Go To Your Study Room

- **Group Name:** ***Synergy***
- **Project Name:** ***Go to your study room***
- **Group Members:**

1. **Roopa Shankargouda Patil**
2. **Sadhana Seetamsetty**
3. **Revati Dhananjayan Lalitha**
4. **Avinash Yachamaneni**
5. **Vasudev Marla Balraj**

Project Report: 12/13/2015

Index

Project Definition:

This web application enables users to view study rooms online and book a study room. In Addition to this User will be able to send the agenda of the meeting to all the participants and send invites.

Scope of Work:

The User will be able to view the study rooms online and book a study room. User will login to the system and search for the rooms available. Once he or she finds of the intended room, user can book the room and invite the participants to the meeting. In addition to this, user will be able to send the agenda of the meeting to the participants.

On the other hand, the admin can have the overview of the booked room history. Admin will have the privileges to view and cancel the rooms booked.

Also includes the functionality of tracking the attendance of the participants

Business Rules:

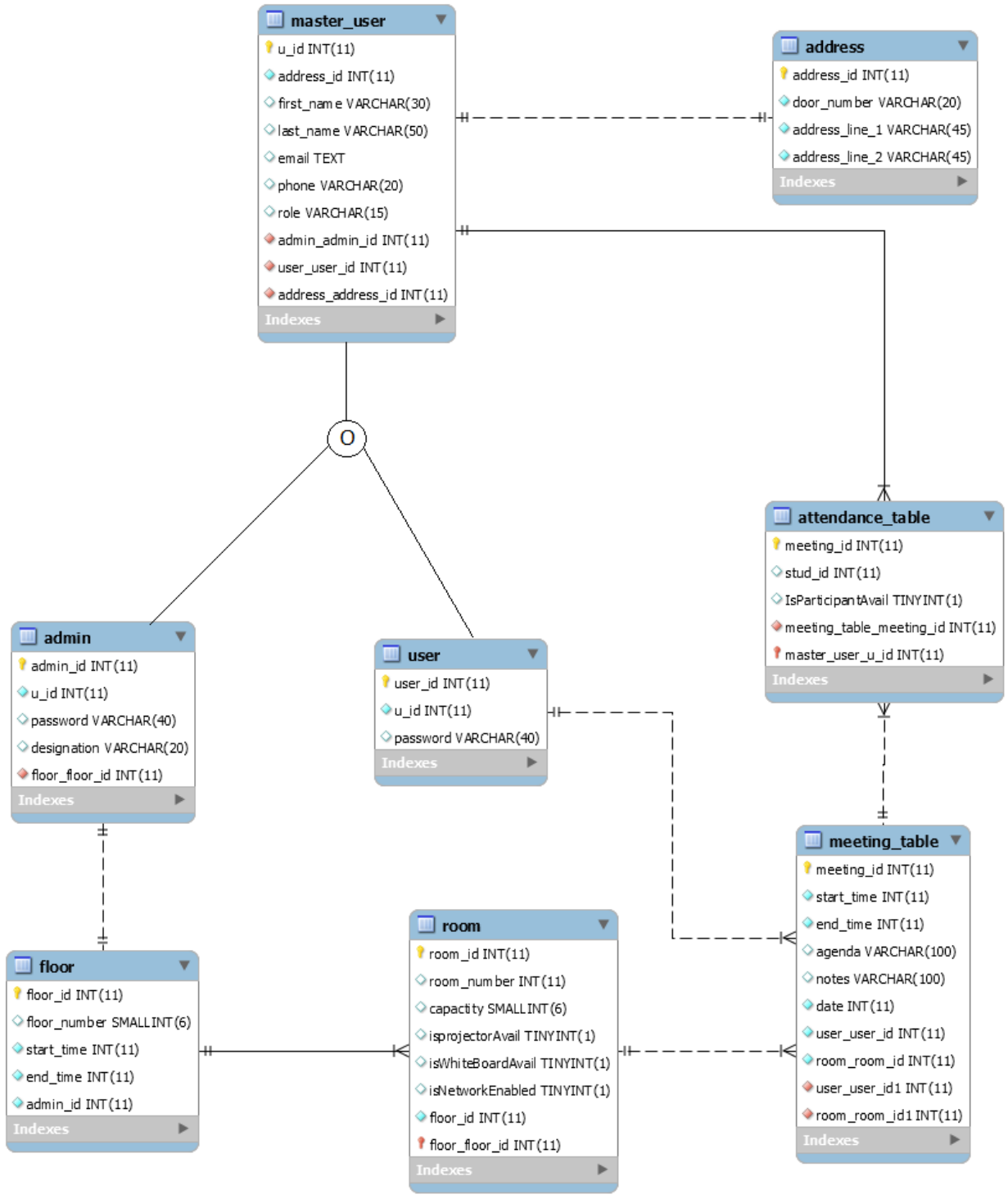
- User needs to be a registered user to access the application
- Master_User can be either user, admin or both
- Room and meeting_table share 1:N relationship, but meeting_table and room share one to one
- User can book room and invite participants
- Admin have access to monitor room booking and cancel room booking for floor level
- Master Admin have access to monitor room booking and cancel room booking in all the floors
- MASTER_USER.ROLE is multi value attribute

Constraints:

Software Limitations: System shall need Windows operating system with Net beans 8.02 for PHP development and XAMPP for Middleware and MYSQL installed.

Hardware Limitations: Each operator shall need a PC capable of running windows operating system with Net beans 8.02 for PHP development and XAMPP for Middleware and MYSQL installed.

Enhanced ER Diagram:



EER Diagram explained:

- Generalization/specialization

  MASTER_USER table is specialization of ADMIN and USER table.

- At least one many to many relationship resolved

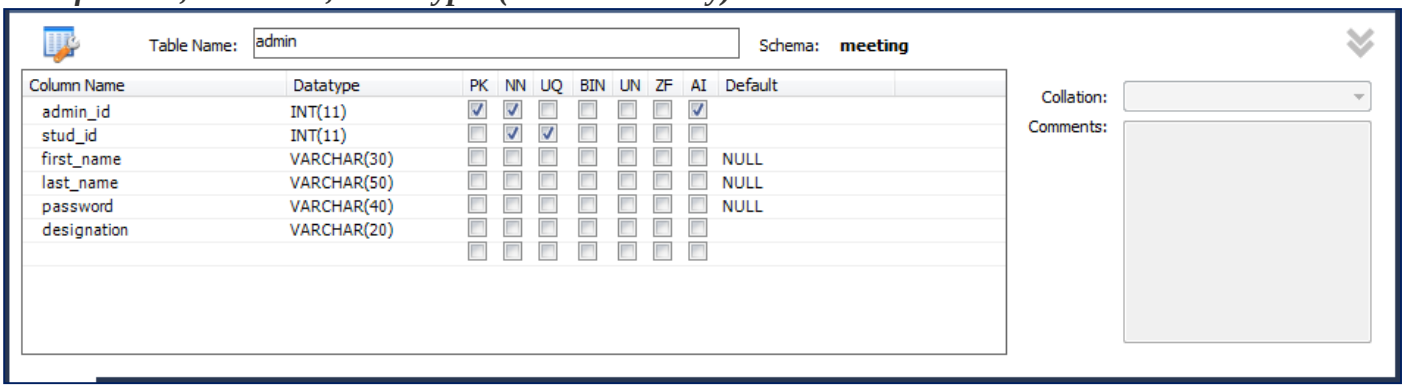
  ROOM and USER share M:M relationship. It is established using MEETING_TABLE.

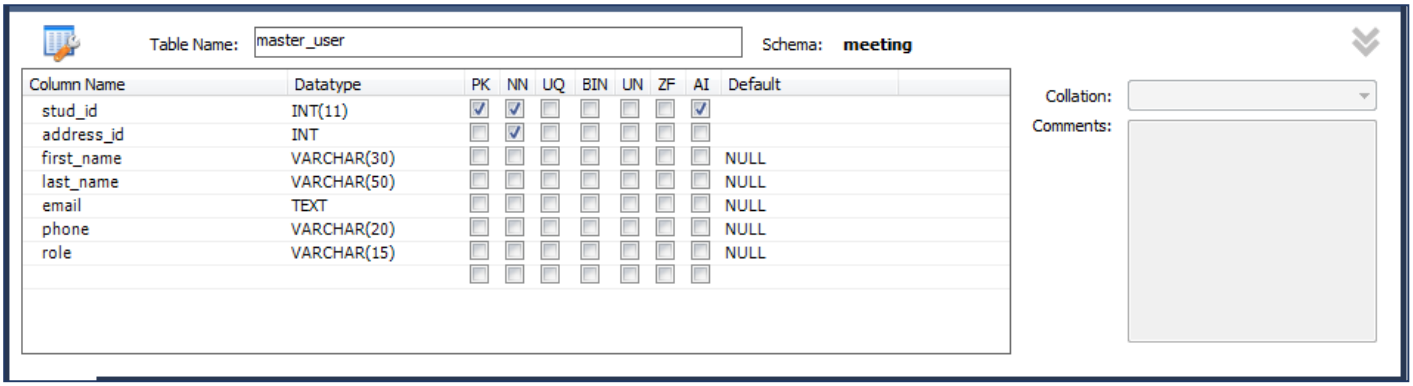
- Login table (if appropriate) – a suggested implementation will be given to you

  USER and ADMIN table stores password that helps in authenticating corresponding users.

