

**Project Report on**

**Room Booking Mobile Application**

**Software Requirements Specification Document**

**Version: 9** **Date: xxxxxx**

**Table of Contents**

1. **Introduction** 3

1.1 Purpose 3

1.2 Scope 3

1. **Specific Requirements**  4

2.1 Functional Requirements 4

2.2 Non-Functional Requirements 5

2.3 Other Requirements 5

1. **Analysis Models** 6

3.1 Use case Diagram 6

3.2 Class Diagram 7

1. **Scenarios** 8

4.1 Login 8

4.2 Registration 9

4.3 Manage Profile 11

4.4 Add Floors 12

4.5 Add Rooms 13

* 1. Room Booking 14

1. Use case Descriptions 15
2. Admin Screen Flow 16
3. User Screen Flow 17
4. **Database** 18

8.1 Database Schema 19

8.2 Data Dictionaries 21

8.3 Data Base Script 23

1.INTRODUCTION

Room Booking App enables Teachers, pedagogical counsellors as well as other CEGEP staff members required to book a room in the Montreal campus either for an exam or a meeting. This makes the app more useful When issues like overbooked spaces and underused areas surface. Room Booking Mobile Application makes it easy for workplace leaders to understand and manage space inventory, from meeting room booking permissions to office, room reservations, and more.

1.1 PURPOSE

Room Booking Mobile Application makes it easy for people to find and use the best-fit meeting room or the perfect room for using exams or meeting. This system increases plan for growth, and make the workplace more flexible and accessible with actionable suggestions.

1.2 SCOPE

This project traverses a lot of areas ranging from business concept to computing field and required to perform several researches to be able to achieve the project objects.

The area covers include:

* Room Booking Mobile Application: This includes study on to understand and manage space inventory, from meeting room booking permissions to office, room reservations, and more.
* CEGEP’s Montreal campus staff will be able to use the system effectively.
* It increases the efficiency of the management at offering quality services to the customers.
* It provides custom features development and support with the software

This section describes the features which are in the scope of the developed application.

#### **Room status**

See instantly if a room is being used. Green for available and red for occupied.

#### **Room capacity**

Shows how many people the room is suited for.

#### **Spontaneous meeting**

Book a room for meetings or for exams! With the ease of two clicks, the available room is yours.

#### **Check in**

To make sure reservations are actually utilized for meetings and free up the time when they're not. Set a time for when the meeting should be cancelled if not checked in.

#### **Room agenda**

Scroll through the meetings of today to see what time slots are left or when the next meeting will start.

#### **Equipment**

Shows what kind of equipment the room offers, such as TV, projector, whiteboard etc. Allows users to report equipment that doesn't work properly.

#### **Custom booking**

Allows you to make personalized bookings. You get to decide the room, time, name of the meeting etc.

#### **[Ro](https://www.meetio.com/products/meetio-room)om usage**

Get valuable data about room usage and booking behavior, enabling you to adapt functionality and plan your space better.

2. REQUIREMENTS

2.1. Functional Requirements

Following are some functional requirements:

* The system allows Admin, Teachers, counsellors and staff to login using their username and password.
* The system allows Admin to add new staff.
* Admin, Teachers, counsellors and staff can view detailed description of rooms based on availability.
* Admin, Teachers, counsellors and staff have access to check the status of the room.
* The system must allow the users to view list of available rooms during booking.
* Users can view the booking confirmation and also cancel the reservations by using booking reference number provided during the reservation process.
* User can view the date and duration of meeting.
* The system allows user to specify the reason for booking and mention capacity.
* User can pick necessary equipment required either for meeting or an exam.
* The system allows Admin to create/modify/delete rooms data.
* Admin can add/delete rooms.
* Admin can update room status and size.
* Admin can cancel booking.
* Admin can add/delete software for meeting and exams.
* Admin can add/delete Hardware equipment for meetings and exams.
* Add any spaces or tools like desks, or other shared resources.

2.2. Non-functional requirements

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users.

Here are some Non-functional requirements:

**Performance**

* Ability to maintain number of users on the server at once.
* Speedy performance / transmission of data
* Display accurately and efficiently on all devices.

**Availability**

* The system is available for 24 hours, 7 days a week.

**Quality Attributes:**

● Maintain a user-friendly environment that is visually appealing

● Easy to see and use navigation

● Maintain readable content.

