# DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

Nishanth S Shastry

## Interactive Student/Faculty Networking Platform

This project is to build an application to manage a student and faculty profile(s) database. It also helps to support the networking of students, faculties and moderators have discussion forums and clubs.

The tables (relations) are implemented in MySQL RDBMS. There are a total of 14 main tables which also form the entity relationship and a total of 10 mapping tables avoid repetition of values in the main table and manage the many –to – many relationships.

The SQL queries are implemented and tested in MySQL RDBMS.

The triggers for this project are when a new event such as, a trigger is required, so that whenever there is an insert or update on the Event table, an alert is created, if alert is already there, it needs to be updated.

This is the design proposal for the application in context and during the actual development there maybe changes to the same due some scenarios missed out from consideration. However, the overall concept would remain the same.

For frontend development PHP will be used to connect the UI and Backend, HTML, CSS, JS for the smooth user-friendly interface.

# Table of Contents

# Introduction

This project is to build an application to manage a student and faculty profile(s) database.
It also helps to support the networking of students, faculties and moderators have discussion forums and clubs.

It is a student and Faculty Database managed by students and faculties.
A student and faculty must first register before they can use the site.
It's a networking site for the school, where students and faculties can get to know each other and interact. It will act as guidance for the new students to get to know his/her school better and to get familiarized with faculties and other activities. It would be a one stop site, no need of Facebook or LinkedIn to know about a particular person of your school or to get info about a particular event or any restaurant/public place to hang around. People can stay updated of who is doing what like who is attending which seminar, who is doing which project etc.

Roles in the DataBase: -

- MainUser – is a registered user of the application with a valid login name and password.

- Student – is also a registered user but with the user type as student.

- Faculty – Is a registered user, of user type "Faculty" – is basically a professor handling many courses.

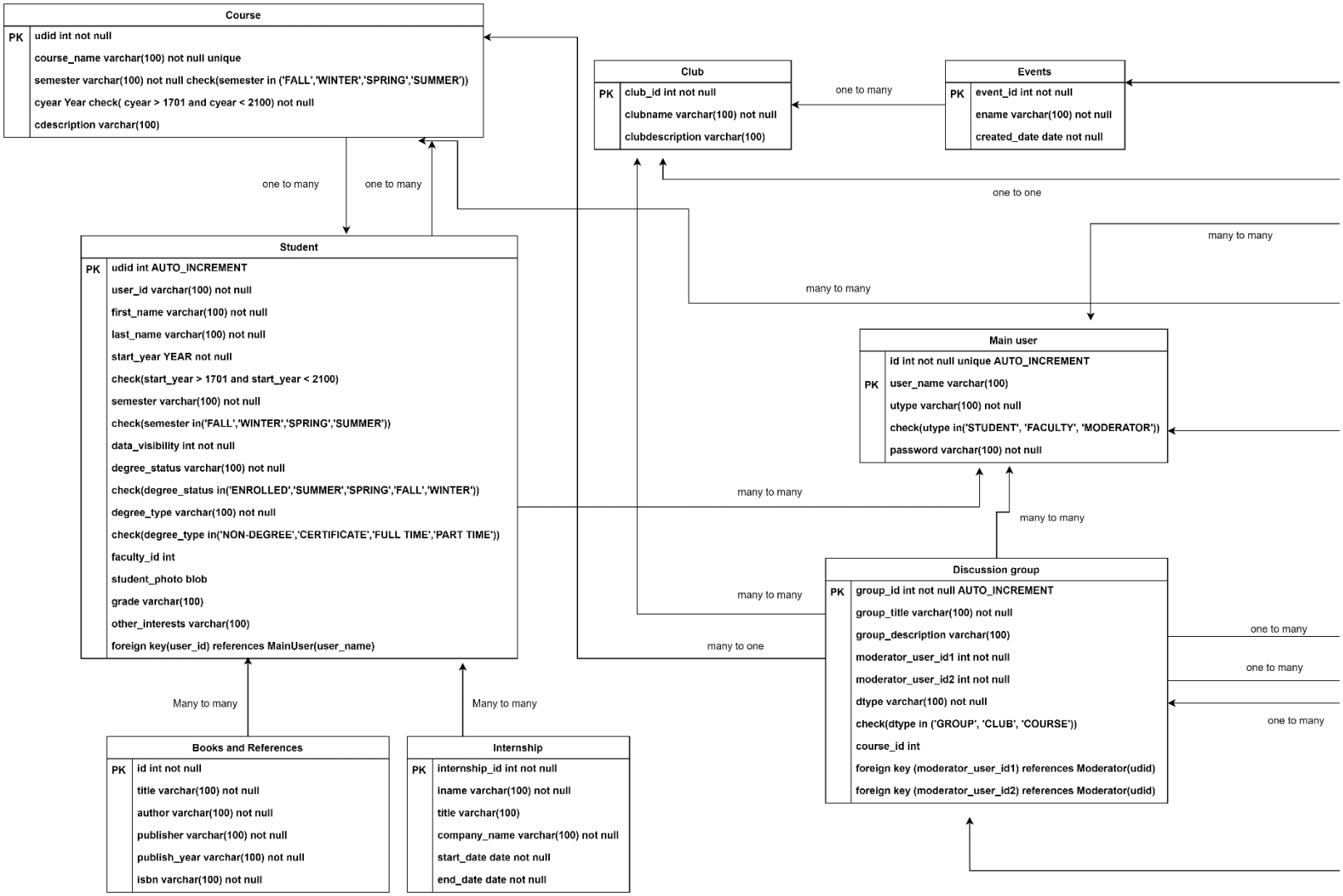
- Moderator – Is a registered user, identified for assigned discussion groups. There will be two moderators for a group, one is mandatory. There will also be a moderator, who is a site-wide administrator.

