**Mini Project Report *on***

**“Student Management System”**

**Submitted by,**

Oishik Mandal PF 32 1032180809

Lakshita Agarwal PF 43 1032181232

Disha Bajaj PF 51 1032181257

**Under Guidance of**

**Prof. Shilpa Sonawani**

At



**School of Computer Engineering and Technology**

**Contents**

Abstract ………………………………………………………………………... 3

List of Figures …………………………………………………………………. 3

List of Tables ……………………………………………………………….…. 3

1. **Introduction …………………………………………………………………..**  4
   1. Motivation ……………………………………………………………... 4
   2. Objectives ………………………………………………………….….. 4
2. **Problem Statement ………………………………………………………… . 5**
   1. Scope ………………………………………………………………….. 5
   2. Planning ……………………………………………………………… . 5
3. **Tools & Technologies Used ………………………………………………… 6**
   1. Software Requirements ………………………………………………... 6
   2. Hardware Requirements ……………………………………………….. 6
4. **Database Design ……………………………………………………………….. 6**
   1. Functional Requirements ……………………………………………..… 6
   2. Non-Functional Requirements …………………………………………. 6
   3. ER Diagram ……………………………………………………………. 7
5. **Database Schema ……………………………………………………………… 8**
6. **DDL……………………………………………………………… 9**
7. **DML……………………………………………………………… 10**
8. **DCL……………………………………………………………… 11**
9. **Triggers……………………………………………………………… 12**
10. **PLSQL Procedure/function……………………………………………………………… 13**
11. **Frontend GUI………………………………………………...… 14**
12. **Relational Database Design ………………………………………………...… 15**
13. **Normalized Database Design ………………………………………………… 16**

Conclusion …………………………………………………………………..….. 17

**Abstract**

An organized and systematic office solution is essential for all universities and organizations. There are many departments of administration for the maintenance of college information and student databases in any institution. All these departments provide various records regarding students. Most of these track records need to maintain information about the students. This information could be the general details like student name, address, performance, attendance etc or specific information related to departments like collection of data.

Our work is useful for easy user interface. We are planning to utilize the powerful database management, data retrieval and data manipulation. Which will provide more ease for managing the data than manually maintaining in the documents. Our work is useful for saving valuable time and reduces the huge paperwork.

**List of figures**

* ER Diagram
* Schema Diagram

**List of tables**

* Fees
* Student
* Exam
* Subject
* Faculty
* Users
* Courses
* Department

**1. Introduction**

1. **Motivation**

Here we shared Student Management System Project Report. Students form a main part of any institution that concerns with. But the institutions find it difficult to keep details of so many students of the organization just in one stretch. It will involve a lot of pen paper work. Sometimes there will be some huge heap of files bundled up and kept together in some corner of the office. If you want any information regarding the particular student then it can be obtained by just entering the roll number or the name of the student to be searched. This student management system will make the work of storing the data in an organized way.

1. **Objectives**

Now these days computer play an important role in any organization because computer can easily manage the great number of data as compare to human. User can easily access data on right time in computer.

FG public school has been working for 17 years. This school has a great number of students. So, it is difficult for the management to get information about this institute on right time. When computer system play in this institute it will make easy to manage and access data of this institute.

The School management system (SMS) based upon the following objectives:

## Record Keeping- The first objective of this system is to maintain the data of all the student and teacher in database. It maintains the personal record as well as the academic record of the student. It maintains student fee record and dues record. Teacher personal record and salary record are also store in the software. SMS also manage class record, subject record, examination record.

1. Reports- Reports are the most important outputs of school management system (SMS). SMS create the following reports like – total no of student in a class, Student admission reports, Character certificate of the student, Submitted Fee report, Submitted due reports etc.

**2. Problem Statement**

1. **Scope -** SMS is intended to help the any institute that wants to store their students and management records into the computer. Our software is specially designed for an educational institute. Our SMS software will fulfil all the requirement of an educational institute. It will store all the personal and academic information of the students. It will also store the fee information of the students. Management will get information of any student who studied/studying in this institute till now easily. This software will also help the management to store their employee's information including their personal information and salary information. Payroll system is very important of any organization .so this software will also handle the pay roll system of this institute. If will also store the results of the students.
2. **Planning** - The school management system facilitates the user in following respects.