2.3. OTHER REQUIREMENTS

**Hardware Requirements**

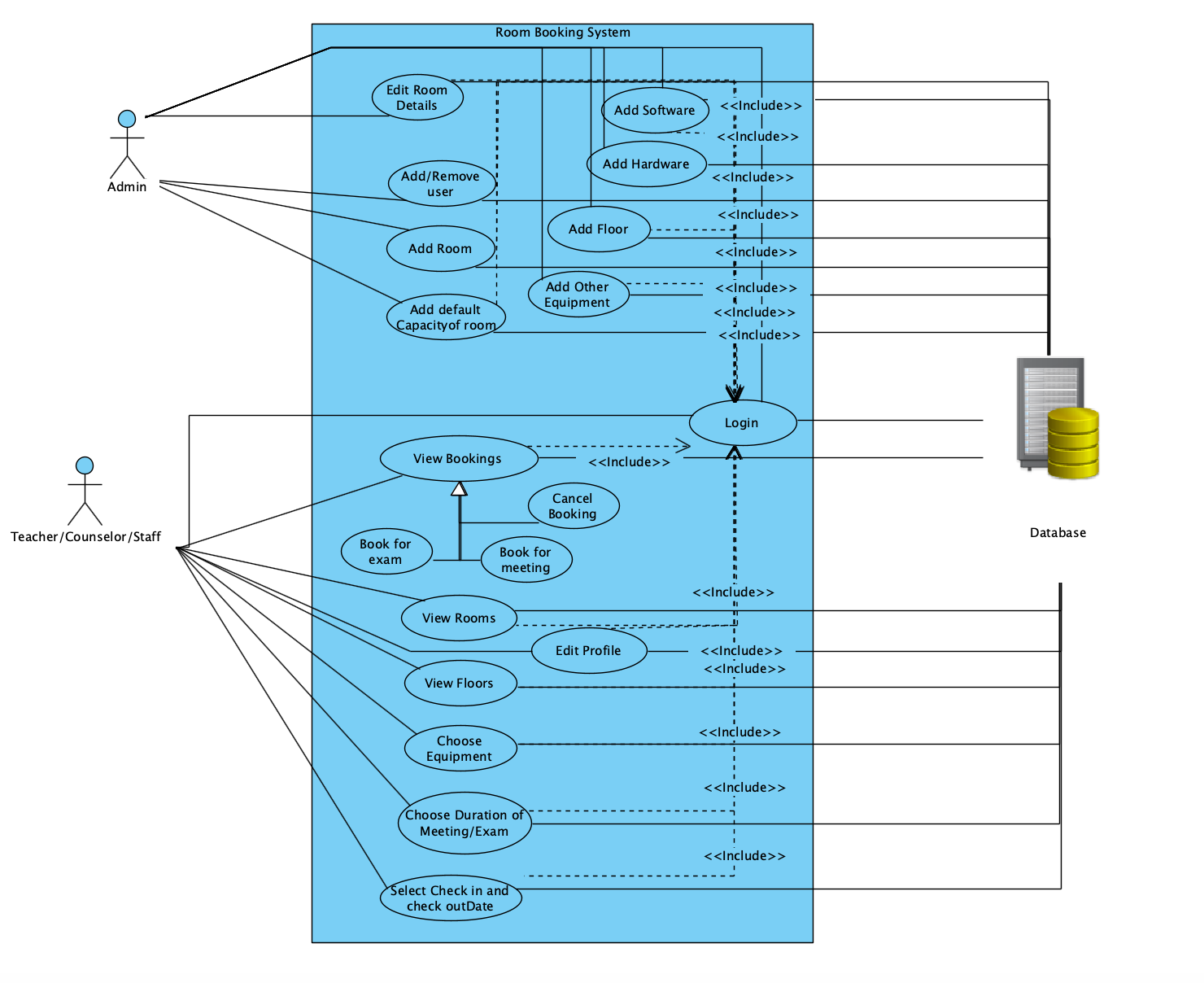
* Device: Android Smartphone
* Memory Space: 32 GB of disk space, 4 GB of RAM for development machine
* Connector: USB cable to export the app from development machine to Android device.
* 1 GB for Android SDK, emulator system images and cache.

**Software Requirements**

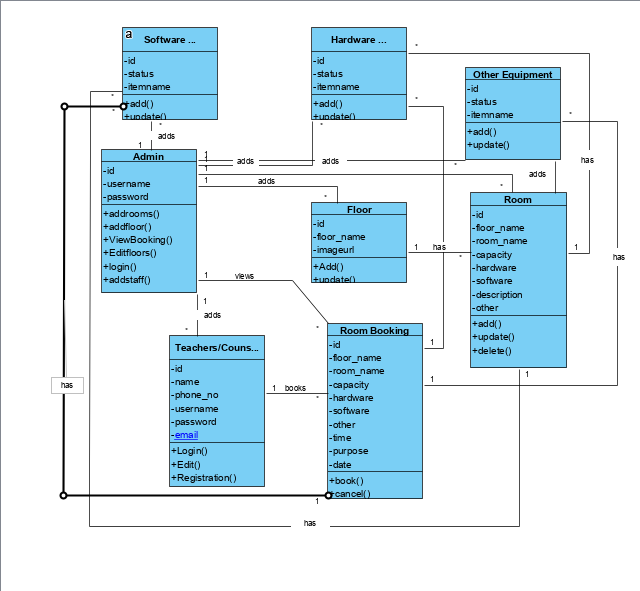
* Eclipse JAVA IDE EE for web developers.
* Android SDK and AVD installed on development machine.
* Java (JDK) 1.6
* Android Plug-in
* MYSQL for database

3. ANALYSIS MODELS

3.1 Use Case Diagram.

****

3.2 Class Diagram



4. SCENARIOS:

4.1. Login:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC1 | | |
| Use Case Name | Log In | | |
| Date created |  | Date last updated : | |
| Actor | Admin, User | | |
| Description | This use case describes how the actor logs in to the app with his registered mail id and password. | | |
| Triggering event | Actor selects the **login** link | | |
| Preconditions | The Actor must have registered mail id before login to the app. | | |
| Flow of events | Actor | | System |
| 1.Actor enters his registered mail id.  2. Actor enters his password.  3. Click Login | | 3.1 All the information (mail id and password) sends to the database.  3.2 Data base authenticates the information and acknowledges to the system as a valid user.  3.3 System displays the menu page to the user. |
| Alternative flow of events | Actor | | System |
| 1. Actor enter his/her mail ID.  2. Actor enter Wrong Password.  3. Clicks Login  4. Actor enter the correct information. | | 3.1 All the information (mail id and Password) sends to the data base.  3.2 Data base check the information and acknowledges to the system as an invalid user.  3.3 System displays the error message saying « Wrong id or Password. Please try again».  3.4 System empty the Fields. |
| Post conditions | The Actor see the menu page after successful login. | | |
| Exception - Conditions |  | | |