# Data Model

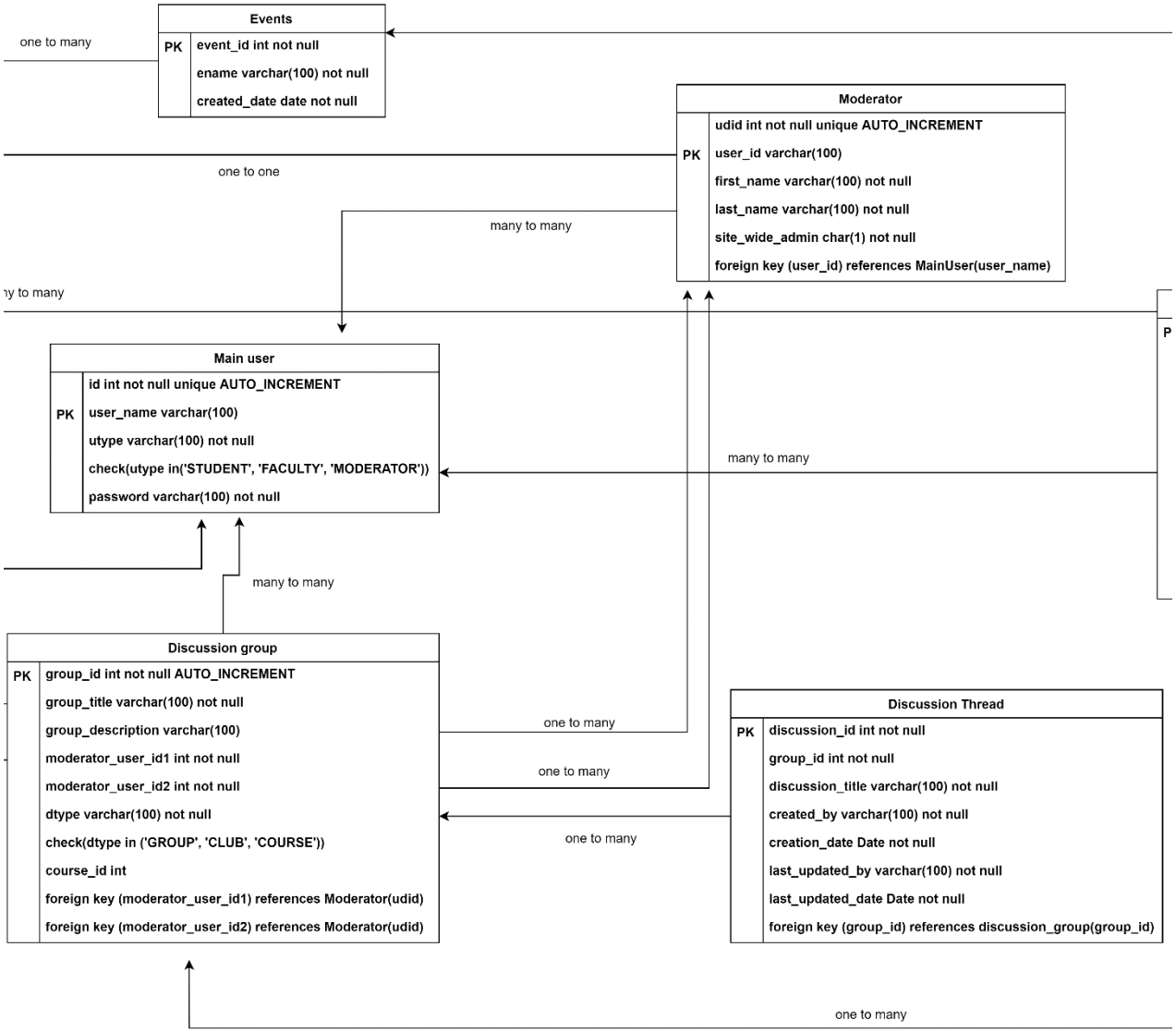
***(SEE THE NEXT PAGE)***

# Data Model

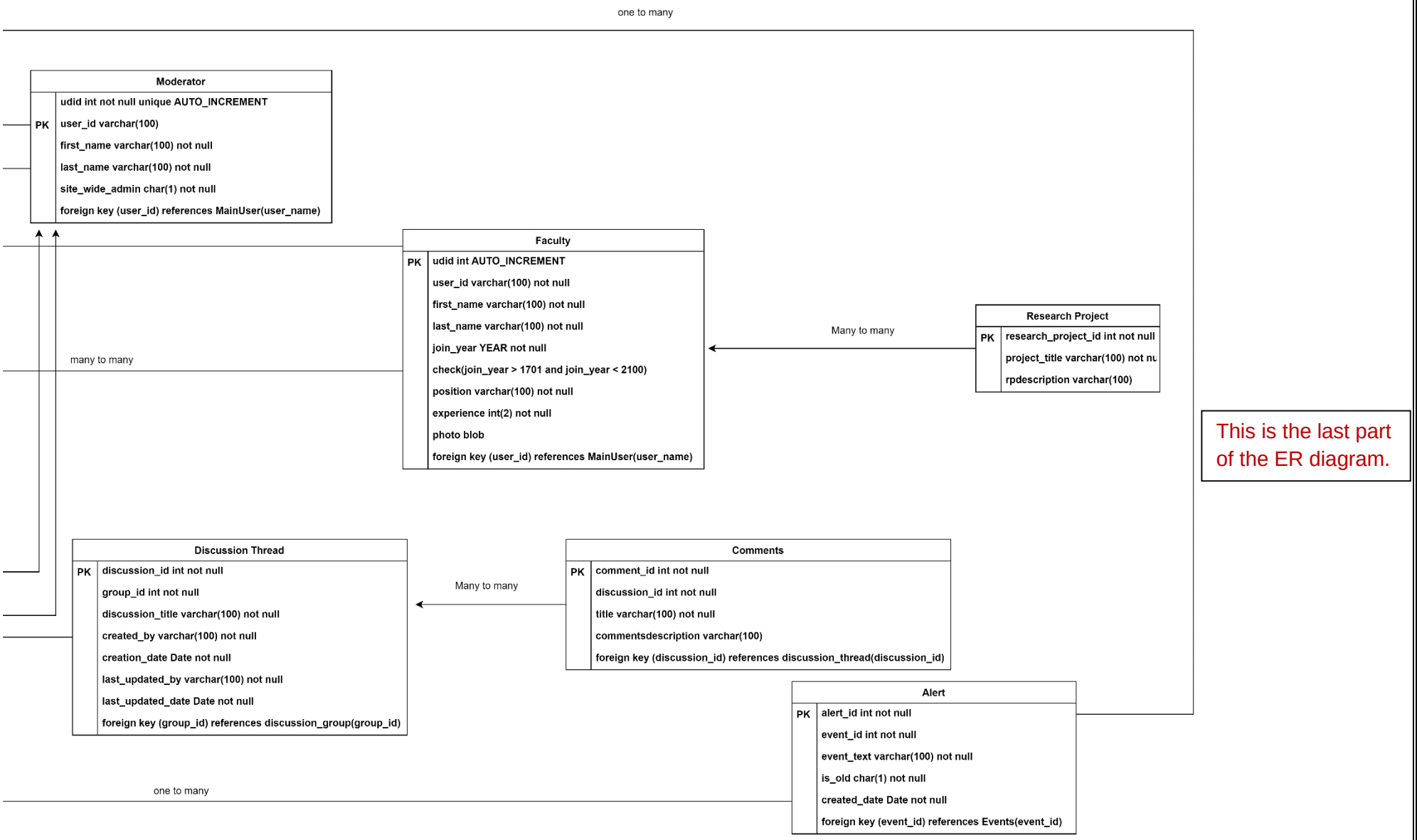
<The ERD developed in draw.io>




This is the first part of the ER diagram.

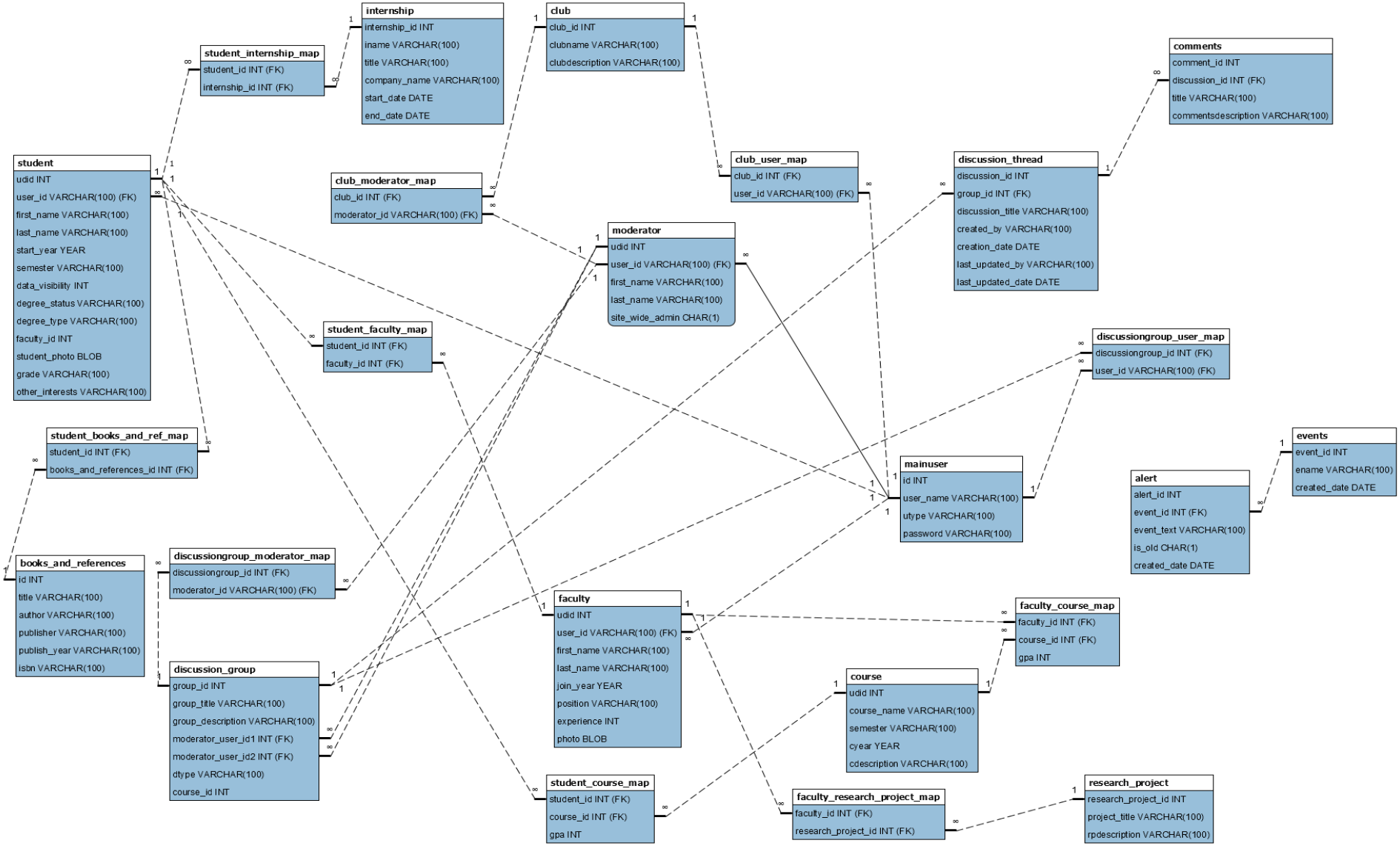



This is the middle part of the ER diagram.


one to many
Moderator
udid int not null unique AUTO_INCREMENT
PK
user_id varchar(100)
first_name varchar(100) not null
last_name varchar(100) not null
site_wide_admin char(1) not null
foreign key (user_id) references MainUser(user_name)
Faculty
PK
udid int AUTO_INCREMENT
user_id varchar(100) not null
first_name varchar(100) not null
last_name varchar(100) not null
join_year YEAR not null
check(join_year > 1701 and join_year < 2100)
position varchar(100) not null
experience int(2) not null
photo blob
foreign key (user_id) references MainUser(user_name)
Many to many
Research Project
PK
research_project_id int not null
project_title varchar(100) not nu
rpdescription varchar(100)
many to many
This is the last part of the ER diagram.
Discussion Thread
PK
discussion_id int not null
group_id int not null
discussion_title varchar(100) not null
created_by varchar(100) not null
creation_date Date not null
last_updated_by varchar(100) not null
last_updated_date Date not null
foreign key (group_id) references discussion_group(group_id)
Many to many
Comments
PK
comment_id int not null
discussion_id int not null
title varchar(100) not null
commentsdescription varchar(100)
foreign key (discussion_id) references discussion_thread(discussion_id)
Alert
PK
alert_id int not null
event_id int not null
event_text varchar(100) not null
is_old char(1) not null
created_date Date not null
foreign key (event_id) references Events(event_id)
one to many

