INDIAN INSTITUTE OF INFORMATION TECHNOLOGY VADODARA

CS 262 DBMS PROJECT

SCHOOL MANAGEMENT SYSTEM

Group Members:-
Rohit Kumar Agrahari(201951129)
Abhijeet Raj(201952201)
Anuj Aglawe(201952203)
Anand Kumar(201952204)
Arjun Pandey(201952206)

Functions Added in the project:-

This all comes under the Admin Section:-

1.Student Registration Form:-By using this form we will take all the required data for student registration using the graphical user interface and insert all the data into a “student” table in the database.

There are two buttons:-

i).Fetch:-This will assign a student Id to the new student who got admitted.

ii).Submit:-This will insert all the values into the student table.

2.Teacher Recruitment Form:-By using this form we will take all the required data for teacher recruitment using the graphical user interface and insert all the data into a “teacher” table in the database.

There are two buttons:-

i).Fetch:-This will assign a Teacher Id to the new teacher who got recruited.

ii).Submit:-This will insert all the values into the teacher table.

3.Student Transfer/Teacher Transfer :-By using this form we will delete the data of the student/teacher from the respective table in the database because they are leaving the school.

Their is one Jcombox in which we will select the entity whose transfer needs to be done i.e. teacher or student

There are two buttons:-

i).Fetch:-This will show the data of the Student/Teacher from the Student/teacher table which need to be deleted.

ii).Transfert:-This button will delete the data from the respective student/teacher table.

4.Accounts Section:- This Section mainly deals with all the money transactions work in the school.
Contains:-
i)Set Fees:-This will be used to set the salary of different classes.
ii)Set Salary:-This will set the salary of the teacher into the salary Table.
iii)Student Fees:-This will be used to pay the fees and simultaneously it will update the amount table and fees table also and it stores the status of the fees paid by the students.
iv)Teacher Salary:-This will be used to pay the salary of the teachers in the school and maintain the status of the salary paid or not in salary table.

5.Course Information Work:-By using this form we will input the courses taught in the school and we will insert the values into the course table in tha database.

This all comes under the Teacher Section:-

1. Student Marks: Here teachers can enter marks of students. Using Student ID

2. Attendance: Teacher can update the attendance using Student Id

3. Courses Enrolled: Teacher can fetch the courses enrolled by student Using Student ID

4. Show Results: Results can be displayed using Student ID

Database Source Code:-

STUDENT TABLE:-

```
  CREATE TABLE `student` (
  `Student_ID` varchar(45) NOT NULL,
  `First_Name` varchar(45) NOT NULL,
  `Middle_Name` varchar(45) NOT NULL,
  `Last_Name` varchar(45) NOT NULL,
  `Father_Name` varchar(45) NOT NULL,
  `Mother_Name` varchar(45) NOT NULL,
  `Contact_No.` varchar(10) NOT NULL,
  `class` int NOT NULL,
  `D.O.B` varchar(45) NOT NULL,
  `Address` varchar(100) NOT NULL,
  PRIMARY KEY (`Student_ID`)
)
```

TEACHER TABLE:-

```
CREATE TABLE `teacher` (
  `Teacher_ID` varchar(45) NOT NULL,
  `First_Name` varchar(45) NOT NULL,
  `Middle_Name` varchar(45) DEFAULT NULL,
  `Last_Name` varchar(45) NOT NULL,
  `D.O.B` varchar(45) NOT NULL,
  `Father_Name` varchar(45) NOT NULL,
  `Mother_Name` varchar(45) NOT NULL,
  `Contact_No.` varchar(45) NOT NULL,
  `Address` varchar(45) NOT NULL,
  PRIMARY KEY (`Teacher_ID`)
)
```

COURSE TABLE:-

```
CREATE TABLE `course` (
  `Course_ID` varchar(45) NOT NULL,
  `Course_Name` varchar(45) NOT NULL,
  `Class` int NOT NULL,
  `Total_Credit` int NOT NULL,
  `Teacher_ID` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`Course_ID`),
  KEY `Teacher_ID_idx` (`Teacher_ID`),
  CONSTRAINT `Teacher_ID` FOREIGN KEY (`Teacher_ID`)
REFERENCES `teacher` (`Teacher_ID`) ON DELETE SET
NULL ON UPDATE CASCADE
)
```