4.2. Registration:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC2 | | |
| Use Case Name | Registration | | |
| Date created |  | Date last updated: | |
| Actor | User | | |
| Description | This use case describes how the User enters all the information for the first-time registration to the app. | | |
| Triggering event | User selects the **register** link | | |
| Preconditions | User must be a new user before registration. | | |
| Flow of events | Actor | | System |
| 1. User enters mail id  2. User enters First name  3.User enters last name  4. User enters password.  5. User reenters the password.  6. Clicks **Submit** | | 1.1 System checks with the database whether the information already exists or not.  4.1 System checks the information whether it is in the range of 6 to 8 characters, includes at least 1 number and 1 special character.  5.1System checks whether the field information matches with the password field.  6.1 System sends all the information to the database and saves it.  6.2 Displays User registered successfully.  6.3 Displays login page |
| Alternative flow of events | Actor | | System |
|  | 1. User enters existing mail id 2. User enters existing first name. 3. User enters existing last name. 4. User enters password that does not satisfy the condition. 5. User re-enters password that does not match with the password field 6. All the above steps will be repeated until user enters appropriate information. 7. User Clicks **Submit** | | 1.1 The information sends to the database.  1.2 Database checks the information and acknowledges to the system as an invalid user.  2.1 System sends the information to the database and acknowledges to the user as first name already exists.  3.1 System sends the information to the database and acknowledges to the user as last name already exists.  4.1 System displays ‘’please enter in the range of 6 to characters, includes at least 1 number and 1 special character ‘‘  5.1 System displays ‘’ password does not match ‘’  7.1 System checks with the database and displays the login page. |
| Post conditions | User must be logged in after successful registration. | | |
| Exception - Conditions |  | | |

4.4. Manage Profile:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC3 | | |
| Use Case Name | Manage Profile | | |
| Date created |  | Date last updated: | |
| Actor | User | | |
| Description | Users able to change all the details of his/her profile in application. | | |
| Triggering event | User selects Edit my profile button | | |
| Preconditions | User must login into application. | | |
| Flow of events | Actor | | System |
| 1. User clicks Edit profile button.  2. User change username and click save button.  3. User change password and click the save button.  4. click the save button. | | 1.1 Display edit profile page which consists of change username, password, email and etc.  2.1. Save the changes.  3.1. Save the changes  4.1 Save the changes. |
| Alternative flow of events | Actor | | System |
| 1. User clicks edit profile button.  2. User change username and click save button.  3. User change password and click the save button.  4. click the save button. | | 1.1 Display edit profile page which consists of change username, password, email and etc.  2.1. Save the changes.  3.1. Save the changes  4.1 Save the changes. Cancelled And empty the fields. |
| Post conditions | User will logout from application after checking his/her profile. | | |
| Exception - Conditions | While updating profile, the details should be correct. | | |

4.5. Add Floors:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC4 | | |
| Use Case Name | Add Floors | | |
| Date created |  | Date last updated: | |
| Actor | Admin | | |
| Description | When Admin wants to add the floor name, by clicking on submit he/she can able to see the floor list. | | |
| Triggering event | Admin clicks add floor option from home menu. | | |
| Preconditions | Admin must exist and floor details also must exist. | | |
| Flow of events | Actor | | System |
| 1. Staff click Add Floor option.  2. Enter the Floor name.  3. Click the submit option. | | 1.1. System display the select Floor option. |
| Alternative flow of events | Actor | | System |
| 1. Select the Floor. | | 1.1 Display all Floors from database related to that category.  2.1 Display a Floor on the screen with all the details available in database. |
| Post conditions | Admin can be able to add Floor name if not, he/she can logout from add Floor option. | | |
| Exception - Conditions |  | | |

4.5. Add Room:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC5 | | |
| Use Case Name | Add Room | | |
| Date created |  | Date last updated: | |
| Actor | Admin | | |
| Description | When Admin wants to select the floor name, add Room name and Add capacity and click submit. | | |
| Triggering event | Admin clicks Add Room option from home menu. | | |
| Preconditions | Admin must exist and room details also must exist. | | |
| Flow of events | Actor | | System |
| 1. Staff click Add Room option.  2. Select Floor name.  3. Add Room name.  4.Add Capacity  5.Click Submit. | | 1.1. System display the select floor, add room and add capacity option. |
| Alternative flow of events | Actor | | System |
| 1. Select the Room. | | 1.1 Display all rooms from database related to that category.  2.1 Display the room name on the screen with all the details available in database. |
| Post conditions | Admin can be able to add room name if not, he/she can logout from add room option. | | |
| Exception - Conditions |  | | |

4.6. Room Booking:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | UC6 | | |
| Use Case Name | Room Booking | | |
| Date created |  | Date last updated: | |
| Actor | User | | |
| Description | When user wants to add the room name, floor name and capacity by clicking on submit he/she can able to see the room list. | | |
| Triggering event | User clicks Room option from home menu. | | |
| Preconditions | User must exist and room details also must exist. | | |
| Flow of events | Actor | | System |
| 1. User click Room option.  2. Enter the floor name.  3. Enter the Room name.  4. Enter the Capacity ofRoom.  5. Enter the equipment.  6. Enter the Reason  7. Enter the Date  8. Click the submit option. | | 1.1. System displays the select floor option.  2.1. System displays the select  Room name option.  3.1. System displays the event equipment category and Display the hardware, software items. |
| Alternative flow of events | Actor | | System |
| 1. Cancel booking | | 1.1 Display all blocks from database related to that category.  1.2 Display the rooms on the screen with all the details available in database. |
| Post conditions | User can be able to select room if not he/she can logout from booking option. | | |
| Exception - Conditions |  | | |
|  |  | | |

**5.Use Case Description:**

**5.1. Login:**

* Admin/User will be able to login into the application using their registered username and password.
* By Clicking login, it directs to the home page of Admin/User.

