****



**Table of Contents**

1. **Introduction**…………………………………………………………………………3
   1. Purpose…………………………………………………………………………...3
   2. Scope……………………………………………………………………………..3
2. **General Description**....……………………………………………………………...4
   1. Product Functions….………………………………………………………….....5
   2. User Characteristics………………………………………………………….......5
3. **Specific Requirements**……………………………………………………………...5
   1. Functional Requirements….……………………………………………………..6
   2. Non-Functional Requirements…………………………………………………...9
4. **Analysis Models**……………………………………………………………………10
   1. Use Case Diagram………………………………………………………………11
   2. Class Diagram…………………………………………………………………..12
5. **Screens**………………….…………………………………………………………..13
6. **Database** ……………..……………………….…….………….….……….....……17
   1. Logical Model.………………….………....…………..………...……...……....17
   2. ER Diagram…....……………………………………………………………….18
   3. Database Script….………………………………………….………………......18
7. **Use Case Scenarios**………....…….…………………………………….………....22
8. **References**……………………………………………………………….………...31
9. **Glossary**…………………………………………………………………….…..…31
10. **Introduction**

Room Booking Mobile Application is a modern digital booking system that enables Teachers, Pedagogical Counsellors as well as other Staff members in CEGEP’s Montreal campus to book a room either for an exam or a meeting. Room Booking Application is essential in an office environment because scheduling the shared spaces in a busy office can be a challenge. So this application is specially designed for the purpose of self-service booking. This application is an easy-to-use room booking tool with a simple and organized overview of the availability of the rooms to prevent miscommunication and scheduling conflicts.

* 1. **Purpose**

The main purpose of this application is to make it easy for users to find a perfect room for upcoming meetings or for exams based on it’s capacity, amenities like hardware equipment and software equipment. This application improves the utilization of rooms with a hassle-free environment and also provides the platform for flexible room management.

* 1. **Scope**

1. The mobile application that we will design will be identified by the name *Room Booking Application*.
2. The objective of this application is to provide a better place to manage the rooms and utilize the room space effectively.
3. Teachers, Pedagogical Counsellors and other staff members in CEGEP’s Montreal campus will be able to use this application to book a room either for an exam or for a meeting.
4. This app is available on the mobile devices that run on Android OS.
5. Features that will be implemented in this application:

* This application will list all the available rooms, to the user.
* Users will book a room for an exam or for a meeting with just 2 clicks of a button.
* Users will see the capacity of the available rooms listed to them.
* Users will see the software equipment and hardware equipment in each available room.
* Users will be able to cancel the booked room.

1. Future Scope of this application:

* This application can allow users to report faulty equipment in the booked room.
* Users can customise the meeting by requesting for additional equipment, users can name their meeting.
* Admins of the application can add feedback options and get feedback from users after they utilize the room space. It will allow admins to manage the room space better.
* Users can send invitations to students or other colleagues to attend the exam or meeting.
* Users can get reminder notifications before the meeting.
* The rooms booked by the users can be cancelled automatically if they didn’t check-in within the timeframe.To do this we can set up a QR code scanner which will allow users to check-in into their booked room by scanning in their mobile.

1. **General Description**

Room Booking App is a modern digital booking system designed for managing, booking exam rooms and conference rooms in CEGEP montreal campus. Smart technology enables effective management and planning of college meetings, makes college processes simpler and helps to use company space in a better way. Colleges are often facing problems with availability of meeting rooms; either they have to deal with overbooked rooms or unused spaces. This can cause confusion and lost time. Thanks to this application you will have an instant overview of the availability of a particular room – simple, easy and stress-free. Users can book a meeting directly on their mobile on their way to work. It takes just a tap or two. All information about rooms availability, booked rooms are centralized and digitized.

* 1. **Product Functions**

Room Booking Application should be an easy to use application which will offer a user friendly process for booking a room. From booking rooms to updating profile details, cancelling booked rooms, viewing available rooms can be done by the user with few computer clicks. Additionally the admin can also add new rooms, view booked rooms, view available rooms, Edit room details within this app.

Room Booking application is supposed to be a mobile based application.

* 1. **User Characteristics**

Room booking application will target different kind of users in CEGEP’s Montreal campus:

* Teachers
* Pedagogical counsellors
* Other staff members in CEGEP’s Montreal campus

1. **Specific Requirements**

The requirements of this mobile application outlines the needs of the users who will use this app. It compiles all the below-listed functional and non-functional requirements and becomes the base reference document for our entire development team.

These requirements list out the technical specifications, and the list of assets(hardware, Third-party softwares) connected with the app.