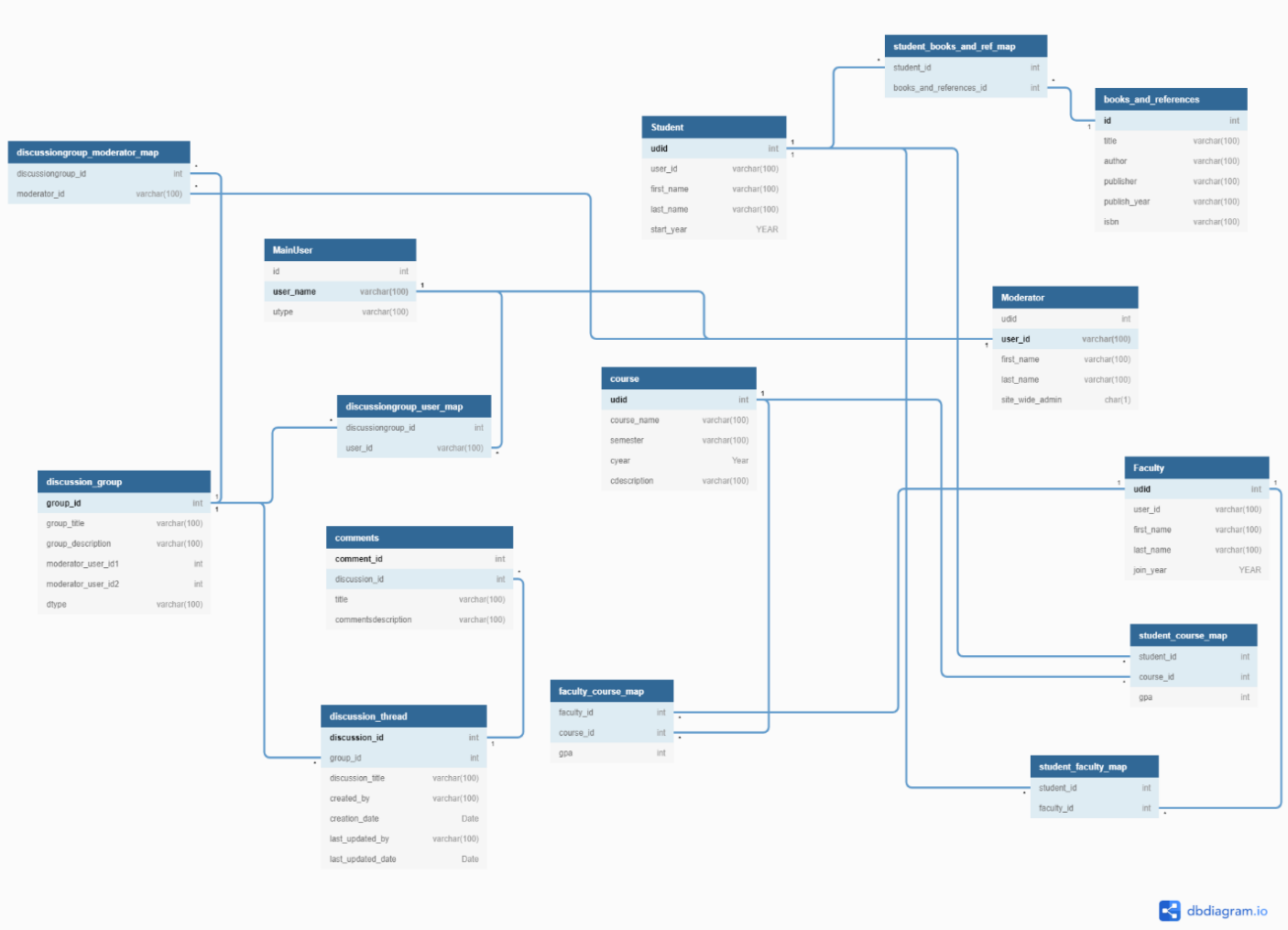
# Data Model

<The ERD developed in MySQL Workbench>



# Schema Diagram

**NOTE: -** The schema below is the one followed as the other tables such as Alert, Events, etc. are considered for triggers and applicable once the frontend part is complete. <The schema is developed in dbdiagram.io>



# FD and Normalization

## 1. Student Schema: -

*Student(MainUser, Name, Degree_Start_Date, Semester_GPA, Data_Visibility, Course, Faculty, Internship, Books and References)*
The set of functional dependencies that we require to hold on *Student* are:

- *Course -> Name, Description*
- *GPA->Grade*
- *Internship -> Name, companyName, Title, StartDate, EndDate*
- *Faculty -> Name, DOJ, Position, Experience, Research Project,*
- *Books And References -> Title, Author, Publisher, ISBN, Year of publishing*

The functional dependency
*User_id ->Login Name, Type, Password*
*Course_id -> Name, Description* holds but need to make course_id superkey.
*Internship_id -> Name, companyName, Title, StartDate, EndDate*
*Faculty_id -> Name, DOJ, Position, Experience, Research Project*
*(Note: This should be decomposed using BCNF, is addressed in the next section)*
*Books_and_references_id -> Title, Author, Publisher, ISBN, Year of publishing*

So Student is replaced by
*Student(User_id, Name, Degree_Start_Date, Semester,Grade ,Data_Visibility, Course_id, Faculty_id, Internship_id, Books_and_references_id)*

Thus the decomposition of *Student* results in five relation schemas -> Main*User, Course*, *Internship, Faculty and Books_and_references*.

## 2. Faculty Schema: -

*Faculty(MainUser, Course, Name, DOJ, Position, Experience, Research Project )*

*Course_id -> Name, Description* holds but need to make course_id superkey.
*ResearchProject_id -> Name, Description*
*User_id ->Login Name, Type, Password*

So the *Faculty* is replaced by,

*Faculty(User_id, Course_id, Name, DOJ, Position, Experience, ResearchProject_id )*

Thus the decomposition of *Faculty* results in three relation schemas -> *User, Course*, *Research_project_id.*

## 3. Discussion Group Schema: -

**Note: For simplicity of the project, Clubs and interest groups are the same.**

*Discussion Group(MainUser, Clubs, event, comments, courses, moderators)*

*User_id -> Login Name, Type, Password*
*Club_id -> Name, Description*
*Event_id->Event name, Event Date*
*Comments_id -> Description*
*Course_id -> Name, Description*
*Moderators ->First name, last name, sitewideadmin, user*

So *DiscussionGroup* is replaced by:

*Discussion Group(User_id, Club_id, event_id, comments_id, courses_id, moderators_id)*

Note that even after applying the BCNF, there is redundancy in data with regard to events for a club and comments for a discussion group and moderators. This redundancy will be addressed by the 3NF.

## Tables and Relations in 3NF: -

As given in the above section, all the relations are in BCNF, which makes them also in 3NF. The details of the Discussion Group schema is given below.

Consider the following schema of *Discussion Group:*

*Discussion Group(MainUser, Clubs, event, comments, courses, moderators)*

The *Discussion Group relation is dependent on the User* and the *moderator* is also dependent on the *user.*

Hence from 3NF, create two different relations *MainUser and Moderator.*

*User_id -> Login Name, Type, Password*
*Moderators -> user_id ,First name, last name, sitewideadmin*
*Discussion Group->User_id,Club_id, event_id,comments_id, ,courses_id, moderator1_id, moderator2_id*

# DDL

**NOTE: - All the Tables are implemented in MySQL RDBMS.**

## Main Tables: -

### Table MainUser: - (id, utype, user_name should not be null)

```
create table MainUser(
        id int not null unique AUTO_INCREMENT,
        user_name varchar(100) primary key,
        utype varchar(100) not null,
        check(utype in('STUDENT', 'FACULTY', 'MODERATOR')),
        password varchar(100) not null
);
```

### Table Student: - (udid, user_id should not be null)

```
create table Student(
        udid int AUTO_INCREMENT primary key,
        user_id varchar(100) not null,
        first_name varchar(100) not null,
        last_name varchar(100) not null,
        start_year YEAR not null,
        check(start_year > 1701 and start_year < 2100),
        semester varchar(100) not null,
        check(semester in('FALL','WINTER','SPRING','SUMMER')),
        data_visibility int not null,
        degree_status varchar(100) not null,
        check(degree_status in('ENROLLED','SUMMER','SPRING','FALL','WINTER')),
        degree_type varchar(100) not null,
        check(degree_type in('NON-DEGREE','CERTIFICATE','FULL TIME','PART TIME')),
        faculty_id int ,
        student_photo blob,
        grade varchar(100),
        other_interests varchar(100),
        foreign key(user_id) references MainUser(user_name)
);
```

### Table Faculty: - (udid, user_id should not be null)

```
create table Faculty(
        udid int AUTO_INCREMENT primary key,
        user_id varchar(100) not null ,
        first_name varchar(100) not null,
        last_name varchar(100) not null,
        join_year YEAR not null,
        check(join_year > 1701 and join_year < 2100),
        position varchar(100) not null,
        experience int(2) not null,
        photo blob,
        foreign key (user_id) references MainUser(user_name)
);
```

**Table Moderator: -** **(udid, user_id should not be null)**

```
create table Moderator(
        udid int not null unique AUTO_INCREMENT,
        user_id varchar(100) primary key,
        first_name varchar(100) not null,
        last_name varchar(100) not null,
        site_wide_admin char(1) not null,
        foreign key (user_id) references MainUser(user_name)
);
```

**Table discussion_group: -** **(group_id, moderator_user_id1, moderator_user_id2 should not be null)**

```
create table discussion_group(
        group_id int primary key not null AUTO_INCREMENT,
        group_title varchar(100) not null,
        group_description varchar(100),
        moderator_user_id1 int not null ,
        moderator_user_id2 int not null ,
        dtype varchar(100)  not null,
        check(dtype in ('GROUP', 'CLUB', 'COURSE')),
        course_id int,
        foreign key (moderator_user_id1) references Moderator(udid),
        foreign key (moderator_user_id2) references Moderator(udid)
);
```

**Table Course: -** **(udid should not be null)**

```
create table course(
        udid int not null primary key,
        course_name varchar(100) not null unique,
        semester varchar(100) not null check(semester in ('FALL','WINTER','SPRING','SUMMER')),
        cyear Year check( cyear > 1701 and cyear < 2100) not null,
        cdescription varchar(100)
);
```

**Table Internship: -** **(internship_id should not be null)**

```
create table internship(
        internship_id int primary key not null,
        iname varchar(100) not null,
        title varchar(100),
        company_name varchar(100) not null,
        start_date date not null,
        end_date date not null
);
```

**Table research_project: -** **(research_project_id should not be null)**

```
create table research_project(
        research_project_id int primary key not null,
        project_title varchar(100) not null,
        rpdescription varchar(100)
);
```

**Table club: - (club_id should not be null)**

```
create table club(
        club_id int primary key not null,
        clubname varchar(100) not null,
        clubdescription varchar(100)
);
```

**Table Events: - (event_id should not be null)**

```
create table Events(
        event_id int primary key not null,
        ename varchar(100) not null,
        created_date date not null
);
```

**Table discussion_thread: - (discussion_id, group_id should not be null)**

```
create table discussion_thread(
        discussion_id int primary key not null,
        group_id int not null ,
        discussion_title varchar(100) not null,
        created_by varchar(100) not null ,
        creation_date Date not null,
        last_updated_by varchar(100) not null ,
        last_updated_date Date not null,
        foreign key (group_id) references discussion_group(group_id)
);
```

**Table comments: - (comment_id, discussion_id should not be null)**

```
create table comments(
        comment_id int primary key not null,
        discussion_id int not null ,
        title varchar(100) not null,
        commentsdescription varchar(100),
        foreign key (discussion_id) references discussion_thread(discussion_id)
);
```

**Table alert: - (alert_id, event_id should not be null)**

```
create table alert(
        alert_id int primary key not null,
        event_id int not null,
        event_text varchar(100) not null,
        is_old char(1) not null,
        created_date Date not null,
        foreign key (event_id) references Events(event_id)
);
```

### Table books_and_references: - (id should not be null)

```
create table books_and_references(
        id int primary key not null,
        title varchar(100) not null,
        author varchar(100) not null,
        publisher varchar(100) not null,
        publish_year varchar(100) not null,
        isbn varchar(100) not null
);
```

## Mapping Tables: -

**Purpose**: These tables are created to avoid repetition of values in the main table and manage mainly many –to – many relationships. All the attributes are the primary keys of the respective table and none of them should be null.