**5.2. Add/Remove User:**

* Only Admin can be able to add or delete the user.
* The requirements to add the user are Name, phone number, email, username and password.

**5.3. Add/Remove Room:**

* Admin can add or delete the room.
* The requirements to add the room are floor name, room name, capacity, software, hardware and description for the room.

**5.4. Add Floor:**

* Admin can add the floor.
* The requirements to add floor is just a floor name and click submit button.
* After clicking the submit button, a new floor will be added to the floor page.

**5.5. Add default capacity of room:**

* Admin can add the default capacity of the room.
* When Admin add a room or edit a room, he can be able to add capacity for the room.

**5.6. Add equipment:**

* Admin can add the equipment to the room upon the requirement of the Staff.
* When Admin add a room or edit a room, he can be able to add equipment to the room.

**5.7. Delete equipment:**

* Admin can be able to delete the equipment from the room.
* When Admin edit a room, he can be able to delete equipment from the room.

**5.8. Add/Delete Software:**

* Admin can be able to Add or delete the software from the room depends upon the requirements of the staff.
* When Admin add a room or edit a room, he can be able to add or delete software from the room.

**5.9. Add/delete Hardware:**

* Admin can be able to Add or delete the hardware from the room depends upon the requirements of the staff.
* When Admin add a room or edit a room, he can be able to add or delete hardware from the room.

**5.10. View Bookings:**

* A user can be able to see his/her booked rooms.
* A user can cancel his/her bookings after viewing.

**5.11. View Rooms:**

* A user can be able to see the available and booked rooms on clicking the rooms button.
* The green rooms indicate the available rooms and the red rooms indicate as the booked rooms.

**5.12. Add capacity:**

* When a user is booking a room, he can add the capacity of the ongoing meeting or exam.

**5.13. Choose equipment:**

* A user can choose the equipment when he is booking a room.
* If the required equipment is not available in the room, he can able to choose the equipment from this functionality.

**5.14. Duration of meeting:**

* When booking a room, user can update the duration of the meeting or exam.

