subject "Mathematics".

**Use Case 4: Posting a Notice**

Post a notice about an upcoming exam scheduled for "2024-12-10".

**SQL Query:**

sql

Copy code

INSERT INTO `notices` (`notice\_id`, `notice\_content`, `post\_date`)

VALUES (401, 'Final examination scheduled for 10th December 2024.', '2024-11-29');

*Explanation:*

* The notice informs students about an important exam date.

**Use Case 5: Viewing Attendance Records**

Retrieve attendance records for "John Doe".

**SQL Query:**

sql

Copy code

SELECT `date`, `status`

FROM `attendance`

WHERE `student\_id` = 101;

*Explanation:*

* This query fetches all attendance records for the student with id 101.

**Use Case 6: Calculating Average Marks**

Calculate the average marks for "John Doe".

**SQL Query:**

sql

Copy code

SELECT AVG(`marks`) AS average\_marks

FROM `marks`

WHERE `student\_id` = 101;

*Explanation:*

* The query computes the average of all marks recorded for "John Doe".

**Use Case 7: Displaying Notices**

List all notices posted in the last month.

**SQL Query:**

sql

Copy code

SELECT `notice\_content`, `post\_date`

FROM `notices`

WHERE `post\_date` >= DATE\_SUB(CURDATE(), INTERVAL 1 MONTH);

*Explanation:*

* This query retrieves notices posted within the last 30 days.

**4. Data Analysis**

This section focuses on analyzing key data elements such as attendance, marks, and notices to extract meaningful insights. Each analysis scenario includes SQL queries and explanations.

**Analysis 1: Attendance Summary for Each Student**

Calculate the total number of days "Present" and "Absent" for each student.

**SQL Query:**

sql

Copy code

SELECT

s.student\_name,

SUM(CASE WHEN a.status = 'Present' THEN 1 ELSE 0 END) AS total\_present,

SUM(CASE WHEN a.status = 'Absent' THEN 1 ELSE 0 END) AS total\_absent

FROM

students s

LEFT JOIN

attendance a ON s.id = a.student\_id

GROUP BY

s.student\_name;

*Explanation:*

* This query sums the number of days each student was "Present" and "Absent."
* It uses conditional aggregation (CASE) for counting attendance statuses.

**Analysis 2: Average Marks Per Subject**

Determine the average marks scored by students in each subject.

**SQL Query:**

sql

Copy code

SELECT

subject,

AVG(marks) AS average\_marks

FROM

marks

GROUP BY

subject;

*Explanation:*

* The query groups marks by subject and calculates the average using the AVG function.

**Analysis 3: Top Performers in Each Subject**

Retrieve the highest marks scored in each subject along with the student's name.

**SQL Query:**

sql

Copy code

SELECT

m.subject,

s.student\_name,

MAX(m.marks) AS highest\_marks

FROM

marks m

JOIN

students s ON m.student\_id = s.id

GROUP BY

m.subject, s.student\_name

ORDER BY

m.subject;

*Explanation:*

* The MAX function retrieves the highest marks per subject.
* JOIN ensures student names are displayed alongside marks.

**Analysis 4: Attendance Percentage of Each Student**

Calculate the attendance percentage for each student.

**SQL Query:**

sql

Copy code

SELECT

s.student\_name,

(SUM(CASE WHEN a.status = 'Present' THEN 1 ELSE 0 END) \* 100 / COUNT(a.status)) AS attendance\_percentage

FROM

students s

LEFT JOIN

attendance a ON s.id = a.student\_id

GROUP BY

s.student\_name;

*Explanation:*

* This query calculates the percentage of days attended by dividing the "Present" count by the total attendance records.

**Analysis 5: Recent Notices**

List the most recent notices posted, limited to the last five.

**SQL Query:**

sql

Copy code

SELECT

notice\_content, post\_date

FROM

notices

ORDER BY

post\_date DESC

LIMIT 5;

*Explanation:*

* The query fetches the latest five notices, ordered by the posting date in descending order.

**Visual Representation of Data (Optional)**

1. **Attendance Trends**: A line graph showing the attendance trends over time for each student.
2. **Marks Distribution**: A bar chart illustrating the distribution of marks for different subjects.
3. **Notice Frequency**: A chart displaying the frequency of notices posted over the academic term.

**Conclusion**

This project demonstrates the implementation of a robust database management system designed to streamline academic processes such as managing subjects, students, coordinators, attendance, marks, and notices. Through the use of a well-structured relational database, key academic data is organized efficiently, ensuring ease of access, data integrity, and meaningful analysis.

Key takeaways from the project include:

1. **Efficient Data Management**: The system efficiently handles large volumes of data related to students, subjects, and academic performance.
2. **Enhanced Monitoring**: Attendance tracking and marks management allow for continuous monitoring of student progress, enabling timely interventions when necessary.
3. **Improved Communication**: The notices module ensures effective dissemination of important information to students.
4. **Data Analysis Capabilities**: SQL queries provide valuable insights into attendance trends, academic performance, and communication patterns.

The system's design highlights the importance of data-driven decision-making in an academic environment, promoting a culture of continuous improvement.

