1. ABSTRACT

This Project is basically aimed in providing latest technologies & trends to the College sector service, so that college can store there student's records like details of student, attendance & semester marks which can be viewed by Administrator It is a comprehensive student information management system developed from the ground up to fulfill the needs of independent Colleges as they guide their students to success. The Education Edge integrated information management system connects operations in the College environment Admissions and Registration, Marks Reports, Attendance Reports, Course Details, Semester details. This reduces data error and ensures that information is always up-to-date throughout the College.

Manual Process of this requires a lot many of records to maintain. College authorities need to take care to store each and every student details and also there examination details .Manual process requires man power. Existing System is manual process .Data Security is not provided in this system. Integrating data is also a problem in this system .It is not User friendly system.

Proposed system is windows application. In this application student details are maintained efficiently admin has a facility to view the student details, edit the details .semester details and marks details are also maintained in this system. This has an enhanced facility.It is a fast, affordable, low-risk solution with easy implementation and lower maintenance and operational costs.

During a college management system the entities are student, admission, time table, lectures subjects, student marks, tuition fee. Also each particular entity has its own particular attribute which are again divided further by multi-valued, single valued, simple and composite

The aim of this thesis is to examine the relationship between different attributes and the student, faculty overall satisfaction with the college administration. It tries to uncover the most influential attributes for the formation of student and faculty satisfaction.

2. REQUIREMENTS SPECIFICATION

The project has six major entities namely Administrator, student, course, faculty, attendance, internal marks. The Administrator table consists of Admin id, user name and password. The student table consists of student Id, first name, last name, mobile no, address date of birth

The course has the following attributes course_id, course name, course description, start and end date of the course for which the student has registered. The faculty has the following attributes the user id to uniquely identify each faculty, faculty name, password for the secure and private access, their address and email

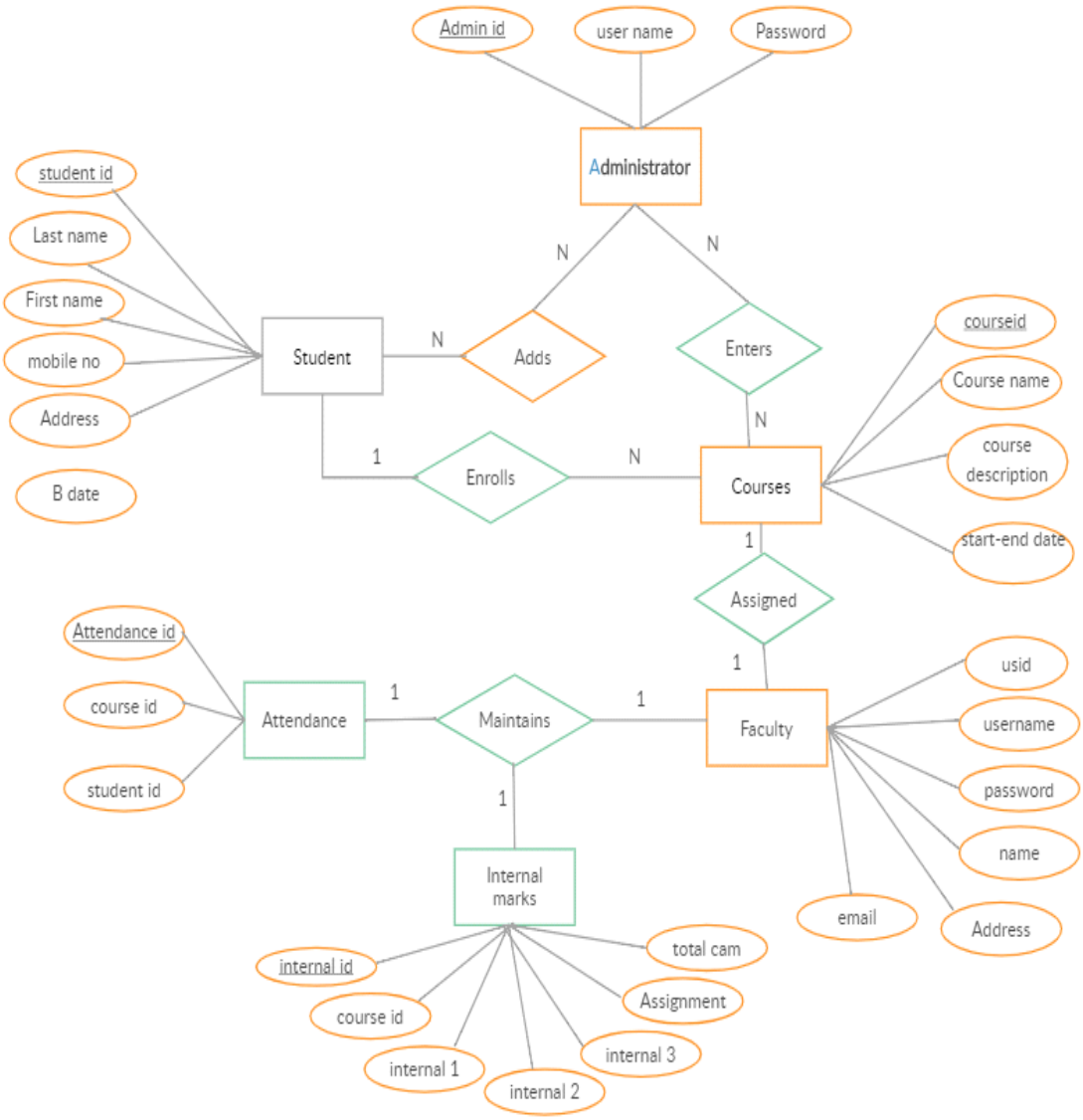
The attendance has the following attributes attendance id, course id to identify the course and the student id which uniquely identify the individual student. The relationships are formed between different modules in our project we have total five relations one is between the student and courses modules as reserves and the second one is between the faculty and course as “assigned” and third one is between the attendance and faculty as “maintains” and fourth is between the administrator and the course as “enters“ and the final is between administrator and student as adds.

For the application development we have two phases as front end and the back end the front end consists of HTML, CSS and PHP and backend as the SQL tables which stores the information from the student and faculty entered in the front end and saves information safely.

Hardware Requirements: Processor, Pentium-IV, Ram 1 GB Hard disk - 20 GB

This are the prerequisites needed for the hotel college management system.

