PROJECT REPORT ON

Auditorium Booking System

MES Institute of Management & Career Courses, Pune

Submitted to

Shilpa Paralikar

Submitted By

Rohan Khare

1711033

Maharashtra Education Society's

INSTITUTE OF MANAGEMENT AND CAREER COURSES (IMCC)

(Recognized by Savitribai Phule Pune University & Approved by AICTE)

131, Mayur Colony, Kothrud Pune 411 038.

Tel. +91-20-25466271, 25463453 • E-mail: director.imcc@mespune.in

DR. SANTOSH DESHPANDE
Director
Web Site: https://imcc.mespune.in

Ref. No : MCA/Project/023/2020-21

Date : 14/09/2020

CERTIFICATE

This is to certify that the Project Report entitled ***"Auditorium Booking System"*** is prepared by ***Rohan Praphulla Khare ,*** a student of ***M.C.A.*** Course for the Academic Year 2019-20 at M.E.Society's Institute of Management & Career Courses (IMCC), Pune - 411 038. M.C.A Course is affiliated to Savitribai Phule Pune University.

To the best of our knowledge, this is original study done by the said student and important sources used by him have been duly acknowledged in this report.

The report is submitted in partial fulfilment of M.C.A. Course for the Academic Year 2019-20 as per the rules & prescribed guidelines of Savitribai Phule Pune University.

Dr. Ravindra Vaidya ***Dr. Santosh Deshpande***

HOD, Department of MCA, IMCC ***Director, IMCC***

Title:

Name: Rohan Khare

Project Name: Auditorium Booking System

INDEX

Chapter 1: Introduction

1.1 Company Profile

1.2 Existing system and need for system:

Firstly, booking process of an auditorium is completely manual in which A customer personally has to visit auditorium office to book an auditorium.

Secondly, there is no online system available to check availability of an auditorium on particular date and time. Problems faced in the current system are:

- A customer has to visit office for booking process.
- A customer cannot check availability of an auditorium at any time.

Need for system:

- Checking availability of auditoriums.
- Online Auditorium booking process.
- Online ticket booking for auditorium visitors.
- Online parking lot booking.

1.3 Scope of system:

- Auditorium Owner:
    - Sign up/Login
    - Register an auditorium
    - Provide multiple packages for booking
    - Check available slots
    - Check booked slots

- Auditorium Customer:
    - Sign up/Login
    - Check available slots
    - Book auditorium
    - Select package as per requirement

- Auditorium Visitor:
    - Ticket booking for particular event

1.4 Operating Environment – Hardware and Software:

1. Android is an open and free mobile platform. This project uses:

a) Google Android mobile SDK and AVD Manager Tool.

2. Need to have android supporting device having at least latest android version. Minimum Android 4.2.2 Jelly Bean required.

3. 265MB of RAM or higher (512 MB is Recommended)

4. Wireless Network Connectivity

5. Some other electronic components and power supply units.

6. Database will be stored using SQ

Chapter 2: Proposed System

2.1 Proposed System

- The proposed system is user friendly and satisfies user's needs regarding Booking and Checking availability of an Auditorium.

- The proposed system is android as well as internet based.

- This application will be used by Three type people
    1. Auditorium Owners
    2. Auditorium Customers
    3. Auditorium visitors