* 1. **Functional Requirements**

Functional requirements specify a function that this application must be able to perform. It can be documented in various ways. It may be Technical details, Data processing and Other specific functionality.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement ID** | **Requirement Statement** | **MUST/SHOULD/COULD/WOULD** | **Comments** |
| FR1 | The application shall have a login screen as a home page. | MUST | The system shall verify the credentials with the database. |
| FR2 | The application shall have a registration page, where the user can enter the details. | MUST | The system shall save the data within the database. |
| FR3 | The Users shall be able to edit/update their profile details. | COULD | The system shall update the profile details of the user in the database. |
| FR4 | The application allows the admin/user to view the list of available rooms. | MUST | The system shall fetch the data of rooms available from the database. |
| FR5 | Users can book a room based on purpose of booking, capacity, hardware/ software requirements. | MUST | The system shall book a room according to the user requirement. |
| FR6 | Users can see their booked rooms, date and duration of exam or meeting. | MUST | The system shall fetch the booked rooms by the user from the database. |
| FR7 | Users can cancel the rooms that are already booked by them. | MUST | The system shall update the room details once the room is cancelled by the booked user. |
| FR8 | The User/Admin shall have the option of Log out. | MUST | The system shall log out the user from the application. |
| FR9 | The application allows the admin to add the rooms, hardware/software equipment. | MUST | The system shall add rooms, hardware/software equipment in the database. |
| FR10 | The application allows the admin to cancel the booked rooms. | MUST | The system shall cancel room booking and update it’s details in the database. |
| FR11 | Admin can edit/update the room details (Room number, Room capacity, Hardware equipment, Software equipment). | MUST | The system shall update the room details in the database. |
| FR12 | Admin can view the booked rooms. | SHOULD | The system shall fetch the details of all booked rooms from the database. |
| FR13 | Admin can view the list of all the rooms in this application (both available & booked rooms). | SHOULD | The system shall fetch the details of all rooms from the database. |
| FR14 | Allow admin to delete the rooms from the app. | MUST | The system shall delete the room details from the database. |
| FR15 | Users can select start date and end date for room booking. | MUST | The system shall display the calendar which allows the user to select dates. |
| FR16 | Users can select start time and end time. | MUST | The system shall display the dropdown list which allows the user to select the timings. |
| FR17 | Duration of the booking will be calculated automatically. | MUST | The system shall calculate the duration of the room booked; based on start time, end time, start date, end date values that are selected by the user. |
| FR18 | Users must be able to see the status of the room booked by them. | MUST | The system shall display the status of the booking room as confirmed, only if no other user has booked that room in the same slot prior to the current user.  If the first user booked a room in a particular time slot, and the second user wants to book the room within the same time slot, it should display the booking status as waiting. Once the first user cancels the booked room, then it will be allocated to the second user and their booking status will be changed to confirmed. |
| FR19 | Users can request for the extra equipment either for meeting or an exam. | SHOULD | The system shall allow users to enter the details of equipment needed by them in their booked room. |

**3.2 Non-Functional Requirements**

Non-Functional Requirements specify the quality attribute of this system. These requirements specify the criteria that can be used to judge the operation of a system, rather than specific behaviors. Functional requirements specify what a system is supposed to do and these Non-Functional requirements specify how a system is supposed to be.

Non-Functional requirements for our system:

**Availability:**

NFR1: The app must be available for 24 hours a day, 7 days a week.

**Reliability:**

NFR2: The app must provide acknowledgement to the user, when the user books a room or cancels a booked room.

**Reusability:**

NFR3: We would use the same codefor the navigation drawer provided for the admin interfaces and user interfaces.

**Throughput:**

NFR4: Users should be able to navigate quickly between screens( maximum navigation time is 5 seconds).

NFR5: The app must have quick response time for every instruction given in this app( maximum response time is 3 seconds).

**Performance:**

NFR6: This App should be able to handle 50 users at a time without affecting its performance or accuracy.

**Quality:**

NFR7: The content in the app must be comprehensible.

NFR8: The app interface should be alluring to the user.

1. **Analysis Models**

We are using three analysis and design models to develop our requirements into a mobile application.