TABLES:

List of Tables, Attributes, Data Types (Data Dictionary):





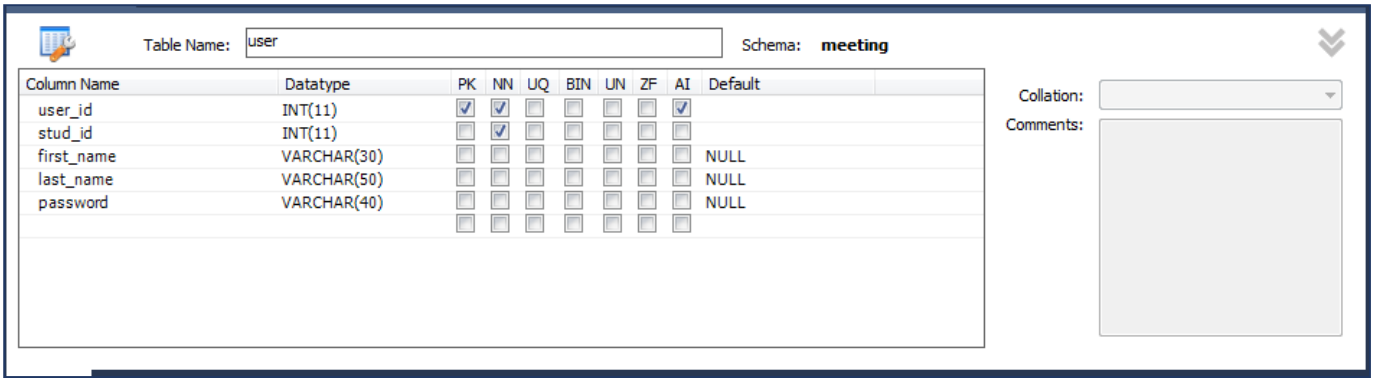


Table Name: user Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
user_id	INT(11)	✓	✓					✓	
stud_id	INT(11)		✓						
first_name	VARCHAR(30)								NULL
last_name	VARCHAR(50)								NULL
password	VARCHAR(40)								NULL

Collation:
Comments:

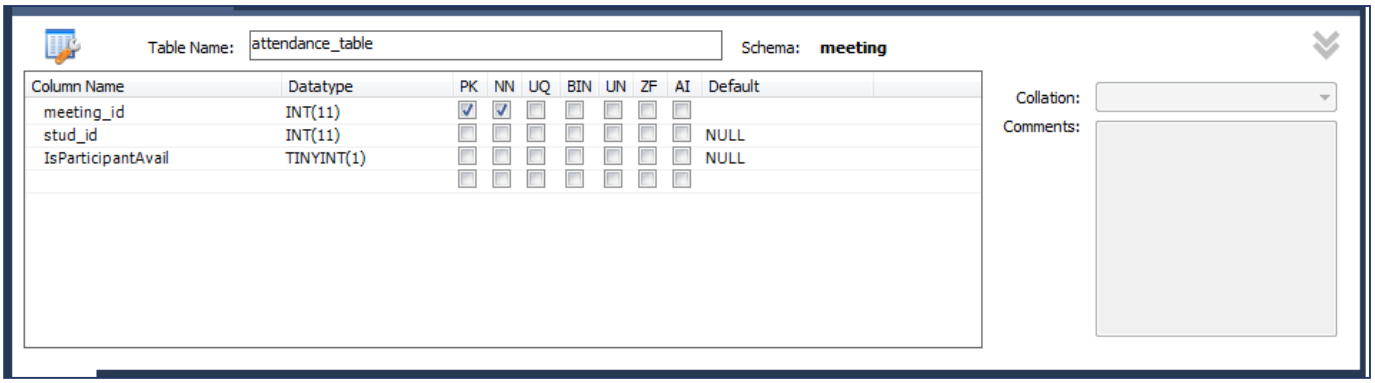


Table Name: attendance_table Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
meeting_id	INT(11)	✓	✓						
stud_id	INT(11)								NULL
IsParticipantAvail	TINYINT(1)								NULL

Collation:
Comments:

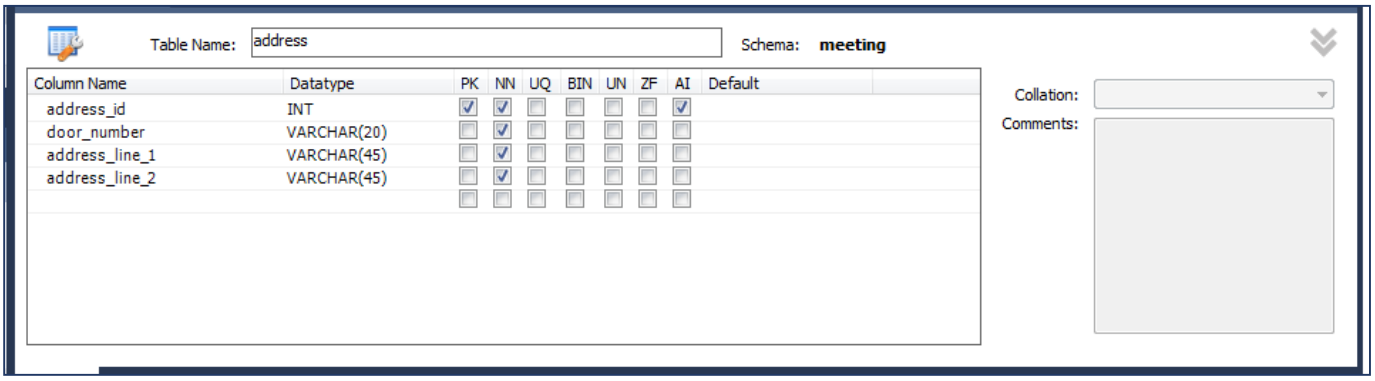


Table Name: address Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
address_id	INT	✓	✓					✓	
door_number	VARCHAR(20)		✓						
address_line_1	VARCHAR(45)		✓						
address_line_2	VARCHAR(45)		✓						

Collation:
Comments:

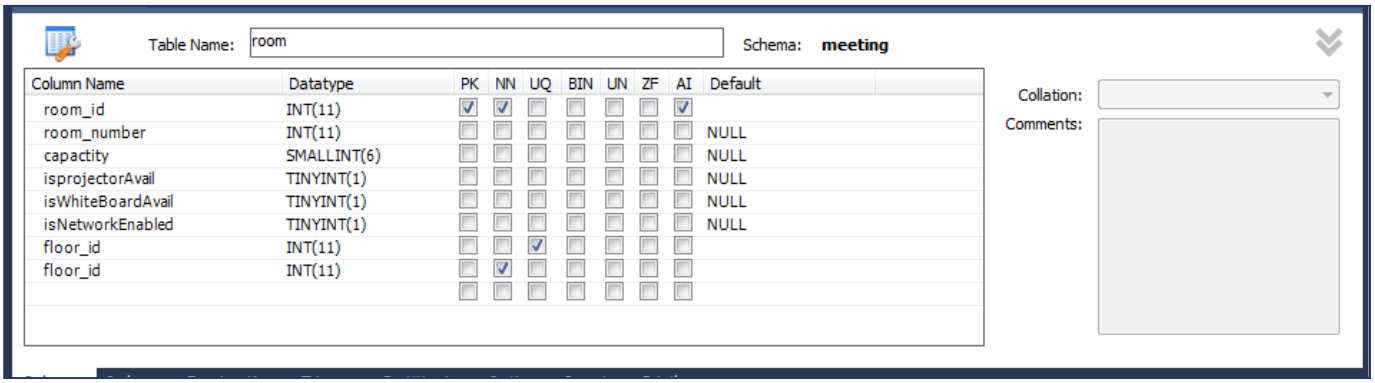


Table Name: room Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
room_id	INT(11)	✓	✓					✓	
room_number	INT(11)								NULL
capactity	SMALLINT(6)								NULL
isprojectorAvail	TINYINT(1)								NULL
isWhiteBoardAvail	TINYINT(1)								NULL
isNetworkEnabled	TINYINT(1)								NULL
floor_id	INT(11)			✓					
floor_id	INT(11)		✓						

Collation:
Comments:

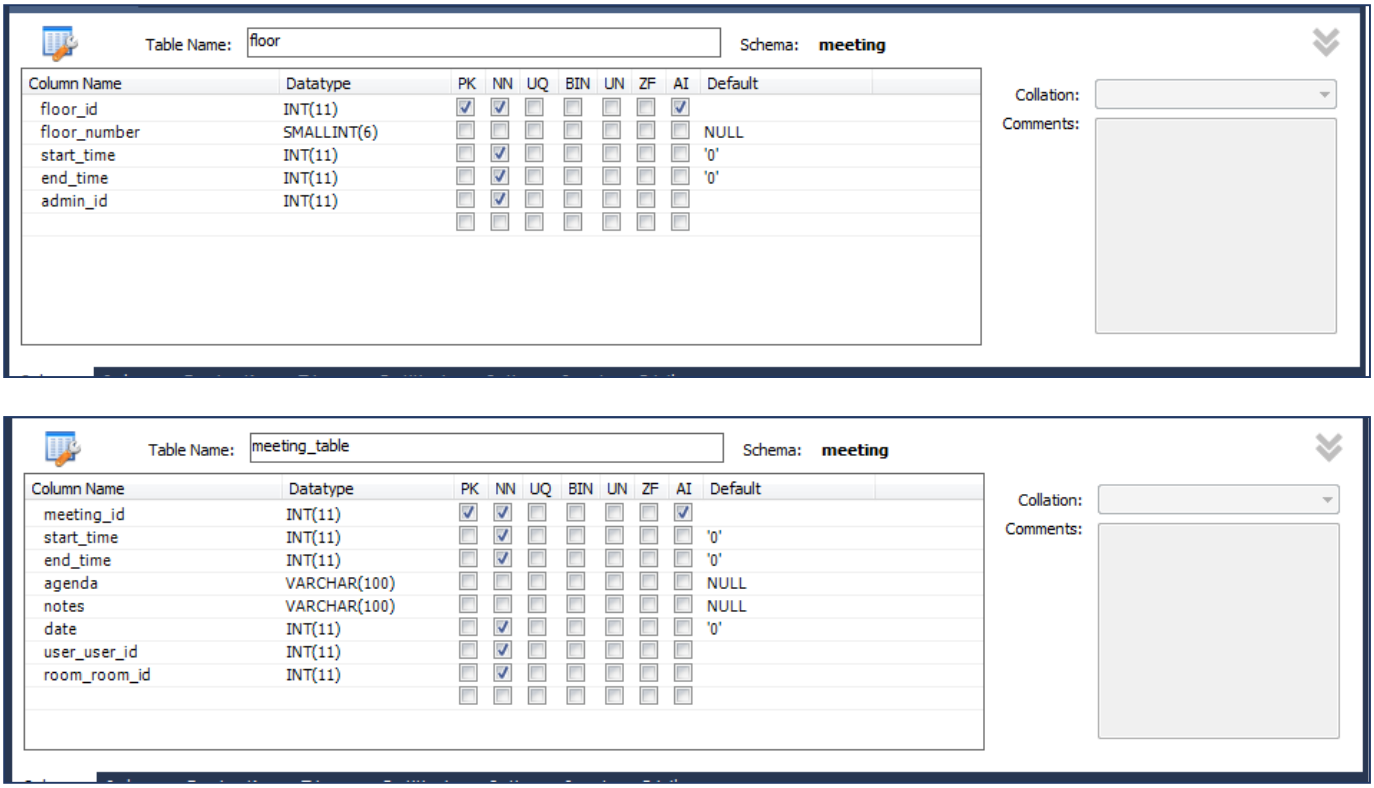


Table Name: floor Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
floor_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	
floor_number	SMALLINT(6)	☐	☐	☐	☐	☐	☐	☐	NULL
start_time	INT(11)	☐	☑	☐	☐	☐	☐	☐	'0'
end_time	INT(11)	☐	☑	☐	☐	☐	☐	☐	'0'
admin_id	INT(11)	☐	☑	☐	☐	☐	☐	☐	

Collation:
Comments:

Table Name: meeting_table Schema: meeting

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
meeting_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	
start_time	INT(11)	☐	☑	☐	☐	☐	☐	☐	'0'
end_time	INT(11)	☐	☑	☐	☐	☐	☐	☐	'0'
agenda	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	NULL
notes	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	NULL
date	INT(11)	☐	☑	☐	☐	☐	☐	☐	'0'
user_user_id	INT(11)	☐	☑	☐	☐	☐	☐	☐	
room_room_id	INT(11)	☐	☑	☐	☐	☐	☐	☐	

Collation:
Comments:

Note:

Above EER diagram and Data dictionary is generated using MYSQL workbench.

SQL Insert Statements:

```
-- -----------------------------------------------------
-- Schema meeting
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `meeting` DEFAULT CHARACTER SET latin1 ;
SHOW WARNINGS;
USE `meeting` ;

-- -----------------------------------------------------
-- Table `meeting`.`address`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`address` (
  `address_id` INT NOT NULL AUTO_INCREMENT COMMENT '',
  `door_number` VARCHAR(20) NOT NULL COMMENT '',
  `address_line_1` VARCHAR(45) NOT NULL COMMENT '',
  `address_line_2` VARCHAR(45) NOT NULL COMMENT '',
  PRIMARY KEY (`address_id`)  COMMENT '')

-- -----------------------------------------------------
-- Table `meeting`.`master_user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`master_user` (
  `u_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `address_id` INT NOT NULL COMMENT '',
  `first_name` VARCHAR(30) NULL DEFAULT NULL COMMENT '',
  `last_name` VARCHAR(50) NULL DEFAULT NULL COMMENT '',
  `email` TEXT NULL DEFAULT NULL COMMENT '',
  `phone` VARCHAR(20) NULL DEFAULT NULL COMMENT '',
  `role` VARCHAR(15) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`stud_id`)  COMMENT '')

-- -----------------------------------------------------
-- Table `meeting`.`admin`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`admin` (
  `admin_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `u_id` INT(11) NOT NULL COMMENT '',
  `password` VARCHAR(40) NULL DEFAULT NULL COMMENT '',
  `designation` VARCHAR(20) NULL COMMENT '',
  PRIMARY KEY (`admin_id`)  COMMENT '')
```

```
- -----------------------------------------------------
-- Table `meeting`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`user` (
  `user_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `u_id` INT(11) NOT NULL COMMENT '',
  `password` VARCHAR(40) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`user_id`)  COMMENT '')



-- -----------------------------------------------------
-- Table `meeting`.`floor`
-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `meeting`.`floor` (
  `floor_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `floor_number` SMALLINT(6) NULL DEFAULT NULL COMMENT '',
  `start_time` INT(11) NOT NULL DEFAULT '0' COMMENT '',
  `end_time` INT(11) NOT NULL DEFAULT '0' COMMENT '',
  `admin_id` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`floor_id`)  COMMENT '')

-- -----------------------------------------------------
-- Table `meeting`.`room`
-- -----------------------------------------------------


CREATE TABLE IF NOT EXISTS `meeting`.`room` (
  `room_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `room_number` INT(11) NULL DEFAULT NULL COMMENT '',
  `capactity` SMALLINT(6) NULL DEFAULT NULL COMMENT '',
  `isprojectorAvail` TINYINT(1) NULL DEFAULT NULL COMMENT '',
  `isWhiteBoardAvail` TINYINT(1) NULL DEFAULT NULL COMMENT '',
  `isNetworkEnabled` TINYINT(1) NULL DEFAULT NULL COMMENT '',
  `floor_id` INT(11) NULL COMMENT '',
  `floor_id` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`room_id`)  COMMENT '')
```

```
-- -----------------------------------------------------
-- Table `meeting`.`meeting_table`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`meeting_table` (
  `meeting_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
  `start_time` INT(11) NOT NULL DEFAULT '0' COMMENT '',
  `end_time` INT(11) NOT NULL DEFAULT '0' COMMENT '',
  `agenda` VARCHAR(100) NULL DEFAULT NULL COMMENT '',
  `notes` VARCHAR(100) NULL DEFAULT NULL COMMENT '',
  `date` INT(11) NOT NULL DEFAULT '0' COMMENT '',
  `user_user_id` INT(11) NOT NULL COMMENT '',
  `room_room_id` INT(11) NOT NULL COMMENT '',
  PRIMARY KEY (`meeting_id`)  COMMENT '')

-- -----------------------------------------------------
-- Table `meeting`.`attendance_table`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `meeting`.`attendance_table` (
  `meeting_id` INT(11) NOT NULL COMMENT '',
  `stud_id` INT(11) NULL DEFAULT NULL COMMENT '',
  `IsParticipantAvail` TINYINT(1) NULL DEFAULT NULL COMMENT '',
  PRIMARY KEY (`meeting_id`)  COMMENT '')
```

Inserting Data:

User Table:

INSERT INTO `user` (`user_id`, `u_id`, `password`) VALUES (1,4,'pass'),(2,5,'pass'),(3,6,'pass');

Master_User Table:

INSERT INTO `master_user` (`u_id`, `address_id`, `first_name`, `last_name`, `email`, `phone`, `role`) VALUES (1,1,'Rajiv','Shukla','raj@gmail.com','999999','floor manager'),
(2,5,'Vinodh ','Khanna','vin@gmail.com','999999','floor manager'),
(3,3,'Sim','Patel','sim@gmail.com','999999','manager')
,(4,2,'Smitha','Rao','smi@gmail.com','999999','student'),
(5,4,'Deep','Rao','dee@gmail.com','999999','student'),
(6,6,'Sai','Vadla','sai@gmail.com','989808','student');