3. ER DIAGRAM


Admin id
user name
Password
Administrator
N
N
student id
Last name
First name
mobile no
Address
B date
Student
N
Adds
Enters
courseid
Course name
course description
start-end date
N
1
Enrolls
N
Courses
1
Assigned
1
Attendance id
course id
student id
Attendance
1
Maintains
1
Faculty
usid
username
password
name
Address
email
1
Internal marks
internal id
course id
internal 1
internal 2
internal 3
Assignment
total cam

Explanation:

Entity-Relationship(ER) model is based on the notion of real world entities and relationships among them. While formulating real world scenario into the database model the ER model creates entity set, relationship set general attributes and constraints.

Entity- An entity in an ER model is a real world entity having properties called attributes. Every attribute is defined by its set of values called domain.

From the above ER diagram the entities are

1. Administrator

2. Courses

3. Faculty

4. Attendance

5. Student

6. Internal marks

Relationship – The logical association among entities is called relationships. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

From the above ER diagram the following relationships are:

1. N number of students enrolls for a single course results in the relationship **many to one** relationship.

2. faculty maintains the attendance i.e. one faculty can maintain one attendance report in **one to one** relationship.

3. one student can be enrolls by N number of courses in **one to many** relationship.

Primary Key: The PRIMARY KEY constraint uniquely identifies each record in a database table. Primary keys must contain UNIQUE values, and cannot contain NULL values. A table can have only one primary key, which may consists of single or multiple fields. From the above ER diagram the primary keys are

1. student_id from the student entity is a primary key.

2. course_id from the courses entity

3. admin_id from the administrator entity

4. faculty_id from the faculty entity

5. Internal_id form the internal marks entity

6. attendance_id is the primary key for the attendance entity

These are the following primary keys from the above ER diagram.

Weak entity: In a relational database, a weak entity is an entity that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a f primary key.

From the above ER diagram the weak entity is Internal marks which has the foreign key as course_id and primary key as internal_id.

4.Table Design with Integrity Constraints

Integrity constraints are set of rules. It is used to maintain the quality of information.

1. **Not null**: Ensures that a column cannot have a NULL values.

```
createtablestudent(
student_idvarchar(4)NOTNULL,
first_namevarchar(255)NOTNULL,
last_namevarchar(255)NOTNULL,
addressvarchar(30)NOTNULL,
DOBdateNOTNULL);
```

Output:

```
Command(s) completed successfully.
```

2. **Unique:** Ensures that all values in column are different.

```
createtablefaculty(
us_idvarchar(4)NOTNULL,
usernamevarchar(255)NOTNULL,
passwordvarchar(255)NOTNULL,
namevarchar(30)NOTNULL,
addressvarchar(20)NOTNULL,
emailvarchar(20),
unique(us_id),
);
```

Output:

```
Command(s) completed successfully.
```

3. **Primary Key**: A combination of a not null and unique.

```
createtableadministrator(
admin_idvarchar(4)NOTNULL,
user_namevarchar(255)NOTNULL,
passwordvarchar(255)NOTNULL,
primarykey(admin_id));
```

Output:

```
Command(s) completed successfully.
```

4.**Foreign Key**: Uniquely identifies a row/record in another table.

```
createtableattendance(
attendance_idvarchar(255)NOTNULL,
course_idvarchar(255)NOTNULL,
student_idvarchar(255)NOTNULL,
primarykey(attendance_id),
foreignkey(course_id)referencesstudent(student_id)
);
```

Output:

```
Command(s) completed successfully.
```

5. Check: Ensures that all values in column are satisfying a specific condition.

```
createtableinternal_marks(
internal_idvarchar(255),
course_idvarchar(255),
```

```
internal_1int,
internal_2int,
internal_3int,
assignmentint,
total_markint,
check(internal_1>=0),
check(internal_2>=0),
check(internal_3>=0));
```

Output:

```
Command(s) completed successfully.
```

6. Default: Sets a default value to column when no value is assigned.

```
createtablecourse(
course_idvarchar(255),
course_namevarchar(255),
start_enddatedefault'0/0/0'
);
```

Output:

```
Command(s) completed successfully.
```

5. DDL QUERIES

1.1.Create table

```
createtablecourse(
course_idvarchar(255),
course_namevarchar(255),
start_enddatedefault'0/0/0'
);
```

Output

```
Table created.
```

1.2. Create table

```
createtablefaculty(
us_idvarchar(4)NOTNULL,
usernamevarchar(255)NOTNULL,
passwordvarchar(255)NOTNULL,
namevarchar(30)NOTNULL,
addressvarchar(20)NOTNULL,
emailvarchar(20),
unique(us_id),
);
```

Output

```
Table created.
```

1.3. Create table

```
createtableinternal_marks(
internal_idvarchar(255),
course_idvarchar(255),
internal_1int,
```

```
internal_2int,
internal_3int,
assignmentint,
total_markint,
check(internal_1>=0),
check(internal_2>=0),
check(internal_3>=0));
```

Output

```
Table created.
```

1.4.create table

```
createtablecourse(
course_idvarchar(255),
course_namevarchar(255),
start_enddatedefault'0/0/0'
);
```

Output

```
Table created.
```

1.5.Create table

```
createtableadministrator(
admin_idvarchar(4)NOTNULL,
user_namevarchar(255)NOTNULL,
passwordvarchar(255)NOTNULL,
primarykey(admin_id));
```

Output

```
Table created.
```

2.1 Alter table

```
altertablecourseADDvalueint;
```

Output

```
Table altered.
```

2.2 modify table

```
altertablecourse modifyvalueint;
```

2.3 drop

```
altertablecoursedropcolumnvalue;
```

Output

```
Table altered.
```

3.drop table

```
droptablecourse;
```

Output

```
Table drop.
```

6.DML QUERIES

INSERT

```
SQL> insert into course values(CA01,'dbms','2017-05-03');

1 row created.

SQL> insert into hotel values('CA02',’pos’,'2017-05-01');

1 row created.

SQL> insert into hotel values('CA03',’SIA,’2017-08-05');

1 row created.


SQL> select distinct course_id from course;

Course_id
-------------------------
CA01
CA02
CA03

SQL> select * from course where course_id =CA01;

Course_id           course_name          start_date
--------------      -----------          -------------
CA01                dbms                 2017-05-11

SQL> select * from internal_marks where internal_1 <0;

no rows selected
```

```
SQL> select * from internal_marks where internal_2 >50;

internal_id   course_id     internal_1    internal_2    internal_3
-----------   --------   ----------  -----------       -------------
IN02          CA02          61            97           89

SQL>select  * from internal where internal_1 between 50 and 100;

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN02          CA02       61          97            89
IN03          CA03       71          77            92
IN04          CA04       53          67            76
  SQL>select  * from internal where internal_1 in (61,71);

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN02          CA02       61          97            89
IN03          CA03       71          77            92
```