## Data Entry: Entering the correct and reliable data to the proposed system is the most important factor. The proposed system will facilitate the user with user-friendly screen, which will help the users to enter the correct data to proposed system easily. Different checks have been applied in the program for the validity of data so that wrong information cannot be entered into the system easily.

## Updating: Mistake in the entry process are possible to occur. These mistakes needed to be corrected at the right time so that the data remain correct. Any mistake during the entry process can be corrected through this option.

## Deletion: We can easily delete a particular record by requesting through entering the identity number or name of an item. The specified record is deleted from the database, which is not required anymore. This rarely occurs because most of the record are stored permanently, which may be referenced in later time.

**3. Tools & Technologies Used**

**Software Requirements**

* 1. HTML/CSS
  2. JavaScript
  3. Node-JS
  4. MySQL

**Hardware Requirement**

1. Processor: - Pentium
2. RAM: - 512MB
3. Memory: 1GB

**4. Database Design**

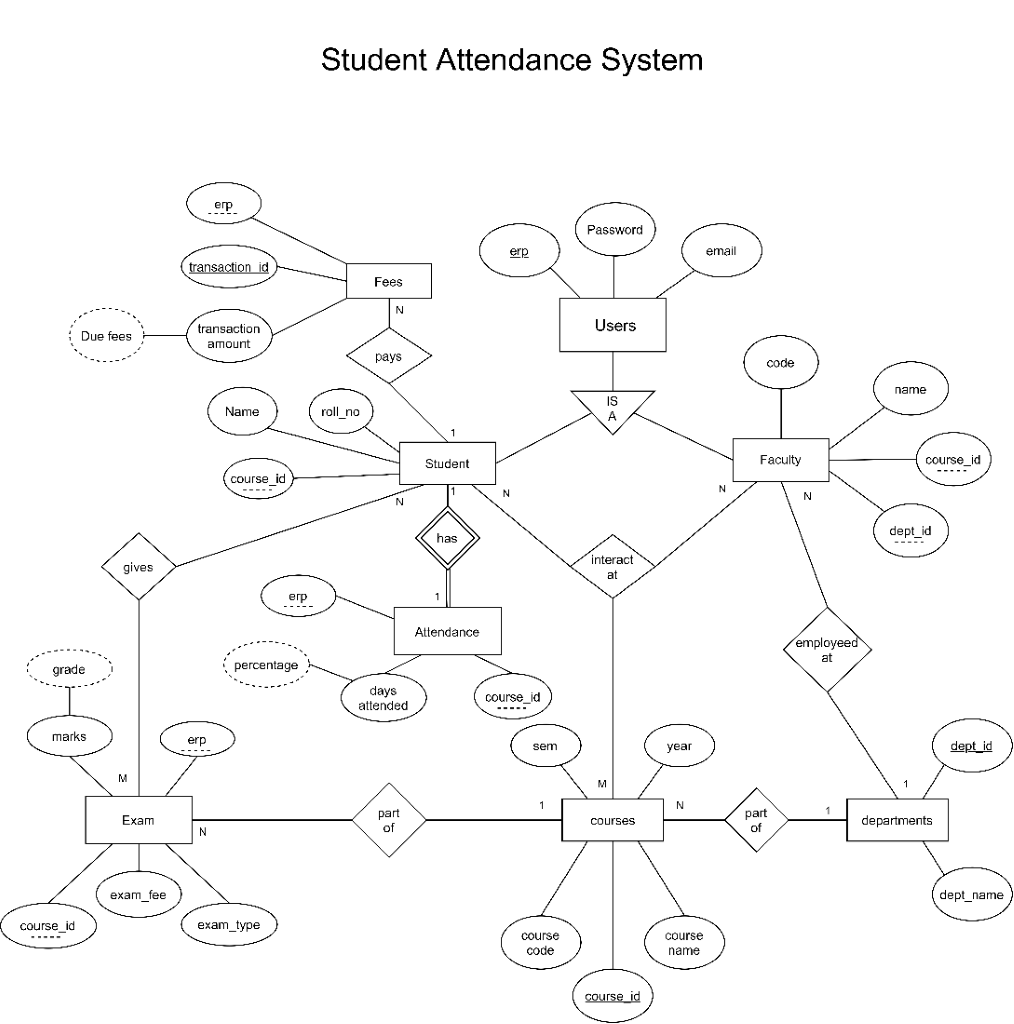
**Functional Requirements**

1. CREATION OF NEW RECORD: This function creates a record for a new student
2. DELETION OF RECORD: This function is used to delete the existing record of any student.
3. UPDATION IN RECORD: This function updates the information in a record of any student.
4. DISPLAY OF DATA IN RECORD: This function displays the record of the students.
5. SEARCHING A RECORD: This function searches for a student record

**Non- Functional Requirements**

1. Security: Only authorized users can access the system with username and password.
2. Performance: Easy tracking of records and updating can be done.
3. User Friendly: The System is very interactive.
4. Maintainability: Backups for database are available.