Address Table:
INSERT INTO `address` (`address_id`, `door_number`, `address_line_1`, `address_line_2`) VALUES (1,'1212','UT Drive','Park'),
(2,'3452','Smith Lane','Concord'),
(3,'300','Drake Avenue','Shamrock'),(4,'121','Keith Avenue','Smahrock'),(5,'450','Kirk Drive','Welmington'),
(6,'920','Smith Lane','Welmington '),
(7,'111','Ramsay Avenue','Balentine');

Admin Table:
INSERT INTO `admin` VALUES (1,1,'pass','floor manager'),(2,2,'pass','floor manager'),(3,3,'pass','manager');
Meeting_table:

INSERT INTO `meeting_table` (`meeting_id`, `start_time`, `end_time`, `agenda`, `notes`, `date`, `user_user_id`, `room_room_id`) VALUES (123,1100,1200,'Project planning','1) abc 2) cde 3) def) xyz',1312015,1,21),(124,1500,1600,'Team Activity','1) abc 2) cde 3) def) xyz',1312015,1,22),(125,1230,1330,'Technical ','1) abc 2) cde 3) def) xyz',1312015,2,21);

Room table:

INSERT INTO `admin` (`admin_id`, `u_id`, `password`, `designation`) VALUES

(1,1,'pass','floor manager'),(2,2,'pass','floor manager'),(3,3,'pass','manager');

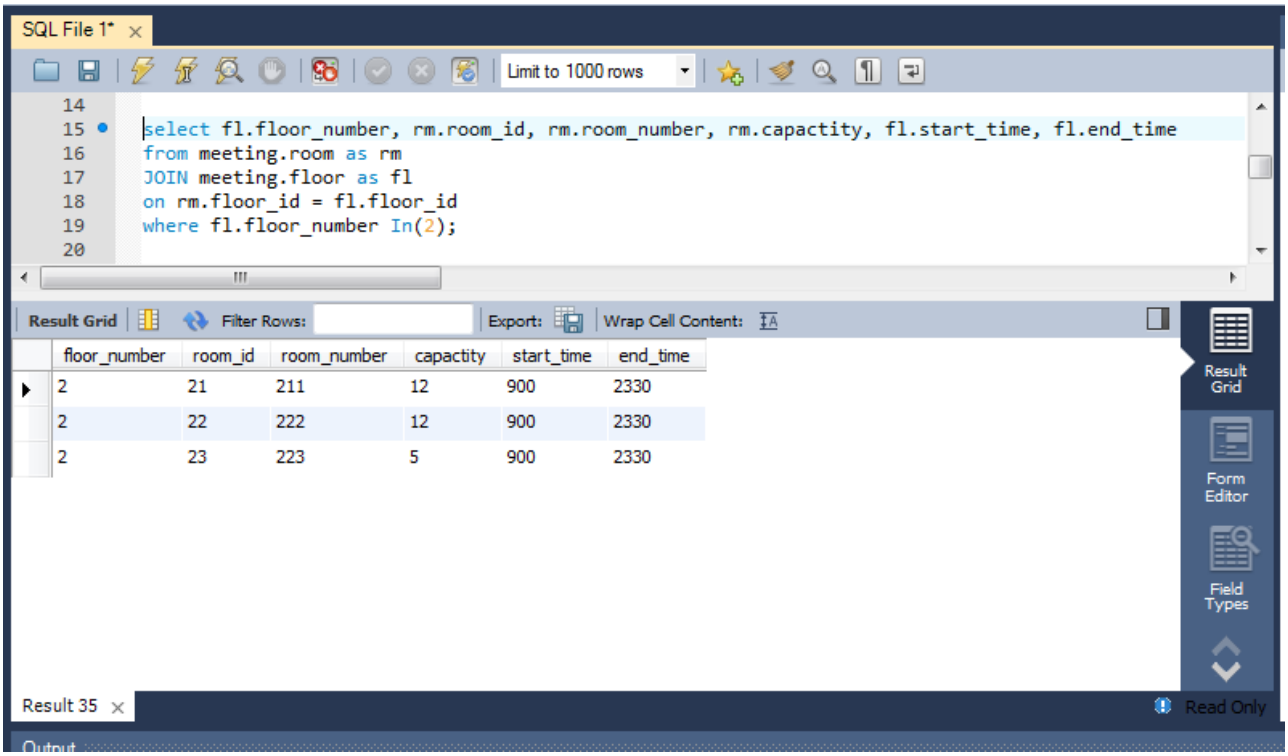
Floor:

INSERT INTO `floor` (`floor_id`, `floor_number`, `start_time`, `end_time`, `admin_id`) VALUES (112,1,800,2130,1),(113,2,900,2330,2),(114,3,800,2300,3);

Advanced Queries:

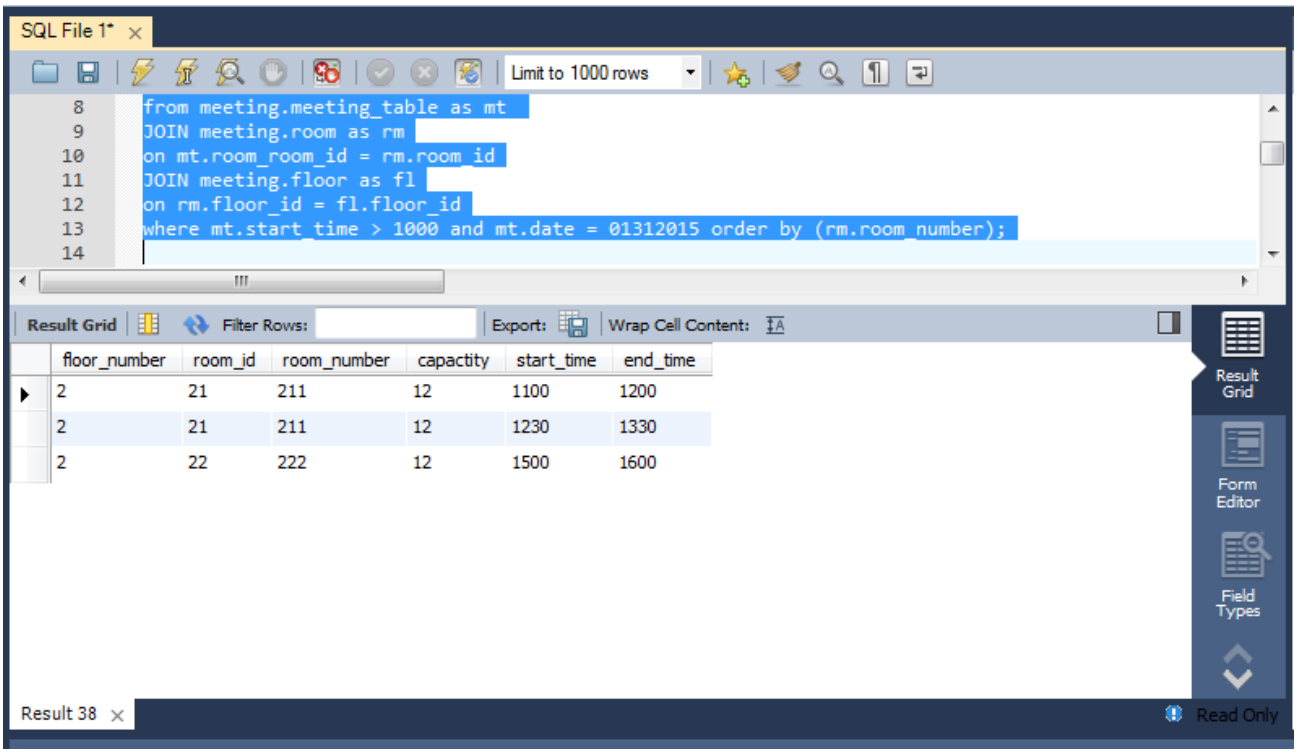
Query 1:
Below query will retrieve rooms available in different floors, their start and end time:

```
select fl.floor_number, rm.room_id, rm.room_number, rm.capactity, fl.start_time, fl.end_time
from meeting.room as rm
JOIN meeting.floor as fl
on rm.floor_id = fl.floor_id
where fl.floor_number In(1,2,3);
```