2.2 Objectives of the System

1. To automate booking and billing process.

2. To search an auditorium as per requirement.

3. To provide different packages to auditorium customer.

4. To check availability of an auditorium on particular date and time.

5. To provide facility to auditorium visitors for Event ticket booking and Parking slot booking.

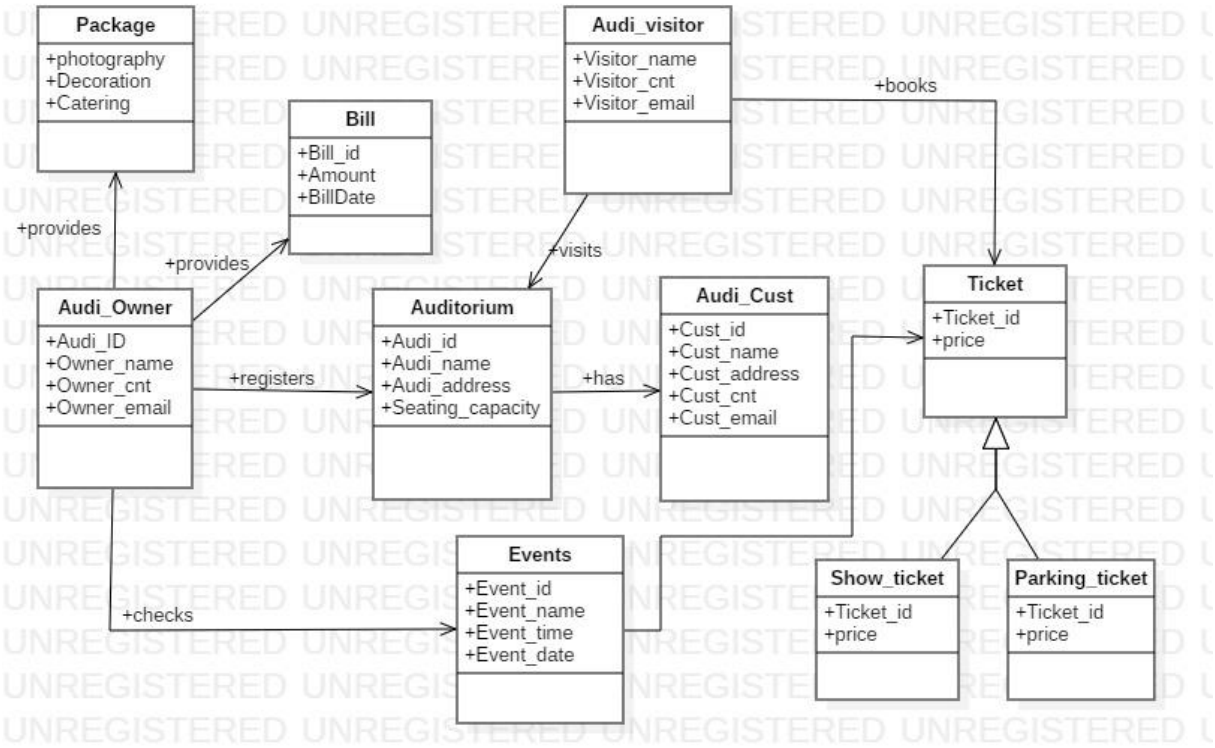
2.3User Requirements

1. To automate booking process.

2. Customer should be able to check availability of an auditorium on there device.

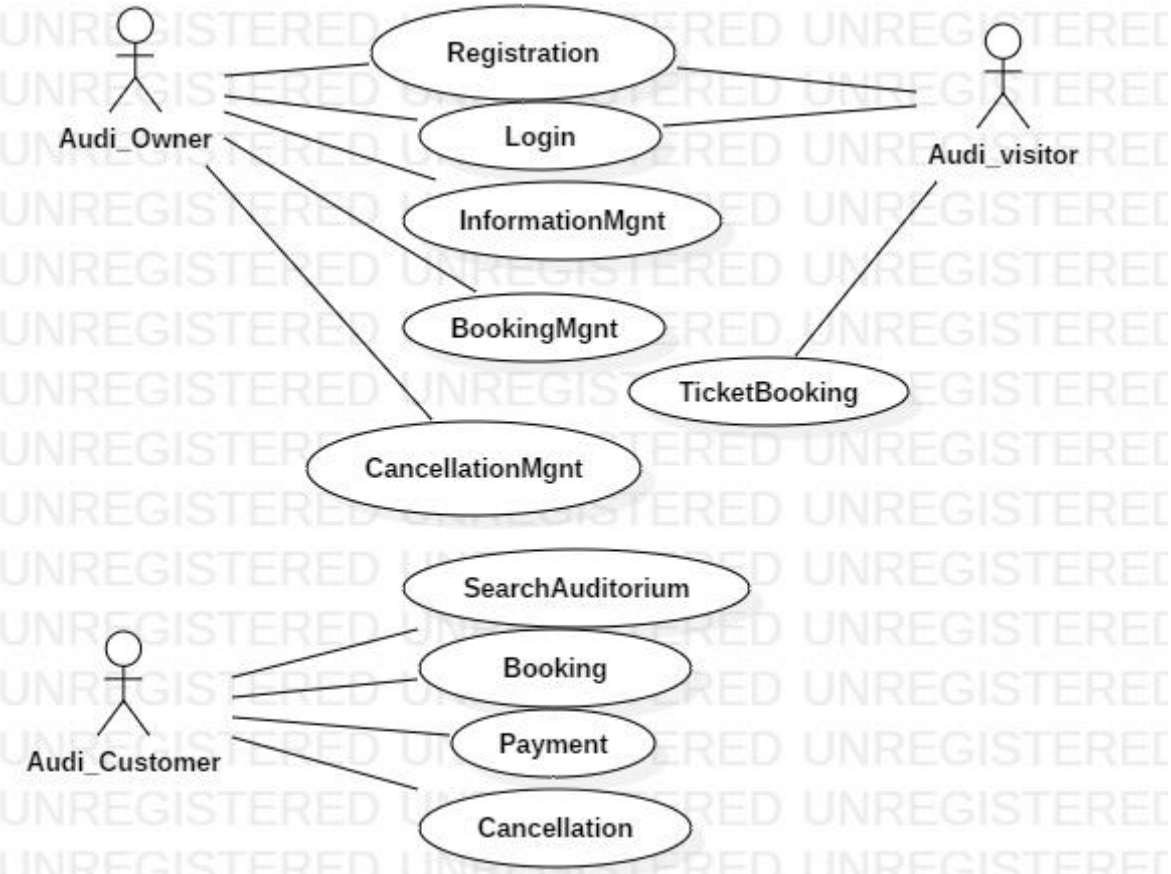
Chapter 3: Analysis and Design

3.2 Class Diagram