### Table students_books_and_ref_map: -

```
create table student_books_and_ref_map(
        student_id int not null,
        books_and_references_id int not null,
        foreign key (student_id) references Student(udid),
        foreign key (books_and_references_id) references books_and_references(id)
);
```

### Table student_course_map: -

```
create table student_course_map(
        student_id int not null,
        course_id int not null,
        gpa int,
        foreign key (student_id) references Student(udid),
        foreign key (course_id) references course(udid)
);
```

### Table student_internship_map: -

```
create table student_internship_map(
        student_id int not null,
        internship_id int not null,
        foreign key (student_id) references Student(udid),
        foreign key (internship_id) references internship(internship_id)
);
```

### Table faculty_course_map: -

```
create table faculty_course_map(
        faculty_id int not null,
        course_id int not null,
        gpa int,
        foreign key (faculty_id) references Faculty(udid),
        foreign key (course_id) references course(udid)
);
```

### Table faculty_research_project_map: -

```
create table faculty_research_project_map(
        faculty_id int not null,
        research_project_id int not null,
        foreign key (faculty_id) references Faculty(udid),
        foreign key (research_project_id) references research_project(research_project_id)
);
```

### Table student_faculty_map: -

```
create table student_faculty_map(
        student_id int not null,
        faculty_id int not null,
        foreign key (faculty_id) references Faculty(udid),
        foreign key (student_id) references Student(udid)
);
```

### Table discussiongroup_user_map: -

```
create table discussiongroup_user_map(
        discussiongroup_id int not null,
        user_id varchar(100) not null,
        foreign key (discussiongroup_id) references discussion_group(group_id),
        foreign key (user_id) references MainUser(user_name)
);
```

### Table discussiongroup_moderator_map: -

```
create table discussiongroup_moderator_map(
        discussiongroup_id int not null,
        moderator_id varchar(100) not null,
        foreign key (discussiongroup_id) references discussion_group(group_id),
        foreign key (moderator_id) references Moderator(user_id)
);
```

### Table club_user_map: -

```
create table club_user_map(
        club_id int not null,
        user_id varchar(100) not null,
        foreign key (club_id) references club(club_id),
        foreign key (user_id) references MainUser(user_name)
);
```

### Table club_moderator_map: -

```
create table club_moderator_map(
        club_id int not null,
        moderator_id varchar(100) not null,
        foreign key (club_id) references club(club_id),
        foreign key (moderator_id) references Moderator(user_id)
);
```

# Triggers

**Trigger to update the ‘STUDENT’ table: -**

```
########TRIGGERS FOR TABLE `STUDENT`########
CREATE TABLE `audit` (
	`id` int unsigned NOT NULL AUTO_INCREMENT,
	`std_id` int NOT NULL,
	`changetype` enum('NEW','EDIT','DELETE') NOT NULL,
	`changetime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
	CURRENT_TIMESTAMP,
	PRIMARY KEY (`id`),
	KEY `ix_std_id` (`std_id`),
	KEY `ix_changetype` (`changetype`),
	KEY `ix_changetime` (`changetime`),
	CONSTRAINT `FK_audit_std_id` FOREIGN KEY (`std_id`) REFERENCES `student` (`udid`)
	ON DELETE CASCADE ON UPDATE CASCADE
);
//TRIGGER WHEN THERE IS AN ENTRY IN THE ‘STUDENT’ TABLE
DELIMITER $$
CREATE
	TRIGGER `std_after_insert` AFTER INSERT
	ON `student`
	FOR EACH ROW BEGIN

		IF NEW.user_id THEN
			SET @changetype = 'DELETE';
		ELSE
			SET @changetype = 'NEW';
		END IF;

		INSERT INTO audit (std_id, changetype) VALUES (NEW.udid, @changetype);

   END$$
DELIMITER ;
//TRIGGER WHEN THERE IS AN UPDATE IN THE ‘STUDENT’ TABLE
DELIMITER $$
CREATE
	TRIGGER `std_after_update` AFTER UPDATE
	ON `student`
	FOR EACH ROW BEGIN

		IF NEW.user_id THEN
			SET @changetype = 'DELETE';
		ELSE
			SET @changetype = 'EDIT';
		END IF;

		INSERT INTO audit (std_id, changetype) VALUES (NEW.udid, @changetype);

   END$$
DELIMITER ;
```

## Test Input and output: -

INSERT INTO MAINUSER(id,USER_NAME,UTYPE,PASSWORD) VALUES(26,'test', 'STUDENT', 'vinayak'); **To avoid referential integrity.**

INSERT INTO STUDENT VALUES(13,'test','test','testtest',2019,'FALL',1,'ENROLLED','NON-DEGREE',5,NULL,'A',NULL);

select * from audit;

Result Grid | Filter Rows: | Edit:

| id | std_id | changetype | changetime |
|---|---|---|---|
| 1 | 13 | NEW | 2020-05-31 18:32:12 |
| NULL | NULL | NULL | NULL |

UPDATE STUDENT SET faculty_id=3 where user_id='test';

select * from audit;

Result Grid | Filter Rows: | Edit:

| id | std_id | changetype | changetime |
|---|---|---|---|
| 1 | 13 | NEW | 2020-05-31 18:32:12 |
| 2 | 13 | EDIT | 2020-05-31 18:32:47 |
| NULL | NULL | NULL | NULL |

update student set faculty_id=5 where udid=13;

select * from audit;

Result Grid | Filter Rows: | Edit:

| id | std_id | changetype | changetime |
|---|---|---|---|
| 1 | 13 | NEW | 2020-05-31 18:32:12 |
| 2 | 13 | EDIT | 2020-05-31 18:32:47 |
| 3 | 13 | EDIT | 2020-05-31 18:33:25 |
| NULL | NULL | NULL | NULL |

# SQL Queries

**NOTE: - ALL Queries are implemented in MySQL RDBMS.**

**Display the most recently discussions/comments from a specific interest group/club/course.**

```
#Query-1:
#Display the most recently discussions/comments
#from a specific interest group/club/course.

Create VIEW recent_discussions AS
select COUNT(*), dg.dtype,dg.group_title, dt.last_updated_date
from discussion_group as dg, discussion_thread as dt
where dg.group_id=dt.group_id  group by dt.last_updated_date;
#dg.dtype='GROUP' or  dg.dtype='CLUB'or dg.dtype='COURSE'and

select * from recent_discussions;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| COUNT(*) | dtype | group_title | last_updated_date |
|---|---|---|---|
| 12 | CLUB | FOCAL Programming Club | 2019-06-21 |
| 10 | GROUP | The Music of Donovan Interest Group | 2019-05-26 |
| 12 | CLUB | C Programming Club | 2019-05-27 |
| 9 | CLUB | Electron Microscopy Club | 2019-05-28 |
| 7 | COURSE | The Music of Donovan Course Group | 2019-05-29 |

**Display the most recently entered discussions/comments from all the interest group/club/course that a student has registered to.**

```
#Query-2:
#Display the most recently entered discussions/comments
#from all the interest group/club/course that a student has registered to.

create view recent_discussion_created as
select COUNT(*), dg.dtype,dg.group_title, dt.creation_date
from discussion_group as dg, discussion_thread as dt
where dg.group_id=dt.group_id  group by dt.creation_date;
#HAVING COUNT(*)<=10

select * from recent_discussion_created;
```

| COUNT(*) | dtype | group_title | creation_date |
|---|---|---|---|
| 19 | CLUB | FOCAL Programming Club | 2019-05-21 |
| 10 | GROUP | The Music of Donovan Interest Group | 2019-05-26 |
| 8 | CLUB | C Programming Club | 2019-05-28 |
| 6 | CLUB | The Music of Donovan Club | 2019-05-29 |
| 7 | COURSE | The Music of Donovan Course Group | 2019-05-25 |

**Display the list of all moderators, the group/club/course that they moderate and are members of.**

```
#Query-3:
#Display the list of all moderators, the group/club/course
#that they moderate and are members of.

select * from moderator;
-- 1st set of Moderators and the groups the are member of.
create view moderator_set_1 as
select m.udid, m.first_name, dg.group_title
from `moderator` as m
right join `discussion_group` as dg
on dg.moderator_user_id1=m.udid
where m.user_id IN (select m.user_id
                    from `moderator` as m, `mainuser` as mu
                    where m.user_id=mu.user_name) order by m.udid;
```

734 20:03:50 select * from moderator_set_1 LIMIT 0, 2000

| udid | first_name | group_title |
|---|---|---|
| 1 | surya | C Programming Interest Group |
| 1 | surya | The Music of Donovan Interest Group |
| 1 | surya | Electron Microscopy Interest Group |
| 1 | surya | International Finance Interest Group |
| 1 | surya | Greek Tragedy Interest Group |
| 1 | surya | Greek Tragedy Interest Group |
| 1 | surya | Virology Interest Group |
| 1 | surya | Compiler Design Interest Group |
| 1 | surya | Geology Interest Group |

……

……..

……..

……..

150 row(s) returned

| | | |
|---|---|---|
| 5 | srinivas | Tort Law Course Group |
| 5 | srinivas | Corporate Law Course Group |
| 5 | srinivas | Video Gaming Course Group |
| 5 | srinivas | World History Course Group |
| 5 | srinivas | Bankruptcy Course Group |
| 5 | srinivas | Organic Chemistry Course Group |
| 5 | srinivas | Existentialism Course Group |