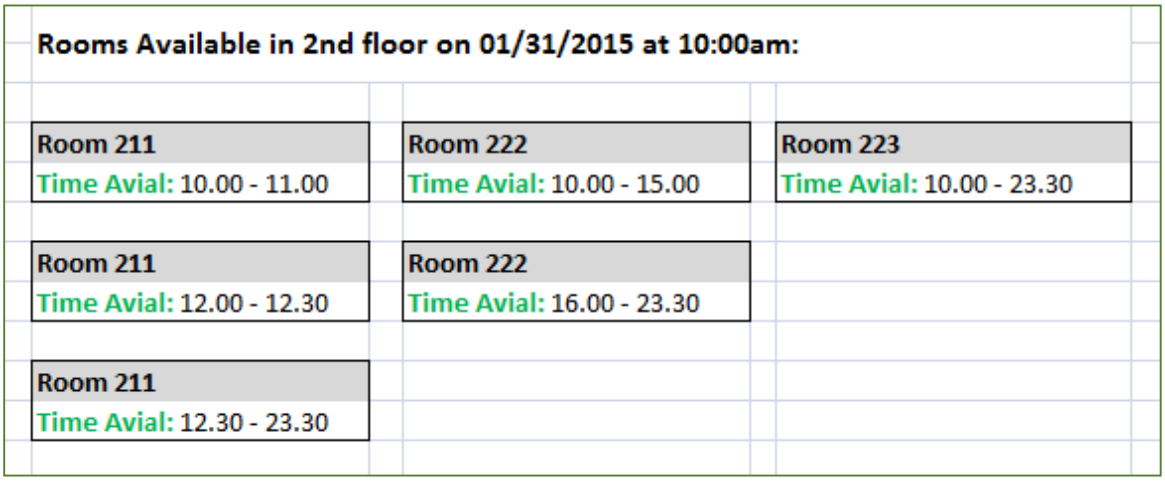


Query 2:
Below query retrieves the list of rooms that are booked on a particular date and time interval:

```
select fl.floor_number, rm.room_id, rm.room_number, rm.capactity, mt.start_time, mt.end_time
from meeting.meeting_table as mt
JOIN meeting.room as rm
on mt.room_room_id = rm.room_id
JOIN meeting.floor as fl
on rm.floor_id = fl.floor_id
where mt.start_time > 1000 and mt.date = 01312015 order by (rm.room_number);
```



We should compare query 1 results with query 2 and display to user slots taken and available in UI as below:



Database Implementation:

Stored Procedures:

A stored procedure takes the common SQL statements used in your application program. MySQL stored procedures can accept parameters from the calling program, and can send a result back. The CALL statement takes the name of the stored procedure and any parameters that need to be passed to it. Typically a stored procedure return results into variables that you specify

We have implemented stored procedures for the following reasons:

1. Stored procedures are more efficient.
2. Stored procedures process are safe.

Book Room:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `bookroom`(IN room INT, IN stime TIME, IN etime TIME, IN userId INT)
BEGIN

INSERT INTO meeting_table(room_room_id, start_time, end_time, user_user_id) VALUES (room, stime, etime, userId);

END
```

Calling the Stored procedure in PHP:

```
$sql = 'CALL bookroom('.$room.','.$stime.','.$etime.','.$userid.')';
```

View History:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `history`(IN userId INT)
BEGIN
        SELECT mt.date, r.room, r.capacity, fl.floor_number, mt.start_time, mt.end_time
   FROM meeting.meeting_table AS mt
   JOIN meeting.room as r
   JOIN meeting.floor AS fl
   ON  mt.user_user_id = userId AND mt.room_room_id = r.room_id and r.floor_id = fl.floor_id
   ORDER BY (mt.date);

END
```

Calling the stored procedure in php:

$sql1 = 'CALL history('.$userid.')';

Triggers:

CREATE TRIGGER updtr1
BEFORE update
ON user FOR EACH ROW
INSERT INTO newuser(ID,main)
values(OLD.stud_id,OLD.user_id) ;

Performance Tuning:

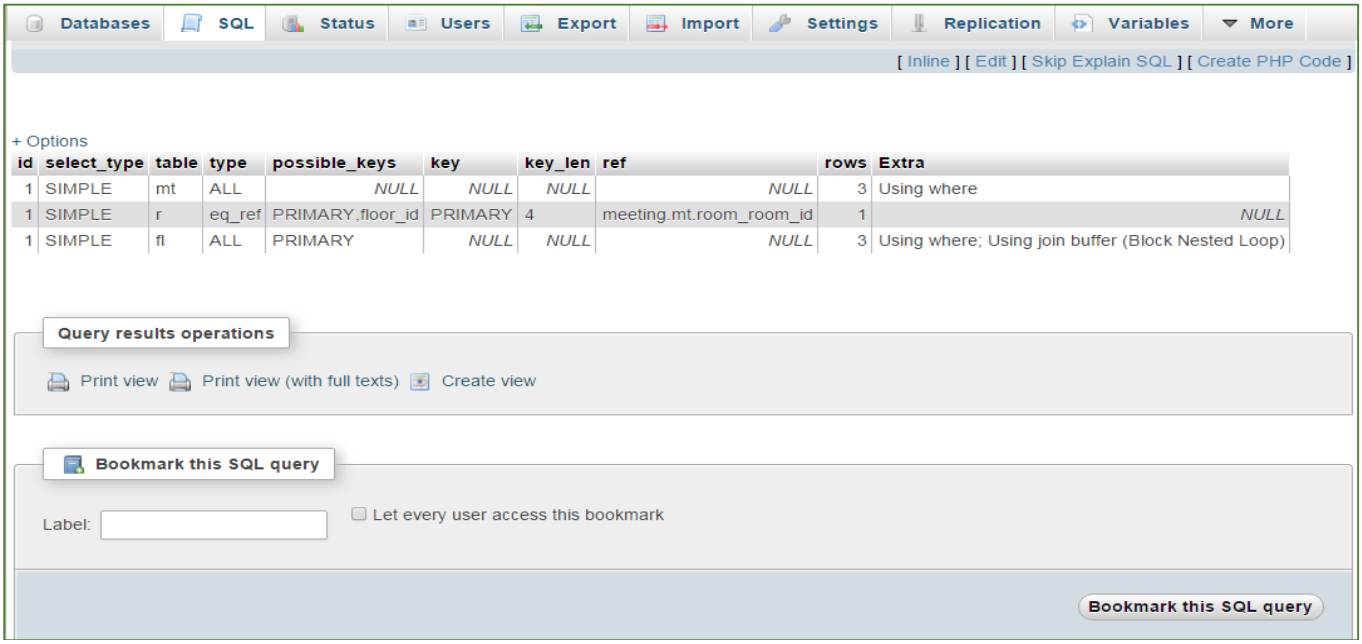
USING SMALLEST DATATYPES:
It takes longer to read in large data types than smaller ones, as the longer data types require more disk sectors to be read into memory.

CREATE TABLE IF NOT EXISTS `meeting`.`user` (
`user_id` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
`u_id` INT(11) NOT NULL COMMENT '',
`password` VARCHAR(40) NULL DEFAULT NULL COMMENT '',
PRIMARY KEY (`user_id`) COMMENT '')

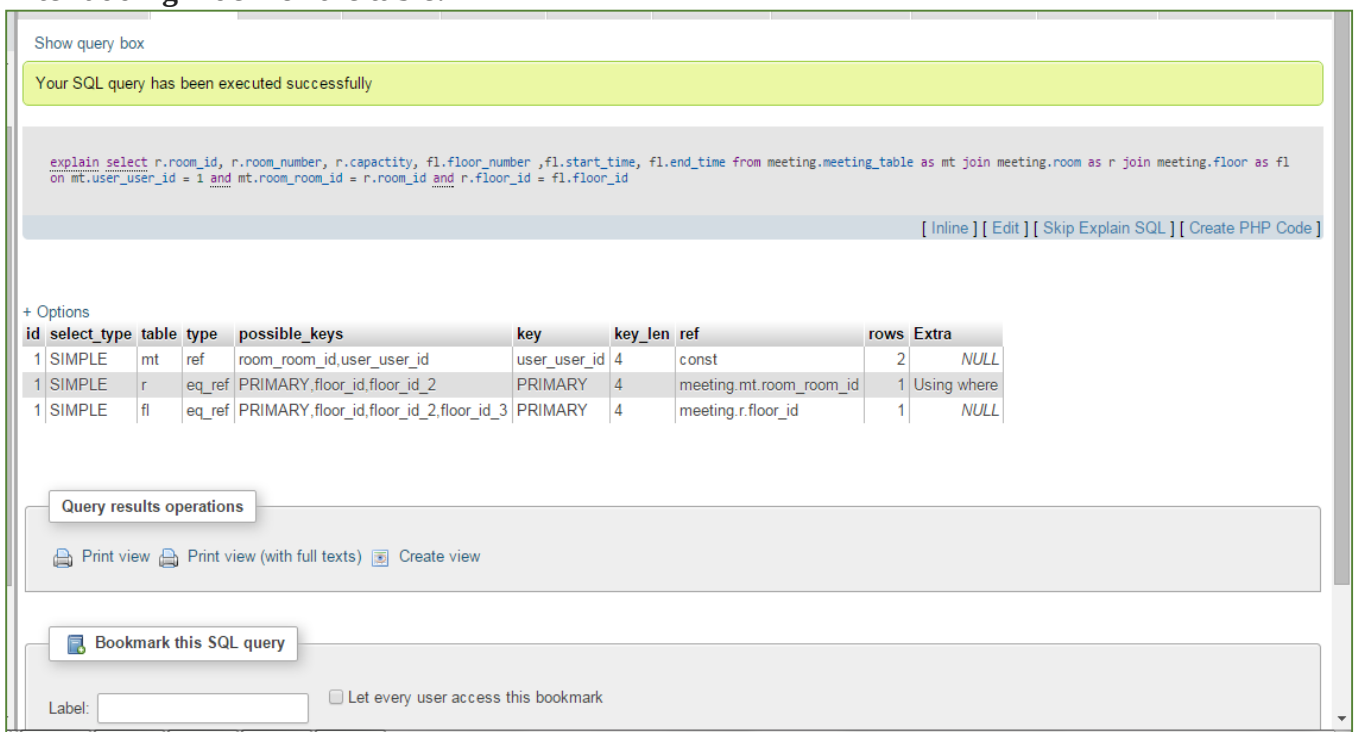
Indexes:

Indexes are used to find rows with specific column values quickly. Without an index, MySQL must begin with the first row and then read through the entire table to find the relevant rows.