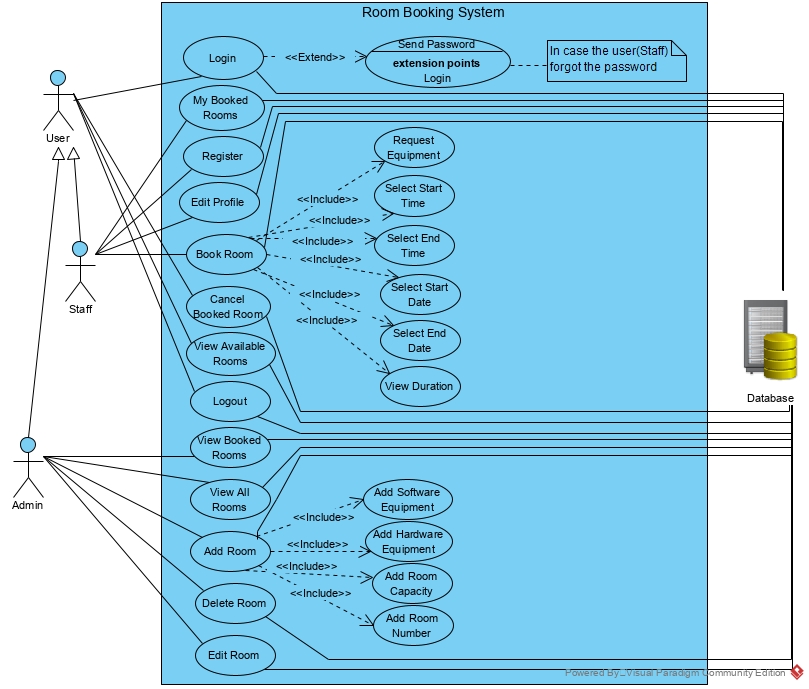
**Use case Diagram**

A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system.

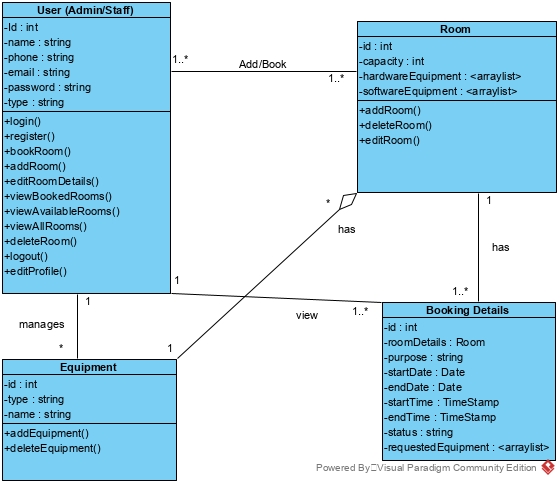
**Class Diagram**

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. The class diagram is the main building block of object-oriented modelling.

* 1. **Use case Diagram**

****

* 1. **Class Diagram**



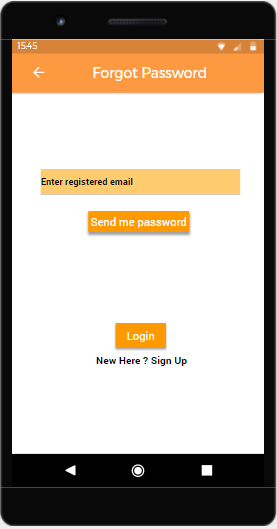
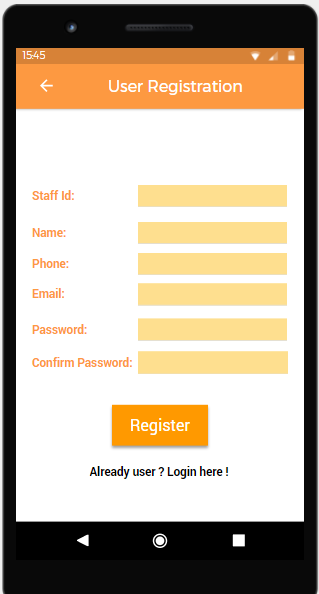
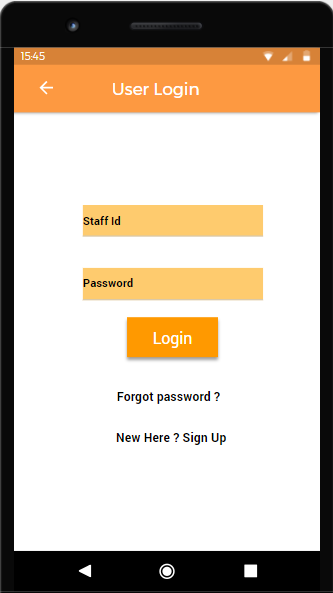
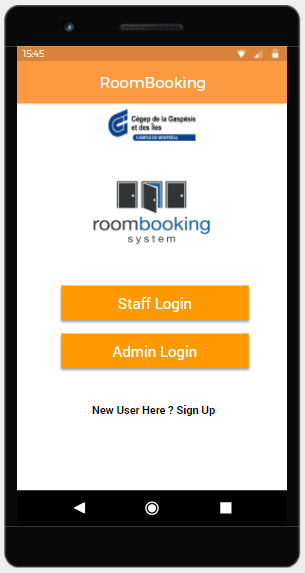
1. **Screens**