**ER Diagram**

Student Management System

**5. Database Schema**

**Schema Table**

Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| erp | password | Roll\_no | name | Course\_id |

Faculty

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| erp | password | code | Dept\_id | name | Course\_id |

Departments

|  |  |
| --- | --- |
| Dept\_id | dept\_name |

Courses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course\_id | Course\_code | Course\_name | sem | year |

Fees

|  |  |  |  |
| --- | --- | --- | --- |
| Transaction\_id | Fees\_due | Transaction\_amt | erp |

Exam

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| erp | marks | grade | Exam\_fees | Exam\_type | Course\_id |

Attendance

|  |  |  |  |
| --- | --- | --- | --- |
| erp | Days\_attended | percentage | Course\_id |

**6.DDL-(Data Definition Language)**

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

**1 .Create-**is used to create the database or its objects (like table, index, function, views, store procedure and triggers).

CREATE DATABASE db\_student;

USE db\_student;

CREATE TABLE IF NOT EXISTS courses (

course\_id INT primary key,

course\_name varchar(20) NOT NULL,

sem INT NOT NULL,

year int

);

CREATE TABLE IF NOT EXISTS users (

erp INT PRIMARY KEY,

password varchar(20) NOT NULL,

email\_id varchar(20) NOT NULL UNIQUE

);

CREATE TABLE IF NOT EXISTS department (

dept\_id INT primary key,

dept\_name varchar(20) NOT NULL

);

CREATE TABLE IF NOT EXISTS faculty (

erp INT PRIMARY KEY NOT NULL,

course\_id INT NOT NULL,

dept\_id INT NOT NULL,

name varchar(20) NOT NULL,

Foreign Key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key (dept\_id) references department(dept\_id) ON DELETE CASCADE ON UPDATE CASCADE

CREATE TABLE IF NOT EXISTS student (

erp INT PRIMARY KEY,

course\_id INT NOT NULL,

name varchar(20) NOT NULL,

roll\_no INT NOT NULL,

Foreign key(course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

-- Attendance for students only

create table IF NOT EXISTS attendance (

attendance\_id INT PRIMARY KEY,

erp INT,

course\_id INT NOT NULL,

days\_attended INT NOT NULL DEFAULT 0,

Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key (erp) references student(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS fees (

transaction\_id INT PRIMARY KEY,

erp INT NOT NULL,

transaction\_amount float NOT NULL CHECK (transaction\_amount > 0),

Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS exam (

exam\_id INT PRIMARY KEY,

erp INT NOT NULL,

course\_id INT NOT NULL,

marks float NOT NULL DEFAULT 0,

exam\_fee float NOT NULL DEFAULT 500 CHECK (exam\_fee < 2000),

exam\_type varchar(10) NOT NULL,

Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key (erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

1. **Drop-**is used to delete objects from the database.

DROP DATABASE IF EXISTS db\_student;

**7.DML(Data Manipulation Language)**

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

1.**INSERT**-is used to insert data into a table.

INSERT INTO courses VALUES (1001,"Physics",01,2011);

INSERT INTO courses VALUES (1002,"DBMS",01,2011);

INSERT INTO courses VALUES (1003,"MIT",01,2011);

INSERT INTO courses VALUES (1004,"DELD",01,2011);

INSERT INTO courses VALUES (1005,"Humanity",01,2011);

-- Student

INSERT INTO users VALUES (103218001,"password","as@gmail.com");

INSERT INTO users VALUES (103218002,"password","bs@gmail.com");

INSERT INTO users VALUES (103218003,"password","cs@gmail.com");

INSERT INTO users VALUES (103218004,"password","ds@gmail.com");

INSERT INTO users VALUES (103218005,"password","es@gmail.com");

-- Teachers

INSERT INTO users VALUES (13218001,"password","john@gmail.com");

INSERT INTO users VALUES (13218003,"password","hp@gmail.com");

INSERT INTO users VALUES (13218005,"password","goblin@gmail.com");