Select r.room_id, r.room_number, r.capactity, fl.floor_number ,fl.start_time, fl.end_time from meeting.meeting_table as mt join meeting.room as r join meeting.floor as fl on mt.user_user_id = 1 and mt.room_room_id = r.room_id and r.floor_id = fl.floor_id ;



After adding index for the table:



Show query box

Your SQL query has been executed successfully

```
explain select r.room_id, r.room_number, r.capactity, fl.floor_number ,fl.start_time, fl.end_time from meeting.meeting_table as mt join meeting.room as r join meeting.floor as fl
on mt.user_user_id = 1 and mt.room_room_id = r.room_id and r.floor_id = fl.floor_id
```

[Inline] [Edit] [Skip Explain SQL] [Create PHP Code]

+ Options

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	mt	ref	room_room_id,user_user_id	user_user_id	4	const	2	*NULL*
1	SIMPLE	r	eq_ref	PRIMARY,floor_id,floor_id_2	PRIMARY	4	meeting.mt.room_room_id	1	Using where
1	SIMPLE	fl	eq_ref	PRIMARY,floor_id,floor_id_2,floor_id_3	PRIMARY	4	meeting.r.floor_id	1	*NULL*

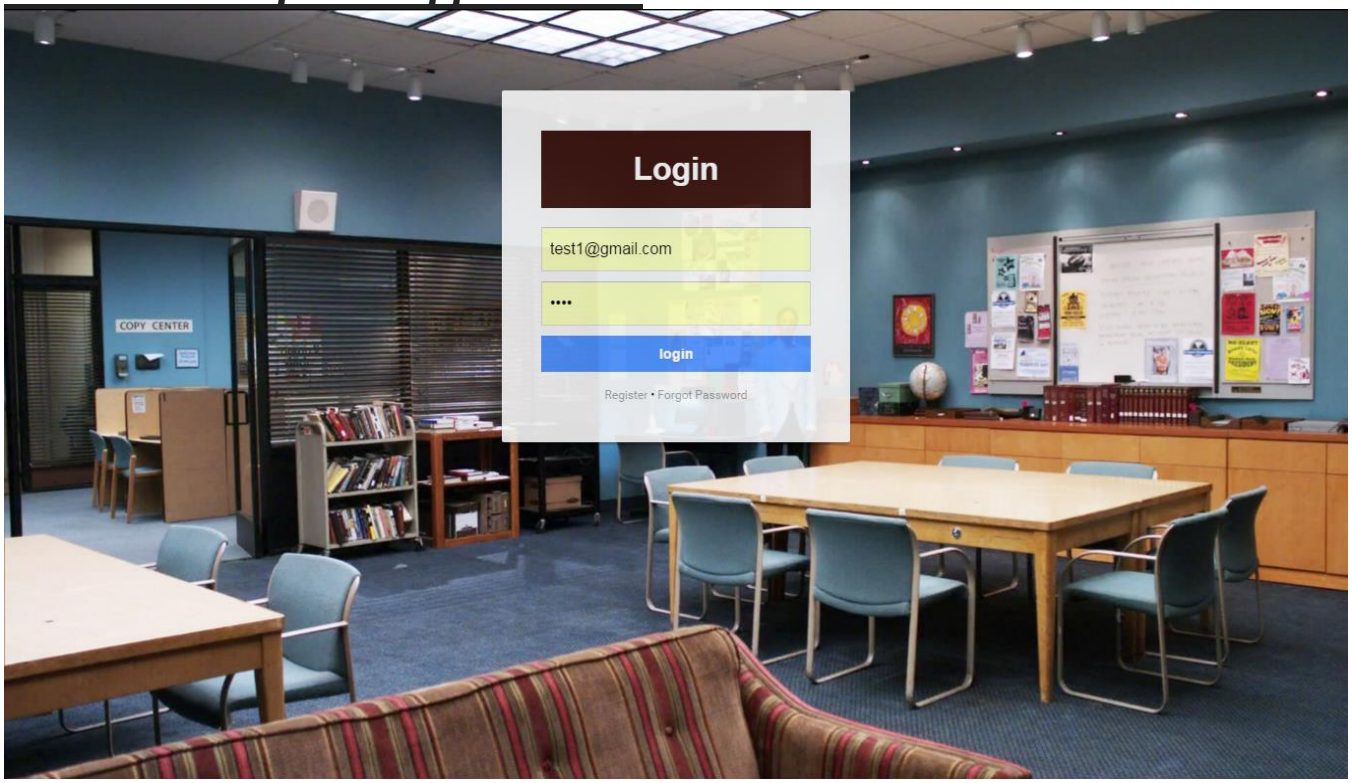
Query results operations

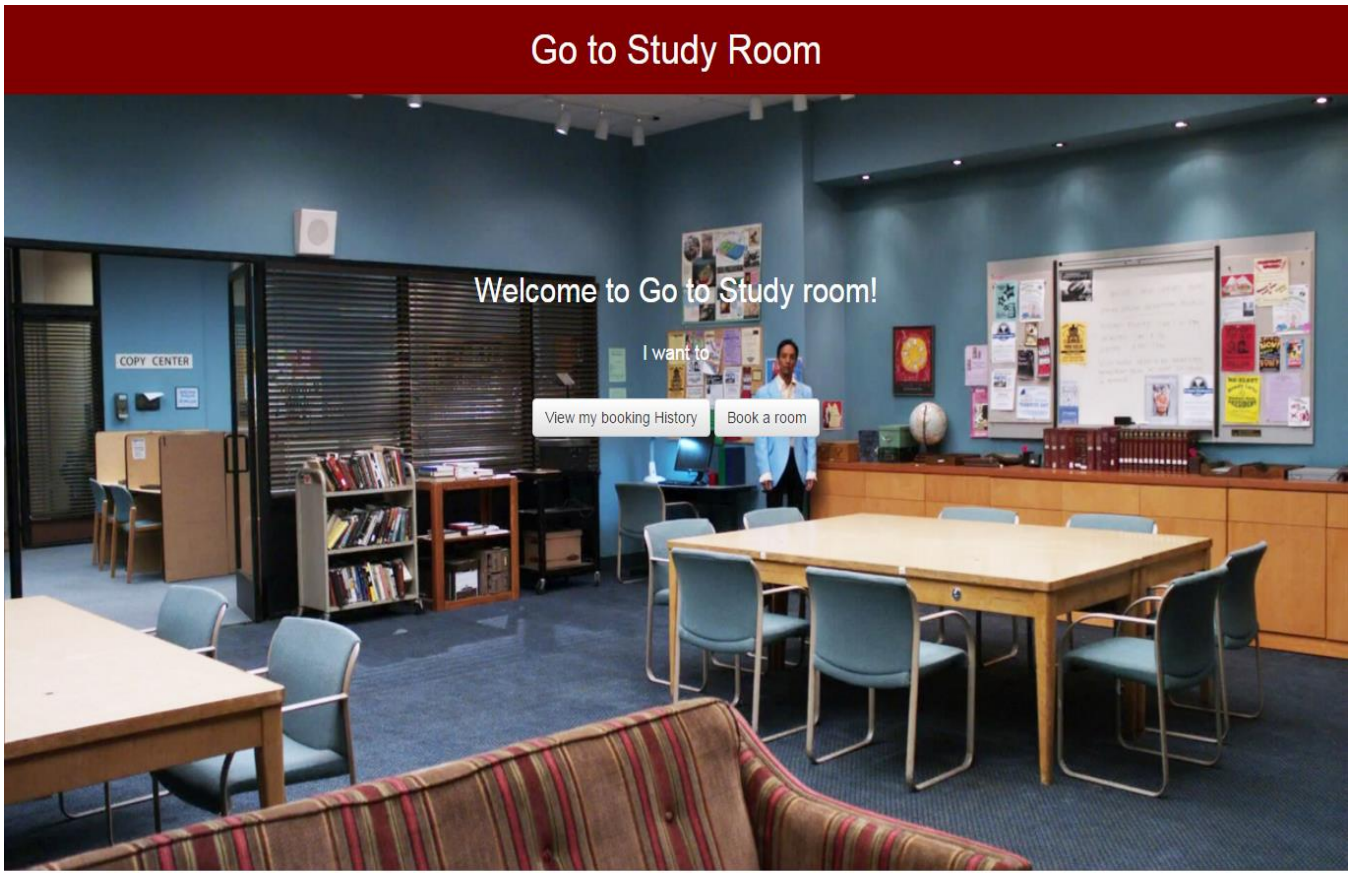
Print view Print view (with full texts) Create view