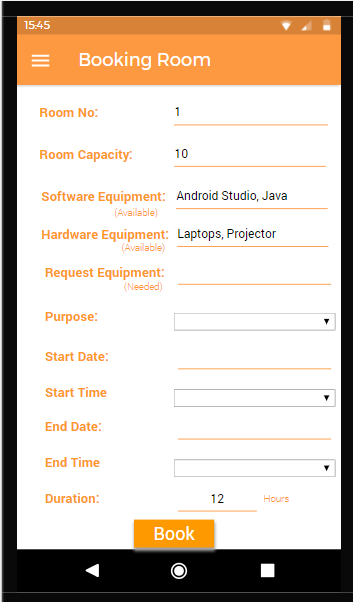
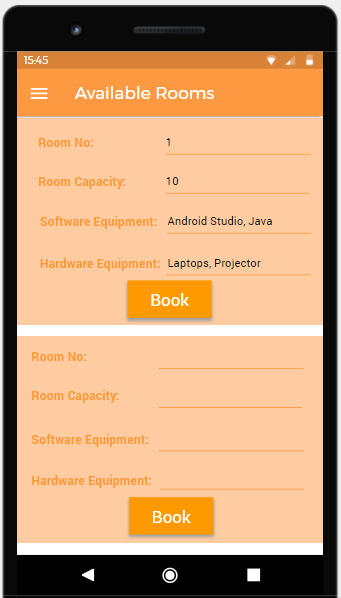
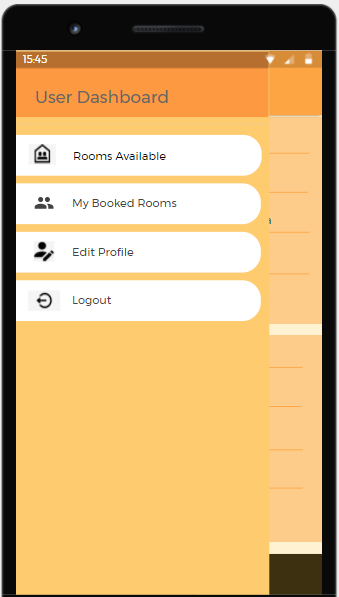
**Screen flow diagram**

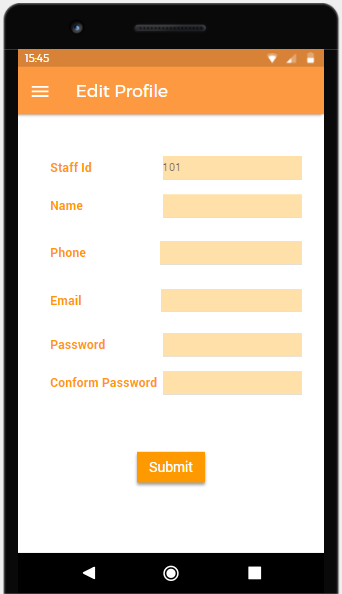
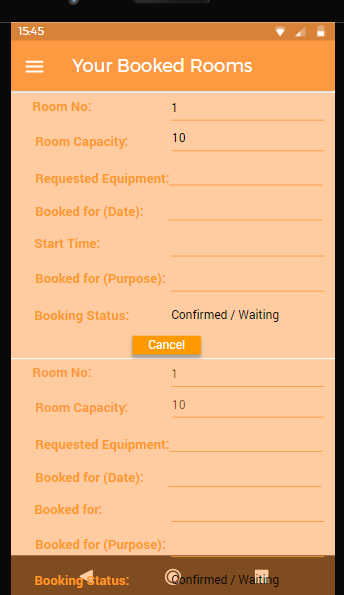
****

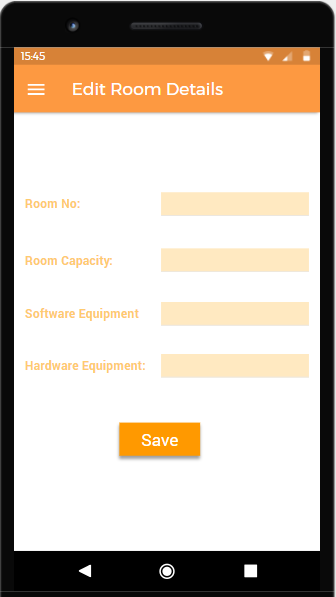
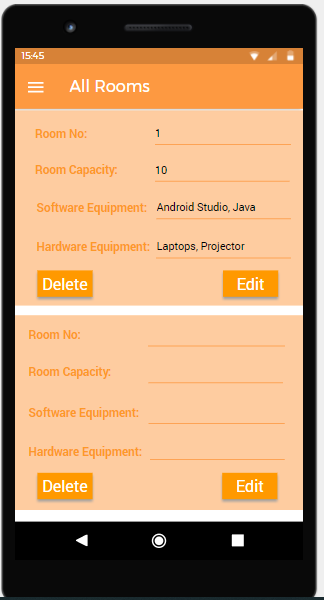
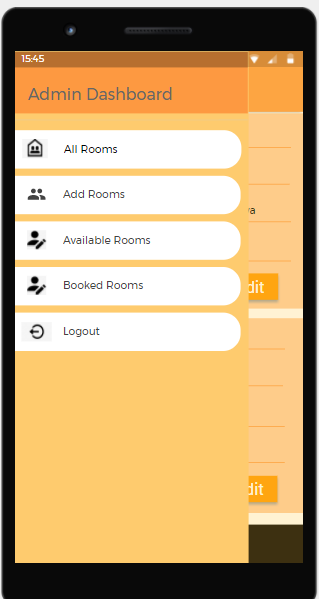
**Screen Design**