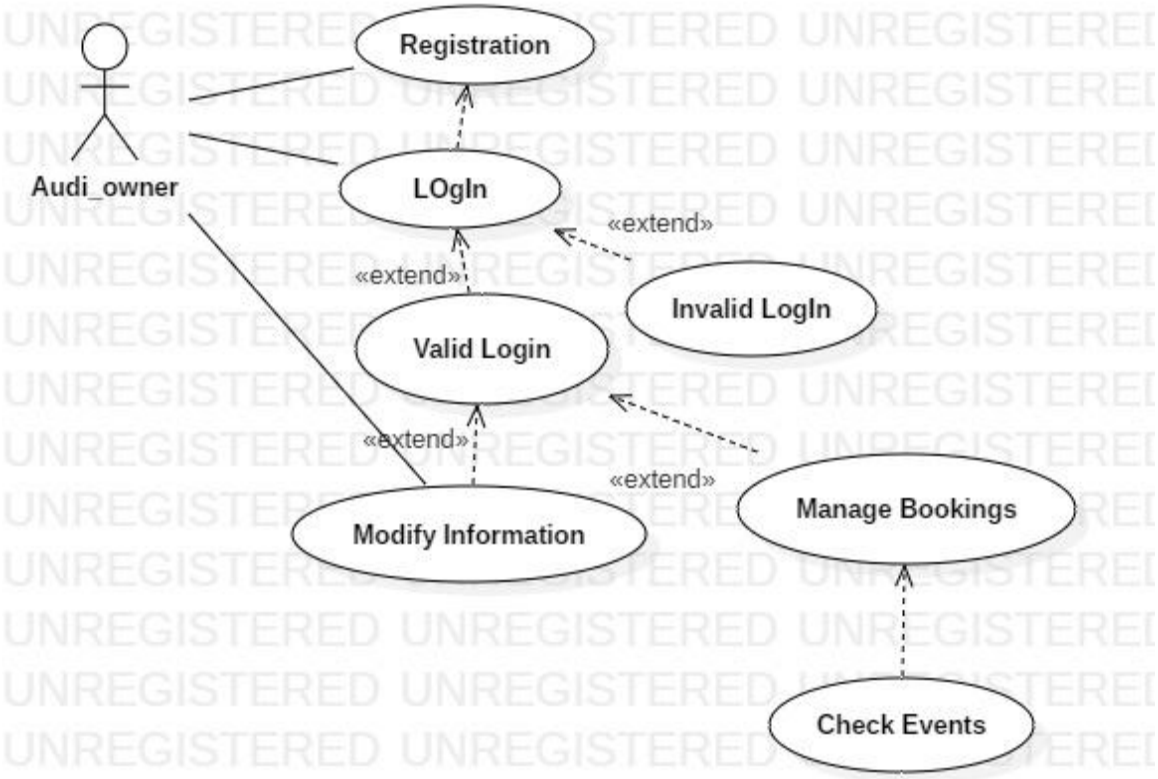


3.3 Use Case Diagram

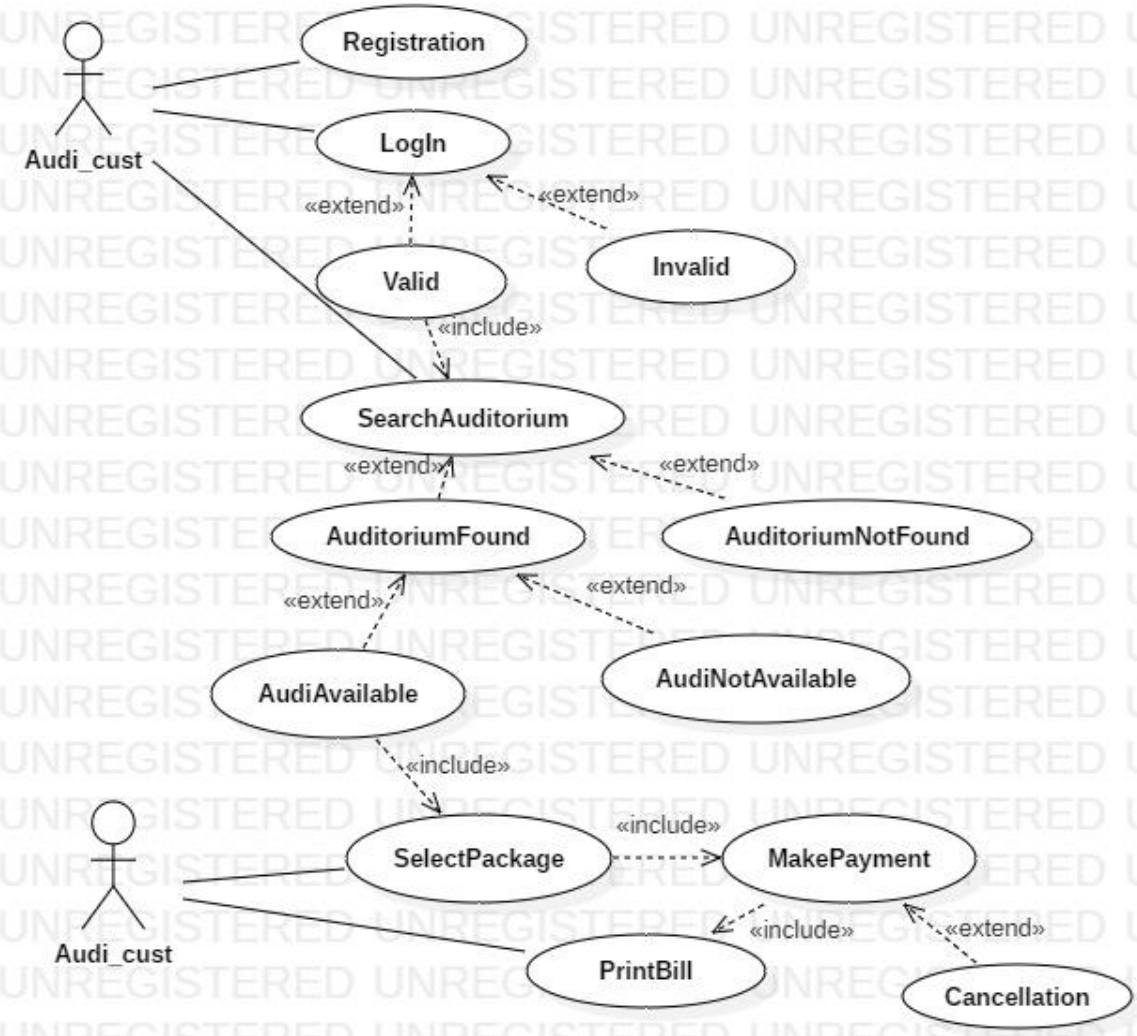
1.Auditorium Booking System



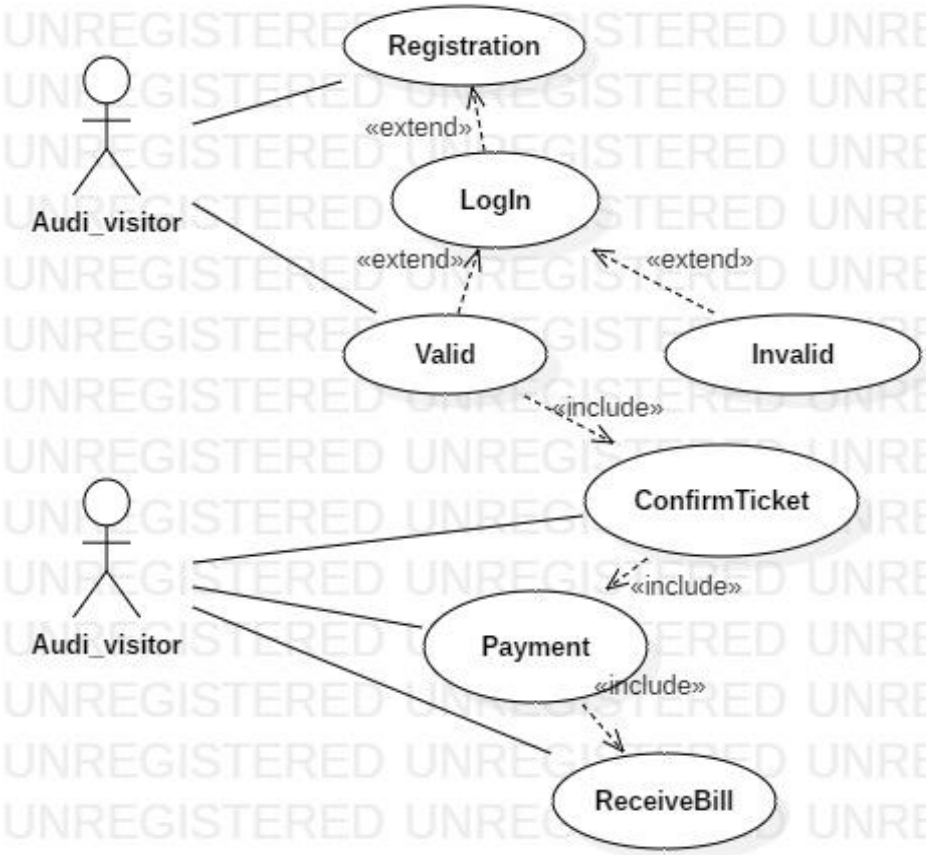
2.Auditorium Owner



3.Auditorium Customer

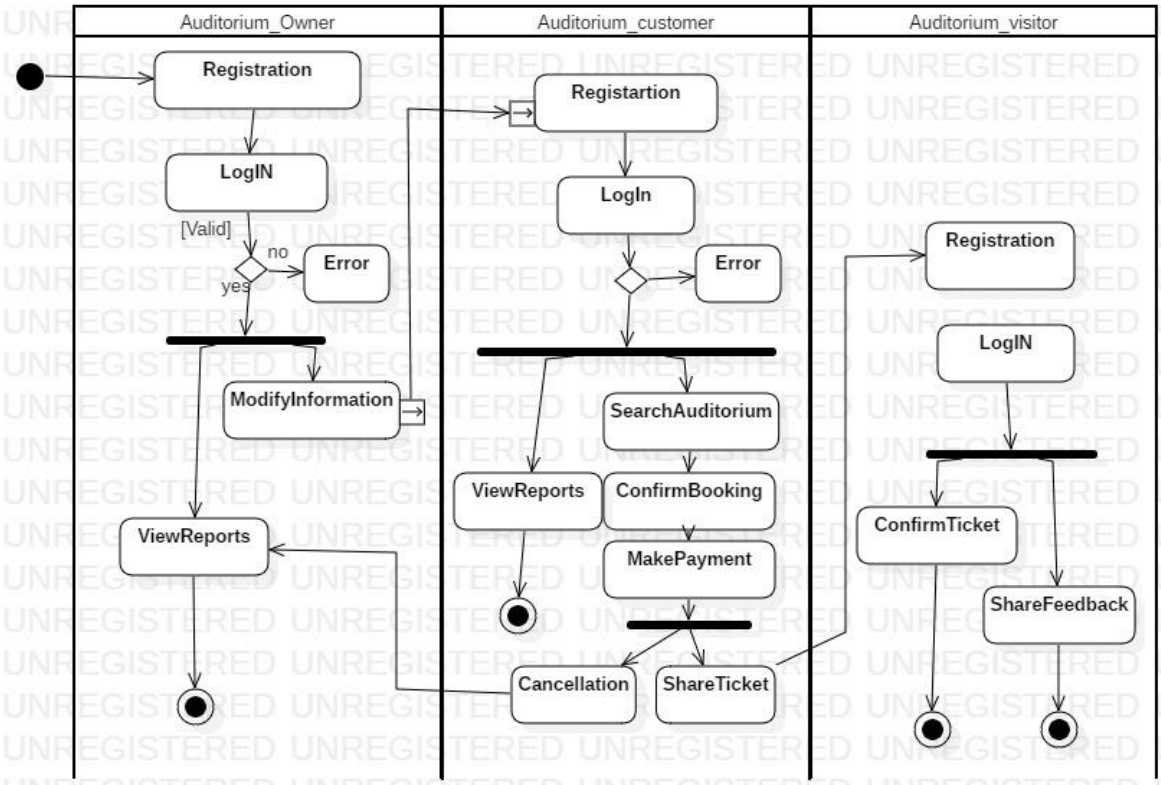


4.Auditorium Visitor

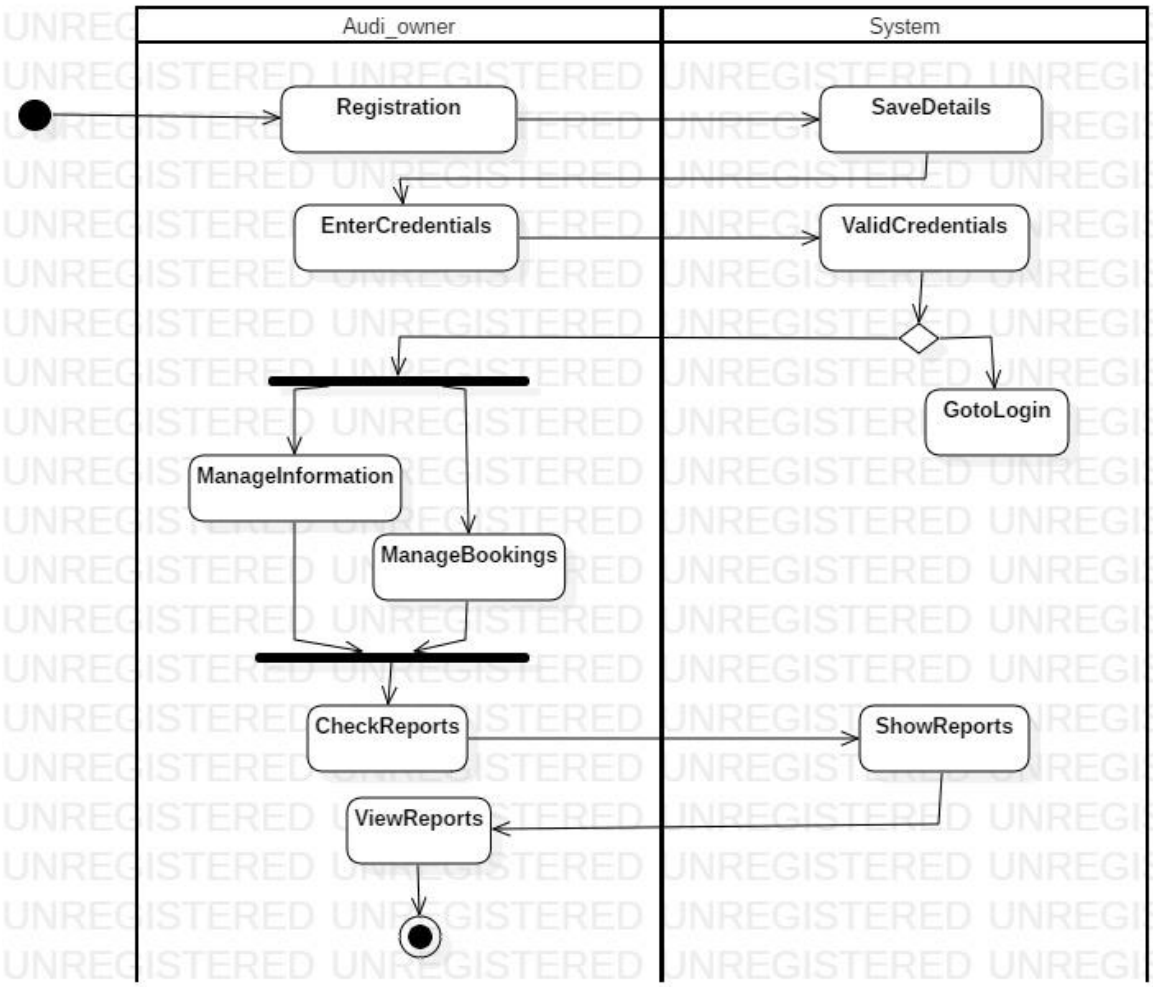


3.4. Activity Diagram

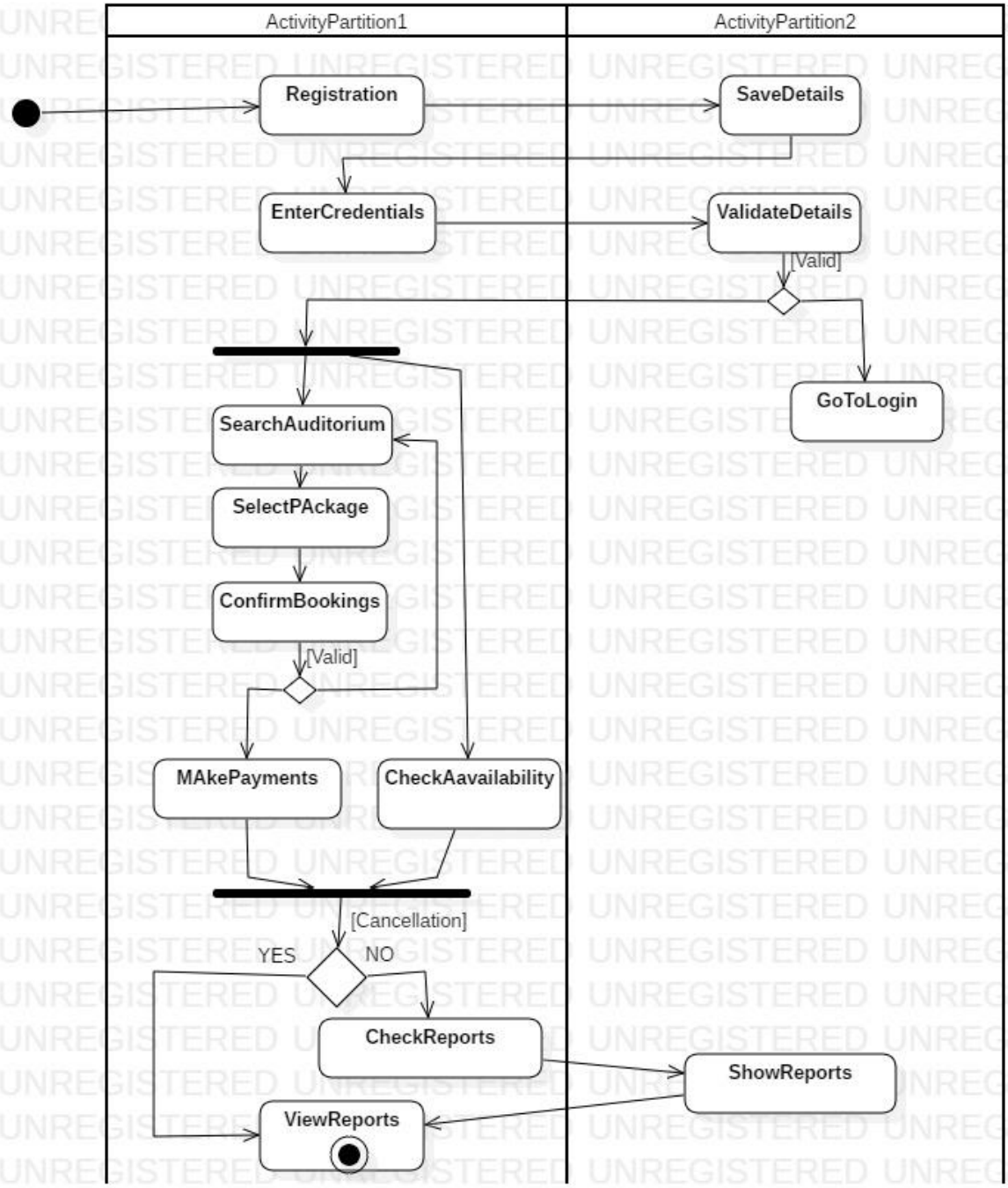
1.Auditorium Booking System



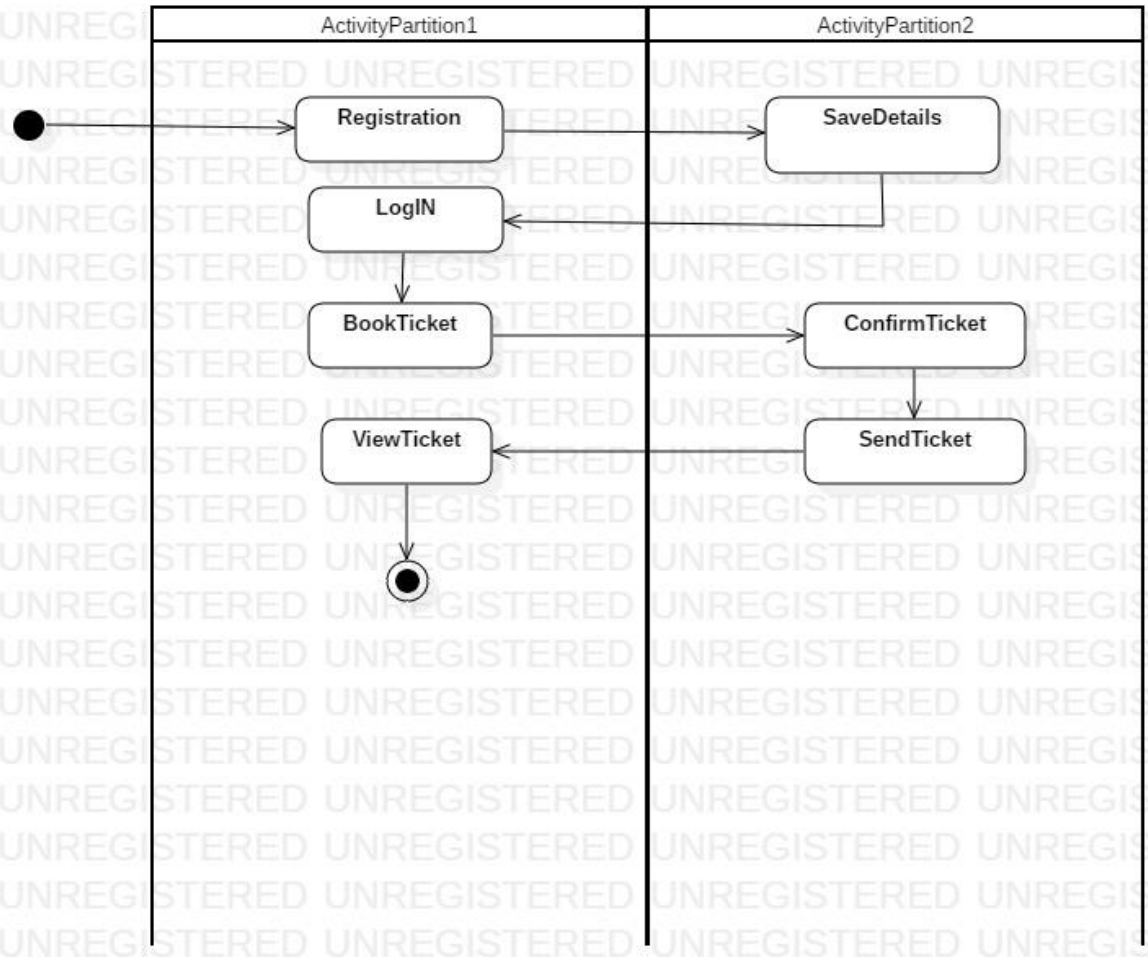
2. Auditorium Owner



3. Auditorium Customer

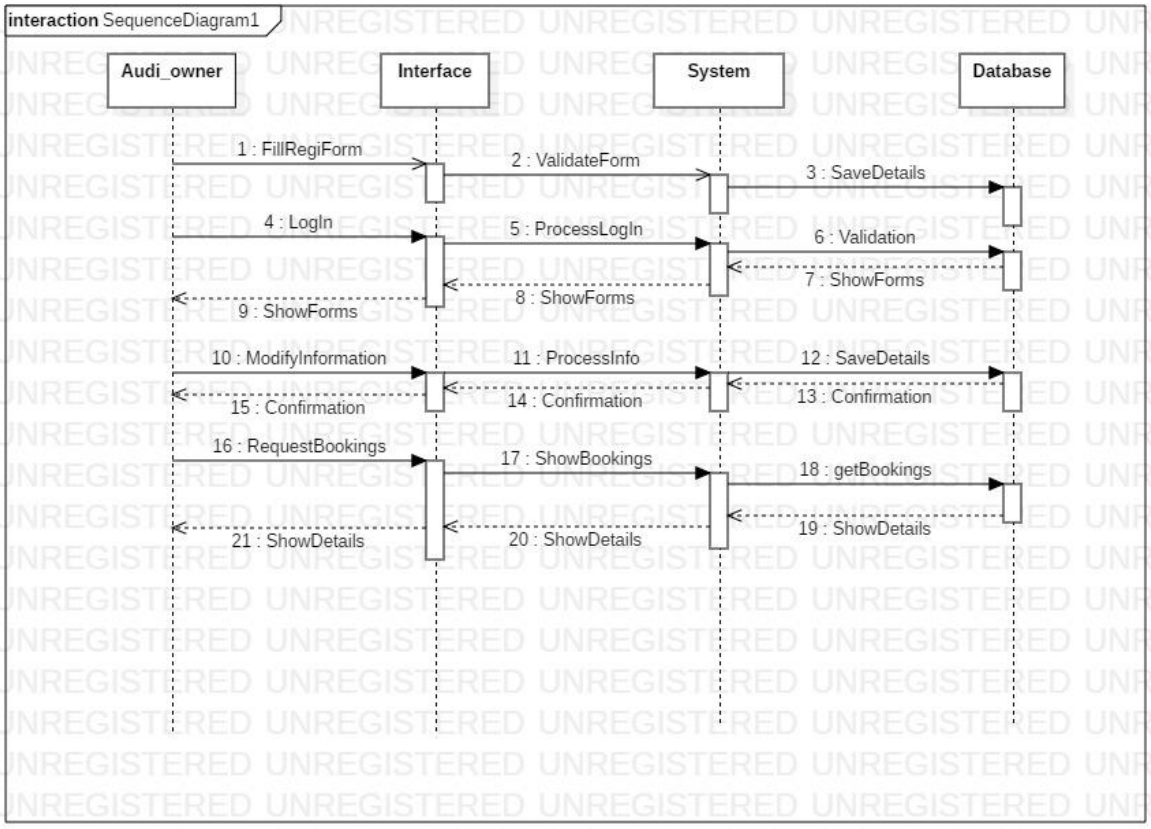


4. Auditorium visitor

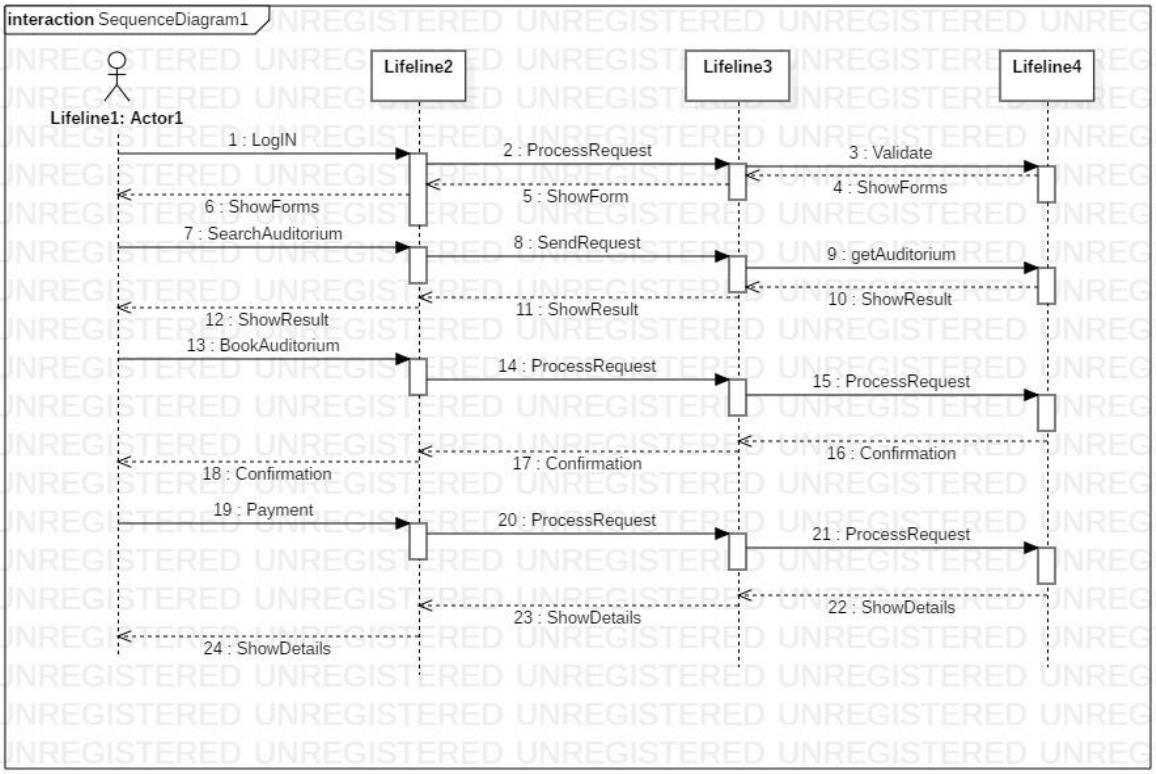


3.5 Sequence Diagram