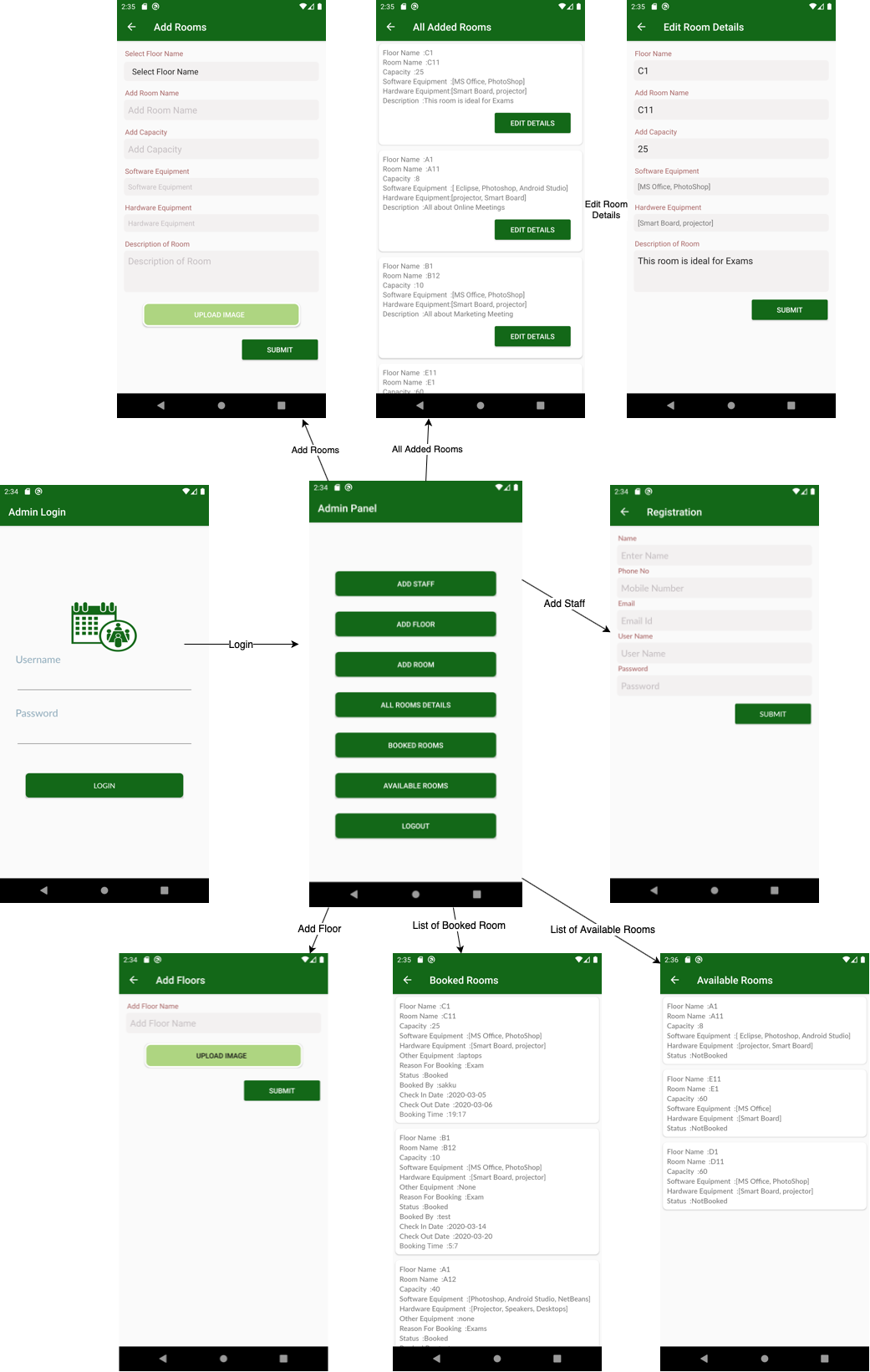
**5.15. Select date:**

* This functionality comes under room booking process.
* A user should give the date input when he wants to book the room for exam or meeting.

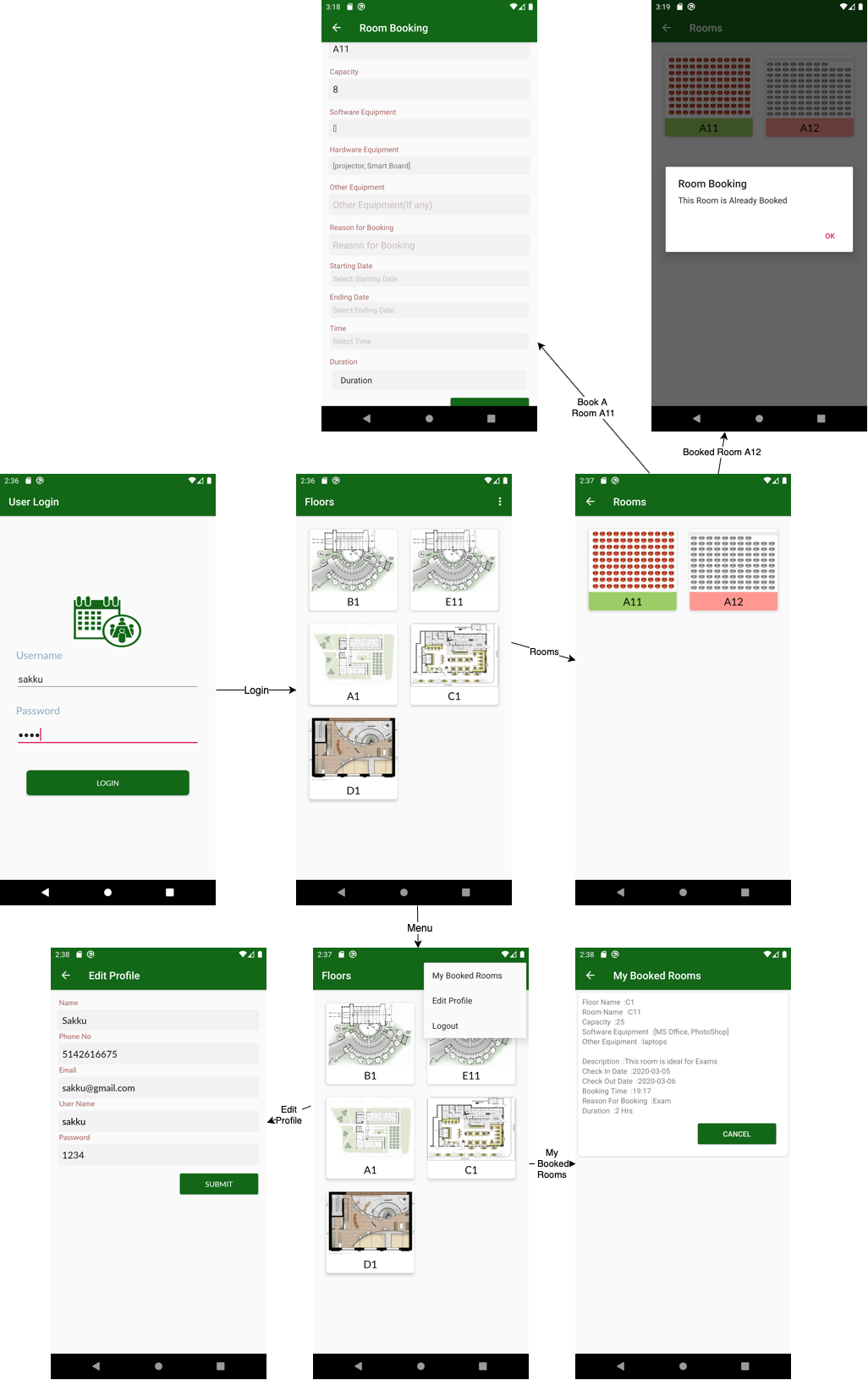
**5.16. Edit profile:**

* User can access this functionality after admin creating his account.
* User can be able to change his personal data and saved it.