**Recommendations**

Based on the analysis conducted, the following recommendations are suggested for further enhancing the system:

1. **Automated Alerts**:
   * Implement automated alerts for students falling below a certain attendance percentage or performance threshold.
2. **User Interface Development**:
   * Develop a user-friendly front-end interface for students, teachers, and administrators to interact with the database without needing SQL knowledge.
3. **Data Security and Backup**:
   * Implement robust data security measures such as user authentication, encryption, and regular database backups to ensure data integrity and confidentiality.
4. **Integration with Learning Management Systems (LMS)**:
   * Integrate the system with existing LMS platforms for seamless data sharing and enhanced academic management.
5. **Advanced Analytics**:
   * Incorporate machine learning algorithms to predict student performance trends and attendance patterns, providing proactive support.
6. **Mobile Access**:
   * Develop a mobile application that provides real-time access to attendance records, marks, and notices, ensuring students stay informed on the go.

**6. SQL Queries & Sample Data Tables**

This section includes sample SQL queries used in the system as well as examples of the data tables generated. The queries demonstrate how the database handles common operations such as inserting data, updating records, and retrieving key information. Sample data tables showcase how the system stores actual records.

**Sample SQL Queries**

1. **Creating the students Table** This query creates the table that stores student information.

sql

Copy code

CREATE TABLE `students` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`student\_name` VARCHAR(100) NOT NULL,

`dob` DATE NOT NULL,

`class\_id` INT(11),

PRIMARY KEY (`id`),

FOREIGN KEY (`class\_id`) REFERENCES `class\_and\_subjects`(`id`)

);

1. **Inserting Data into students Table** Insert a new student, "John Doe", into the students table.

sql

Copy code

INSERT INTO `students` (`student\_name`, `dob`, `class\_id`)

VALUES ('John Doe', '2007-05-15', 3);

1. **Fetching All Students with Their Class Information** Retrieve the list of all students along with their class and subjects.

sql

Copy code

SELECT

s.student\_name,

c.class,

c.subject\_1, c.subject\_2, c.subject\_3, c.subject\_4, c.subject\_5, c.subject\_6

FROM

students s

JOIN

class\_and\_subjects c ON s.class\_id = c.id;

1. **Updating a Student's Class** Update the class of a student (e.g., move "John Doe" to class "12").

sql

Copy code

UPDATE `students`

SET `class\_id` = 4

WHERE `student\_name` = 'John Doe';

1. **Deleting a Student Record** Delete a student record (e.g., remove "John Doe" from the database).

sql

Copy code

DELETE FROM `students`

WHERE `student\_name` = 'John Doe';

1. **Fetching Attendance Records for a Specific Student** Retrieve attendance records for "John Doe" to check if the student was present or absent.

sql

Copy code

SELECT `date`, `status`

FROM `attendance`

WHERE `student\_id` = 101;

1. **Getting Marks for a Specific Subject** Retrieve all marks for the subject "Mathematics".

sql

Copy code

SELECT `student\_id`, `marks`

FROM `marks`

WHERE `subject` = 'Mathematics';

1. **Posting a Notice** Insert a new notice about an upcoming exam.

sql

Copy code

INSERT INTO `notices` (`notice\_content`, `post\_date`)

VALUES ('The final exam is scheduled for 10th December 2024.', '2024-11-29');

**Sample Data Tables**

1. **students Table**

| **id** | **student\_name** | **dob** | **class\_id** |
| --- | --- | --- | --- |
| 101 | John Doe | 2007-05-15 | 3 |
| 102 | Jane Smith | 2006-09-20 | 3 |
| 103 | Bob Johnson | 2005-12-25 | 4 |

1. **class\_and\_subjects Table**

| **id** | **class** | **subject\_1** | **subject\_2** | **subject\_3** | **subject\_4** | **subject\_5** | **subject\_6** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 10 | Math | Science | English | History | Geography | Physical Ed |
| 4 | 12 | Math | Biology | Chemistry | Physics | English | Computer Sci |

1. **attendance Table**

| **attendance\_id** | **student\_id** | **date** | **status** |
| --- | --- | --- | --- |
| 201 | 101 | 2024-11-29 | Present |
| 202 | 102 | 2024-11-29 | Absent |
| 203 | 103 | 2024-11-29 | Present |

1. **marks Table**

| **mark\_id** | **student\_id** | **subject** | **marks** |
| --- | --- | --- | --- |
| 301 | 101 | Mathematics | 85 |
| 302 | 102 | Science | 90 |
| 303 | 103 | Chemistry | 78 |

1. **notices Table**

| **notice\_id** | **notice\_content** | **post\_date** |
| --- | --- | --- |
| 401 | Final exam scheduled for 10th December 2024. | 2024-11-29 |
| 402 | Mid-term exams on 15th November 2024. | 2024-11-10 |

**Final Notes**

This project highlights the practical application of a relational database for managing key academic processes. By storing and querying data efficiently, administrators, coordinators, and educators can access real-time information that aids in decision-making, communication, and performance monitoring.

In future versions, the system can be enhanced by adding user interfaces, automated reporting, and integration with other academic systems.