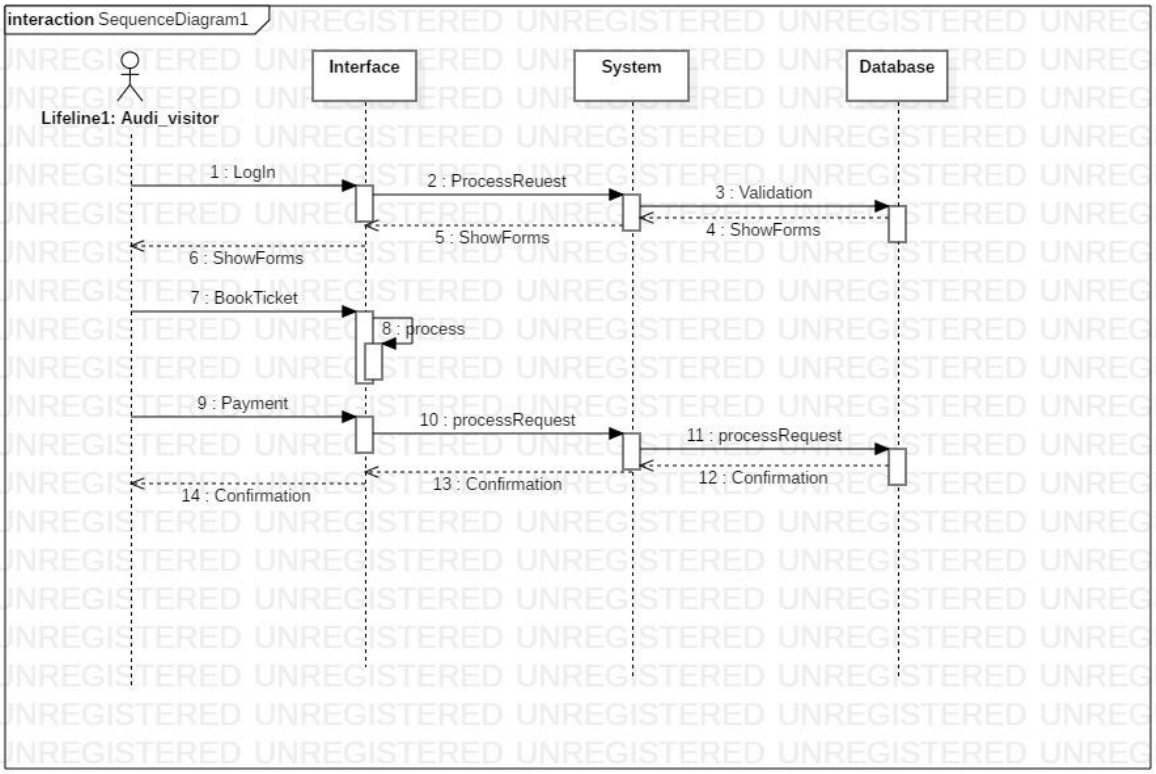
1.Auditorium Owner



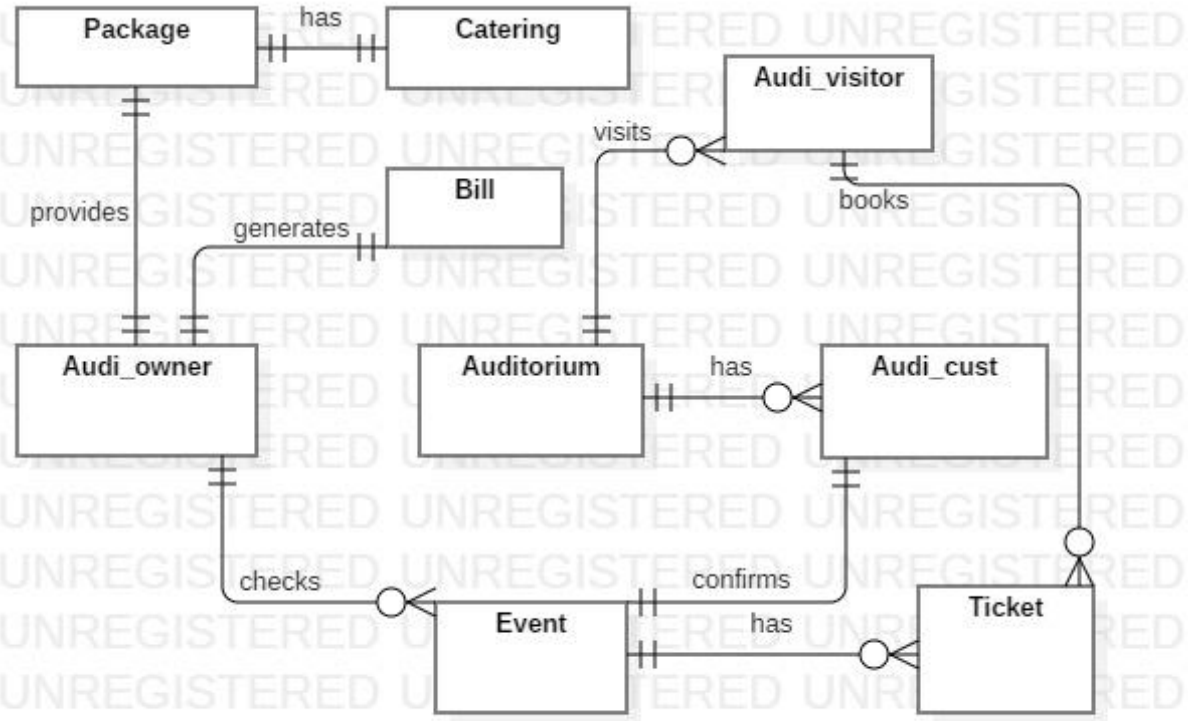
2.Auditorium Customer



3.Auditorium Visitor

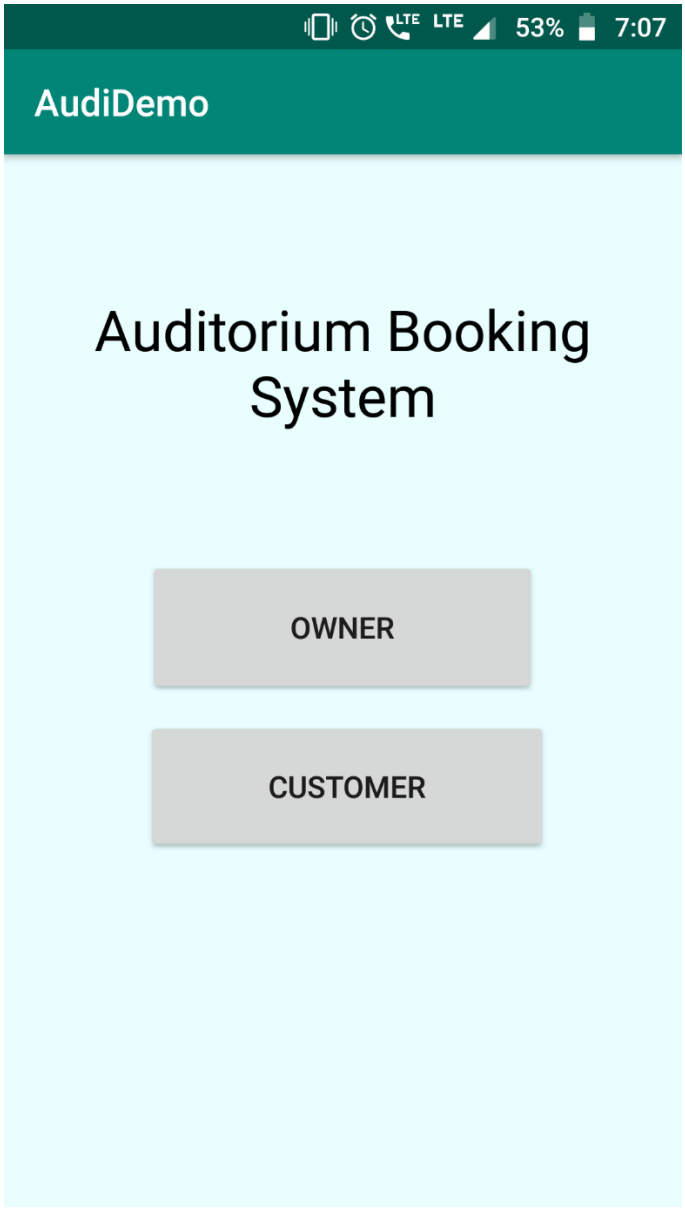


3.6 E R Diagram

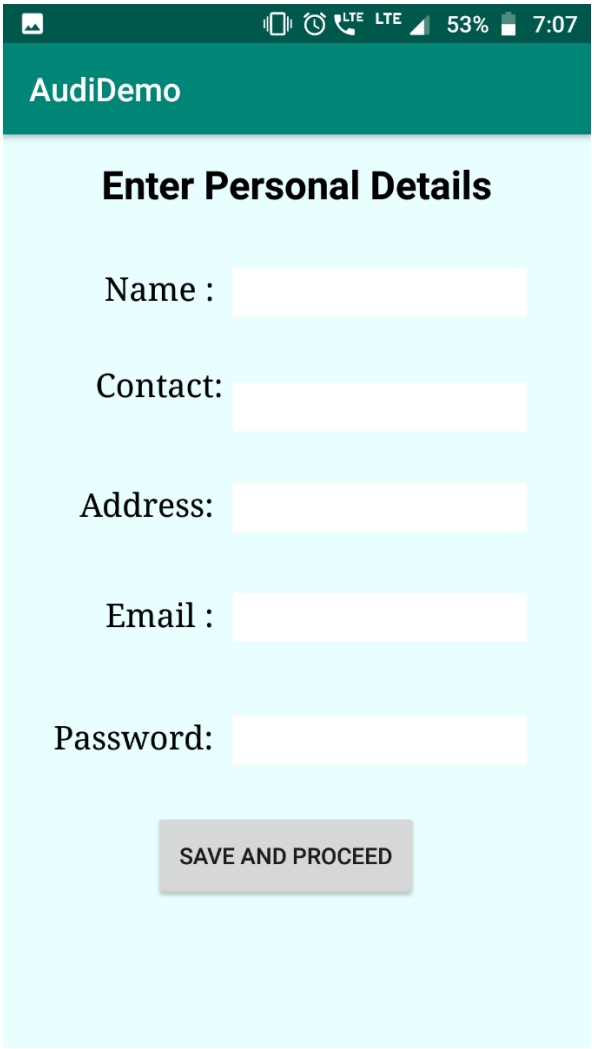


3.13 User Interface Design :

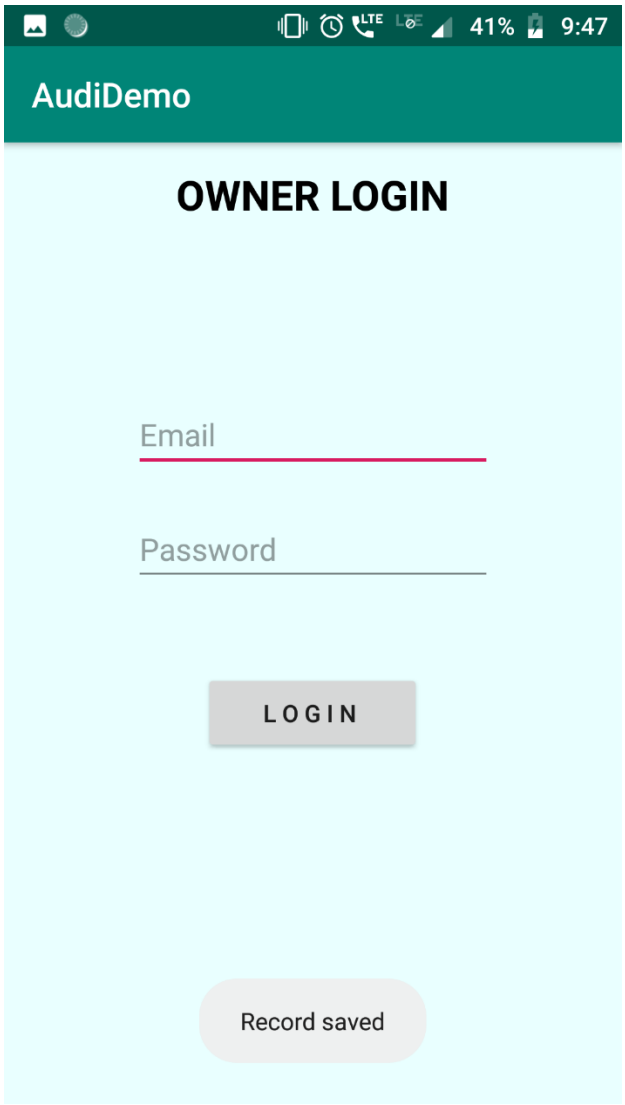
- Main Screen



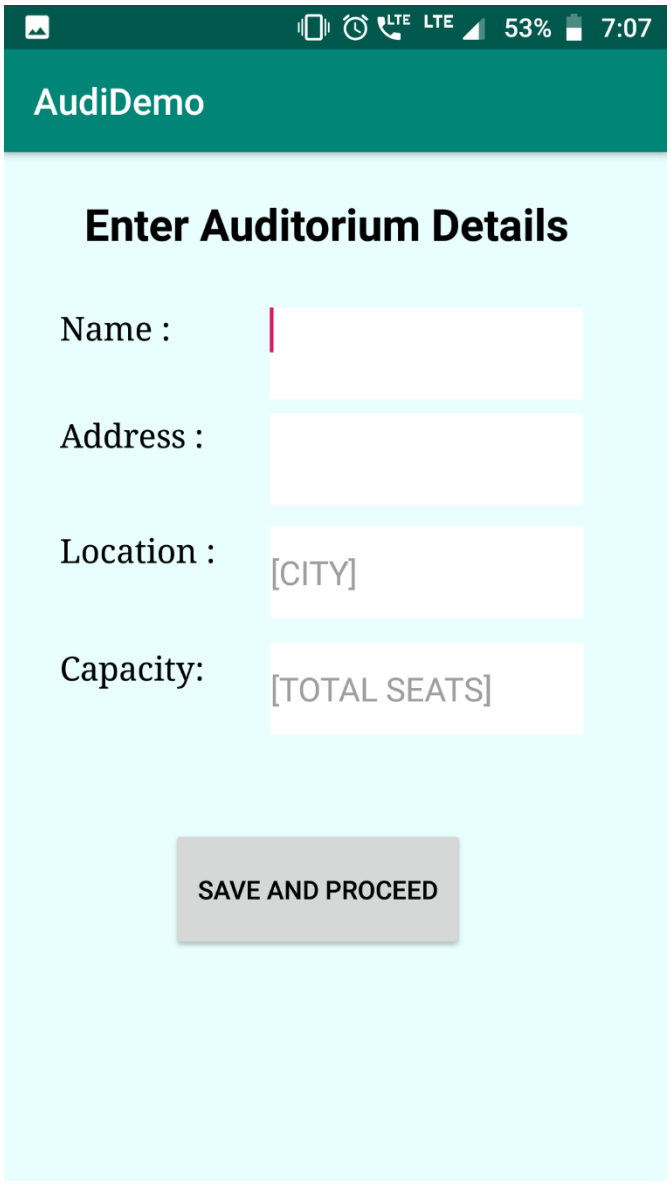
- **Owner registration screen :**



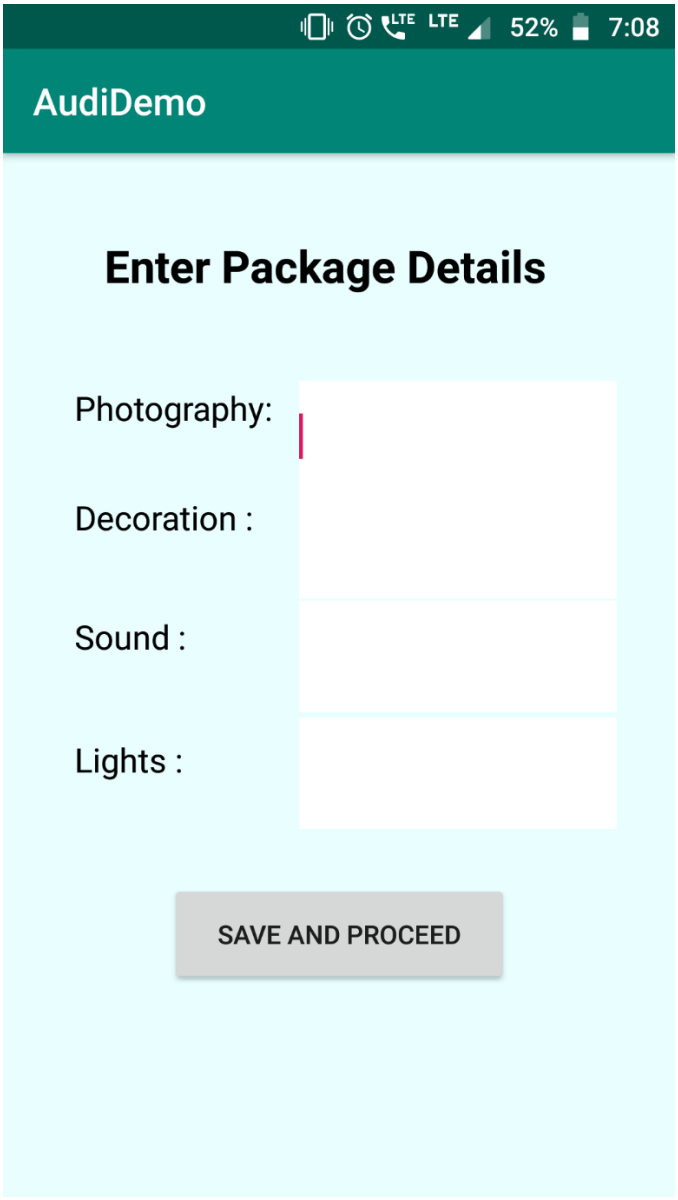
- **Owner login screen :**



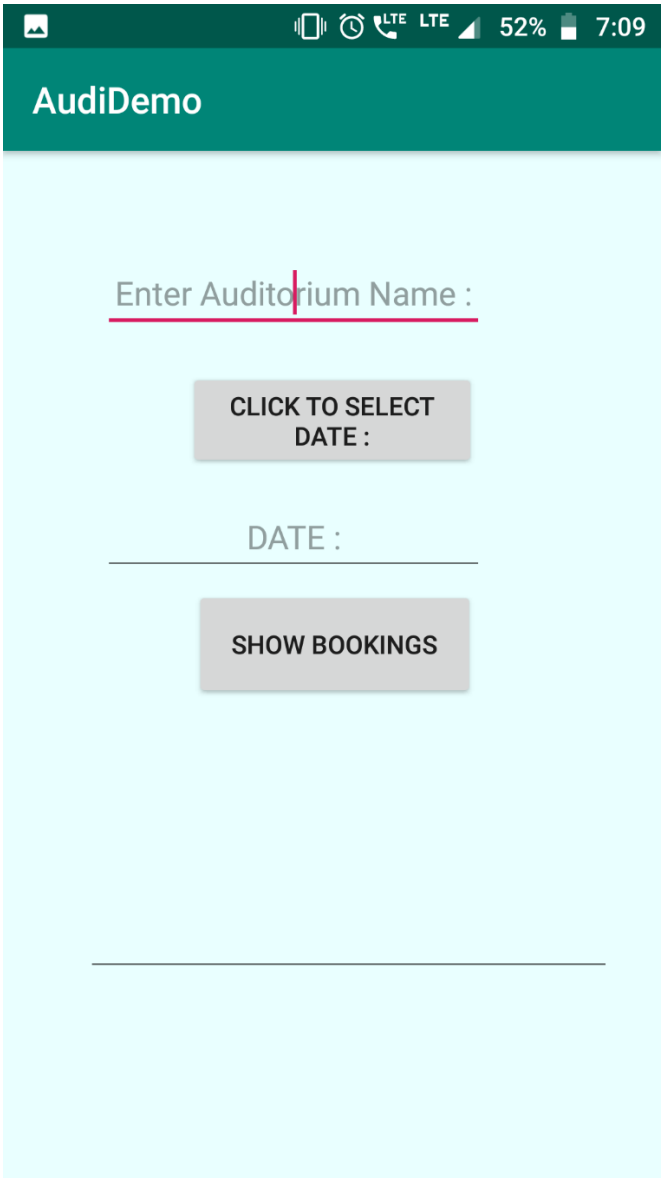
- **Auditorium registration screen :**



- **Package information screen :**