INSERT INTO users VALUES (13218007,"password","bjp@gmail.com");

-- Departments

INSERT INTO department VALUES (01,"Engineering");

INSERT INTO department VALUES (02,"Pure Science");

INSERT INTO department VALUES (03,"Arts");

INSERT INTO department VALUES (04,"Humanities");

INSERT INTO department VALUES (05,"Politics");

INSERT INTO department VALUES (06,"Music");

INSERT INTO department VALUES (07,"Commerce");

-- Faculty

INSERT INTO faculty VALUES (13218001,1001,01,"John Wick");

INSERT INTO faculty VALUES (13218003,1002,02,"Harry Potter");

INSERT INTO faculty VALUES (13218005,1003,07,"Goblin");

INSERT INTO faculty VALUES (13218007,1004,05,"Narendra Modi");

-- Student

INSERT INTO student VALUES (103218001,1001,"Clone - 11",21);

INSERT INTO student VALUES (103218002,1001,"Clone - 12",22);

INSERT INTO student VALUES (103218003,1001,"Clone - 13",23);

INSERT INTO student VALUES (103218004,1001,"Clone - 14",24);

**2.UPDATE-**is used to update existing data within a table.

CREATE TABLE IF NOT EXISTS faculty (

erp INT PRIMARY KEY NOT NULL,

course\_id INT NOT NULL,

dept\_id INT NOT NULL,

name varchar(20) NOT NULL,

Foreign Key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign Key (dept\_id) references department(dept\_id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS student (

erp INT PRIMARY KEY,

course\_id INT NOT NULL,

name varchar(20) NOT NULL,

roll\_no INT NOT NULL,

Foreign key(course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

-- Attendance for students only

create table IF NOT EXISTS attendance (

attendance\_id INT PRIMARY KEY,

erp INT,

course\_id INT NOT NULL,

days\_attended INT NOT NULL DEFAULT 0,

Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key (erp) references student(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS fees (

transaction\_id INT PRIMARY KEY,

erp INT NOT NULL,

transaction\_amount float NOT NULL CHECK (transaction\_amount > 0),

Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TABLE IF NOT EXISTS exam (

exam\_id INT PRIMARY KEY,

erp INT NOT NULL,

course\_id INT NOT NULL,

marks float NOT NULL DEFAULT 0,

exam\_fee float NOT NULL DEFAULT 500 CHECK (exam\_fee < 2000),

exam\_type varchar(10) NOT NULL,

Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,

Foreign key (erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE

);

1. **DELETE-**is used to delete records from a database table.
2. CREATE TABLE IF NOT EXISTS courses (
3. course\_id INT primary key,
4. course\_name varchar(20) NOT NULL,
5. sem INT NOT NULL,
6. year int
7. );
8. CREATE TABLE IF NOT EXISTS users (
9. erp INT PRIMARY KEY,
10. password varchar(20) NOT NULL,
11. email\_id varchar(20) NOT NULL UNIQUE
12. );
13. CREATE TABLE IF NOT EXISTS department (
14. dept\_id INT primary key,
15. dept\_name varchar(20) NOT NULL
16. );
17. CREATE TABLE IF NOT EXISTS faculty (
18. erp INT PRIMARY KEY NOT NULL,
19. course\_id INT NOT NULL,
20. dept\_id INT NOT NULL,
21. name varchar(20) NOT NULL,
22. Foreign Key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,
23. Foreign Key (dept\_id) references department(dept\_id) ON DELETE CASCADE ON UPDATE CASCADE
24. );
25. CREATE TABLE IF NOT EXISTS student (
26. erp INT PRIMARY KEY,
27. course\_id INT NOT NULL,
28. name varchar(20) NOT NULL,
29. roll\_no INT NOT NULL,
30. Foreign key(course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,
31. Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE
32. );
33. -- Attendance for students only
34. create table IF NOT EXISTS attendance (
35. attendance\_id INT PRIMARY KEY,
36. erp INT,
37. course\_id INT NOT NULL,
38. days\_attended INT NOT NULL DEFAULT 0,
39. Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,
40. Foreign key (erp) references student(erp) ON DELETE CASCADE ON UPDATE CASCADE
41. );
42. CREATE TABLE IF NOT EXISTS fees (
43. transaction\_id INT PRIMARY KEY,
44. erp INT NOT NULL,
45. transaction\_amount float NOT NULL CHECK (transaction\_amount > 0),
46. Foreign key(erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE
47. );
48. CREATE TABLE IF NOT EXISTS exam (
49. exam\_id INT PRIMARY KEY,
50. erp INT NOT NULL,
51. course\_id INT NOT NULL,
52. marks float NOT NULL DEFAULT 0,
53. exam\_fee float NOT NULL DEFAULT 500 CHECK (exam\_fee < 2000),
54. exam\_type varchar(10) NOT NULL,
55. Foreign key (course\_id) references courses(course\_id) ON DELETE CASCADE ON UPDATE CASCADE,
56. Foreign key (erp) references users(erp) ON DELETE CASCADE ON UPDATE CASCADE
57. );