```
#2nd set of Moderatorsand the groups the are member of.
create view moderator_set_2 as
select m.udid, m.first_name, dg.group_title
from `moderator` as m
right join `discussion_group` as dg
on dg.moderator_user_id2=m.udid
where m.user_id IN (select m.user_id
                    from `moderator` as m, `mainuser` as mu
                    where m.user_id=mu.user_name) order by m.udid;

select * from moderator_set_1;
select * from moderator_set_2;
```

735 20:07:35 select * from moderator_set_2 LIMIT 0, 2000

| udid | first_name | group_title |
|---|---|---|
| 1 | surya | Calculus Interest Group |
| 1 | surya | Environmental Law Interest Group |
| 1 | surya | The Beatles Interest Group |
| 1 | surya | Marine Mammals Interest Group |
| 1 | surya | Game Programming Interest Group |
| 1 | surya | Shakespeare Interest Group |
| 1 | surya | World History Interest Group |
| 1 | surya | Tort Law Interest Group |

.....

.....

.....

.....

.....

150 row(s) returned

| | | |
|---|---|---|
| 5 | srinivas | Electricity and Magnetism Course ... |
| 5 | srinivas | Elastic Structures Course Group |
| 5 | srinivas | Transaction Processing Course Gr... |
| 5 | srinivas | Computational Biology Course Group |
| 5 | srinivas | Cost Accounting Course Group |
| 5 | srinivas | Journalism Course Group |
| 5 | srinivas | Geology Course Group |

**Find the most commented on group/club/course.**

```
#Query-4:
#Find the most commented on group/club/course.

create view most_comments as
select COUNT(*),  dg.dtype ,c.title,dt.discussion_title
from comments as c, discussion_group as dg, discussion_thread as dt
where c.discussion_id=dt.discussion_id
and dt.discussion_title IN (select dt.discussion_title
                            from discussion_thread as dt
                            where dt.group_id=dg.group_id) group by dt.discussion_title;

select * from most_comments;
```

| COUNT(*) | dtype | title | discussion_title |
|---|---|---|---|
| 12 | CLUB | C Programming Interest Group Comment 1 | C Programming Interest Group Thread 1 |
| 7 | GROUP | The Music of DonovanComment 1 | The Music of Donovan GroupThread 1 |
| 5 | CLUB | The Music of Donovan ClubComment 1 | The Music of Donovan Club Thread 1 |
| 3 | CLUB | Electron Microscopy Comment 1 | Electron Microscopy Club Thread 1 |
| 3 | COURSE | C Programming Course GroupComment 1 | C Programming Course GroupThread 1 |
| 3 | COURSE | The Music of Donovan Course GroupComment 1 | The Music of Donovan Course GroupThread 1 |

**Find whether anyone is interested in a particular book**

```
#Query-5:
#Find whether anyone is interested in a particular book

create view books_interested as
select COUNT(*), s.first_name, br.title
from `student` as s, `student_books_and_ref_map` as sbrm
left join `books_and_references` as br
on br.id=sbrm.books_and_references_id
where sbrm.student_id=s.udid group by br.title;

select * from books_interested;
```

| COUNT(*) | first_name | title |
|---|---|---|
| 2 | surya | world is flat 1 |
| 2 | surya | world is flat 2 |
| 4 | surya | world is flat 3 |
| 4 | surya | world is flat 4 |
| 3 | larry | world is flat 5 |
| 3 | srinivas | world is flat 6 |

**Display the past average GPA of all the courses taught by a faculty**

```
#Query-6:
#Display the past average GPA of all the courses taught by a faculty

-- change the faculty_id for different results

create view gpa_faculty as
select fcp.faculty_id, fcp.course_id, c.course_name, f.first_name,
c.cdescription, avg(fcp.gpa)
from faculty_course_map as fcp, course as c, faculty as f
where fcp.course_id = c.udid and fcp.faculty_id = f.udid
and fcp.faculty_id = 1 group by fcp.course_id ;

select * from gpa_faculty;
```

| faculty_id | course_id | course_name | first_name | cdescription | avg(fcp.gpa) |
|---|---|---|---|---|---|
| 1 | 1 | CS787 | vinayak | C Programming | 60.0000 |
| 1 | 2 | CS238 | vinayak | The Music of Donovan | 70.0000 |
| 1 | 3 | CS608 | vinayak | Electron Microscopy | 60.0000 |
| 1 | 4 | CS539 | vinayak | International Finance | 80.0000 |
| 1 | 5 | CS278 | vinayak | Greek Tragedy | 40.0000 |
| 1 | 6 | CS972 | vinayak | Greek Tragedy | 60.0000 |
| 1 | 7 | CS391 | vinayak | Virology | 90.0000 |
| 1 | 8 | CS814 | vinayak | Compiler Design | 35.0000 |
| 1 | 9 | CS272 | vinayak | Geology | 40.0000 |

**Display the past average GPA of all the courses taken by a student.**

```
#Query-7:
#Display the past average GPA of all the courses taken by a student.

-- change the student_id for different results

create view gpa_student as
select scp.student_id, scp.course_id, c.course_name, s.first_name,
c.cdescription, avg(scp.gpa)
from student_course_map as scp, course as c, student as s
where scp.course_id = c.udid and scp.student_id = s.udid
and scp.student_id = 1 group by scp.course_id ;

select * from gpa_student;
```

| student_id | course_id | course_name | first_name | cdescription | avg(scp.gpa) |
|---|---|---|---|---|---|
| 1 | 1 | CS787 | surya | C Programming | 80.0000 |
| 1 | 2 | CS238 | surya | The Music of Donovan | 80.0000 |
| 1 | 3 | CS608 | surya | Electron Microscopy | 90.0000 |
| 1 | 4 | CS539 | surya | International Finance | 80.0000 |
| 1 | 5 | CS278 | surya | Greek Tragedy | 70.0000 |
| 1 | 6 | CS972 | surya | Greek Tragedy | 80.0000 |
| 1 | 7 | CS391 | surya | Virology | 40.0000 |
| 1 | 8 | CS814 | surya | Compiler Design | 80.0000 |
| 1 | 9 | CS272 | surya | Geology | 60.0000 |
| 1 | 10 | CS612 | surya | Mobile Computing | 80.0000 |
| 1 | 11 | CS237 | surya | Surfing | 80.0000 |
| 1 | 12 | CS313 | surya | International Trade | 70.0000 |
| 1 | 13 | CS887 | surya | Latin | 80.0000 |
| 1 | 14 | CS328 | surya | Composition and Liter... | 80.0000 |
| 1 | 15 | CS984 | surya | Music of the 50s | 80.0000 |
| 1 | 16 | CS241 | surya | Biostatistics | 90.0000 |
| 1 | 17 | CS338 | surya | Graph Theory | 80.0000 |

**Display Average GPA's of all the courses taken by a Student and all Students: -**

```
#Query-8:
#Display Avereage GPA's of all the courses taken by a Student and all Students: -

/*
select scp.student_id, scp.course_id, c.course_name, c.cdescription,
    s.first_name, avg(scp.gpa)
    from student_course_map as scp, course as c, student as s
    where scp.course_id = c.udid and scp.student_id = s.udid
    group by scp.student_id;
*/
create view avg_gpa_students as
select scp.student_id, s.first_name, avg(scp.gpa)
    from student_course_map as scp, course as c, student as s
    where scp.course_id = c.udid and scp.student_id = s.udid
    group by scp.student_id;

select * from avg_gpa_students;
```

| student_id | first_name | avg(scp.gpa) |
|---|---|---|
| 1 | surya | 76.4706 |
| 2 | harry | 73.0769 |
| 3 | larry | 62.2222 |
| 4 | benson | 57.7778 |
| 5 | srinivas | 68.1818 |
| 7 | shakti | 73.6364 |

**Display Average GPA's of all the courses taught by a Faculty and by all Faculties: -**

```
#Query-9:
#Display Avereage GPA's of all the courses taught by a Faculty and by all Faculties: -

create view avg_gpa_faculty_each_course as
select fcp.faculty_id, fcp.course_id, c.course_name, c.cdescription,
    f.first_name, avg(fcp.gpa)
    from faculty_course_map as fcp, course as c, faculty as f
    where fcp.course_id = c.udid and fcp.faculty_id = f.udid
    group by fcp.course_id;

select * from avg_gpa_faculty_each_course;
```

| | | |
|---|---|---|
| 741 | 20:16:57 | select * from avg_gpa_faculty_each_course LIMIT 0, 2000 |

| faculty_id | course_id | course_name | cdescription | first_name | avg(fcp.gpa) |
|---|---|---|---|---|---|
| 1 | 1 | CS787 | C Programming | vinayak | 60.0000 |
| 1 | 2 | CS238 | The Music of Donovan | vinayak | 70.0000 |
| 1 | 3 | CS608 | Electron Microscopy | vinayak | 60.0000 |
| 1 | 4 | CS539 | International Finance | vinayak | 80.0000 |
| 1 | 5 | CS278 | Greek Tragedy | vinayak | 40.0000 |
| 1 | 6 | CS972 | Greek Tragedy | vinayak | 60.0000 |
| 1 | 7 | CS391 | Virology | vinayak | 70.0000 |
| 1 | 8 | CS814 | Compiler Design | vinayak | 52.5000 |
| 1 | 9 | CS272 | Geology | vinayak | 55.0000 |
| 2 | 10 | CS612 | Mobile Computing | saychatt | 40.0000 |
| 2 | 11 | CS237 | Surfing | saychatt | 70.0000 |
| 2 | 12 | CS313 | International Trade | saychatt | 60.0000 |

.....
.....
.....
.....
.....
.....

40 row(s) returned

| | | | | | |
|---|---|---|---|---|---|
| 5 | 38 | CS730 | Quantum Mechanics | eileen | 65.0000 |
| 5 | 39 | CS362 | Embedded Systems | eileen | 44.0000 |
| 5 | 40 | CS341 | Quantum Mechanics | eileen | 48.0000 |
| 5 | 41 | CS582 | Marine Mammals | eileen | 89.0000 |
| 5 | 42 | CS867 | The IBM 360 Architec... | eileen | 98.0000 |
| 5 | 43 | CS169 | Marine Mammals | eileen | 78.0000 |
| 5 | 44 | CS680 | Electricity and Magne... | eileen | 76.0000 |
| 7 | 29 | CS591 | Shakespeare | shashi | 87.0000 |

## ANOTHER WAY QUERY - 9