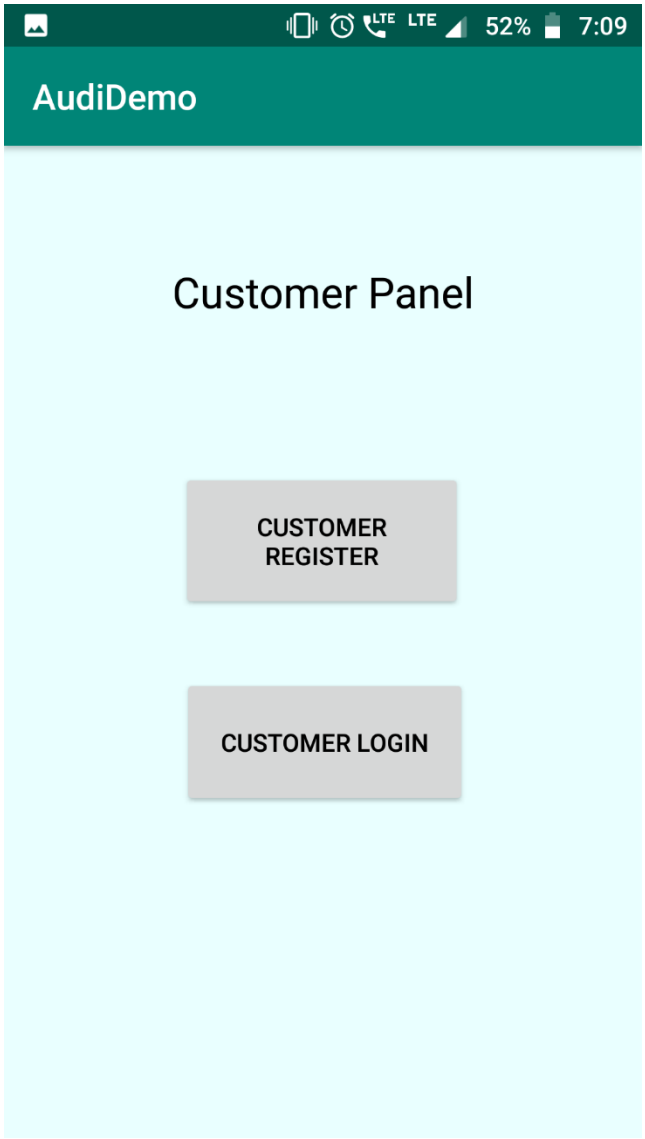


- **Owner auditorium availability screen :**

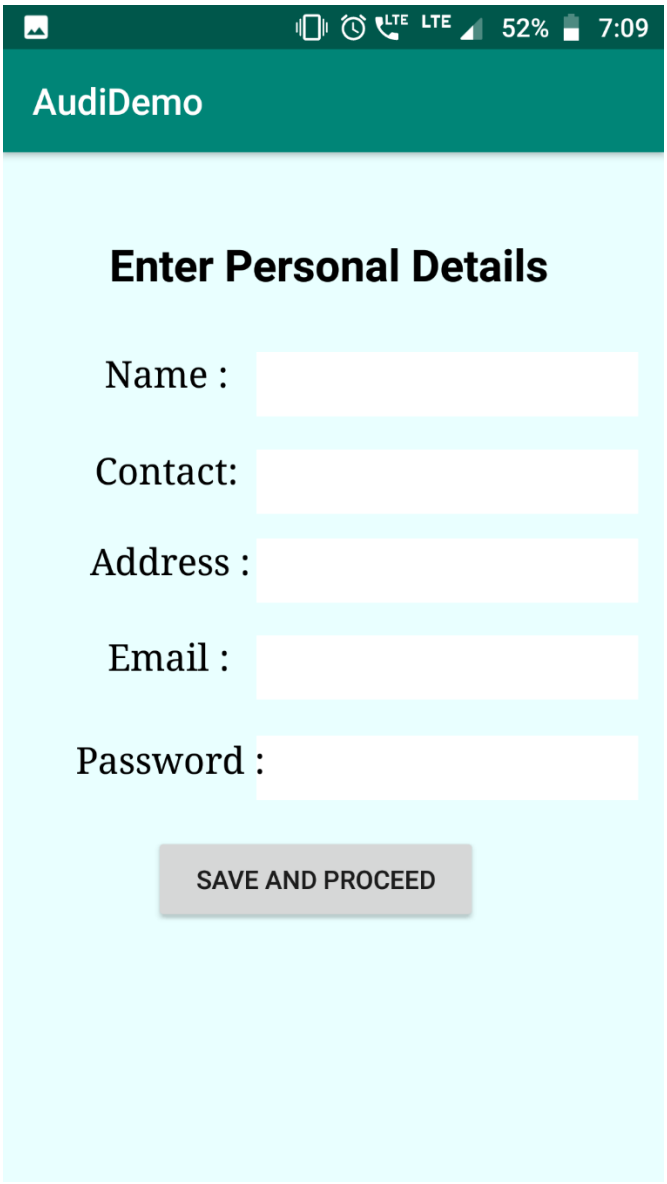


A. Auditorium Customer :

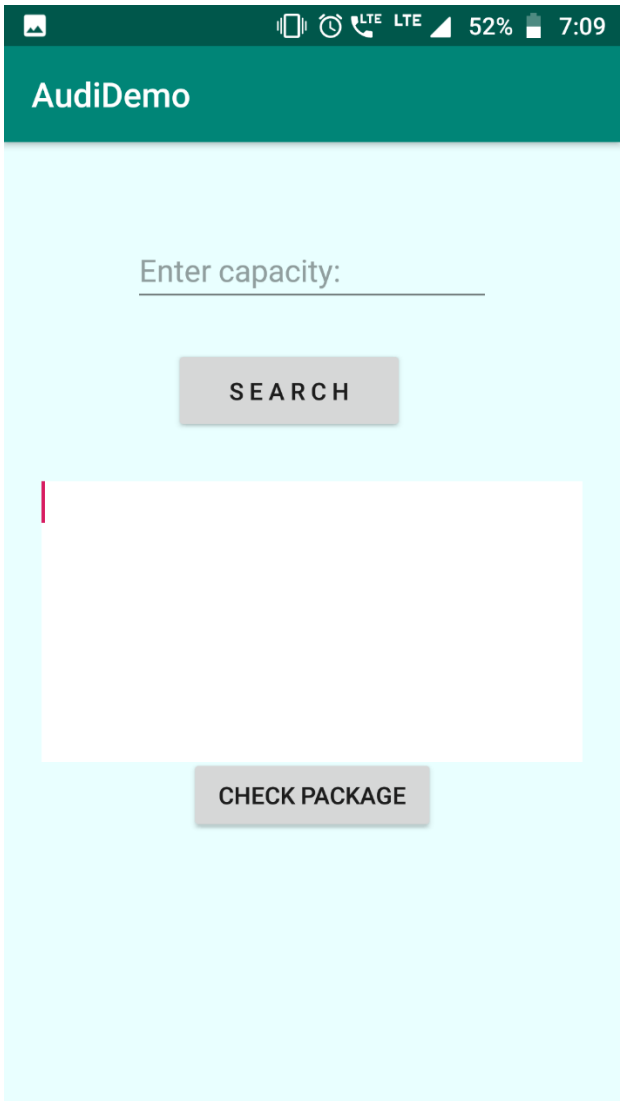
- **Customer search panel :**



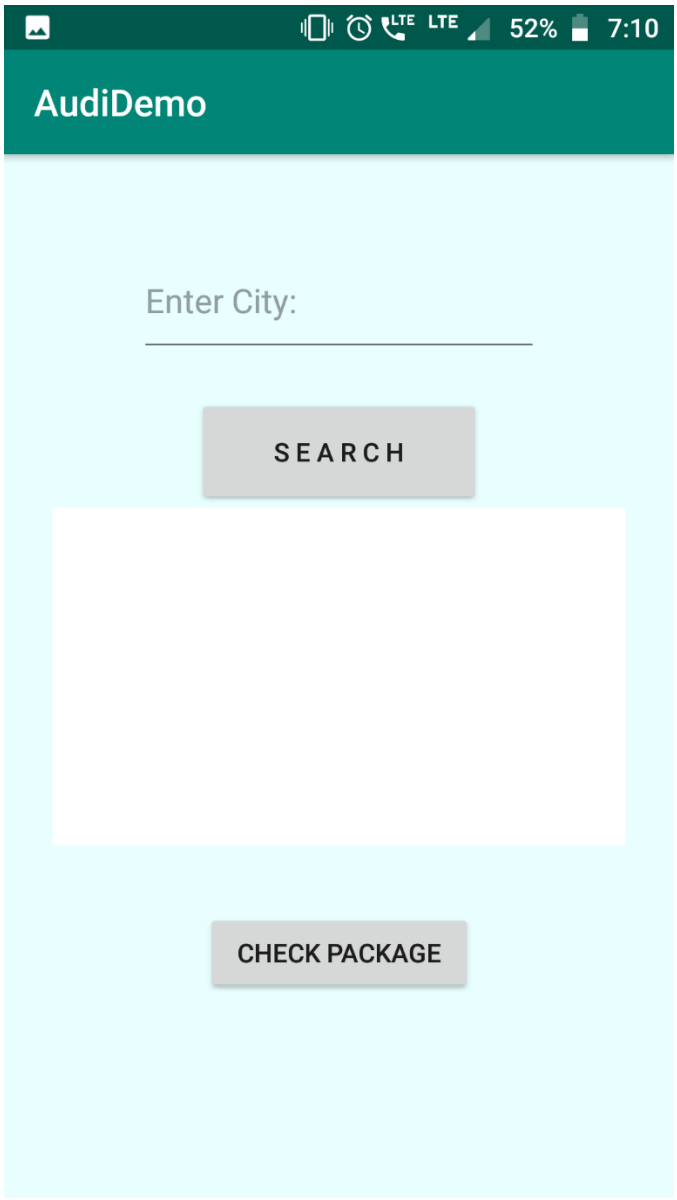
- **Customer registration screen :**



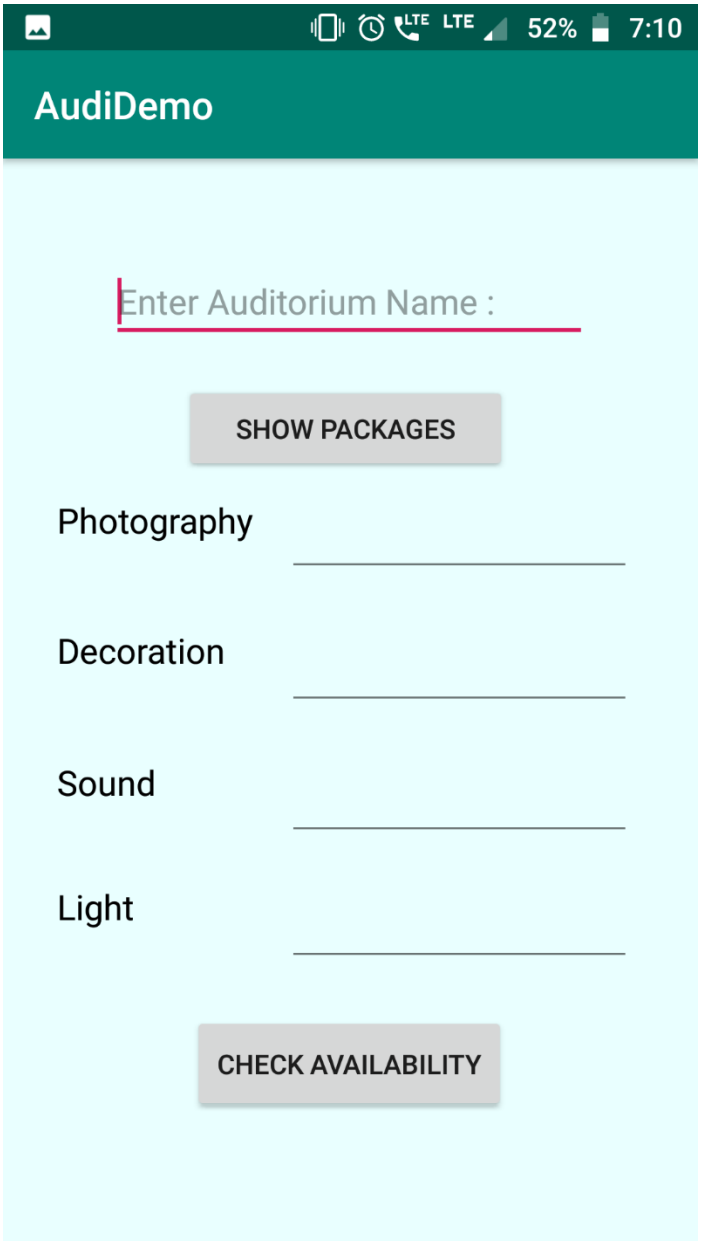
- **Search by capacity screen:**



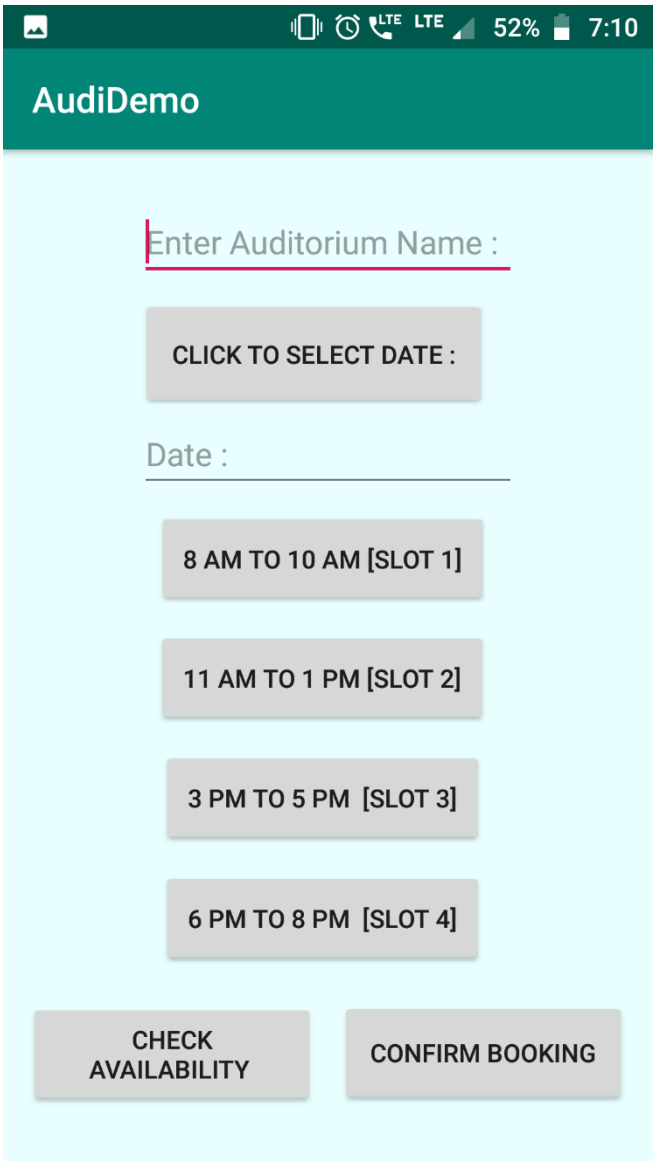
- **Search by Location:**



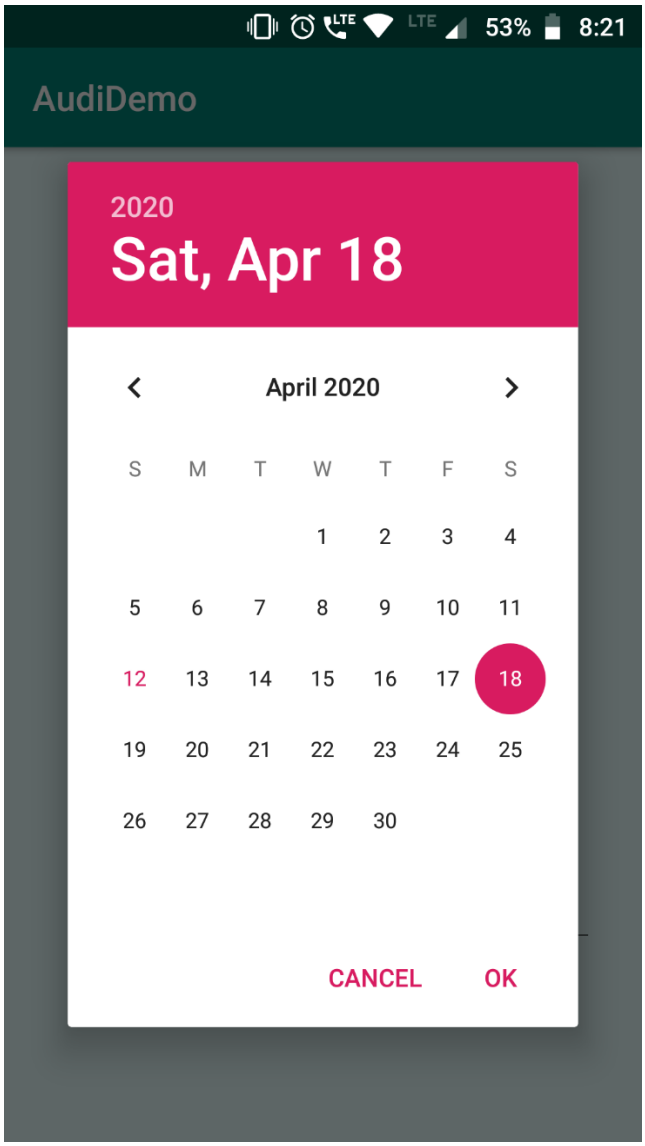
- **Check package information screen:**


52% 7:10
AudiDemo
Enter Auditorium Name :
SHOW PACKAGES
Photography
Decoration
Sound
Light
CHECK AVAILABILITY