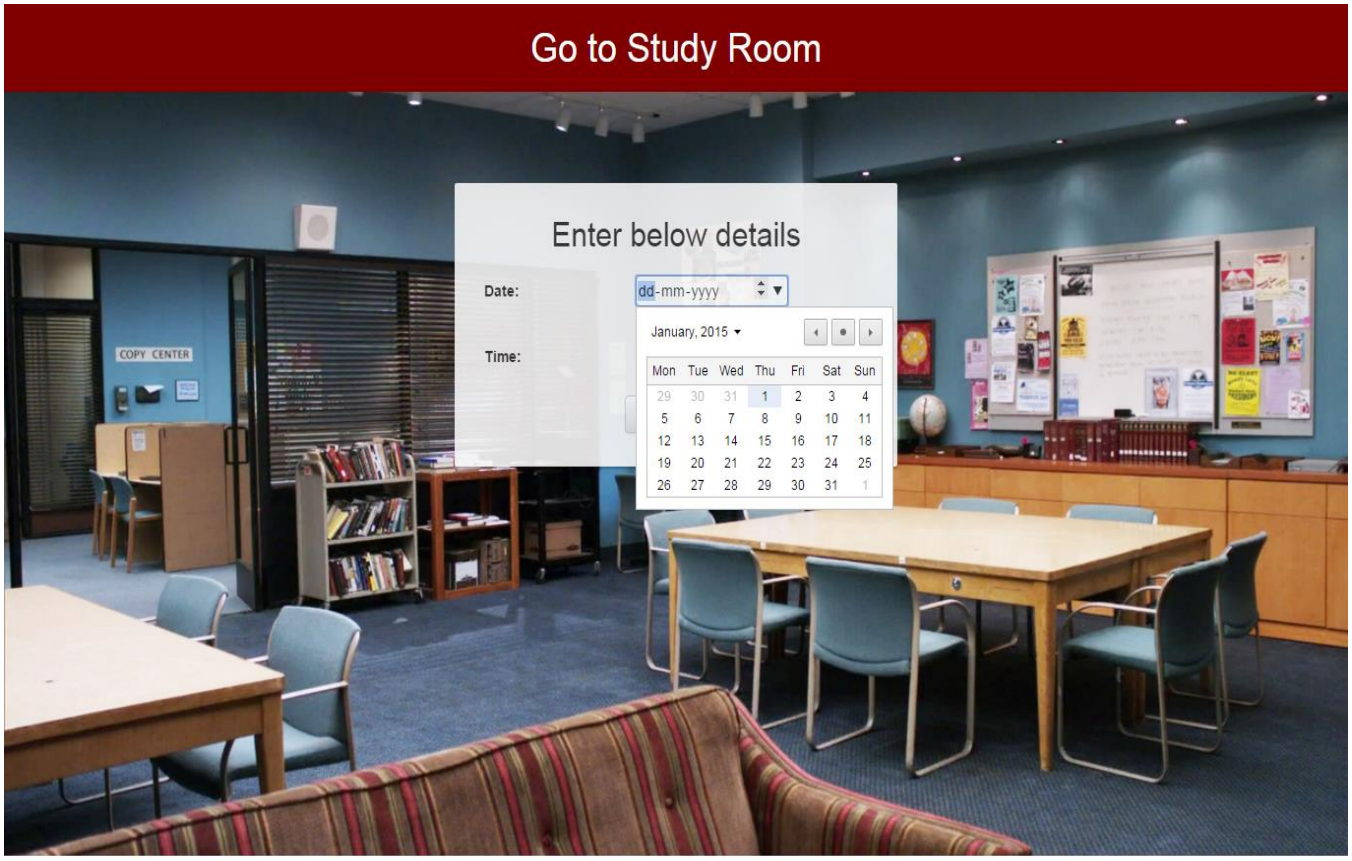
Bookmark this SQL query

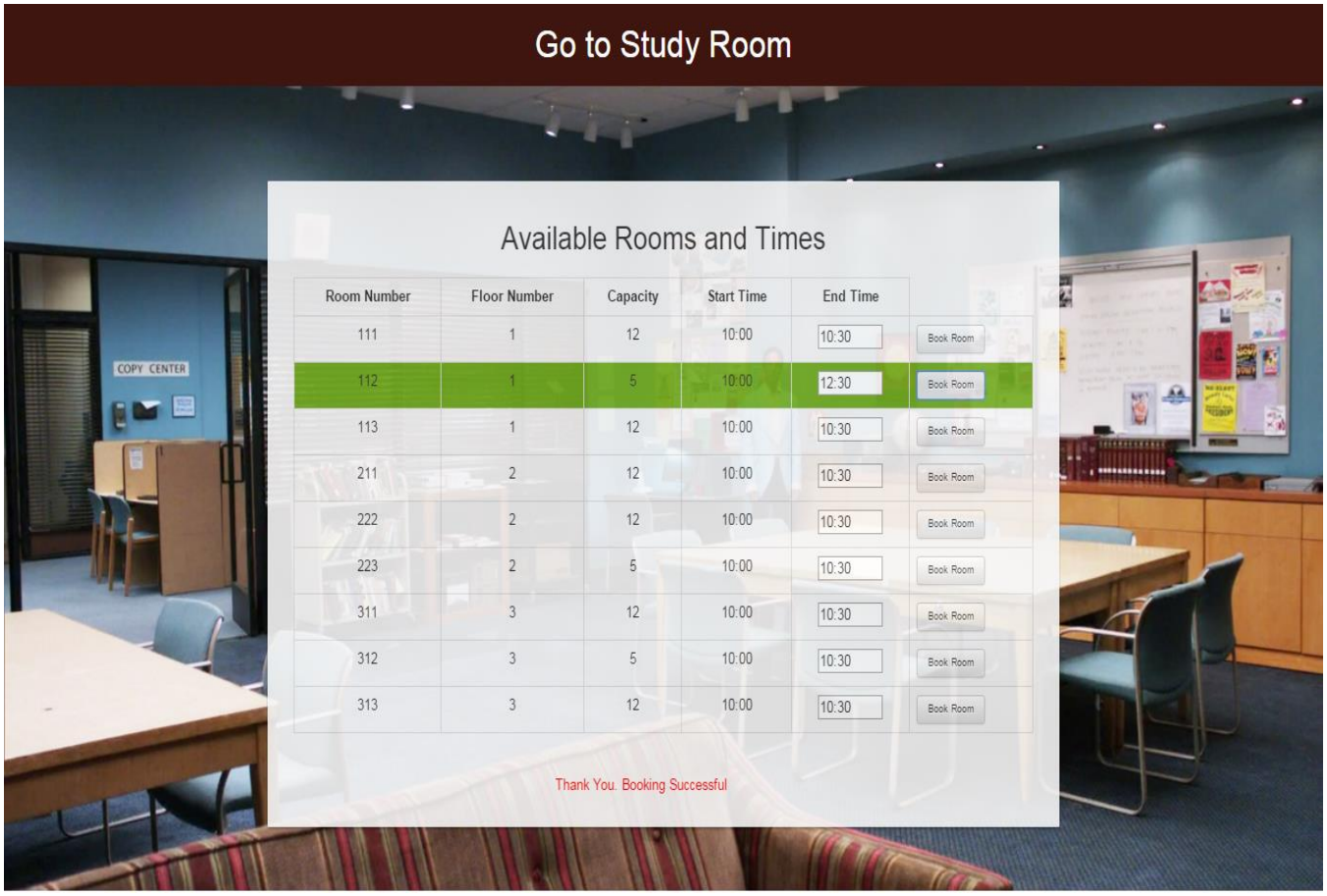
Label: Let every user access this bookmark