6.Admin Screen flows:

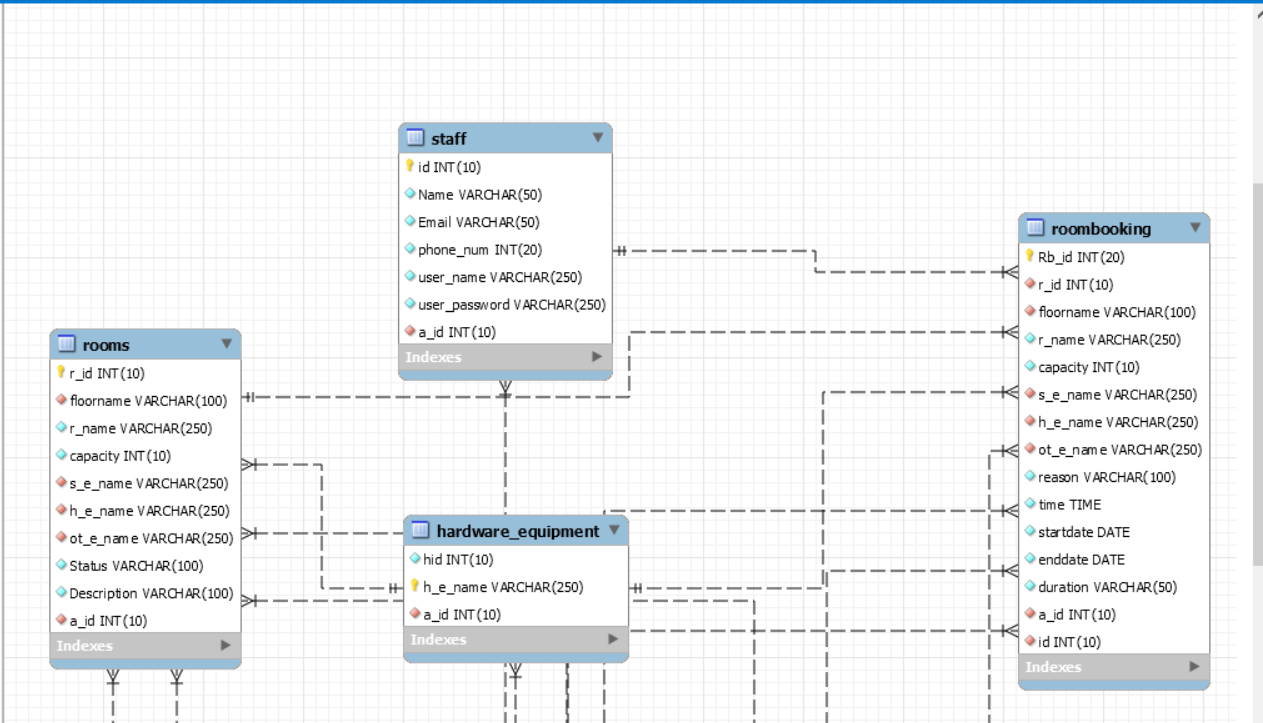
**

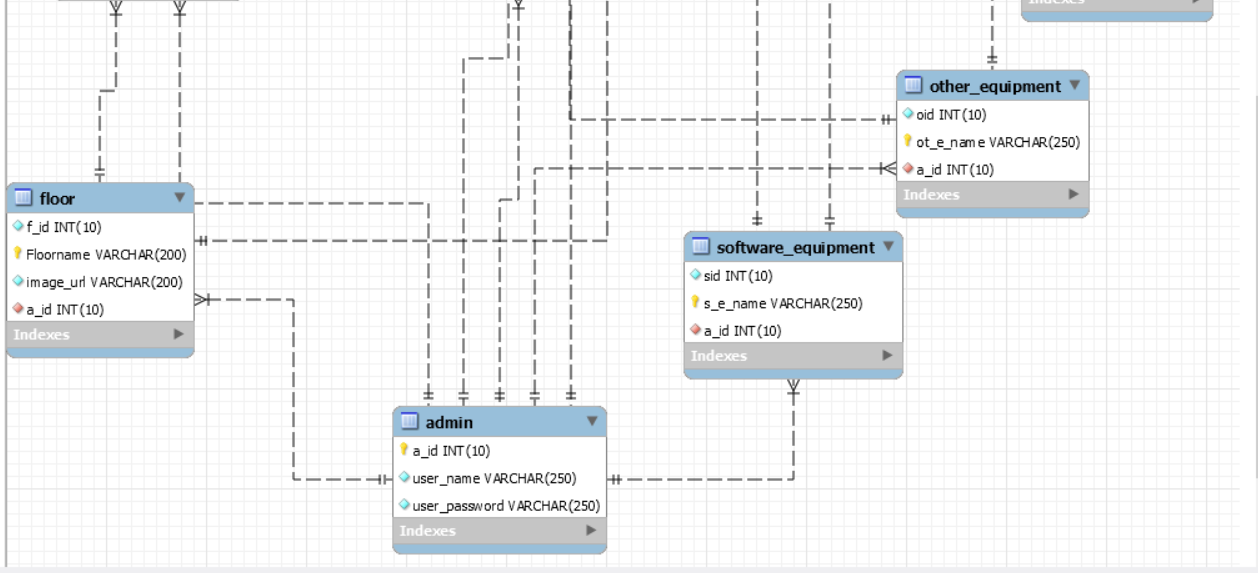
7.User Screen flows:

****

8.Database:

8.1 Data Base Schema Diagram:

****

****

8.2. Data Dictionaries:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***ADMIN*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | *Admin id* | INT | 10 | N | Contains Admin id |
| 2 | *username* | VARCHAR | 50 | N | Username for account login |
| 3 | *password* | VARCHAR | 50 | N | Password for account login |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STAFF REGISTRATION** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Staff\_id | INT | 10 | N | Contains staff id |
| 2 | *Name* | VARCHAR | 50 | N | Username for account login |
| 3 | *Phone* | VARCHAR | 50 | N | *Phone number* |
| 4 | *Email ID* | VARCHAR | 50 | N | *E-mail of Staff* |
| 5 | *Password* | VARCHAR | 50 | N | *Password for account login* |
| 6 | *Admin\_id* | INT | 10 | N | *Contains Admin id for staff registration* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***FLOORS*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Floor\_id | INT | 10 | N | Contains Floor id number |
| 2 | *Floor Name* | *VARCHAR* | *50* | N | Contains Floor name |
| 3 | *Admin\_id* | *INT* | *10* | N | Contains Admin id for adding floors |
| 4 | *imageurl* | *VARCHAR* | *50* | N | Contains Floor image url |
| 5 | *Bookingid* | *INT* | *10* | N | Contains room Booking id |
|  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***ROOMS*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Room\_id | INT | 10 | N | Contains Room id number |
| 2 | *Room Name* | *VARCHAR* | *50* | N | Contains Room name |
| 3 | *Admin\_id* | *INT* | *10* | N | Contains Admin id for adding rooms |
| 4 | *Image url* | *VARCHAR* | *50* | N | Contains image url |
| 5 | *Capacity* | *INT* | *10* | N | Capacity of Room |
| 6 | *Status* | *VARCHAR* | *50* | N | Status of room |
| 7 | *Description* | *VARCHAR* | *50* | N | Description of room |
| 8 | *Software* | *VARCHAR* | *50* | N | Contains equipped software |
| 9 | *Hardware* | *VARCHAR* | *50* | N | Contains equipped hardware |
| 10 | *Other* | *VARCHAR* | *50* | Y | Contains other equipment |
| 11 | *BookingID* | *INT* | *10* | N | Bookingid of room |
| 12 | *FloorFId* | *INT* | *10* | N | Contains floor id |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Hardware Equipment*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Equipment\_id | INT | 10 | N | Contains Equipment id |
| 2 | *Item Name* | VARCHAR | 50 | N | Contains Equipment name |
| 3 | *Status* | *VARCHAR* | *50* | N | *Status of equipment* |
| 4 | *RoomBooking\_id* | *INT* | *10* | N | *ID of room that contains equipment* |
| 5 | *Admin\_id* | *INT* | *10* | N | *Contains Admin id for adding Equipment* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Software Equipment*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Equipment\_id | INT | 10 | N | Contains Equipment id |
| 2 | *Item Name* | VARCHAR | 50 | N | Contains Equipment name |
| 3 | *Status* | *VARCHAR* | *50* | N | *Status of equipment* |
| 4 | *RoomBooking\_id* | *INT* | *10* | N | *ID of room that contains equipment* |
| 5 | *Admin\_id* | *INT* | *10* | N | *Contains Admin id for adding Equipment* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Other Equipment*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Equipment\_id | INT | 10 | N | Contains Equipment id |
| 2 | *Item Name* | VARCHAR | 50 | N | Contains Equipment name |
| 3 | *Status* | *VARCHAR* | *50* | N | *Status of equipment* |
| 4 | *RoomBooking\_id* | *INT* | *10* | N | *ID of room that contains equipment* |
| 5 | *Admin\_id* | *INT* | *10* | N | *Contains Admin id for adding Equipment* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***ROOM BOOKING*** | | | | | |
|  | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Rb\_id | INT | 10 | N | Contains Room Booking id number |
| 2 | *Floor name* | VARCHAR | 50 | N | *Contains Floor Name* |
| 3 | Room name | VARCHAR | 50 | N | *Contains Room Name* |
| 4 | *Capacity* | INT | 10 | N | *Contains Capacity of room* |
| 5 | *Software* | VARCHAR | 50 | N | *Equipped Software* |
| 6 | *Hardware* | VARCHAR | 50 | N | *Equipped hardware* |
| 7 | *Other Equipment* | VARCHAR | 50 | Y | *Other Equipment* |
| 4 | *Reason* | VARCHAR | 50 | N | *Reason for Booking* |
| 5 | *Start Date* | *DATE* | 10 | N | *Contains start date of meeting* |
| 6 | *End Date* | *DATE* | 10 | N | *Contains end date of meeting* |
| 5 | *Duration* | *TIME* | 50 | N | *Duration for how long the room is booked for* |
| 7 | *Staff\_id* | *INT* | *10* | *N* | *ID of staff who booked the room* |

### 8.3. Database Script:

drop database if exists RoombookingDB;

create database RoombookingDB;

use RoombookingDB;

/\* Admin table \*/

create table if not exists Admin(

a\_id int(10) not null,

user\_name varchar(250) not null,

user\_password varchar(250) not null,

primary key(a\_id)

);

insert into Admin(a\_id,user\_name,user\_password)

values(111,"swaran","swaran123");

insert into Admin(a\_id,user\_name,user\_password)

values(222,"surya","surya123");

/\* Staff table \*/

create table if not exists Staff(

id int(10) not null,

Name varchar(50) not null,

Email varchar(50) not null,

phone\_num int(20) not null,

user\_name varchar(250) not null,

user\_password varchar(250) not null,

a\_id int(10) not null,

primary key (id),

foreign key (a\_id) references admin (a\_id)