**8. DCL-(Data Control Language)**

DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

**10. Triggers-**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

-- Triggers

DELIMITER $$

CREATE TRIGGER insert\_attendance

AFTER INSERT ON student

FOR EACH ROW

BEGIN

IF NOT EXISTS (select \* from attendance where erp=new.erp) THEN

insert INTO attendance values (new.erp%100,new.erp,new.course\_id,0);

END IF;

END$$

DELIMITER ;

**11. Procedure/Function-**"A **procedures** or **function** is a group or set of SQL and PL/SQL statements that perform a specific task."  
A function and  procedure is a named PL/SQL Block which is similar . The major difference between a procedure and a function is, a function must always return a value, but a procedure may or may not return a value.

CREATE PROCEDURE ins\_exam()

BEGIN

DECLARE DONE INT DEFAULT 0;

DECLARE CR\_ID INT;

DECLARE ST\_ID INT;

DECLARE S\_NAME VARCHAR(20);

DECLARE SEM INT;

DECLARE YR DECIMAL(10,3);

DECLARE CU1 CURSOR FOR SELECT \* FROM courses;

DECLARE CU2 CURSOR FOR SELECT erp FROM student;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET DONE = 1;

OPEN CU1;

OPEN CU2;

REPEAT

FETCH CU1 INTO CR\_ID,S\_NAME,SEM,YR;

-- INSERT INTO courses VALUES (1001,"Physics",01,2011);

FETCH CU2 INTO ST\_ID;

-- INSERT INTO student VALUES (103218001,1001,"Clone - 11",21);

IF NOT EXISTS(SELECT \* FROM exam WHERE erp = ST\_ID) THEN

IF NOT EXISTS(SELECT exam\_id FROM exam WHERE exam\_id = (SEM\*1000) + (CR\_ID%100) + (ST\_ID%10)) THEN

INSERT INTO exam VALUES ((SEM\*1000) + (CR\_ID%100) + (ST\_ID%10),ST\_ID,CR\_ID,0,500,"Subjective");

END IF;

END IF;