String operations:

```
SQL> select * from student where first_name like '%h';

Student_idfirst_namelast_name       address    DOB
----------   ----------      ---------       -------     ---
ST_01        harishkumar            #102     03-05-1998
ST_02        ganeshnithya           #103     09-03-1998

SQL> select * from student where first_name like 'h%';

Student_idfirst_namelast_name       address    DOB
----------   ----------      ---------       -------     ---
ST_01        harishkumar            #102     03-05-1998
```

```
ST_03        hariharan            #104     06-12-1998

SQL> select * from student where first_name like '%h%';

Student_idfirst_namelast_name       address    DOB
----------   ----------      ---------       -------     ---
ST_01        harishkumar            #102     03-05-1998
ST_02        ganeshnithya           #101     02-12-1998
ST_03        hariharan            #104     06-12-1998
```

Arithmetic operations:

```
SQL> select internal_id,course_id,internal_1+2  from internal;

internal_id   course_id   internal_1
-----------   --------   ----------
IN02          CA02       64
IN03          CA03       73
IN04          CA04       55
```

```
SQL> select internal_id,course_id,internal_1-2  from internal;

internal_idcourse_id   internal_1
-----------   --------   ----------
IN02          CA02       60
IN03          CA03       69
IN04          CA04       51
```

```
SQL> select internal_id,course_id,internal_1*1 from internal;

internal_idcourse_id   internal_1
-----------   --------   ----------
IN02          CA02       62
IN03          CA03       71
IN04          CA04       53
```

CONDITION CLAUSE

```
SQL>select  * from internal where internal_1='IN02 and internal_1>61;

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN02          CA02       61          97            89
```

RENAME

```
SQL> select internal_id,course_id as id,internal_1*1 from internal;

internal_id    id          internal_1
-----------   --------   ----------
IN02          CA02       62
IN03          CA03       71
IN04          CA04       53


SQL> select internal_id as I_ID,course_id ,internal_1*1 from internal;

I_id          course_id   internal_1
-----------   --------   ----------
IN02          CA02       62
IN03          CA03       71
IN04          CA04       53
```

UPDATE

```
SQL> update student set DOB='31-07-1997' where ST_ID='SA01';

1 row updated

SQL> update internal set internal_1=100 where internal_2<50;

1 row updated

Sql> update student set last_name='kumar.R' where first_name='harish';

1 row updated
```

DELETE

```
Sql> delete from student where Student_id='ST_ID';

1 row deleted
```

ORDER BY CLAUSE

```
sql> select from internal order by internal_1;

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN04          CA04       53          67            76
IN02          CA02       61          97            89
IN03          CA03       71          77            92
```

```
sql> select from internal order by internal_1  asc;

internal_id   course_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN04          CA04       53          67            76
IN02          CA02       61          97            89
IN03          CA03       71          77            92

sql> select from internal order by internal_1 desc;

internal_id   course_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN03          CA03       71          77            92
IN02          CA02       61          97            89
IN04          CA04       53          67            76
```

AGGREGATION functions

```
sql> select sum(internal_1) from internal ;

sum(internal_1)
---------------

185

sql> select * from internal ;

count(*)
--------
3

sql> select avg(internal_1) from internal ;

avg(internal_1)
```

```
--------------
61.66666666
```

SET OPERATION

Union

```
sql>select student_id from student
union
selectstudent_id from attendance


student_id
----------
ST01
ST01
ST02
ST02
ST03
ST04
ST04
ST05
ST05
ST06
ST06
ST07
ST08
ST08
```

INTERSECTION

```
SQL>Select student_id from student
Intersection
Select student_id from attendance
```

```
Student_id
----------
ST01
ST02
ST03
ST04
ST05
ST06
ST07
ST08
`
```

EXCEPT

```
SQL>select internal_1 from internal_marks
Except
Select internal_2 from internal_marks

Internal_1
----------
68

89

56
```

IN

```
SQL> select * from internal where internal_1 in (61,71);

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN02          CA02       61          97            89
IN03          CA03       71          77            92
```

NOT IN

```
SQL> select * from internal where internal_1 not in (61,71);

internal_idcourse_id   internal_1  internal_2    internal_3
-----------   --------   ----------  -----------  -------------
IN02          CA02       81          97            89
IN03          CA03       21          77            92
IN04          CA04       99          98            87
```

NATURAL JOIN

```
Sql>select student.student_id, faculty_id
From student
Inner join faculty on student.student_id=student.faculty_id

Student_idfaculty_id
-----------       ------------
ST01               FA04
ST02               FA05
ST03               FA01
ST04               FA02
ST05               FA07
ST07               FA06
```

Group by

```
Sql>select s.student_id,m.internal_1
from student s,internal_mark m
group by student_id
```

```
student_id  internal_1
----------  -----------
ST01          56
ST01          54
ST02          71
ST02          99
ST03          99
ST03          87
ST04          76
ST04          87
ST05          98
ST05          75
```

GROUP BY having clause

```
Sql>select s.student_id,m.internal_1
from student s,internal_mark m
group by student_id
having internal_1>50;
```

```
student_id  internal_1
----------  -----------
ST01          56
ST01          54
ST02          71
ST03          87
```

```
ST04          76
ST04          87
ST05          98
ST05          75
```

AGGREGATION WITH GROUP BY

```
Sql>select student.student_id, faculty_id
From student
Group by student_id
Having count(*)>1;

Student_idfaculty_id
-----------       ------------
ST01               FA04
ST02               FA05
ST03               FA01
ST04               FA02
ST05               FA07
ST07               FA06
```

7. SUB QUERIES

```
sql>select * from student

where

student_id=(select student_id

from student

wherestudent_id=ST01);
```

output

```
student_id      first_name      last_name       address      dob

----------      ----------      ---------     -------       ----

ST01            harish          kumar           #201       1998-05-03
```

```
Sql>Select count(*)

From ( select * from

studentfirst_name like 'h%');
```

OUTPUT

```
count(*)

-------

3

Sql>Select student_id, avg(internal_mark)

From student

Group by student_id

Having avg(internal_mark)>(select avg(internal_mark)
```

```
From student);
```

OUTPUT

```
Student_id      avg(internal_mark)
----------      ------------------
ST01            97.7
ST02            99.9
ST03            86.7
ST04            87.6
ST05            87.5
ST06            99.9
ST07            97.8
```

```
sql>select (
selectavg(internal_mark)
from internal) as avg_mark;
avg_mark
-------
99.7
87.7
78.9
77.8
87.9
78.8
```

```
87.8

Sql>select count(*)

From

(select internal_mark_1

                  (selectavg(internal_mark_1) as avg(internal_mark)

                  From internal_makr

)

Where internal_1>avg(internal_1);

Count(*)

-------

6
```