);

insert into Staff(id,Name,Email,phone\_num,user\_name,user\_password,a\_id)

values(1001,"sakku","sakku@gmail.com",+1514252525,"sakarvarthy","A123",111);

insert into Staff(id,Name,Email,phone\_num,user\_name,user\_password,a\_id)

values(1002,"gill","gill@gmail.com",+1514250025,"Georgia","B123",222);

/\* floor table \*/

create table if not exists floor(

f\_id int(10) not null,

Floorname varchar(200) not null,

image\_url varchar(200) not null,

a\_id int(10) not null,

primary key (Floorname),

foreign key(a\_id) references admin(a\_id)

);

insert into floor(f\_id,Floorname,image\_url,a\_id) values (0001,"A1","image1.png",111);

insert into floor(f\_id,Floorname,image\_url,a\_id) values (0002,"B1","image2.png",222);

/\* Software\_Equipment table \*/

create table if not exists Software\_Equipment(

sid int(10) not null,

s\_e\_name varchar(250) not null,

a\_id int(10) not null,

primary key (s\_e\_name),

foreign key (a\_id) references admin(a\_id)

);

insert into Software\_Equipment(sid,s\_e\_name,a\_id) values (505,"photoshop,Netbeans",111);

insert into Software\_Equipment(sid,s\_e\_name,a\_id) values (506,"Eclipse and python idle",222);

/\* Hardware\_Equipment table\*/

create table if not exists Hardware\_Equipment(

hid int(10) not null,

h\_e\_name varchar(250) not null,

a\_id int(10) not null,

primary key (h\_e\_name),

foreign key (a\_id) references admin(a\_id)

);

insert into Hardware\_Equipment(hid,h\_e\_name,a\_id) values (605,"Laptops and board",111);

insert into Hardware\_Equipment(hid,h\_e\_name,a\_id) values (606,"Laptops and chairs",222);

/\* other equipment table \*/

create table if not exists Other\_Equipment(

oid int(10) not null,

ot\_e\_name varchar(250) not null,

a\_id int(10) not null,

primary key (ot\_e\_name) ,

foreign key (a\_id) references admin(a\_id)

);

insert into Other\_Equipment(oid,ot\_e\_name,a\_id) values (705,"chairs and tables",111);

insert into Other\_Equipment(oid,ot\_e\_name,a\_id) values (706,"tables and air conditioners",222);

/\* Rooms table \*/

create table if not exists Rooms(

r\_id int(10) not null,

floorname varchar(100) not null,

r\_name varchar(250) not null,

capacity int(10) not null,

s\_e\_name varchar(250) not null,

h\_e\_name varchar(250) not null,

ot\_e\_name varchar(250) not null,

Status varchar(100) not null,

Description varchar(100) not null,

a\_id int(10) not null,

primary key (r\_id),

foreign key (a\_id) references admin(a\_id),

foreign key (floorname) references floor(floorname),

foreign key (s\_e\_name) references Software\_Equipment(s\_e\_name),

foreign key (h\_e\_name) references Hardware\_Equipment(h\_e\_name),

foreign key (ot\_e\_name) references Other\_Equipment(ot\_e\_name)

);

insert into Rooms(r\_id,floorname,r\_name,capacity,s\_e\_name,h\_e\_name,ot\_e\_name,Status,Description,a\_id)

values (1212,"A1","lab101",100,"photoshop,Netbeans","Laptops and board","chairs and tables","available/not available","Exam purpopse",111);

insert into Rooms(r\_id,floorname,r\_name,capacity,s\_e\_name,h\_e\_name,ot\_e\_name,Status,Description,a\_id)

values (1313,"B1","lab201",150,"Eclipse and python idle","laptops and chairs","tables and air conditioners","available/not available","Exam/meeting purpose",222);

/\* Room booking details table \*/

create table if not exists Roombooking(

Rb\_id int(20) not null,

r\_id int(10) not null,

floorname varchar(100) not null,

r\_name varchar(250) not null,

capacity int(10) not null,

s\_e\_name varchar(250) not null,

h\_e\_name varchar(250) not null,

ot\_e\_name varchar(250) not null,

reason varchar(100) not null,

time time not null,

startdate date not null,

enddate date not null,

duration varchar(50) not null,

a\_id int(10) not null,

id int(10) not null,

primary key (Rb\_id),

foreign key (a\_id) references admin(a\_id),

foreign key (r\_id) references Rooms(r\_id),

foreign key (floorname) references floor(floorname),

foreign key (s\_e\_name) references Software\_Equipment(s\_e\_name),

foreign key (h\_e\_name) references Hardware\_Equipment(h\_e\_name),

foreign key (ot\_e\_name) references Other\_Equipment(ot\_e\_name),

foreign key (id) references Staff (id)

);

insert into Roombooking(Rb\_id,r\_id,Floorname,r\_name,capacity,s\_e\_name,h\_e\_name,ot\_e\_name,Reason,time,startdate,enddate,Duration,a\_id,id)

Values (1111,1212,"A1","Lab101",100,"photoshop,Netbeans","Laptops and board","chairs and tables","Exam","04:00:00","2020-02-10","2020-02-12","3 hrs per day",111,1001);

insert into Roombooking(Rb\_id,r\_id,Floorname,r\_name,capacity,s\_e\_name,h\_e\_name,ot\_e\_name,Reason,time,startdate,enddate,Duration,a\_id,id)

Values (2222,1313,"B1","Lab201",150,"Eclipse and python idle","laptops and chairs","tables and air conditioners","Exam/meeting","10:30:00","2020-02-11","2020-02-15","5 hrs per day",222,1002);