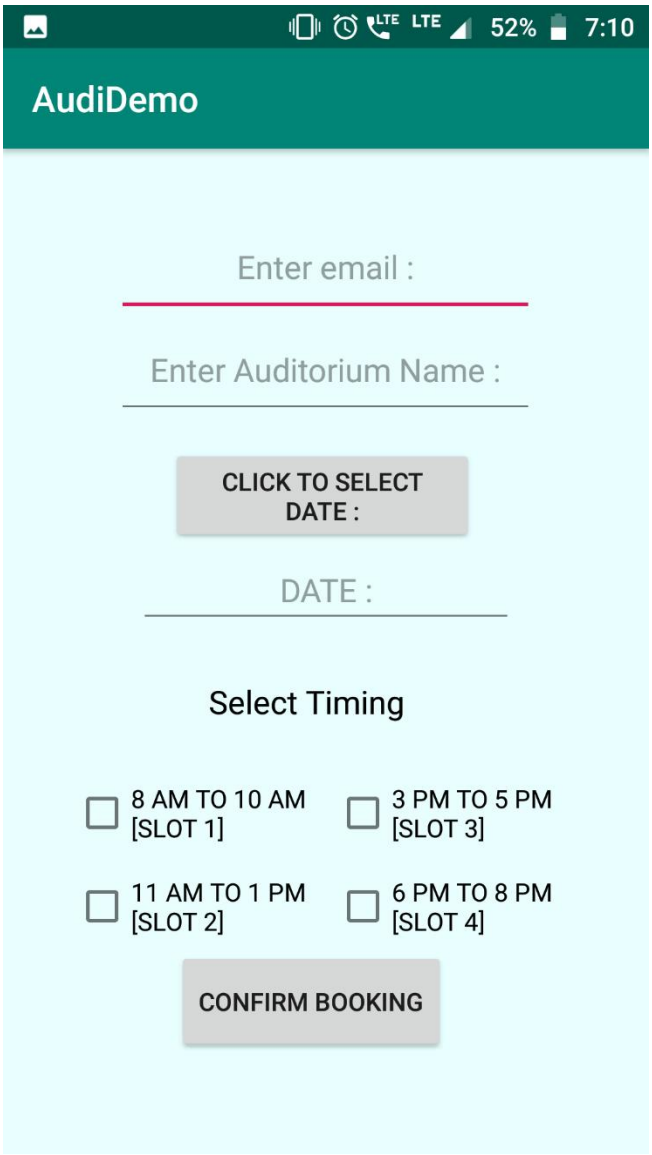
- **Check auditorium availability screen :**



- **Customer select date screen :**



- **Confirm booking screen:**



3.14 Data Dictionary

Sr.No.	Field Name	Field Size	Data Type	Description
1	Audi_id	5	Integer	Auditorium ID
2	Audi_name	20	Text	Auditorium Name
3	Audi_ addr	30	Text	Auditorium Address
4	Audi_City	30	Text	Auditorium City
5	Audi_capacity	5	Text	Auditorium Capacity
6	OWNER_ID	5	INTEGER	Owner's ID
7	Name	20	TEXT	Owner's Name
8	Conntact	10	TEXT	Owner's Contact
9	ADDRESS	30	TEXT	Owner's Address
10	EMAIL	10	TEXT	Email
11	PASSWORD	10	TEXT	Password
12	CustID	5	Integer	Customer ID
13	CustName	30	Text	Customer Name
14	CustContact	10	Integer	Customer Contact
15	CustAddress	20	Text	Customer Address
16	CustEmail	20	Text	Email
17	CustPassword	20	Text	Password
18	BookingID	5	Integer	Event Booking ID
19	CustEMAIL	30	Text	Customer Email
20	AUDI_NAME	30	TEXT	Auditorium Name
21	Date	20	TEXT	Event Date
22	Slot_ONE	5	INTEGER	Slot one
23	Slot_TWO	5	INTEGER	Slot Two
24	Slot_THREE	5	INTEGER	Slot Three
25	Slot_FOUR	5	INTEGER	Slot Four
26	CustID	10	INTEGER	Customer ID
27	AUDI_ID	10	INTEGER	Auditorium ID
28	PACKAGE_ID	5	Integer	PACKAGE_ID
29	PHOTOGRAPHY	30	Text	Photography
30	DECORATION	10	TEXT	Decoration
31	SOUND	20	TEXT	Sound
32	LIGHT	20	TEXT	Light

3.15 Table Specification

1. AudiInfo

Sr.No.	Field Name	Field Size	Data Type	Constraints	Description	Table
1	Audi_id	5	Integer	PK	Auditorium ID	Auditorium
2	Audi_name	20	Text	Not Null	Auditorium Name	Auditorium
3	Audi_ addr	30	Text	Not Null	Auditorium Address	Auditorium
4	Audi_City	30	Text	Not Null	Auditorium City	
5	Audi_capacity	5	Text	Not Null	Auditorium Capacity	Auditorium

2.Audi_owner

Sr.No.	Field Name	Field Size	Data Type	Constraints	Description	Table
1	OWNER_ID	5	INTEGER	NOT NULL	Owner’s ID	AudiOwner
2	Name	20	TEXT	NOT NULL	Owner’s Name	AudiOwner
3	Conntact	10	TEXT	NOT NULL	Owner’s Contact	AudiOwner
4	ADDRESS	30	TEXT	NOT NULL	Owner’s Address	AudiOwner
5	EMAIL	10	TEXT	NOT NULL	Email	AudiOwner
6	PASSWORD	10	TEXT	NOT NULL	Password	AudiOwner

3. Audi_cust

Sr.No.	Field Name	Field Size	Data Type	Constraints	Description	Table
1	CustID	5	Integer	PK	Customer ID	Audi_cust
2	CustName	30	Text	Not Null	Customer Name	Audi_cust
3	CustContact	10	Integer	Not Null	Customer Contact	Audi_cust
4	CustAddress	20	Text	Not Null	Customer Address	Audi_cust
5	CustEmail	20	Text	Not Null	Email	Audi_cust
6	CustPassword	20	Text	Not Null	Password	Audi_cust

4. AudiBooking

Sr.No.	Field Name	Field Size	Data Type	Constraints	Description	Table
1	BookingID	5	Integer	PK	Event Booking ID	AudiBooking
2	CustEMAIL	30	Text	Not Null	Customer Email	AudiBooking
3	AUDI_NAME	30	TEXT	Not Null	Auditorium Name	AudiBooking
4	Date	20	TEXT	Not Null	Event Date	AudiBooking
5	Slot_ONE	5	INTEGER	Not Null	Slot one	AudiBooking
6	Slot_TWO	5	INTEGER	Not Null	Slot Two	AudiBooking
7	Slot_THREE	5	INTEGER	Not Null	Slot Three	AudiBooking
8	Slot_FOUR	5	INTEGER	Not Null	Slot Four	AudiBooking
9	CustID	10	INTEGER	Not Null	Customer ID	AudiBooking
10	AUDI_ID	10	INTEGER	Not Null	Auditorium ID	AudiBooking

5.PackageInfo

Sr.No.	Field Name	Field Size	Data Type	Constraints	Description	Table
1	PACKAGE_ID	5	Integer	PK	PACKAGE_ID	PackageID
2	PHOTOGRAPHY	30	Text	Not Null	Photography	PackageID
3	DECORATION	10	TEXT	Not Null	Decoration	PackageID
4	SOUND	20	TEXT	Not Null	Sound	PackageID r
5	LIGHT	20	TEXT	Not Null	Light	PackageID

3.15 Test Procedures and Implementation

1. Unit Testing

Unit testing concentrates verification on the smallest element of the program – the module. Using the detailed design description important control paths are tested to establish errors within the bounds of the module.

In this system each sub module is tested individually as per the unit testing such as campaign, lead, contact etc are tested individually. Their input field validations are tested.

2. Integration testing

Once all the individual units have been tested there is a need to test how they were put together to ensure no data is lost across interface, one module does not have an adverse impact on another and a function is not performed correctly.

After unit testing each and every sub module is tested with integrating each other.

System testing for the current system:

In this level of testing we are testing the system as a whole after integrating all the main modules of the project. We are testing whether system is giving correct output or not. All the modules were integrated and the flow of information among different modules was checked. It was also checked that whether the flow of data is as per the requirements or not. It was also checked that whether any particular module is non-functioning or not i.e. once the integration is over each and every module is functioning in its entirety or not.

In this level of testing we tested the following: -

- Whether all the forms are properly working or not.
- Whether all the forms are properly linked or not.
- Whether all the images are properly displayed or not.
- Whether data retrieval is proper or not.

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

Test Cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non-functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

Test Design Techniques

Typical black-box test design techniques include:

- Decision table testing
- All-pairs testing
- State transition Analysis
- Equivalence partitioning
- Boundary value analysis
- Cause–effect graph
- Error guessing

Advantages

- Efficient when used on large systems.
- Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
- Tester can be non-technical.
- There is no need for the tester to have detailed functional knowledge of system.
- Tests will be done from an end user's point of view, because the end user should accept the system. (This testing technique is sometimes also called Acceptance testing.)
- Testing helps to identify vagueness and contradictions in functional specifications.
- Test cases can be designed as soon as the functional specifications are complete.

Disadvantages

- Test cases are challenging to design without having clear functional specifications.
- It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
- It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
- There are chances of having unidentified paths during the testing process.

Test Case:

Test case Id	Description	Test Steps	Expected Result	Pass/Fail
1.	To test blank Email and password	Email="", Password="" Click on login	Display error message to enter username and password	Pass
2	To test incorrect Email and password	Email="abcd@g.com" password="123"	Display error message for wrong password	Pass
3	To test correct Email and password	Enter valid data	Login successful and navigate to home page	Pass
4.	To test blank textbox for entering fields	Keep text box blank	Should display error message	Pass

5.	To test for adding multiple auditorium information by same owner	Add information	Should display Error message	Pass
6.	To test for adding multiple auditorium information by same owner	Add information	Should display error message	Pass
7.	Test to check availability of different auditorium	Check availability	Should display error message	Pass
8.	Test to check auditorium By capacity above 10000	Check auditorium	Should display error message	Pass
9.	Test to check auditorium by wrong name	Check auditorium	Should display error message	Pass

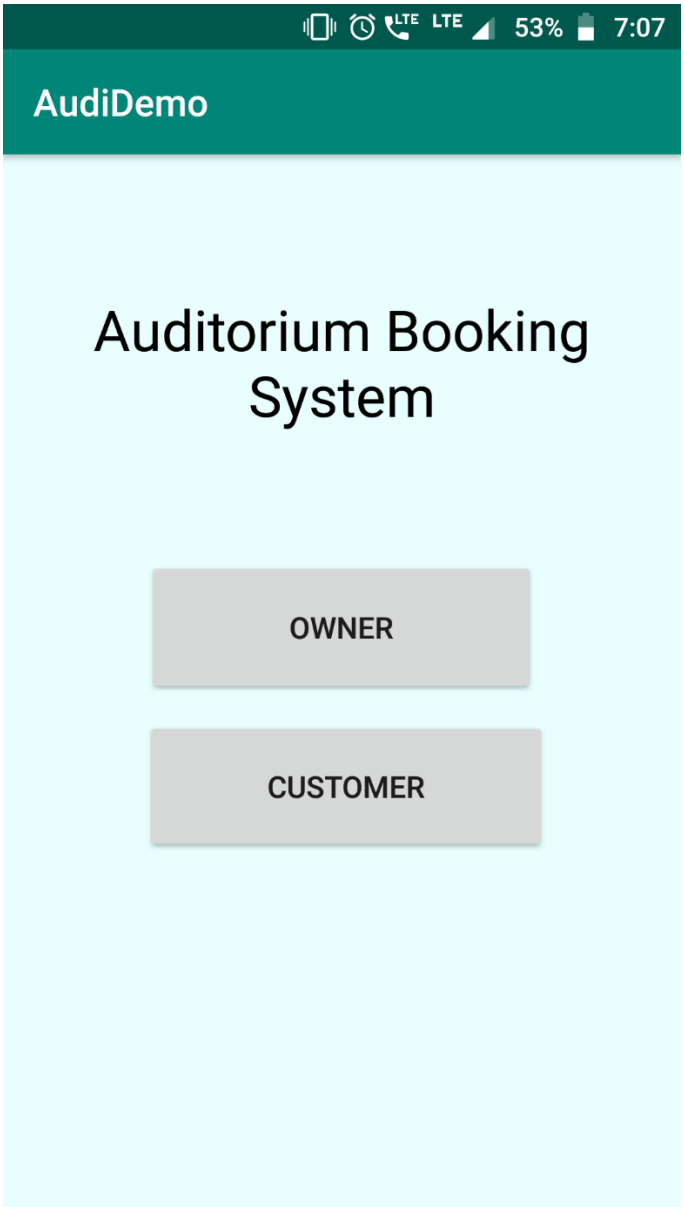
10.	Test to check package with wrong name	Check package	Should display error message	Pass
11.	Test to check auditorium availability with wrong date	Check availability	Should display error message	Pass
12.	Test to confirm booking on wrong date	Confirm booking	Should display error message	Pass

Chapter 4: USER MANUAL

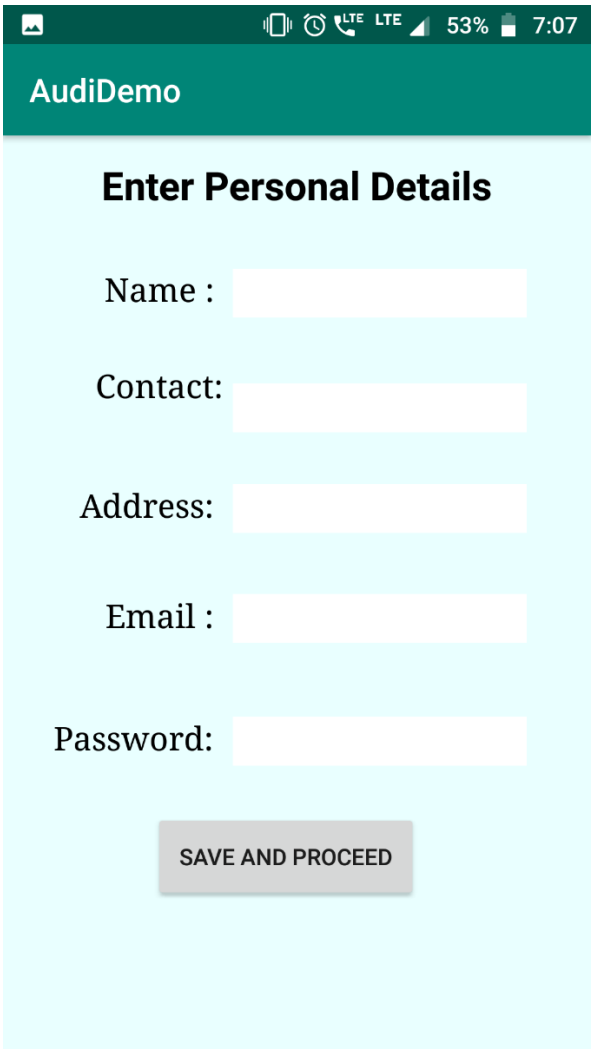
4.1User Manual:

A. User Manual for Auditorium Owner:

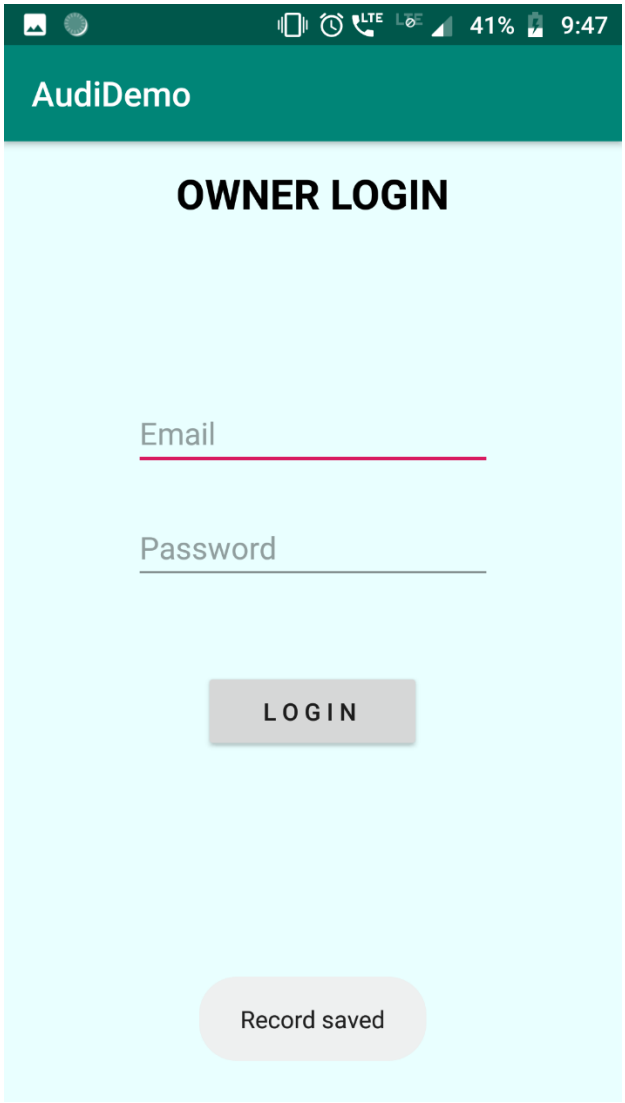
- Owner should open the application
- Owner should get register first.