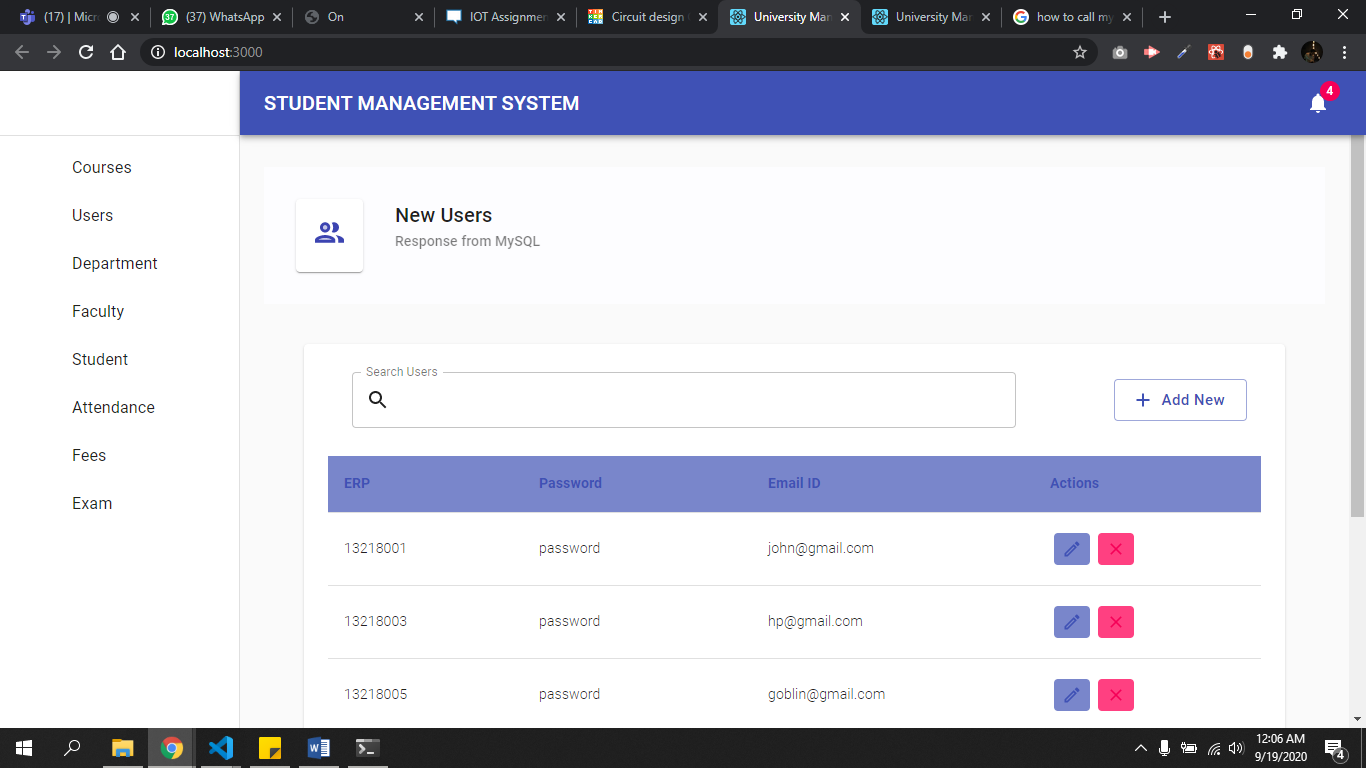
UNTIL DONE -- Till Done = 1

END REPEAT;

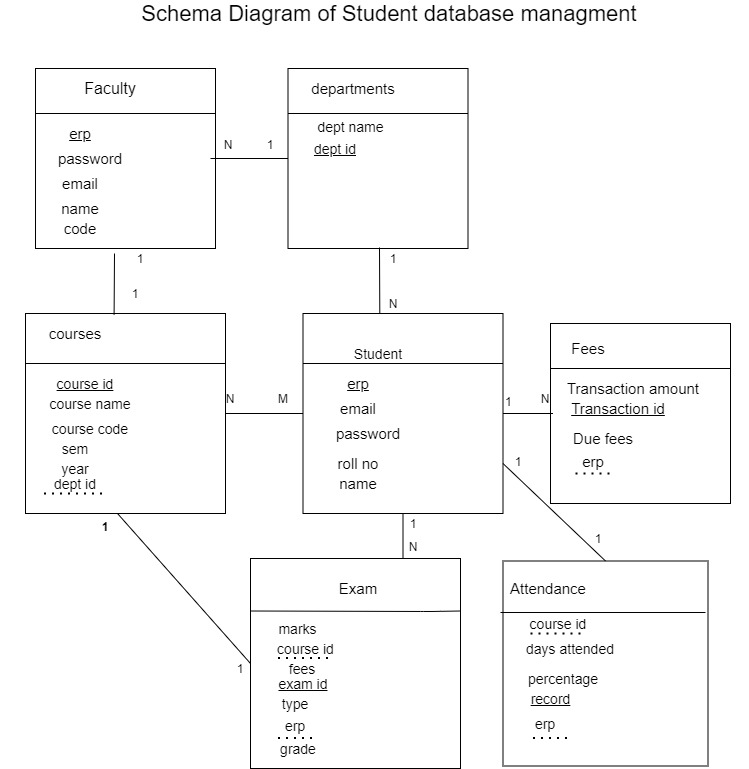
END $$

DELIMITER ;

**12 .Frontend UI-**



**13. Relational Database Design**



**14. Normalized Database Design**

* **1st Normal Form:**

Student can have multiple course\_id so First normalized form is needed and therefore a separate courses table was created with course\_id as primary key.

Due to inexistence of candidate keys

* **2nd Normal Form:**

Already in 2nd Normal Form

* **3rd Normal Form:**

Already in 3rd Normal Form

**Conclusion**

In the end a management system is only as effective as the number of features it can provide safely and consistently. To that end a student management system provides us with only half of the picture. The full university database management will consider student as well as faculty members.