8.SQL FUNCTIONS

1. Concat : The CONCAT() function adds two or more expressions together.

```
Select concat(FirstName," ",LastName) as Name from
    Student;
```

Output:

```
Name

----------

Harish Kumar

VarunDatta

KaulSidha
```

2. Lower : The LOWER() function converts a string to lower-case.

```
SELECT LOWER(FirstName) AS LowercaseFirstName

FROM Student;
```

Output:

```
Number of Records: 91

LowercaseCustomerName

varun

harish

aayush
```

3. Reverse: The REVERSE() function reverses a string and returns the result.

```
SELECT REVERSE(FirstName)

FROM Student;
```

Output:

```
Number of Records: 3

nuraV

hsiraH

luaK
```

4. Substring: The SUBSTRING() function extracts some characters from a string.

```
SELECT SUBSTRING(FirstName, 1, 3) AS ExtractString

FROM Student;
```

Output:

```
Number of Records: 3

ExtractString

Var

Har
```

```
Aay
```

5.Abs: The ABS() function returns the absolute value of a number.

```
SELECT Abs(-243.5) AS AbsNum;
```

Output:

```
Number of Records: 1

AbsNum

243.5
```

6.Ceiling/floor: The CEILING() function returns the smallest integer value that is >= a number.

```
SELECT CEILING(25) AS CeilValue;
```

Output:

```
Number of Records: 1

CeilValue

26
```

7.Current Timestamp: The CURRENT_TIMESTAMP function returns the current date and time, in a 'YYYY-MM-DD hh:mm:ss.mmm' format.

```
SELECT CURRENT_TIMESTAMP;
```

Output:

```
Number of Records: 1

2018-10-09 01:47:36.477
```

8.**Dateadd**: The DATEADD() function adds a time/date interval to a date and then returns the date.

```
SELECT DATEADD(year, 1, '2017/08/25') AS DateAdd;
```

Output:

```
Number of Records: 1

DateAdd

2018-08-25 00:00:00.000
```

9.Datediff: The DATEDIFF() function returns the difference between two dates.

```
SELECT DATEDIFF(year, '2017/08/25', '2011/08/25') AS DateDiff;
```

Output:

```
Number of Records: 1

DATEDIFF("2017-06-25", "2017-06-15")

10
```

10.**Getdate**: The GETDATE() function returns the current database system date and time, in a 'YYYY-MM-DD hh:mm:ss.mmm' format.

```
SELECT GETDATE();
```

Output: 2018-10-09 01:41:41.943

9. VIEWS

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

CREATE VIEW Syntax

CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;

1.
```
CREATE VIEW StudentList AS

SELECT FirstName, LastName

FROM Student;
```

OUTPUT:

```
You have made changes to the database.
```

2.CREATE VIEW AdmissionList AS

SELECT CourseName, StudentName

FROM Admission;

OUTPUT:

You have made changes to the database.

3.CREATE VIEW Lectures AS

SELECT Subject, Lecture

FROM Lectures;

OUTPUT:

You have made changes to the database.

4.CREATE VIEW AdmissionList AS

SELECT FirstName

FROM Admission;

OUTPUT:

You have made changes to the database.

5.CREATE VIEW Subjects AS

SELECT CourseName

FROM Subject;

OUTPUT:

You have made changes to the database.

10. NORMALISATION

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

The most commonly used normal forms:

First normal form(1NF)

Second normal form(2NF)

Third normal form(3NF)

Boyce &Codd normal form (BCNF)

First normal form (1NF):

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a college wants to store the names and contact details of its students. It creates a table that looks like this:

stu_idstu_namestu_addressstu_mobile

| --- | --- | --- | --- |
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
			9900012222
103	Ron	Chennai	7778881212

This table is not in 1NF as the rule says “each attribute of a table must have atomic (single) values”, the emp_mobile values for employees Jon & Lester violates that rule.

To make the table complies with 1NF we should have the data like this:

stu_idstu_namestu_addressstu_mobile

-------	----------	------------	------------
101	Herschel	New Delhi	8912312390
102	Jon	Kanpur	8812121212
102	Jon	Kanpur	9900012222
103	Ron	Chennai	7778881212

Second normal form (2NF):

A table is said to be in 2NF if both the following conditions hold:

-Table is in 1NF (First normal form)

-No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

teacher_id	subject	teacher_age
-------	----------	------------
111	Maths	38
111	Physics	38
222	Biology	38
333	Physics	40

Candidate Keys: {teacher_id, subject}

Non prime attribute: teacher_age

To make the table complies with 2NF we can break it in two tables like this:

teacher_subjet table:

teacher_id	subject
111	Maths
111	Physics
222	Biology
333	Physics

teacher_details table:

teacher_id	teacher_age
111	38
222	38
333	40

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

-Table must be in 2NF

-Transitive functional dependency of non-prime attribute on any super key should be removed.

-An attribute that is not part of any candidate key is known as non-prime attribute.

In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency X-> Y at least one of the following conditions hold:

-X is a super key of table

-Y is a prime attribute of table

An attribute that is a part of one of the candidate keys is known as prime attribute.

Example: Suppose a college wants to store the complete address of each student, they create a table named stu_details that looks like this:

stu_id	stu_name	stu_zip	stu_state
1001	John	282005	UP
1002	Ajeet	222008	TN
1006	Lora	282007	TN
1101	Lilly	292008	UK

Super keys: {stu_id}, {stu_id, stu_name}, {stu_id, stu_name, stu_zip}…so on

Candidate Keys: {stu_id}

Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

Student table:

stu_id	stu_name	stu_zip
1001	John	282005
1002	Ajeet	222008
1003	Lora	282007
1101	LillY	292008

Student_zip table:

stu_zip	stu_state
282005	UP
222002	TN
282007	TN
292008	UK

Boyce Codd normal form (BCNF)

It is an advance version of 3NF that's why it is also referred as 3.5NF. BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency X->Y, X should be the super key of the table.