```
create view avg_gpa_faculty as
select fcp.faculty_id, f.first_name, avg(fcp.gpa)
    from faculty_course_map as fcp, course as c, faculty as f
    where fcp.course_id = c.udid and fcp.faculty_id = f.udid
    group by fcp.faculty_id;

select * from avg_gpa_faculty;
```

| faculty_id | first_name | avg(fcp.gpa) |
|---|---|---|
| 1 | vinayak | 59.4444 |
| 2 | saychatt | 58.1818 |
| 3 | anjali | 63.2222 |
| 4 | richard | 74.6667 |
| 5 | eileen | 62.9091 |
| 7 | shashi | 73.2727 |

## ANOTHER WAY QUERY - 9

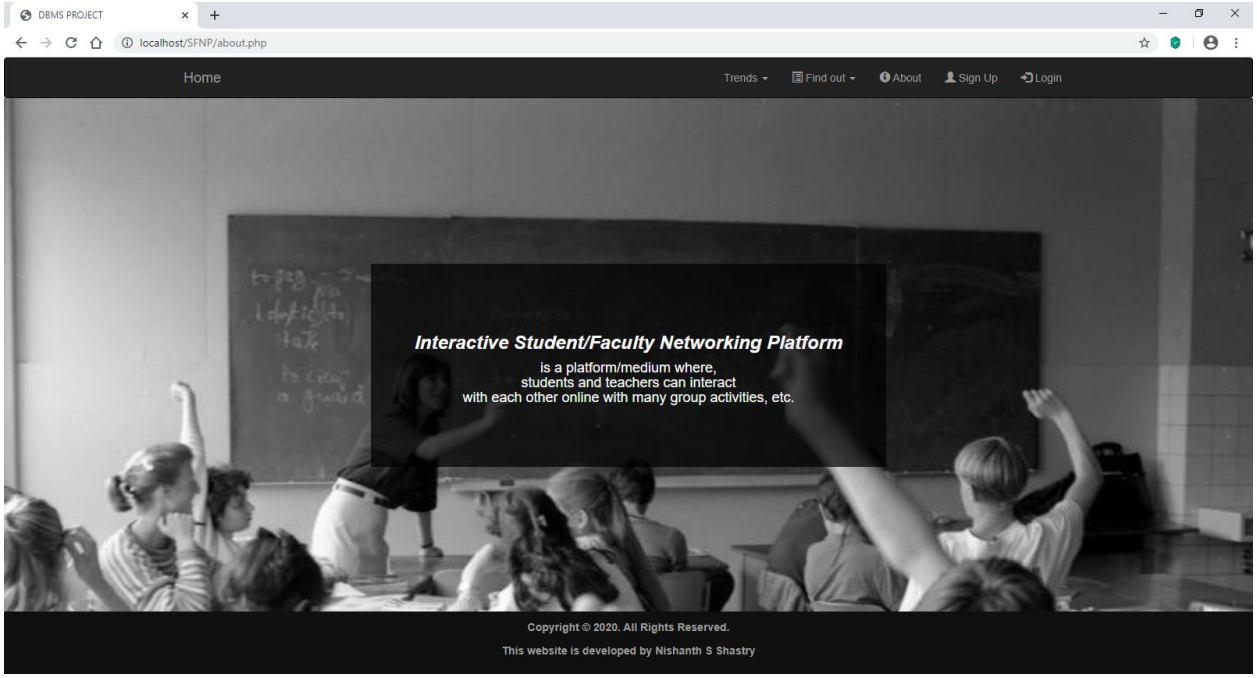
```
CREATE VIEW Faculty_Avg_GPA AS
SELECT fcp.faculty_id,f.first_name,AVG(fcp.gpa)
FROM faculty_course_map as fcp, course as c, faculty as f
WHERE fcp.course_id = c.udid and fcp.faculty_id = f.udid and
fcp.gpa > (SELECT AVG(fcp.gpa) FROM faculty_course_map as fcp)
group by fcp.faculty_id;