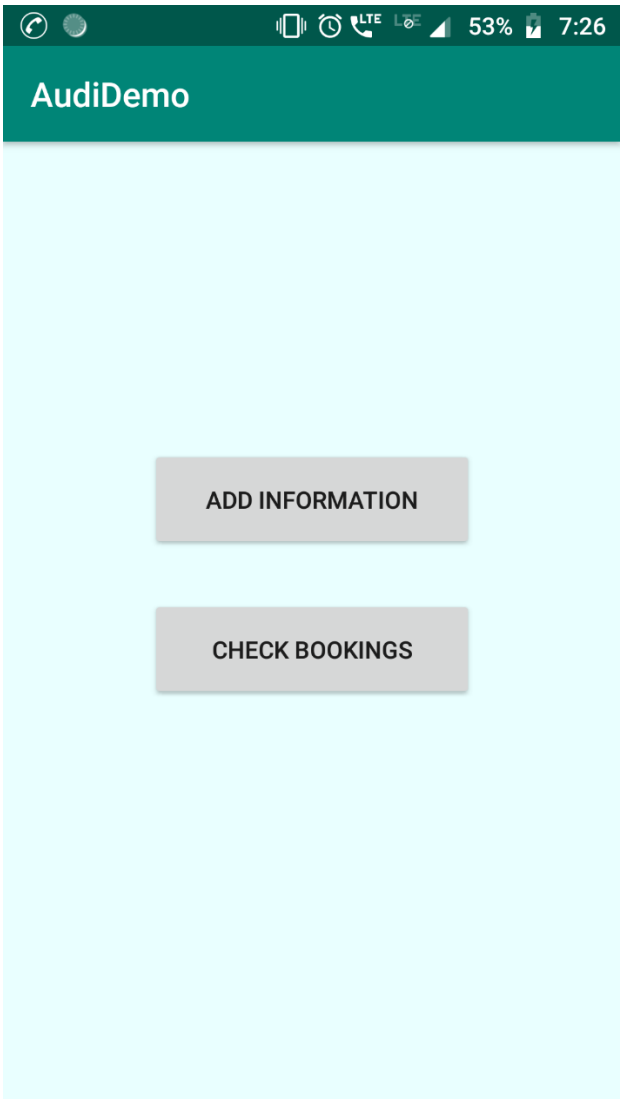
- **Owner should submit personal details**



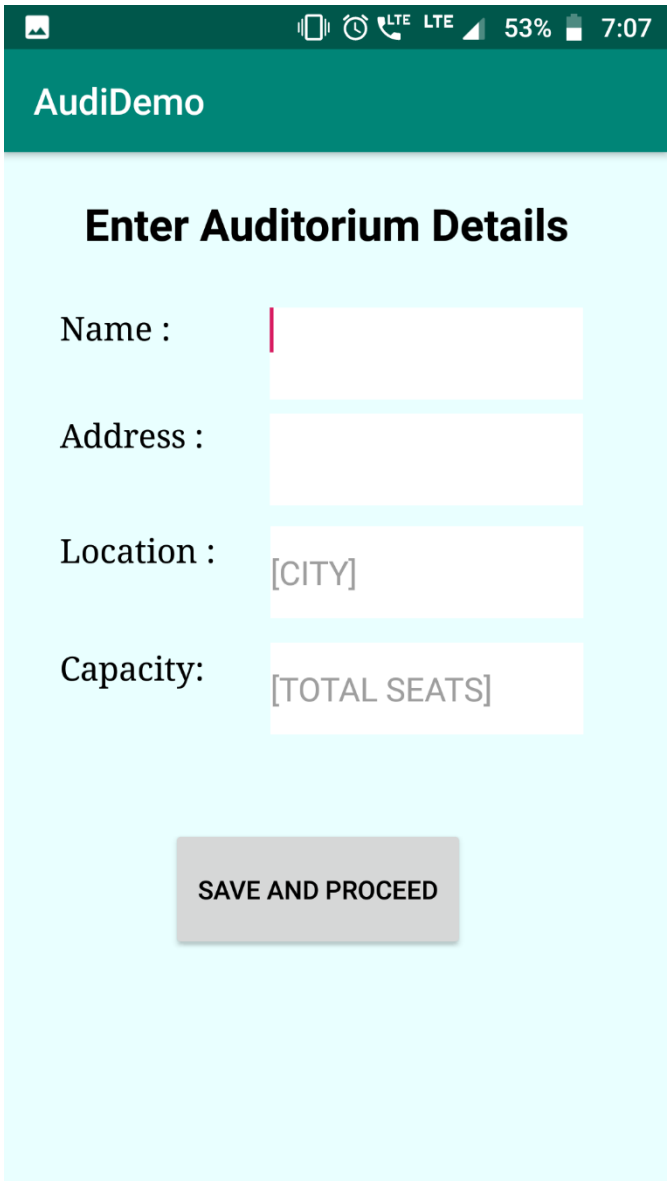
- **Owner can Login with their credentials**



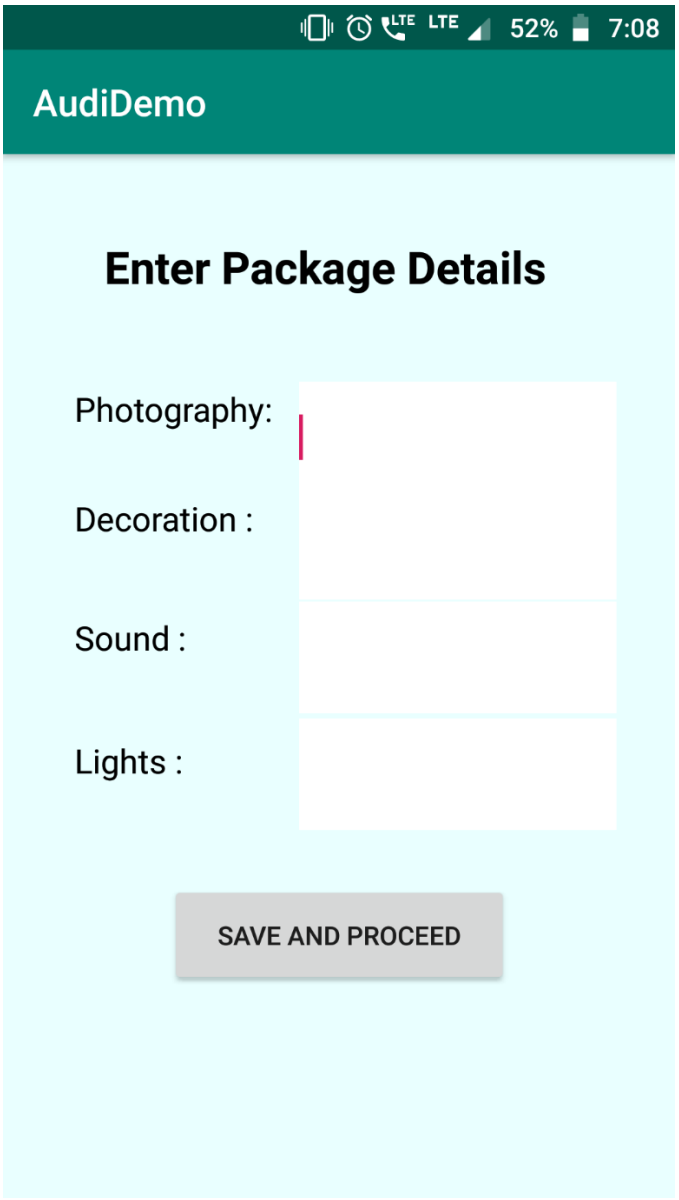
- **Owner Can select one options**



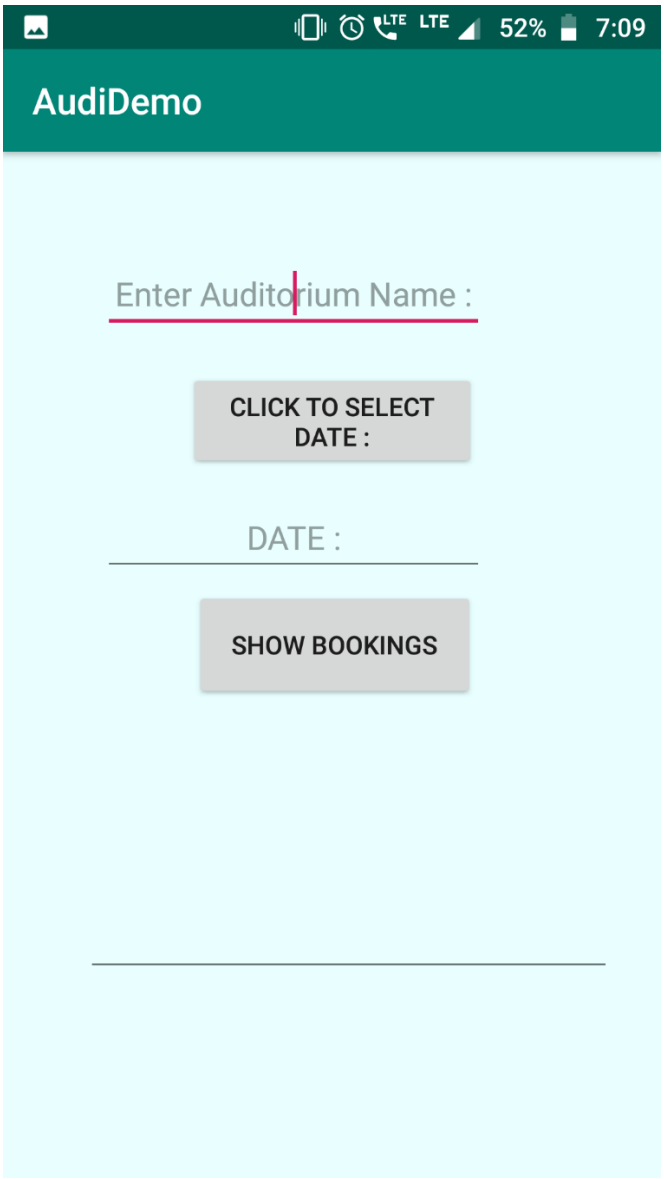
- **Owner can provide auditorium details**