Example: Suppose there is a college wherein students are in more than one department. They store the data like this:

stu_id	stu_nationality	stu_dept	dept_type
1001	Austrain	Production and planning	D001
1001	Austrain	file	D001
1002	Amerian	design and technical support	D134
1102	American	Purchasing department	D134

Functional dependencies in the table above:

stu_id ->stu_nationality

stu_dept -> {dept_type}

Candidate key: {stu_id, stu_dept}

To make the table comply with BCNF we can break the table in three tables like this:

stu_nationality table:

stu_idstu_nationality

1001	Austrian
1002	American

stu_dept table:

stu_deptdept_type

Production and planning	D001
Stores	D001
design and technical support	D134
Purchasing department	D134

emp_dept_mapping table:

stu_id	stu_dept
1001	Production and planning
1001	stores
1002	design and technical support
1002	Purchasing department

Functional dependencies:

stu_id ->stu_nationality

stu_dept -> {dept_type}

C**andidate keys:**

For first table: stu_id

For second table: stu_dept

For third table: {stu_id, stu_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.

11.APPLICATION DEVELOPMENT

PHP and MySQL Development

PHP is a fast and feature-rich open source scripting language used to develop Web Applications or Internet / Intranet Applications.

MySQL is a powerful open source database server built based on a relational database management system (RDBMS) and is capable of handling a large concurrent database connection.

When combined together, talented PHP and MySQL developers can build very powerful and scalable Web / Internet / Intranet Applications.

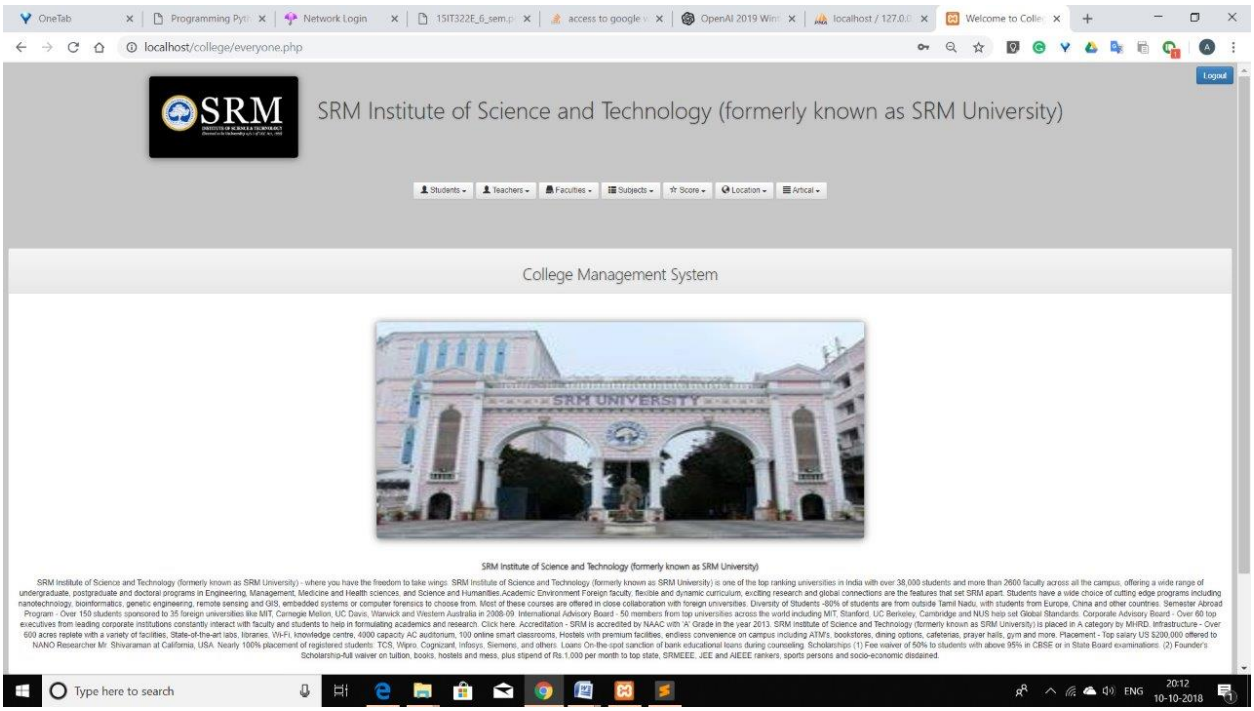
PHP and MySQL are referred to as development tools.

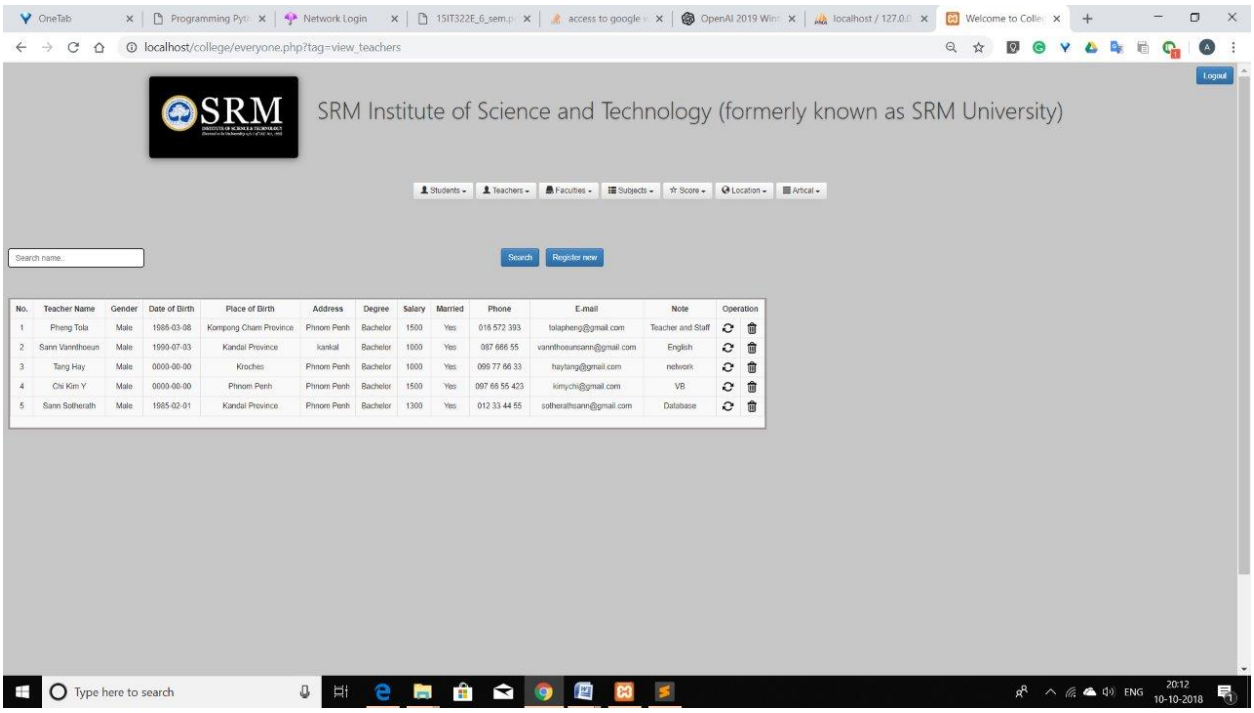
PHP and MySQL are Open Source, meaning that they are free development tools, and there is a large community of dedicated volunteer programmers who contribute to make improvements and are continuously adding features to it. The development tools and database servers that require licensing costs have limited programming resources compared to open source development tools, which have an enormous and fast growing dedicated and knowledgeable community that extends around the world.