select * from Faculty_Avg_GPA;
```

| faculty_id | first_name | AVG(fcp.gpa) |
|---|---|---|
| 1 | vinayak | 80.0000 |
| 2 | saychatt | 74.0000 |
| 3 | anjali | 76.2000 |
| 4 | richard | 84.5000 |
| 5 | eileen | 81.4000 |
| 7 | shashi | 77.2222 |

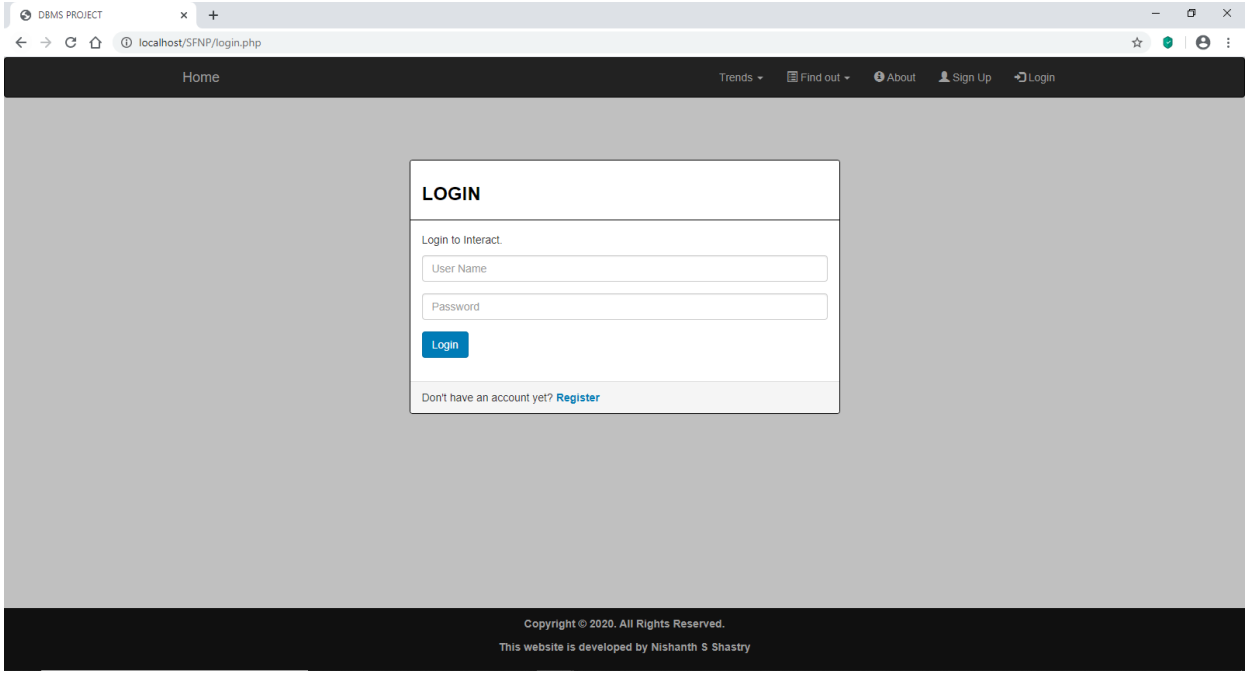
# UI with PHP (Screen Shots)

## MAIN PAGE: *(WHEN A USER VISITS THE WEBSITE)*

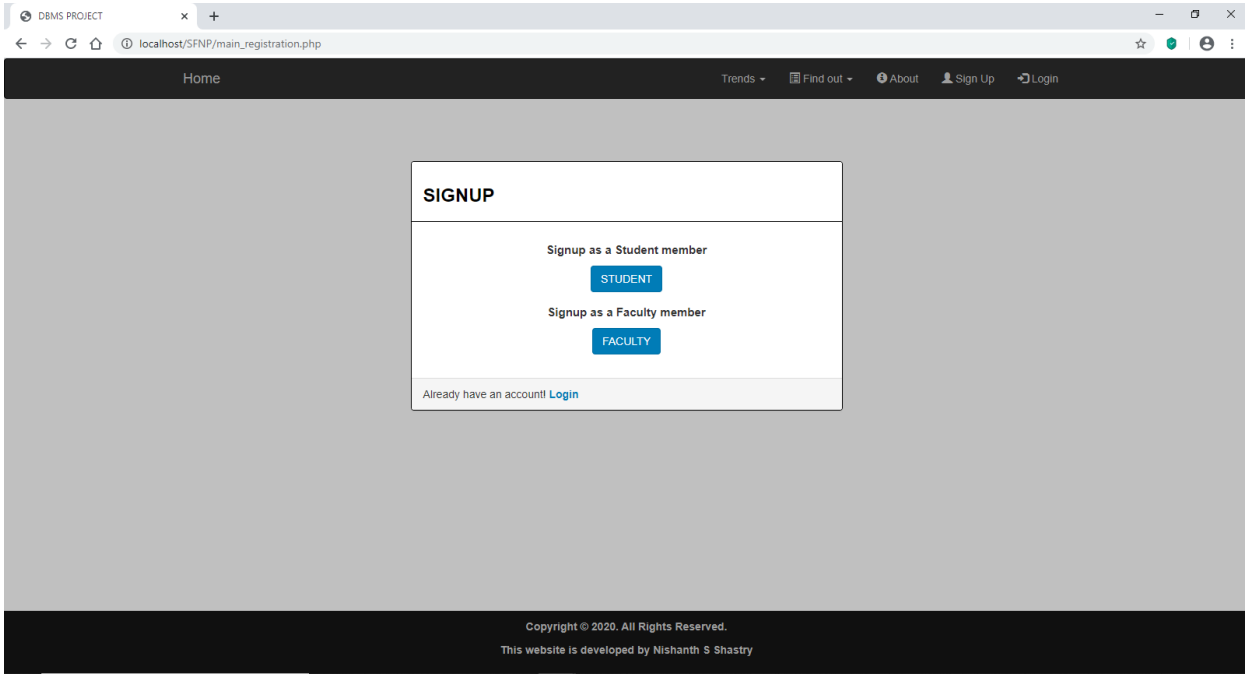

## ABOUT PAGE: *(COMMON TO ALL USERS)*



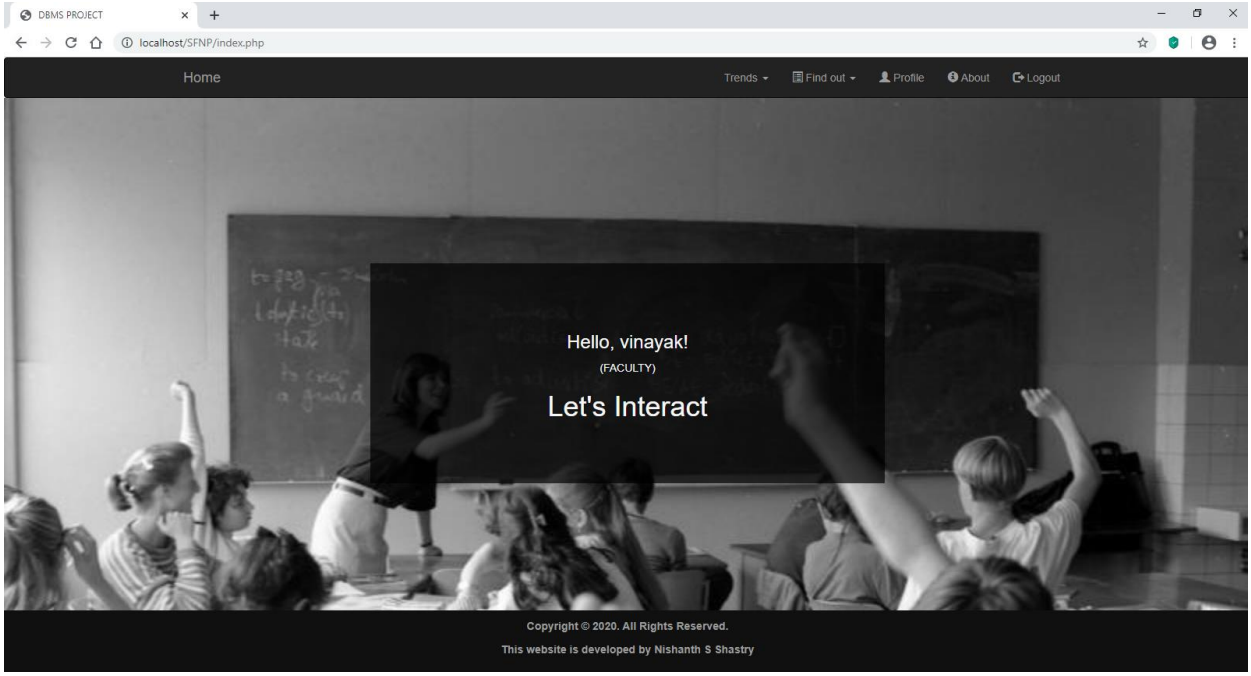
## LOGIN PAGE: *(COMMON TO ALL USERS BEFORE LOGIN)*



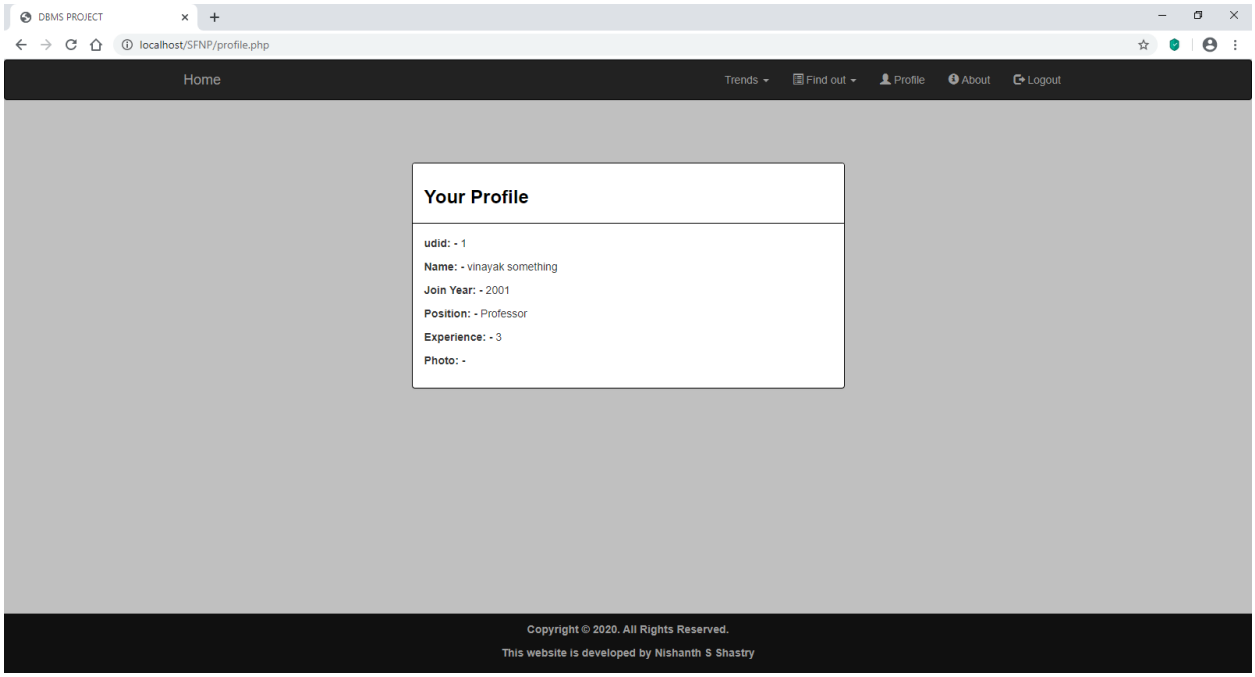

## SIGN UP PAGE: *(UNDER DEVELOPMENT - CAN BE DEVELOPED)*



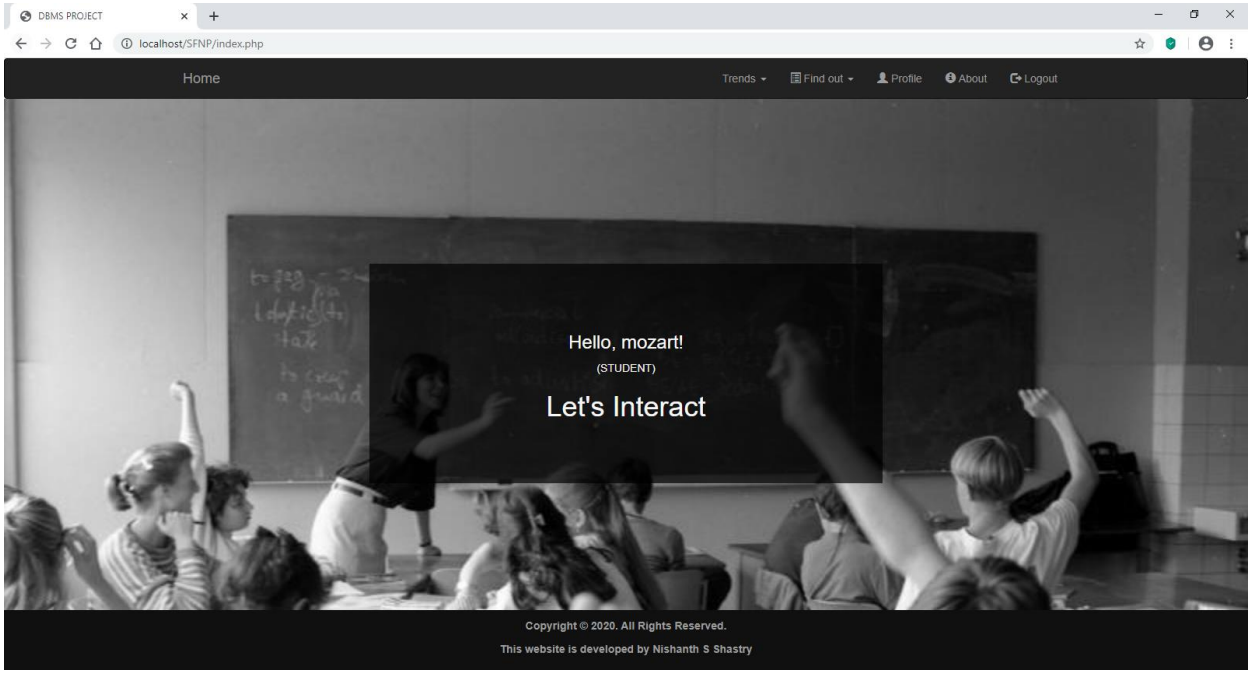
**MAIN PAGE:** ***(AFTER FACULTY LOGIN – NOTICE THE NAVBAR CHANGES)***



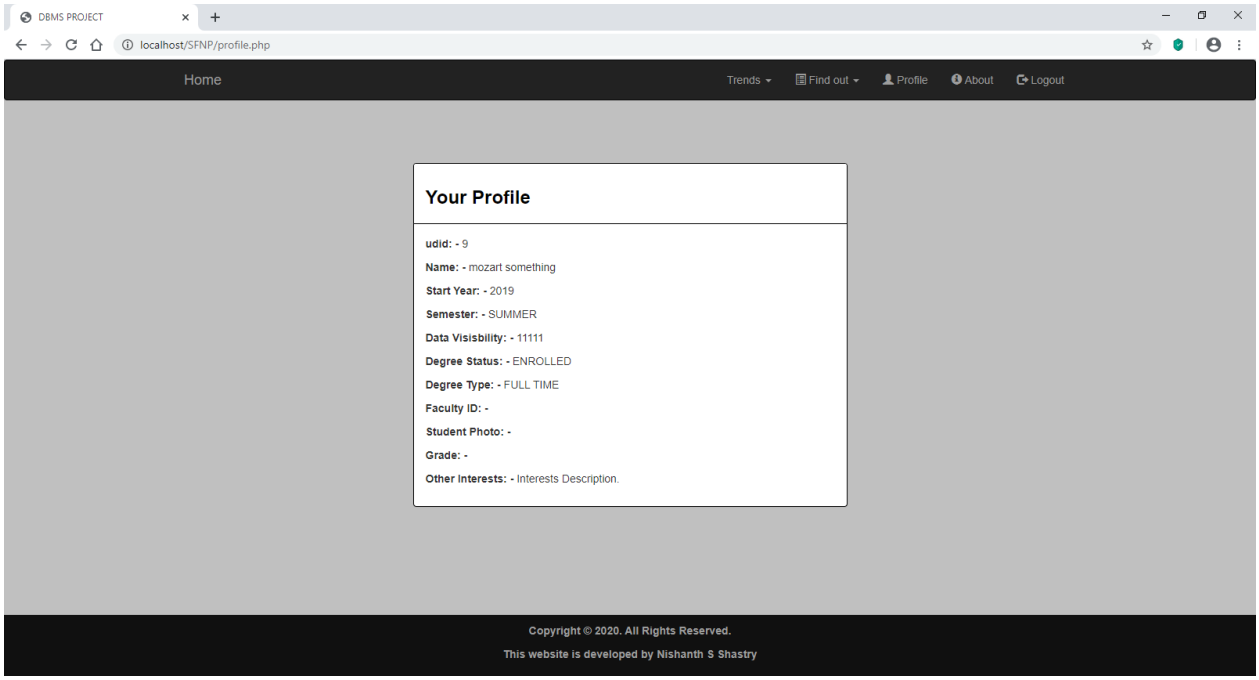

**PROFILE PAGE:** ***(FACULTY)***



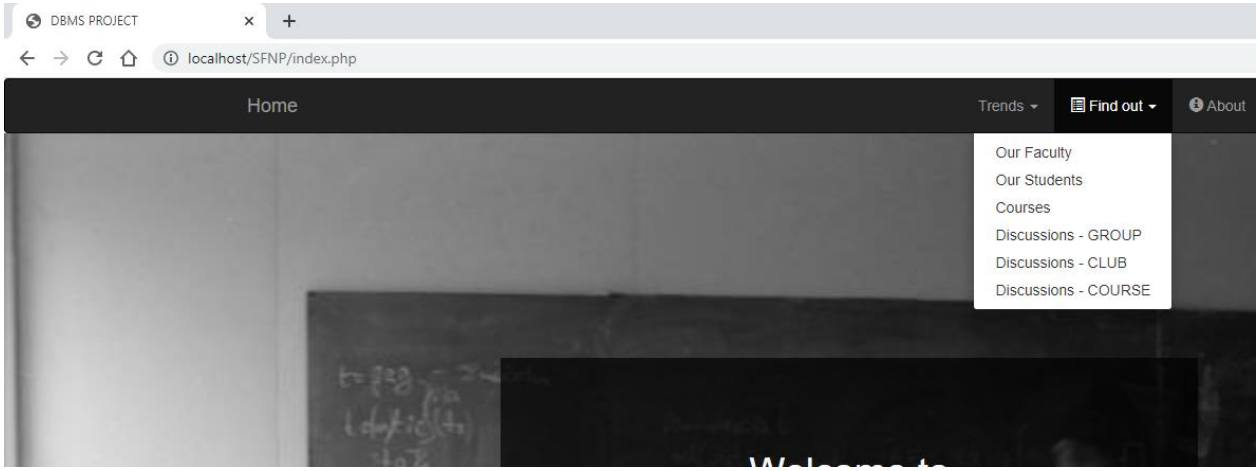
## MAIN PAGE: *(AFTER STUDENT LOGIN – NOTICE THE NAVBAR CHANGES)*

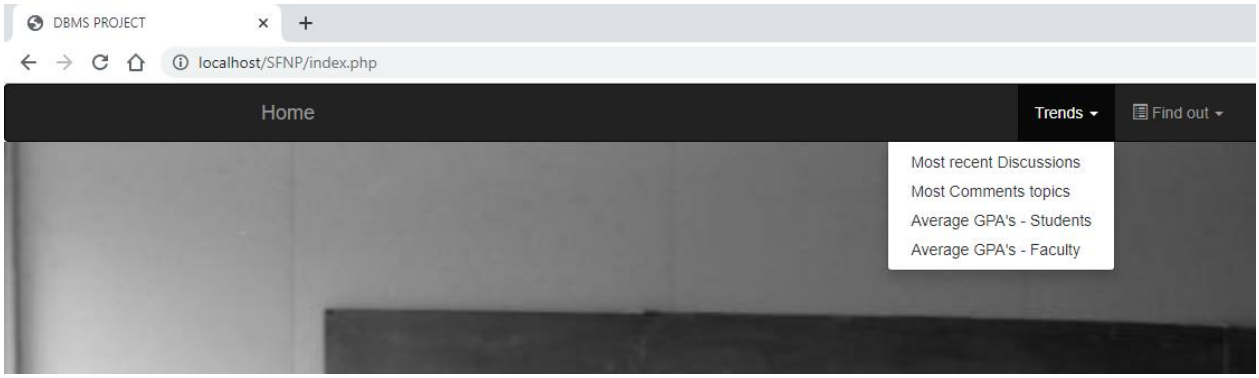



## PROFILE PAGE: *(STUDENT)*

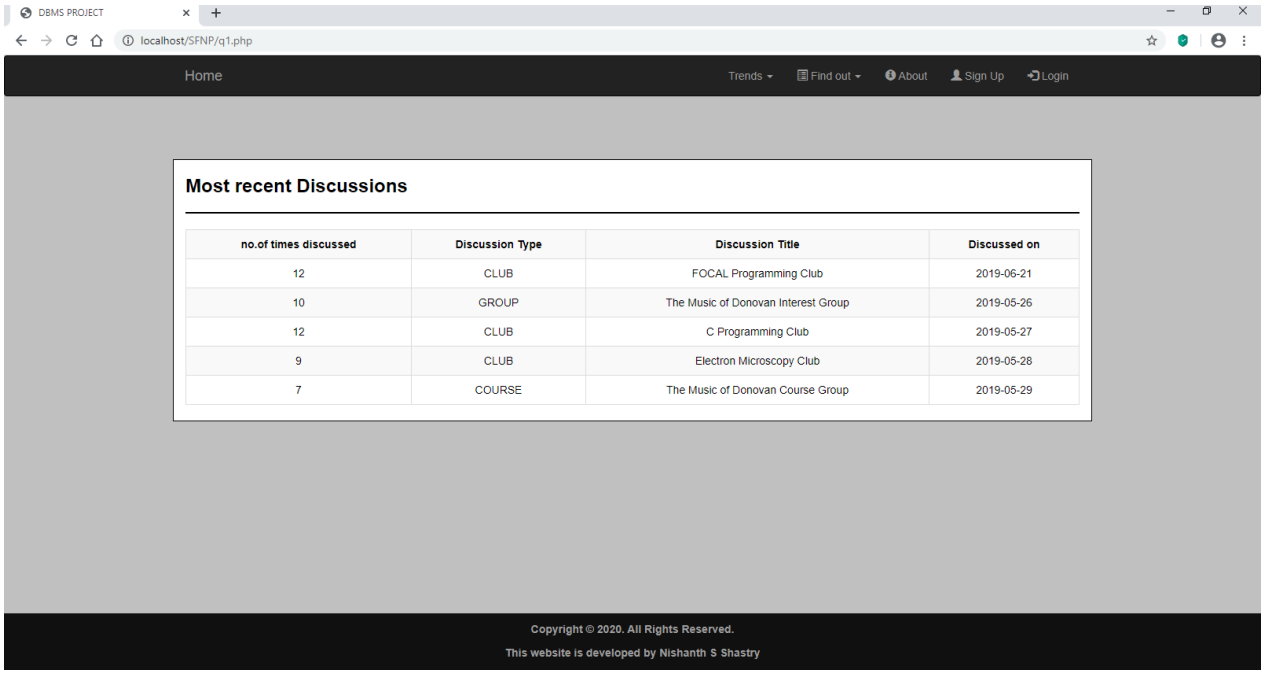


## COMMON WEB-PAGES TO ALL USERS:




## FEW SCREEN SHOTS OF WEBPAGES WITH SQL QUERIES:




| no.of times discussed | Discussion Type | Discussion Title | Discussed on |
|---|---|---|---|
| 12 | CLUB | FOCAL Programming Club | 2019-06-21 |
| 10 | GROUP | The Music of Donovan Interest Group | 2019-05-26 |
| 12 | CLUB | C Programming Club | 2019-05-27 |
| 9 | CLUB | Electron Microscopy Club | 2019-05-28 |
| 7 | COURSE | The Music of Donovan Course Group | 2019-05-29 |

DBMS PROJECT

localhost/SFNP/q4.php

Home | Trends | Find out | About | Sign Up | Login

## Most Comments topics

| no.of times commented | Group Type | Comment title | Discussion Title |
|---|---|---|---|
| 12 | CLUB | C Programming Interest Group Comment 1 | C Programming Interest Group Thread 1 |