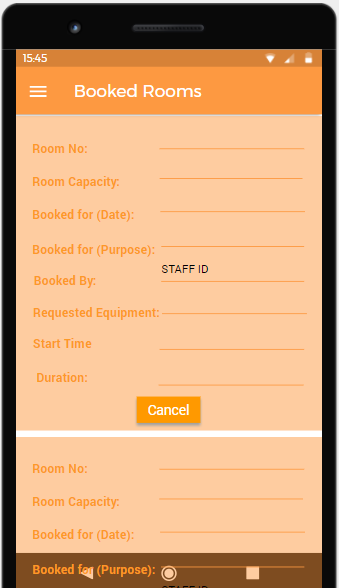
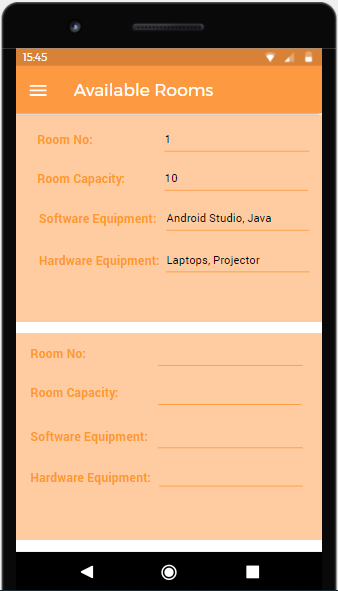
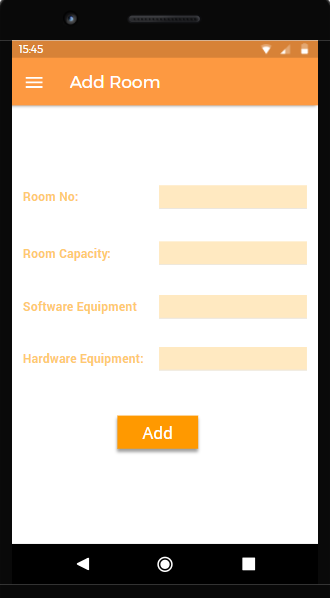
We can simulate the screens of the app in this tool and verify how it looks when the App is designed. We use drag and drop widgets, screens to design the screens. We used Justinmind Prototyper 8.7.7 to design the Screens. We can simulate these screens with this tool or we can view on our mobile device by connecting it.





****

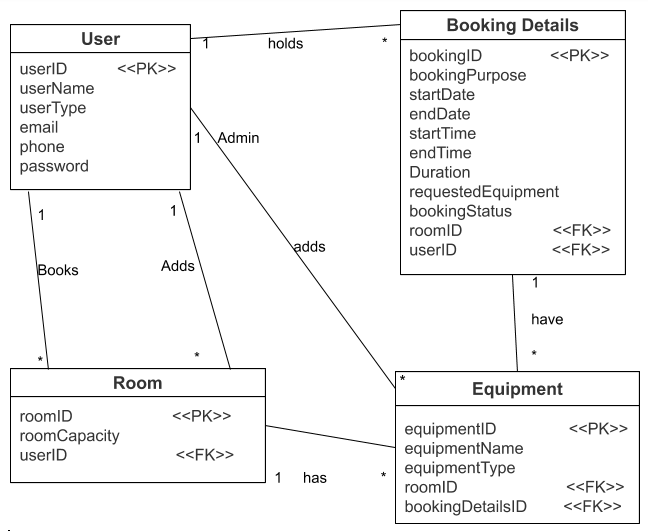
****



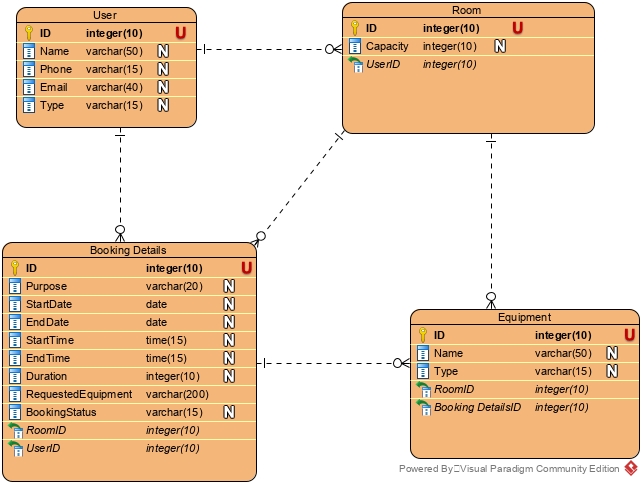
1. **Database**

Database is absolutely an integral part of software systems. To fully utilize ER Diagram in database engineering guarantees you to produce high-quality database design to use in database creation, management, and maintenance. An ER model also provides a means for communication.