FEES TABLE:-

```
CREATE TABLE `fees` (
  `St_ID` varchar(45) NOT NULL,
  `Jan-March` varchar(45) DEFAULT 'NOT PAID',
  `April-Jun` varchar(45) DEFAULT 'NOT PAID',
  `Jul-Sep` varchar(45) DEFAULT 'NOT PAID',
  `Oct-Dec` varchar(45) DEFAULT 'NOT PAID',
  PRIMARY KEY (`St_ID`),
  CONSTRAINT `St_ID` FOREIGN KEY (`St_ID`)
REFERENCES `student` (`Student_ID`) ON DELETE
CASCADE
)
```

ATTENDANCE TABLE:-

```
CREATE TABLE `attendance` (
  `S_ID` varchar(45) DEFAULT NULL,
  `Date` varchar(45) DEFAULT NULL,
  `Present/Absent` varchar(45) DEFAULT NULL,
  KEY `S_ID` (`S_ID`),
  CONSTRAINT `S_ID` FOREIGN KEY (`S_ID`)
REFERENCES `student` (`Student_ID`) ON DELETE
CASCADE ON UPDATE CASCADE
)
```

MARKS TABLE:-

```
CREATE TABLE `marks` (
  `Student_ID` varchar(45) NOT NULL,
  `Course_ID` varchar(45) NOT NULL,
  `Obtained_Marks` int DEFAULT '0',
  `Status` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`Student_ID`,`Course_ID`),
  KEY `Student_ID_idx` (`Student_ID`),
  KEY `Course_ID_idx` (`Course_ID`),
  CONSTRAINT `Student_ID` FOREIGN KEY (`Student_ID`)
REFERENCES `student` (`Student_ID`) ON DELETE
CASCADE ON UPDATE CASCADE
)
```

SALARY TABLE:-

```
CREATE TABLE `salary` (
  `T_ID` varchar(45) NOT NULL,
  `January` varchar(45) DEFAULT 'NOT PAID',
  `February` varchar(45) DEFAULT 'NOT PAID',
  `March` varchar(45) DEFAULT 'NOT PAID',
  `April` varchar(45) DEFAULT 'NOT PAID',
  `May` varchar(45) DEFAULT 'NOT PAID',
  `June` varchar(45) DEFAULT 'NOT PAID',
  `July` varchar(45) DEFAULT 'NOT PAID',
  `August` varchar(45) DEFAULT 'NOT PAID',
  `September` varchar(45) DEFAULT 'NOT PAID',
  `October` varchar(45) DEFAULT 'NOT PAID',
  `November` varchar(45) DEFAULT 'NOT PAID',
  `December` varchar(45) DEFAULT 'NOT PAID',
  `SalaryAmount` int DEFAULT '0',
  PRIMARY KEY (`T_ID`),
  CONSTRAINT `T_ID` FOREIGN KEY (`T_ID`)
REFERENCES `teacher` (`Teacher_ID`) ON DELETE
CASCADE ON UPDATE CASCADE
)
```

FEES TABLE:-

```
CREATE TABLE `fees` (
  `St_ID` varchar(45) NOT NULL,
  `Jan-March` varchar(45) DEFAULT 'NOT PAID',
  `April-Jun` varchar(45) DEFAULT 'NOT PAID',
  `Jul-Sep` varchar(45) DEFAULT 'NOT PAID',
  `Oct-Dec` varchar(45) DEFAULT 'NOT PAID',
  PRIMARY KEY (`St_ID`),
  CONSTRAINT `St_ID` FOREIGN KEY (`St_ID`)
REFERENCES `student` (`Student_ID`) ON DELETE
CASCADE
)
```

FEES AMOUNT:-

```
CREATE TABLE `fees_amount` (
  `Class` int NOT NULL,
  `Amount` int DEFAULT NULL,
  PRIMARY KEY (`Class`)
)
```