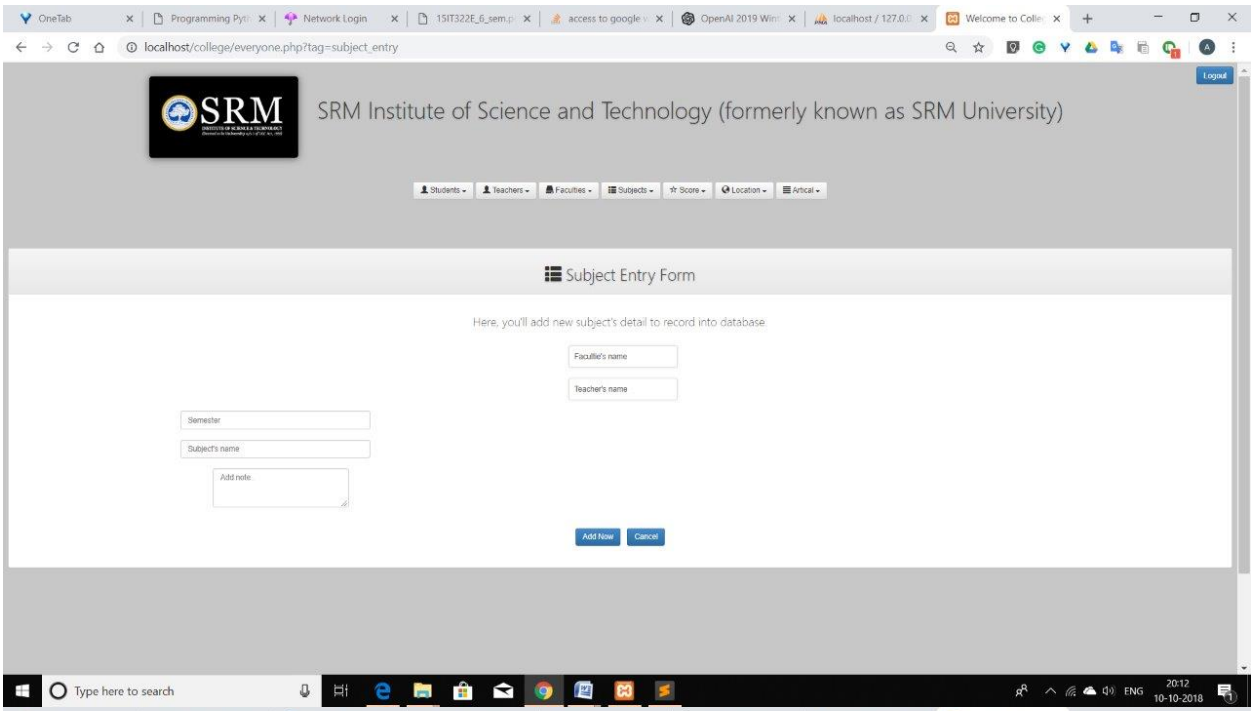
There has been disagreement about which tool is better. Naturally, the developer who is more familiar with one tool over the other will stand behind the tool that he or she has experience with.

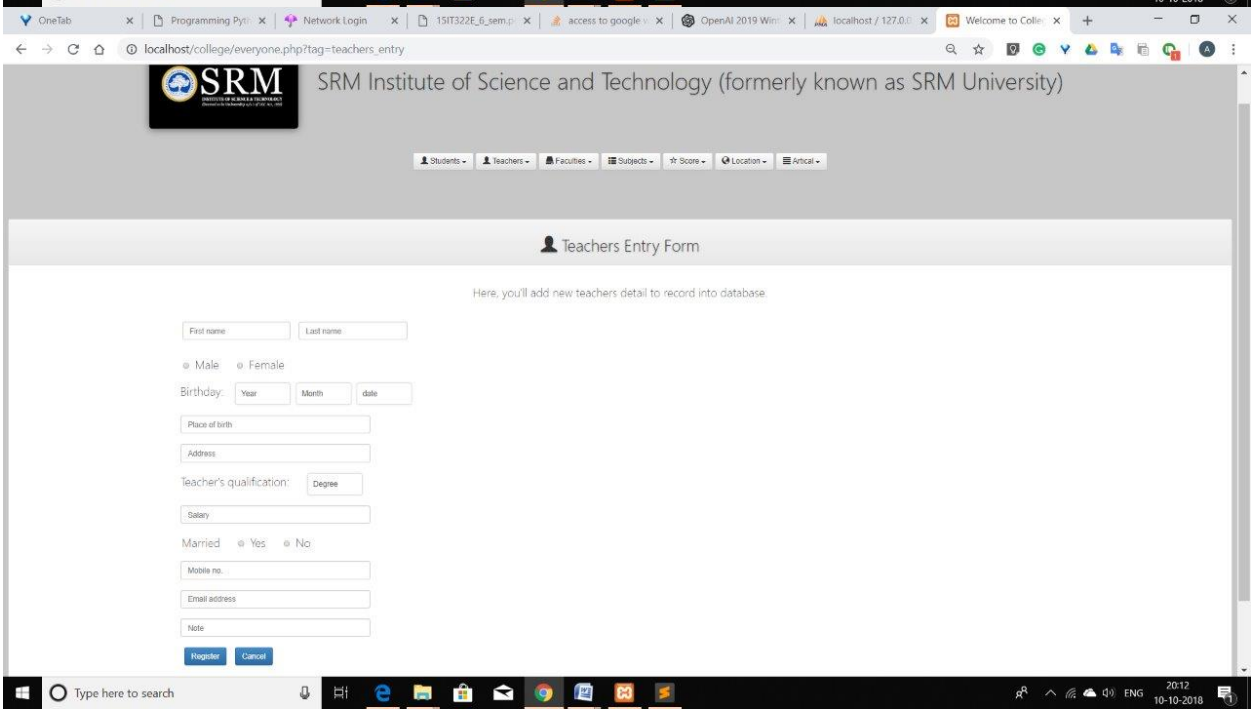
With our experience, we have found that, PHP and MySQL are the best development tools. When developed correctly, applications can be built with clean and simple usability, complex functionality, speed, power and scalability.