* 1. **Logical Model**

****

* 1. **ER Diagram**



* 1. **Database Script**

|  |
| --- |
| create database RoomBookingSystemDB2;  Use RoomBookingSystemDB2;  create table user(  id int(10) not null,  name varchar(50) not null,  phone varchar(15) not null,  email varchar(40) not null,  type varchar(15) not null,  password varchar(30) not null,  primary key (id)  );  insert into user(id,name,phone,email,type,password) values(1000,"tagore","+15142699000","tagore@gmail.com","admin","Tagore@1993");  insert into user(id,name,phone,email,type,password) values(2000,"tanya","+15142568000","tanya@gmail.com","admin","Tanya@1994");  insert into user(id,name,phone,email,type,password) values(3000,"arshdeep","+15148906000","arshdeep@gmail.com","admin","Arshdeep@1995");  insert into user(id,name,phone,email,type,password) values(9001,"akhil","+15147896000","akhil@gmail.com","staff","Akhil@9001");  insert into user(id,name,phone,email,type,password) values(9002,"mahesh","+15147896111","mahesh@gmail.com","staff","Mahesh@9002");  insert into user(id,name,phone,email,type,password) values(9003,"aditya","+15147896222","aditya@gmail.com","staff","Aditya@9003");  create table equipment(  id int(10) not null,  name varchar(50) not null,  type varchar(15) not null,  room\_id int(10),  booking\_details\_id int(10),  primary key (id),  foreign key(room\_id) references room(id),  foreign key(booking\_details\_id) references booking\_details(id)  );  insert into equipment(id,name,type,room\_id,booking\_details\_id) values (1,"android studio","software",2001,5001);  insert into equipment(id,name,type,room\_id) values (2,"mysql work bench","software",2002);  insert into equipment(id,name,type,room\_id,booking\_details\_id) values (3,"docker","software",2001,5001);  insert into equipment(id,name,type,room\_id,booking\_details\_id) values (101,"laptops","hardware",2001,5001);  insert into equipment(id,name,type,room\_id) values (102,"projectors","hardware",2002);  insert into equipment(id,name,type,room\_id,booking\_details\_id) values (103,"printers","hardware",2002,5002);  create table room(  id int(10) not null,  capacity int(10) not null,  primary key (id)  );  insert into room(id,capacity)values (2001,20);  insert into room(id,capacity)values (2002,30);  insert into room(id,capacity)values (2003,10);  insert into room(id,capacity)values (2004,50);  create table booking\_details(  id int(10) not null,  purpose varchar(20) not null,  startdate date not null,  enddate date not null,  starttime time not null,  endtime time not null,  duration int(10) not null,  requested\_equipment varchar(100) not null,  booking\_status varchar(15) not null,  roomid int(10) not null,  userid int(10) not null,  primary key (id),  foreign key(roomid) references room(id),  foreign key(userid) references user(id)  );    insert into booking\_details(id,purpose,startdate,enddate,starttime,endtime,duration,requested\_equipment,booking\_status,roomid,userid)  Values (5001,"exam","2020-05-16","2020-05-17","18:00:00","06:00:00",12,"Chairs","Confirmed",2001,9001);  insert into booking\_details(id,purpose,startdate,enddate,starttime,endtime,duration,requested\_equipment,booking\_status,roomid,userid)  Values (5002,"meeting","2020-05-17","2020-05-18","12:00:00","06:00:00",18,"Fans","Waiting",2002,9002);  select \* from equipment e  inner join booking\_details bkdt on e.booking\_details\_id = bkdt.id  inner join room r on e.room\_id = r.id where bkdt.id=5001;  select \*, GROUP\_CONCAT(name SEPARATOR ',') as equipments from (select e.id as eq\_id, bkdt.id as bkd\_id, e.type, e.name from equipment as e  inner join booking\_details bkdt on e.booking\_details\_id = bkdt.id  inner join room as r on e.room\_id = r.id where bkdt.id=5001) as T group by type; |

1. **Use Case Scenarios**

**UC-7.1: Register**

|  |  |  |
| --- | --- | --- |
| System: | A Mobile Application – Room Booking System | |
| Identifier: | UC-1 | |
| Name: | Register New User | |
| Pre-Condition(s): | The user is non-registered and does not have an account on Room Booking Application | |
| Post- Condition(s): | The user is directed to the Login Page of the Mobile application. | |
| Trigger: | The user clicks on the Register button. | |
| Normal Flow: | User | System |
| 1. The user enters the details in the form and sets up the password. 2. User submit the Register form. | 2.1 System sends all the details to the database.  2.2 System checks with the database whether the details already exist or not.  2.3 If details already exist in the database, the system will display a prompt message that the user is already registered.  2.4 If the user is not registered earlier, then the system stores the data in the database.  2.5 Displays User is registered successfully.  2.6 Displays login page. |
| Alternate Flow: | None | |
| Exceptional Flow(s): | **Exception:** There is an occurrence of internet connection failure.  1. The system will display the message “No Internet Connection, try Again”. | |
| Related Actor(s): | Non-Registered User(Staff) | |
| Related Use Case(s): | Login | |