ACCOUNT TABLE:-

```
CREATE TABLE `account` (
  `Amount` int DEFAULT '0',
  `activity` varchar(45) DEFAULT NULL
)
```

Queries Used In This Project:

TRANSFER

- **FOR STUDENT SECTION**

  On clicking Fetch button fetch details of student:

  SELECT * FROM sms_dbms.student where Student_ID = "S1";

  After that click on transfer for that query is :

  Delete from student where Student_ID = "S1";

- **FOR TEACHER SECTION**

  On clicking Fetch button fetch details of teacher:

  SELECT * FROM sms_dbms.teacher where Teacher_ID = "T1";

  After that click on transfer for that query is :

  Delete from teacher where Teacher_ID = "T1";

FEES SECTION

On clicking Fetch button fetch details of fees paid of a particular student:

select * from fees where St_ID = "S2";

On clicking Fetch button fetch fees amount of a particular student:

SELECT amount FROM fees_amount where class in (SELECT class from student where student_ID = "S3");

On clicking Pay Fees button insert amount of fees paid and activity= sf in account entity:

INSERT INTO account VALUES ("300" , "sf");

For updating status of student fees from Not Paid to Paid:

UPDATE FEES SET `Jan-March` = 'paid' WHERE St_id = "S3";

UPDATE FEES SET `April-Jun` = 'paid' WHERE St_id = "S3";

UPDATE FEES SET `Jul-Sep` = 'paid' WHERE St_id = "S3";

UPDATE FEES SET `Oct-Dec` = 'paid' WHERE St_id = "S3";

SALARY SECTION

For Insert salary amount or update salary amount in salary table:

update salary set SalaryAmount = 2000 where T_ID = "T2";

On clicking Fetch button fetch details of salary paid of a particular teacher:

select * from salary where T_ID = "T2";

On clicking Fetch button fetch salary amount of a particular teacher:

select SalaryAmount from salary where T_ID = "T2"; // Put this query in variable s

On clicking Pay Salary button insert amount of salary paid and activity= ts in account entity:

```
select sum(amount) from account where activity = "sf"; // Put this query in variable x

select sum(amount) from account where activity = "ts"; // Put this query in variable y

Then initialise z = x-y

if(z > s) then execute below query

INSERT INTO ACCOUNT VALUES ("500", "ts");
```

For updating status of salary from Not Paid to Paid:

```
UPDATE SALARY SET JANUARY = "PAID" where T_ID = "T2";

UPDATE SALARY SET FEBRUARY = "PAID" where T_ID = "T2";

UPDATE SALARY SET MARCH = "PAID" where T_ID = "T2";

UPDATE SALARY SET APRIL = "PAID" where T_ID = "T2";

UPDATE SALARY SET MAY = "PAID" where T_ID = "T2";

UPDATE SALARY SET JUNE = "PAID" where T_ID = "T2";

UPDATE SALARY SET JULY = "PAID" where T_ID = "T2";

UPDATE SALARY SET AUGUST = "PAID" where T_ID = "T2";

UPDATE SALARY SET SEPTEMBER = "PAID" where T_ID = "T2";

UPDATE SALARY SET OCTOBER = "PAID" where T_ID = "T2";

UPDATE SALARY SET NOVEMBER = "PAID" where T_ID = "T2";

UPDATE SALARY SET DECEMBER = "PAID" where T_ID = "T2";
```

For showing student marks:

```
SELECT * FROM marks where Student_ID = "S3";

SELECT SUM(Obtained_Marks) FROM marks where Student_ID = "S3";
```

For showing student attendance:

select distinct count(*) from attendance where S_ID = "S3" and `Present/Absent` = "Present";

For showing list of COURSE Enrolled of a particular student:

SELECT COURSE_NAME, COURSE_ID, TOTAL_CREDIT FROM COURSE WHERE CLASS IN (SELECT CLASS FROM STUDENT WHERE STUDENT_ID = "S3");

After registration of student, insert student ID into fees table:

INSERT INTO fees (S_ID) values ("S1");

After registration of teacher, insert teacher ID into salary table:

INSERT INTO salary (T_ID) values ("T1");