| 7 | GROUP | The Music of Donovan Comment 1 | The Music of Donovan Group Thread 1 |
| 5 | CLUB | The Music of Donovan Club Comment 1 | The Music of Donovan Club Thread 1 |
| 3 | CLUB | Electron Microscopy Comment 1 | Electron Microscopy Club Thread 1 |
| 3 | COURSE | C Programming Course Group Comment 1 | C Programming Course Group Thread 1 |
| 3 | COURSE | The Music of Donovan Course Group Comment 1 | The Music of Donovan Course Group Thread 1 |



DBMS PROJECT

localhost/SFNP/q2.php

Home | Trends | Find out | About | Sign Up | Login

## Average GPA's - Students

| Student ID | Name of Students | Average GPA |
|---|---|---|
| 1 | surya | 76.4706 |
| 2 | harry | 73.0769 |
| 3 | larry | 62.2222 |
| 4 | benson | 57.7778 |
| 5 | srinivas | 68.1818 |
| 7 | shakti | 73.6364 |

# Conclusion

- This is the design proposal for the application in context and during the actual development there maybe changes to the same due some scenarios missed out from consideration. However, the overall concept would remain the same.
- Some of the relations have been broken down to 4NF due to multiple redundancy even after applying BCNF and 3NF,
    - Example: *student_books_and_ref_map*
- The mapping functions are designed mainly to implement many-to-many relationship among the main tables.
- Future Scope would include,
    - A fully functional website can be developed with functionalities like commenting on different discussions/topics, being part of various discussion groups irrespective of the user being a Faculty or Student.

      ***(For frontend development PHP is used to connect the UI and Backend, and HTML, CSS, JS for the smooth user-friendly interface.)***

    - The functionality of the tables such as Alert, Club, etc. to be part of the functionality in the website once the website is developed in a larger scale.
    - This system once developed can be used by any educational institutions.