Table 7.1. UC-1: Register New User

**UC-7.2: Login**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-2 | | |
| Name: | Login | | |
| Pre-Condition(s): | The Actor is registered and has an account on Room Booking application. | | |
| Post- Condition(s): | The Actor is directed to the Room Booking page specific to the user/admin. | | |
| Trigger: | The Actor has clicked on the Login button. | | |
| Normal Flow: | User | System | |
| 1. The User enters the email and the password in the form. 2. User clicks the login button. | 1.1 System sends the details given by the user, both Email and password to the database.  1.2 System validate user details and  Database authenticates the user input details whether it is staff or admin  2.1 System displays the dashboard screen to the user. | |
| Alternate Flow: | User | | System |
| 1. User clicks on “forgot password”. 2. The user is directed to the “forgot password” screen. 3. User entered their registered email id. 4. User clicks on “Send me Password” 5. User navigates to the login screen again & enters the received password. | | 3.1 System sends the user entered email id to the database.  3.2 Database checks the email, if the details already exists, system displays “password sent to your email”.  3.3 If the entered email is wrong, the system prompts the error message “Email not registered!”.  3.3 System empty the input Field. |
| Exceptional Flow(s): | **Exception:** The user enters the wrong Password or User Id.  1. The system will prompt the message “Invalid User id or password, try again”. | | |
| Related Actor(s): | User(Staff or Admin) | | |
| Related Use Case(s): | Register | | |

Table 7.2. UC-2: Login

**UC-7.3 – Edit Profile**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-3 | | |
| Name: | Edit Profile | | |
| Pre-Condition(s): | 1. The user is registered and has an account on Room Booking Application. 2. User logged in with the correct id and password. | | |
| Post- Condition(s): | The user can log out from the system after going through their profile. | | |
| Trigger: | The User has clicked on the Edit Profile. | | |
| Normal Flow: | User | System | |
| 1. User clicks on the edit profile button. 2. User can change individual fields like Name, Phone, Email, and Password. 3. User clicks on the submit button. | 1.1 System displays edit profile screen which consists of change name, phone, email, password.  3.1 Changes made will be sent to the database by the system.  3.2 Confirmation message will be displayed by the system as “Changes are successful” | |
| Alternate Flow: | None | | |
|
| Exceptional Flow(s): | **Exception:** The User enters different values for password and confirm password.  1. The system will display the message “Please confirm the password”. | | |
| Related Actor(s): | User (Staff) | | |
| Related Use Case(s): | Login | | |

Table 7.3. UC-3: Edit profile

**UC-7.4: View Rooms Available**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-4 | | |
| Name: | View Rooms Available | | |
| Pre-Condition(s): | The user must be registered.  Room details must exist in the database. | | |
| Post- Condition(s): | The User can be able to view the available rooms in the system. | | |
| Trigger: | The User has clicked on the “Rooms Available” in the dashboard. | | |
| Normal Flow: | User | System | |
| 1. User navigates to their dashboard.  2. User clicks on the “Rooms available”. | 2.1 System displays the list of the rooms available for booking.  2.2 System displays the room number, room capacity, software equipment available, hardware equipment available for the rooms. | |
| Alternate Flow: | User | | System |
| 1. User clicks the back button. | | 1.1 System display User dashboard. |
| Exceptional Flow(s): | None | | |
| Related Actor(s): | User (Admin or Staff) | | |
| Related Use Case(s): | Login | | |

Table 7.4. UC-4: Rooms available.

**UC-7.5: Book Room**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-5 | | |
| Name: | Book Room | | |
| Pre-Condition(s): | The user must be a registered user and room details must exist in the database. | | |
| Post- Condition(s): | The User can be able to view the booked room details and the booking status of the room. | | |
| Trigger: | The user has clicked on the “Book” button. | | |
| Normal Flow: | User | System | |
| 1. The user clicks on the “Book” button. 2. The room id, capacity, hardware equipment and software equipment available in the room will be displayed 3. User enters the purpose of the booking. 4. User enters the request equipment. 5. User enters the Start date, End date. 6. User enters the Start time, End time. 7. Clicks on the book option. | * 1. System displays the select request equipment option.   2. System displays the select purpose option.   3. System displays the select start date option.   4. System displays the select start time option.   5. System displays the select end date option.   6. System displays the select end time option.   7.1System calculates booking duration and will be displayed in the duration field.  7.2 Displays prompt message “Successfully Booked”. | |
| .Alternate Flow: | User | | System |
| 1. User clicks on the back Button. | | 1.1 System displays User dashboard |
| Exceptional Flow(s): | **Exception:** The User gave wrong details while booking the room.  1. The User can cancel the booked room and book the same room again. | | |
| Related Actor(s): | User (Staff) | | |
| Related Use Case(s): | Login | | |