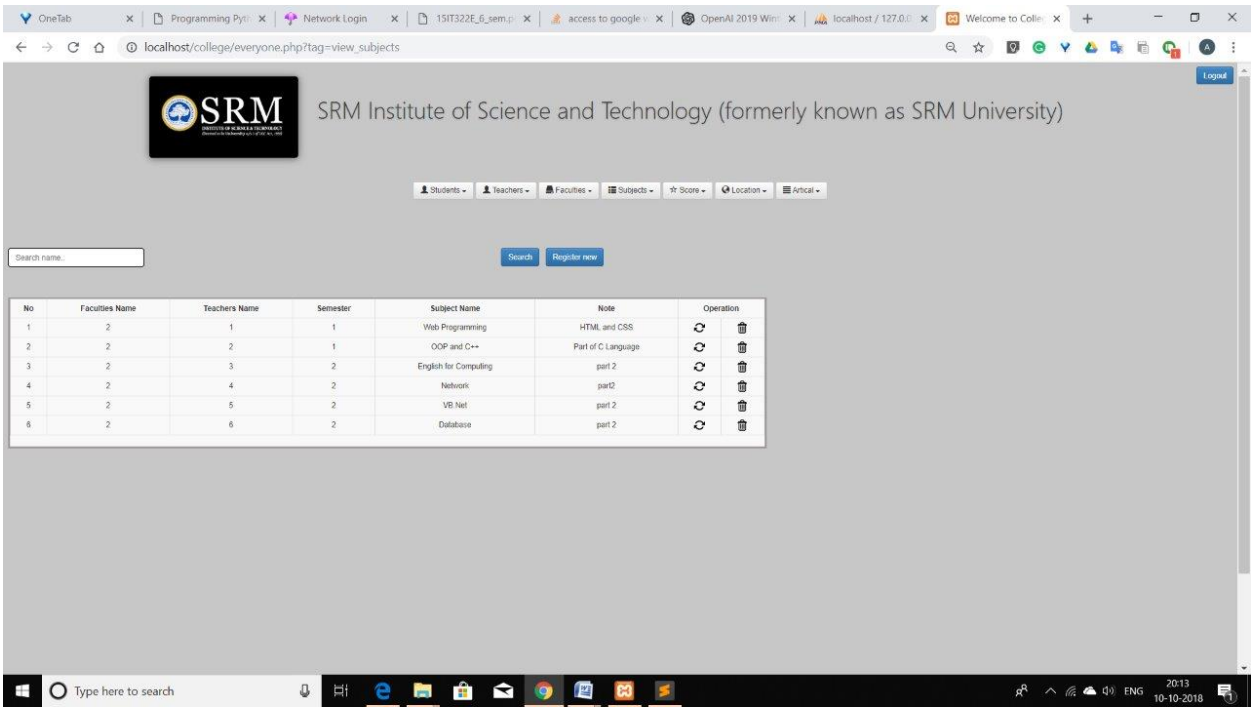
Tools Used: The tools used in making the project are HTML5, CSS3, JavaScriptTechniqueused : Bootstrap, PHP, MySQL for database creation management, PHP development environment provided by xampp and PhPMyAdmin and JQuery.