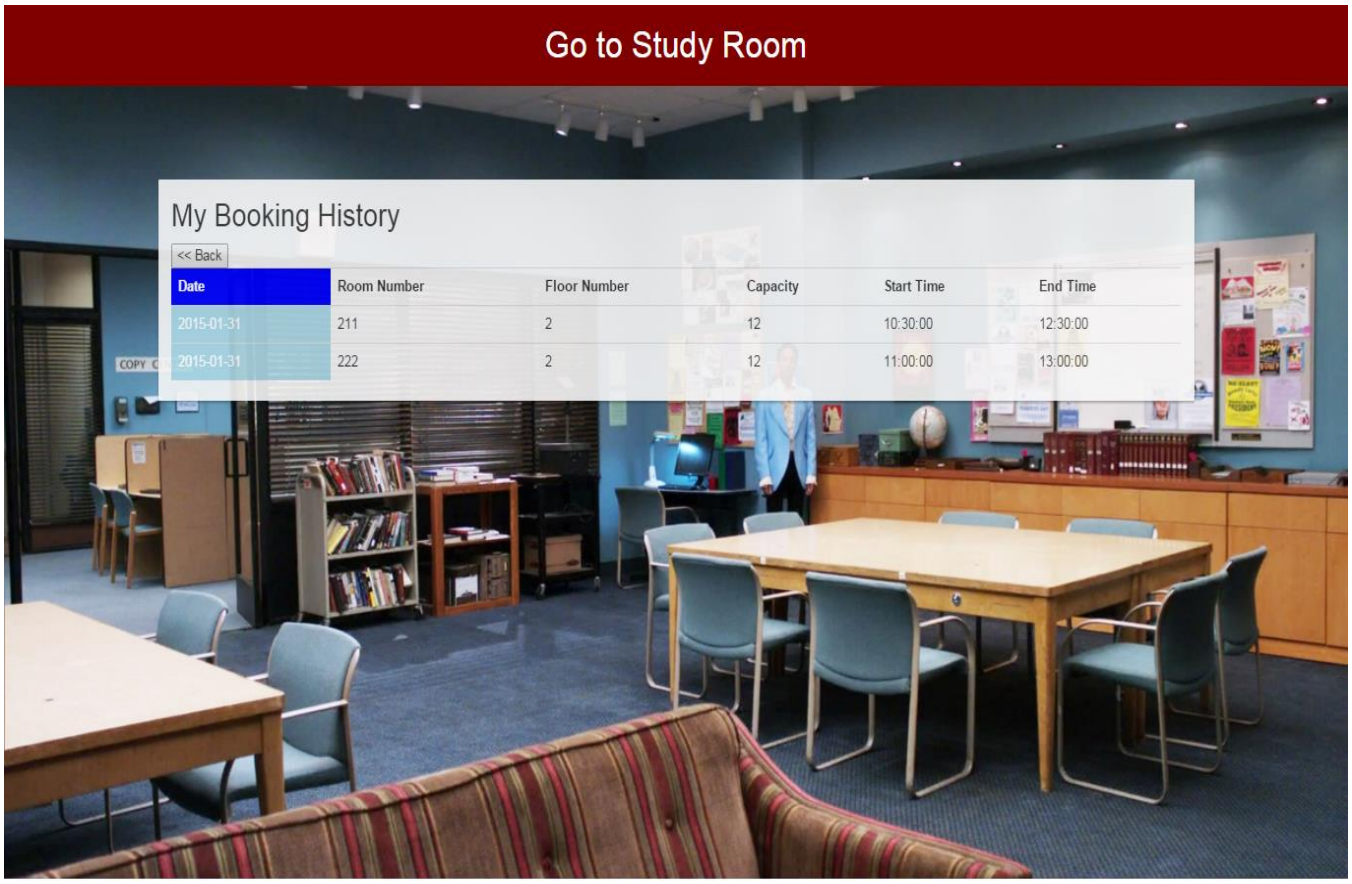
Screenshots of Our Application:

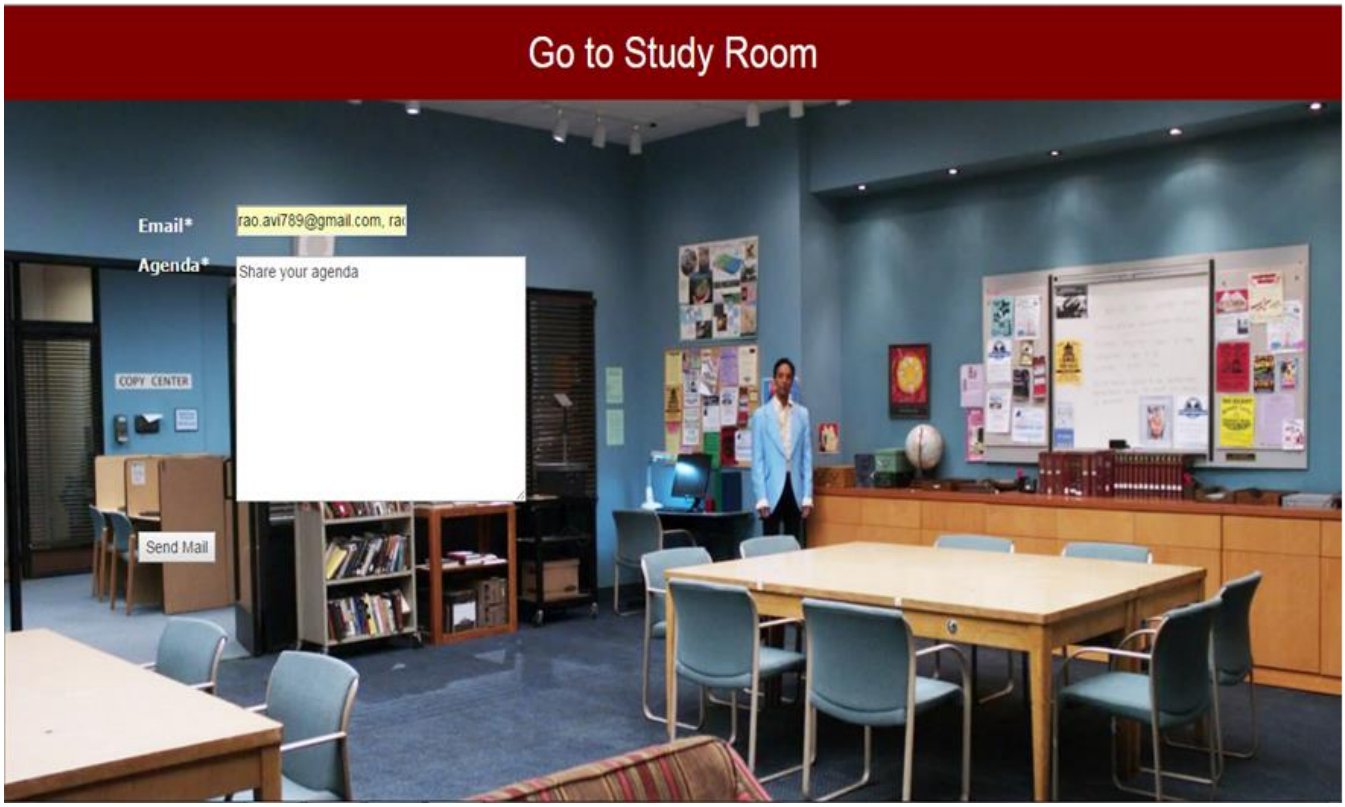



Go to Study Room
COPY CENTER
Welcome to Go to Study room!
I want to
View my booking History
Book a room


Go to Study Room
Enter below details
Date:
dd-mm-yyyy
Time:
January, 2015
Mon Tue Wed Thu Fri Sat Sun
29 30 31 1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31 1
COPY CENTER


Go to Study Room
COPY CENTER
Available Rooms and Times
Room Number
Floor Number
Capacity
Start Time
End Time
111 1 12 10:00 10:30 Book Room
112 1 5 10:00 12:30 Book Room
113 1 12 10:00 10:30 Book Room
211 2 12 10:00 10:30 Book Room
222 2 12 10:00 10:30 Book Room
223 2 5 10:00 10:30 Book Room
311 3 12 10:00 10:30 Book Room
312 3 5 10:00 10:30 Book Room
313 3 12 10:00 10:30 Book Room
Thank You. Booking Successful


Go to Study Room
My Booking History
<< Back
Date
Room Number
Floor Number
Capacity
Start Time
End Time
2015-01-31
211
2
12
10:30:00
12:30:00
2015-01-31
222
2
12
11:00:00
13:00:00


Go to Study Room
Email*
rao.avi789@gmail.com, ra
Agenda*
Share your agenda
COPY CENTER
Send Mail

Future Scope:

- **Mobile App:** The current system can be viewed on any Web Browser. But a stand-alone Mobile application would help access to the database system from mobile in a more efficient way.

- UI responsive: Make it more UI responsive

- **Integration with University Database:** The next step for the project would be University-wide integration which would produce a large database.

- **Security:** Considering sensitive data related to the University, the database would implement measures that make the whole system secure.

Contact Information for all the members:

Sl No	Name	Email Id	Phone No
1	Roopa Shankargouda Patil	Rshanka8@uncc.edu	510789716
2	Sadhana Seetamsetty	sseetams@uncc.edu	9725336663
3	Revati Dhananjayan Lalitha	rdhananj@uncc.edu	9805855758
4	Avinash Yachamaneni	ayachama@uncc.edu	9805855326
5	Vasudev Marla Balraj	vmarlaba@uncc.edu	9803199265

Meeting Dates:

Sl no	Activity	Date
1	Creating project part 1 Report	1st Sep 2015
2	Create ER diagram	28th Sep 2015
3	Project part 2	8th oct 2015
4	Project part 3 Plan	28th Oct 2015
5	Presentation Plan	Dec 1st 2015
6	Project Part 4 documentation plan	Dec 4th 2015

Method of Communication and File Sharing for Team

1. Method of Communication: WhatsApp group, Email

2. Method of File sharing: Google Drive