Table 7.5. UC-5: Book Room

**UC-7.6: Add Rooms**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-6 | | |
| Name: | Add rooms | | |
| Pre-Condition(s): | Admin details must exist in the database.  Admin must be logged in into the system.. | | |
| Post- Condition(s): | Rooms will be added to the list of available rooms in the database. | | |
| Trigger: | Admin select add rooms from admin dashboard. | | |
| Normal Flow: | Admin | System | |
| 1. Admin clicks on Add Rooms in the dashboard. 2. Admin can add a room number. 3. Admin can add room capacity. 4. Admin can add software and hardware equipment details. 5. Admin select submit button. | * 1. System displays the screen with the empty fields to add room number, room capacity, software equipment, hardware equipment.   2. Displays prompt message “Rooms added successfully”. | |
| Alternate Flow: | Admin | | System |
| 1. Admin can select the back button. | | 1.1 System displays Admin dashboard’s first screen. |
| Exceptional Flow(s): | None | | |
| Related Actor(s): | Admin | | |
| Related Use Case(s): | Login | | |

Table 7.6. UC-6: Add Rooms

**UC-7.7: Delete Rooms**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-7 | | |
| Name: | Delete Rooms | | |
| Pre-Condition(s): | Admin must exist and Room details also must exist in the database. | | |
| Post- Condition(s): | Admin will be redirected to the list of the rooms and can see the updated details with the room deleted. | | |
| Trigger: | The admin has clicked on the “Delete” button. | | |
| Normal Flow: | Admin | System | |
| 1. The Admin clicks on the “Delete” button. 2. Admin deletes the room. 3. The room details will be deleted from the list of rooms. | 3.1 System displays prompt message “Room successfully deleted: | |
| Alternate Flow: | Admin | | System |
| 1. Admin can select the back button. | | 1.1 System displays Admin dashboard. |
| Exceptional Flow(s): | None | | |
| Related Actor(s): | Admin | | |
| Related Use Case(s): | Login. | | |

Table 7.7. UC-7: Delete Rooms

**UC-7.8: Edit Rooms**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-8 | | |
| Name: | Edit Rooms | | |
| Pre-Condition(s): | Admin must exist, Admin must login and Room details also must exist. | | |
| Post- Condition(s): | Admin can see the updated room details in the list of rooms. | | |
| Trigger: | The admin has clicked on the “Edit” button. | | |
| Normal Flow: | Admin | System | |
| 1. The Admin clicks on the “Edit” button.   1. Admin edits the room number. 2. Admin edits the software equipment. 3. Admin edits the hardware equipment. 4. Clicks the save button | * 1. System displays updated details of the room number, software equipment, and hardware equipment   5.2 Displays a prompt message “Successfully Edited” | |
| Alternate Flow: | Admin | | System |
| 1. Admin can select the back button. | | 1.1 System displays Admin dashboard’s first screen. |
| Exceptional Flow(s): | 1. A room is already booked by the user. 2. The Admin clicks on the “Edit” button of that room. | 2.1 System displays the message “The room is booked!” | |
| Related Actor(s): | Admin | | |
| Related Use Case(s): | Login. | | |

Table 7.8. UC-8: Edit Rooms

**UC-7.9: Logout**

|  |  |  |  |
| --- | --- | --- | --- |
| System: | A Mobile Application – Room Booking System | | |
| Identifier: | UC-8 | | |
| Name: | Logout | | |
| Pre-Condition(s): | Users must have their credentials registered in the database. | | |
| Post- Condition(s): | The user is directed to the Main screen with the login button. | | |
| Trigger: | The actor has clicked on “Logout” in their dashboard | | |
| Normal Flow: | Admin | System | |
| 1. User clicks on the Logout. | 1.1 The system logouts the actor out of the system,  1.2 User will be redirected to Initial screen. | |
| Alternate Flow: | None | | |
|
| Exceptional Flow(s): | None | | |
| Related Actor(s): | User (Staff or Admin) | | |
| Related Use Case(s): | Login. | | |

Table 7.9. UC-9: Logout

1. **References**

<https://en.wikipedia.org/wiki/Non-functional_requirement>

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/>

<https://www.geeksforgeeks.org/association-composition-aggregation-java/>

<https://docs.oracle.com/javase/8/docs/api/java/util/Date.html>

<https://www.tutorialspoint.com/how-to-compare-two-dates-in-java>

<https://creately.com/blog/diagrams/class-diagram-relationships/>

<http://agiledata.org/essays/dataModeling101.html>

<http://www.agiledata.org/essays/umlDataModelingProfile.html>

1. **Glossary**

|  |  |
| --- | --- |
| **App** | Application |
| **System** | Mobile Application |
| **User** | Admin/ Teachers/ Pedagogical Counsellors/ Other Staff members |