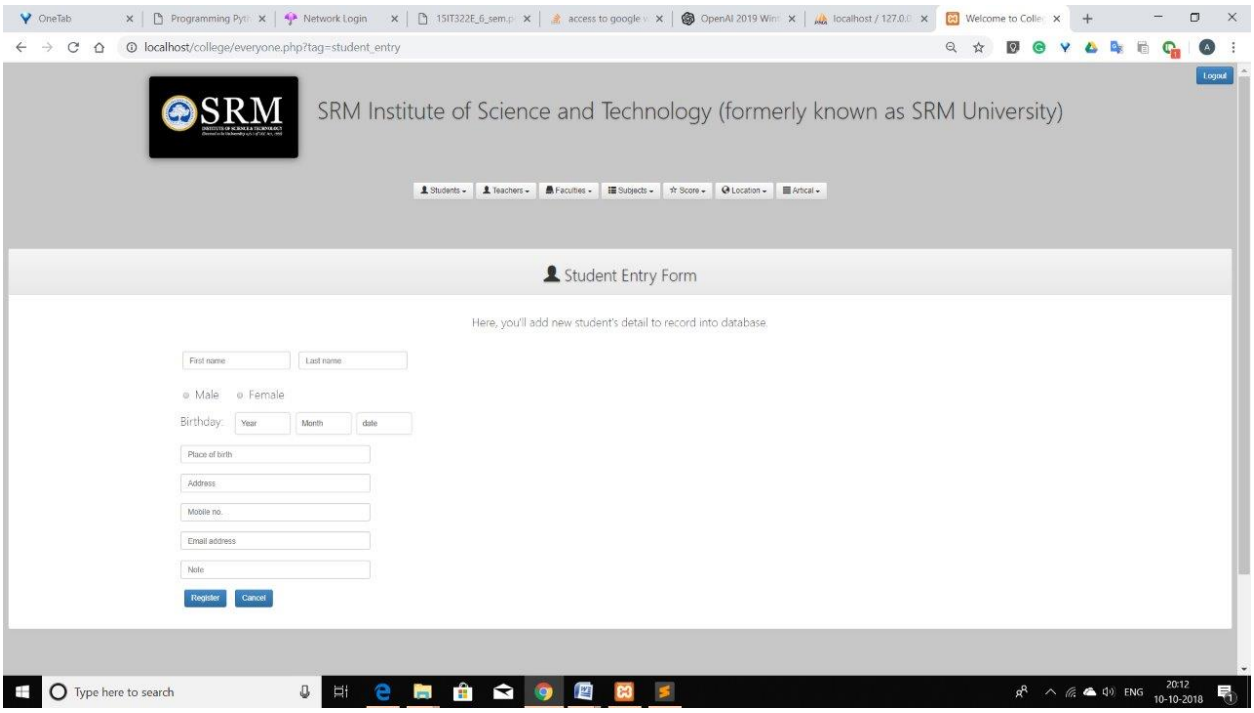

localhost/college/everyone.php
Logout
SRM Institute of Science and Technology (formerly known as SRM University)
Students
Teachers
Faculties
Subjects
Score
Location
Artical
College Management System
SRM UNIVERSITY
SRM Institute of Science and Technology (formerly known as SRM University)
SRM Institute of Science and Technology (formerly known as SRM University) - where you have the freedom to take wings. SRM Institute of Science and Technology (formerly known as SRM University) is one of the top ranking universities in India with over 38,000 students and more than 2600 faculty across all the campus, offering a wide range of undergraduate, postgraduate and doctoral programs in Engineering, Management, Medicine and Health sciences, and Science and Humanities.Academic Environment Foreign faculty, flexible and dynamic curriculum, exciting research and global connections are the features that set SRM apart. Students have a wide choice of cutting edge programs including nanotechnology, bioinformatics, genetic engineering, remote sensing and GIS, embedded systems or computer forensics to choose from. Most of these courses are offered in close collaboration with foreign universities. Diversity of Students -80% of students are from outside Tamil Nadu, with students from Europe, China and other countries. Semester Abroad Program - Over 150 students sponsored to 35 foreign universities like MIT, Carnegie Mellon, UC Davis, Warwick and Western Australia in 2008-09. International Advisory Board - 50 members from top universities across the world including MIT, Stanford, UC Berkeley, Cambridge and NUS help set Global Standards. Corporate Advisory Board - Over 60 top executives from leading corporate institutions constantly interact with faculty and students to help in formulating academics and research. Click here. Accreditation - SRM is accredited by NAAC with 'A' Grade in the year 2013. SRM Institute of Science and Technology (formerly known as SRM University) is placed in A category by MHRD. Infrastructure - Over 600 acres replete with a variety of facilities, State-of-the-art labs, libraries, Wi-Fi, knowledge centre, 4000 capacity AC auditorium, 100 online smart classrooms, Hostels with premium facilities, endless convenience on campus including ATM's, bookstores, dining options, cafeterias, prayer halls, gym and more. Placement - Top salary US $200,000 offered to NANO Researcher Mr. Shivaraman at California, USA. Nearly 100% placement of registered students: TCS, Wipro, Cognizant, Infosys, Siemens, and others. Loans On-the-spot sanction of bank educational loans during counselling. Scholarships (1) Fee waiver of 50% to students with above 95% in CBSE or in State Board examinations. (2) Founder's Scholarship-full waiver on tuition, books, hostels and mess, plus stipend of Rs.1,000 per month to top state, SRMEEE, JEE and AIEEE rankers, sports persons and socio-economic disdained.
Type here to search
ENG
20:12
10-10-2018


localhost/college/everyone.php?tag=view_teachers
Logout
SRM Institute of Science and Technology (formerly known as SRM University)
Students
Teachers
Faculties
Subjects
Score
Location
Artical
Search name..
Search
Register new
No. Teacher Name Gender Date of Birth Place of Birth Address Degree Salary Married Phone E-mail Note Operation
1 Pheng Tola Male 1986-03-06 Kompong Cham Province Phnom Penh Bachelor 1500 Yes 016 572 393 tolapheng@gmail.com Teacher and Staff
2 Sann Vannthoeun Male 1990-07-03 Kandal Province kankal Bachelor 1000 Yes 087 666 55 vannthoeunsann@gmail.com English
3 Tang Hay Male 0000-00-00 Kroches Phnom Penh Bachelor 1000 Yes 099 77 66 33 haytang@gmail.com network
4 Chi Kim Y Male 0000-00-00 Phnom Penh Phnom Penh Bachelor 1500 Yes 097 66 55 423 kimychi@gmail.com VB
5 Sann Sotherath Male 1985-02-01 Kandal Province Phnom Penh Bachelor 1300 Yes 012 33 44 55 sotherathsann@gmail.com Database
Type here to search
ENG
20:12
10-10-2018


localhost/college/everyone.php?tag=subject_entry
SRM Institute of Science and Technology (formerly known as SRM University)
Logout
Students
Teachers
Faculties
Subjects
Score
Location
Artical
Subject Entry Form
Here, you'll add new subject's detail to record into database
Faculties's name
Teacher's name
Semester
Subject's name
Add note
Add Now
Cancel
Type here to search
ENG
20:12
10-10-2018


localhost/college/everyone.php?tag=teachers_entry
SRM Institute of Science and Technology (formerly known as SRM University)
Students
Teachers
Faculties
Subjects
Score
Location
Artical
Teachers Entry Form
Here, you'll add new teachers detail to record into database
First name
Last name
Male
Female
Birthday:
Year
Month
date
Place of birth
Address
Teacher's qualification:
Degree
Salary
Married
Yes
No
Mobile no.
Email address
Note
Register
Cancel
Type here to search
ENG
20:12
10-10-2018


localhost/college/everyone.php?tag=view_subjects
SRM Institute of Science and Technology (formerly known as SRM University)
Logout
Students
Teachers
Faculties
Subjects
Score
Location
Artical
Search name..
Search
Register new
No
Faculties Name
Teachers Name
Semester
Subject Name
Note
Operation
1 2 1 1 Web Programming HTML and CSS
2 2 2 1 OOP and C++ Part of C Language
3 2 3 2 English for Computing part 2
4 2 4 2 Network part2
5 2 5 2 VB.Net part 2
6 2 6 2 Database part 2
Type here to search
ENG
20:13
10-10-2018


localhost/college/everyone.php?tag=student_entry
SRM Institute of Science and Technology (formerly known as SRM University)
Logout
Students
Teachers
Faculties
Subjects
Score
Location
Artical
Student Entry Form
Here, you'll add new student's detail to record into database
First name
Last name
Male
Female
Birthday:
Year
Month
date
Place of birth
Address
Mobile no.
Email address
Note
Register
Cancel
Type here to search
ENG
20:12
10-10-2018

12. Conclusion

College management system is an integrated web application that handles various academic and non academic activities of a College/Academic Institute. The system can access by every students/faculties/employees of the institution through internet connected computers or internet enabled mobile devices with the aid of his user name and password. Every user will have a customized home page with his/her profile management facilities. Through links that displays in the home page the user can access different options of the website assigned to him.Though the system allows access to every one there is a significant security risk involved in this project. To tackle this problem we suggest a modular structure in the proposed system and a complete isolation of the financial and administrative modules from the public portal. Only trusted IPs can access these modules. Web services will interact to the financial and administrative modules to fetch necessary information to display in the public portal. Although a standard password policy will be followed in the designing of the system to prevent the possibilities of malicious activities of itching users.Aself driven module in the proposed system will accomplish the automated tasks such as Email Alerts, SMS alerts, Notifications to the administrator etc.