- **Owner can provide auditorium package details**

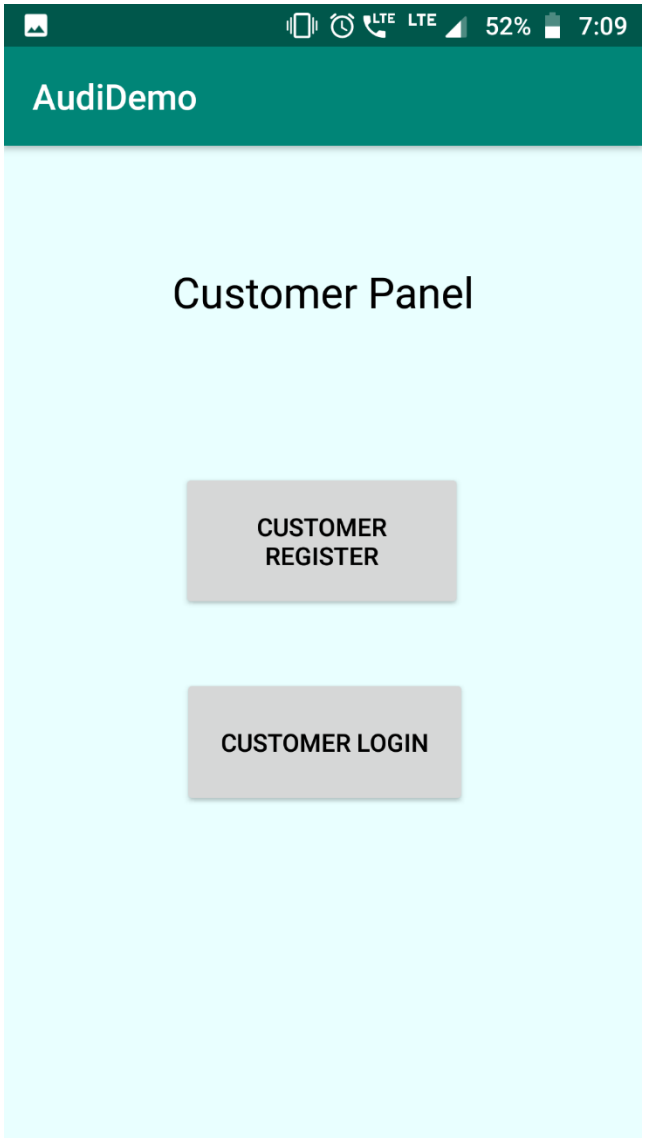


- **Owner can check auditorium bookings on particular date**

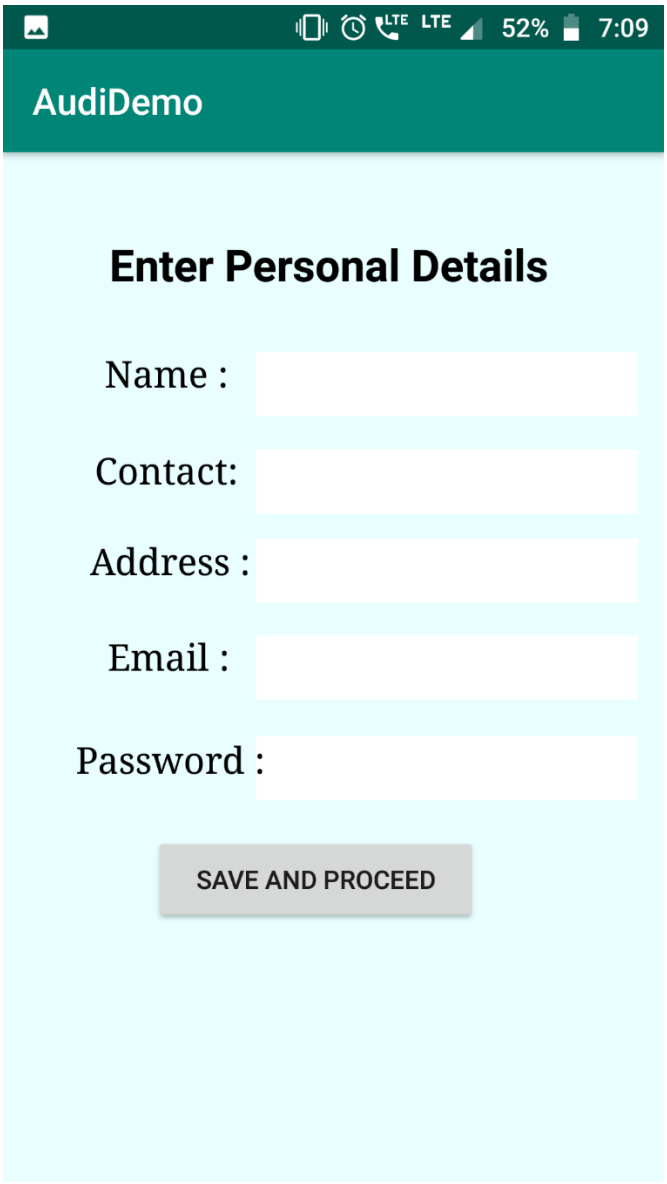


B. User Manual for Auditorium Customer:

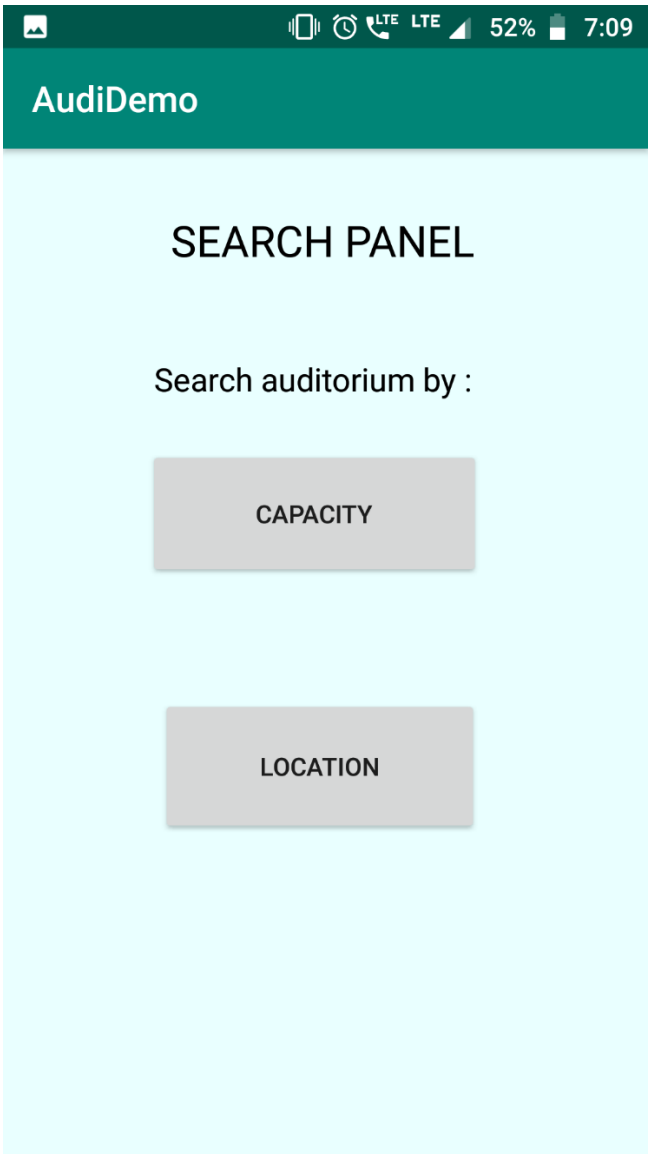
- **Customer can select one option.**



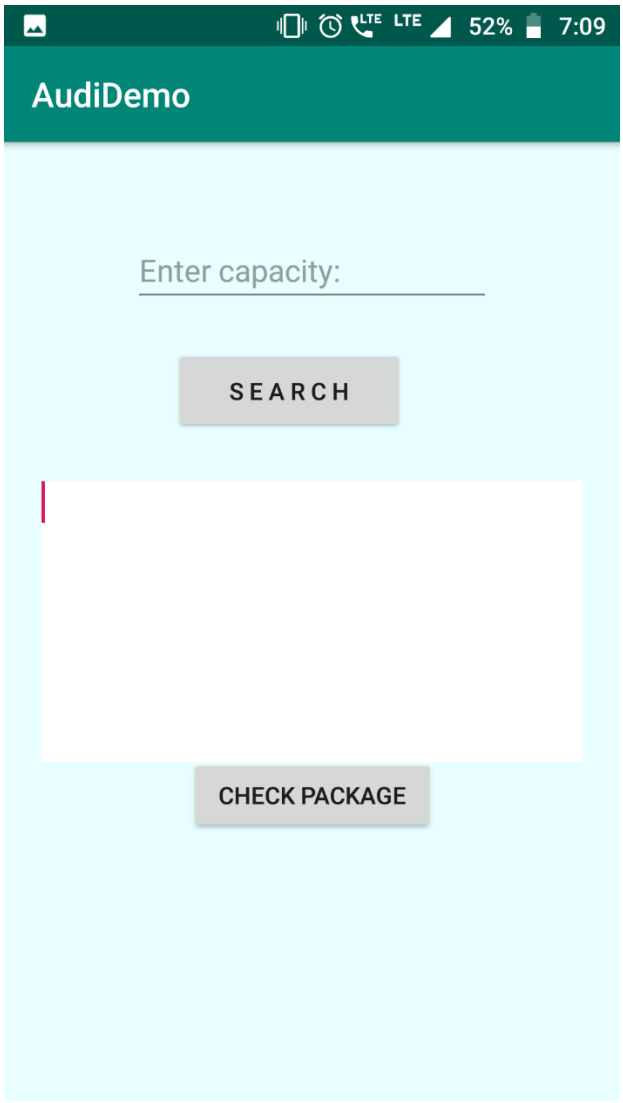
- **Customer can provide personal information:**



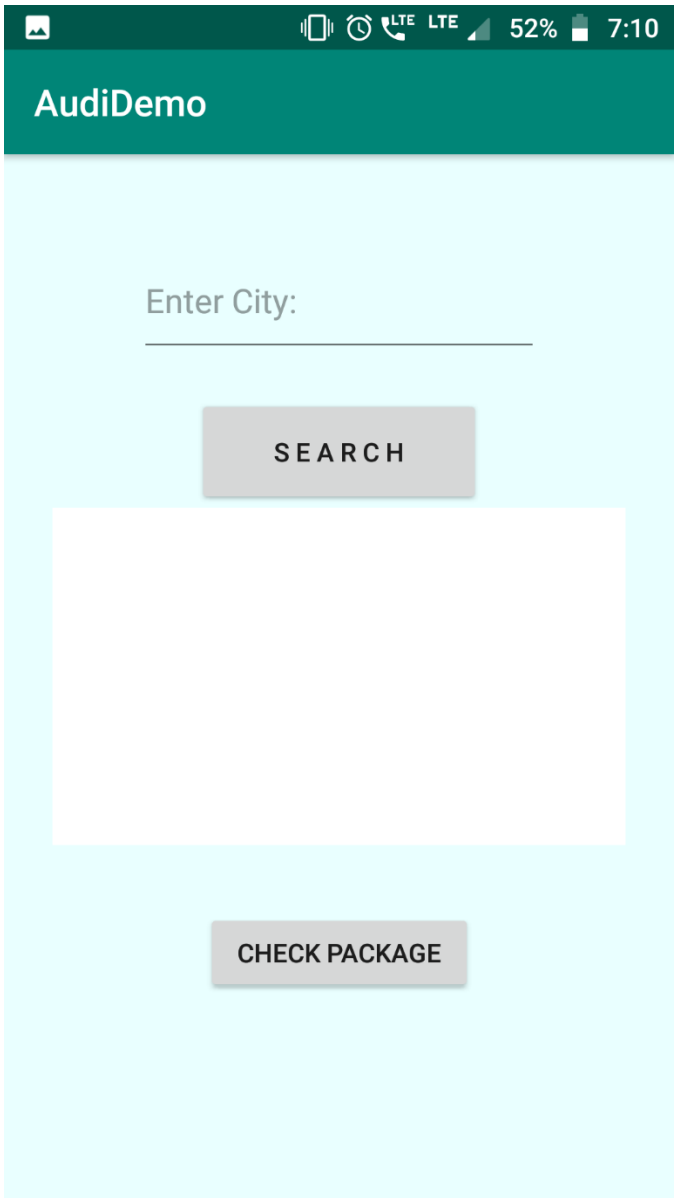
- **Customer search panel:**



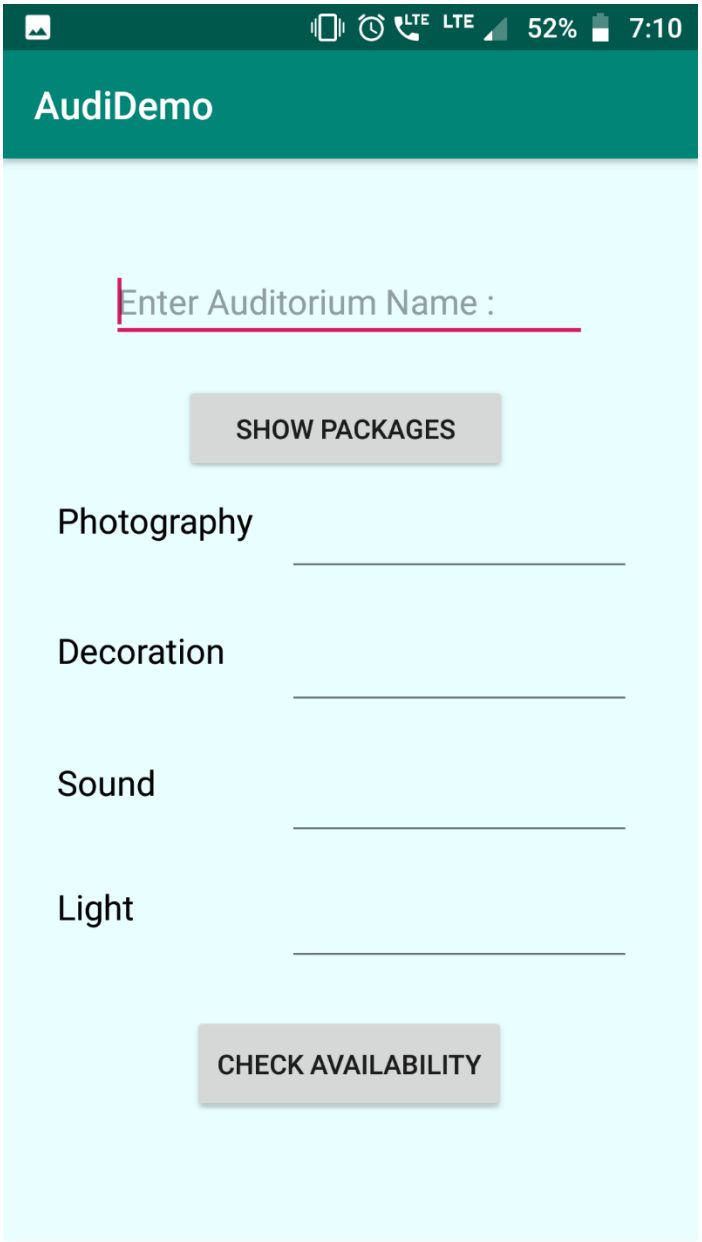
- **Customer can search auditorium by Capacity:**



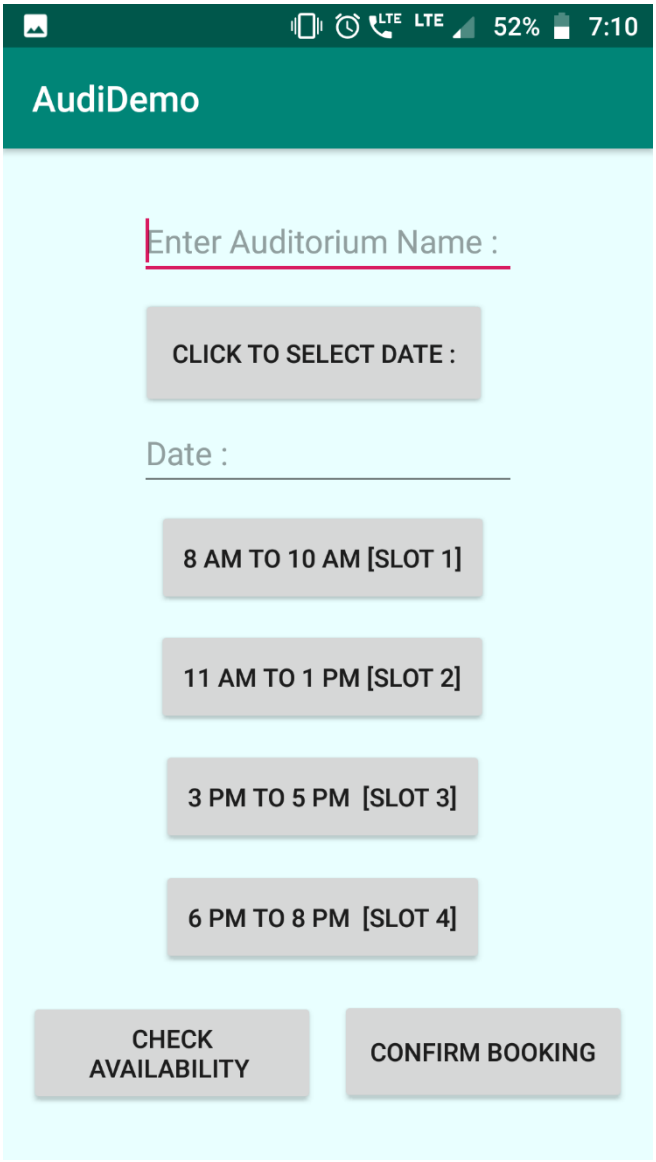
- **Customer can search auditorium by Location:**



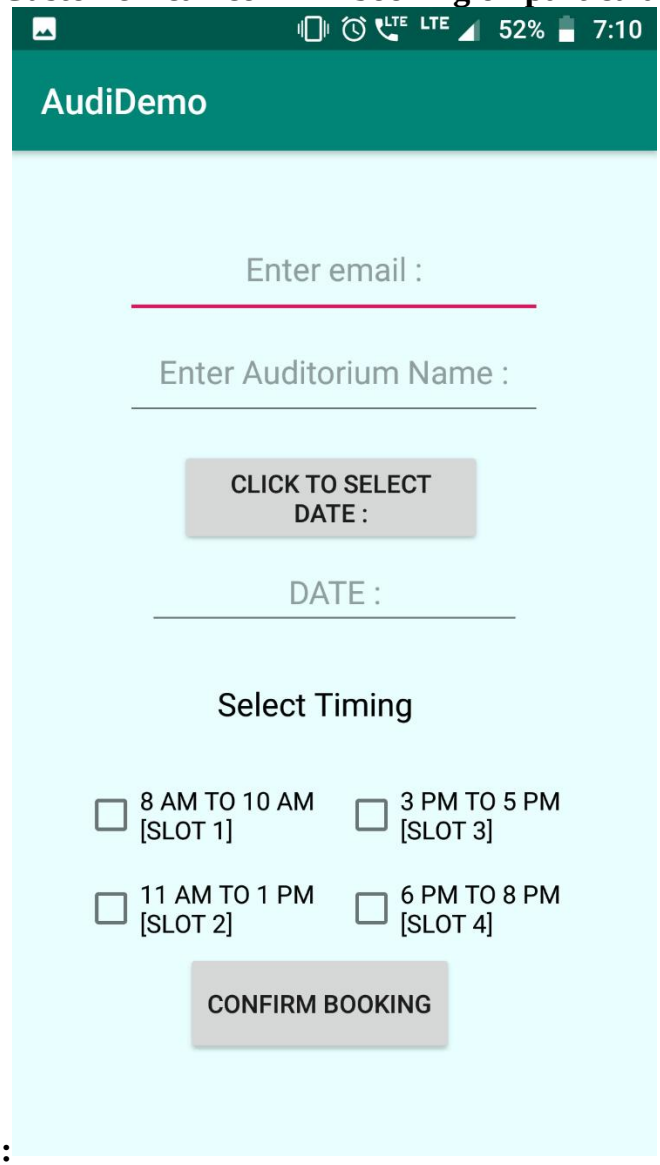
- **Customer can check package information on as per auditorium:**



- **Customer can check auditorium is available on particular date and time :**



- **Customer can confirm booking on particular date and time**



:

4.2 Operations manual –

A) There are 2 users of this system-

- Auditorium Owner
- Auditorium Customer

B) The Auditorium Owner has the following functions :

- Registration
- Login
- Provide auditorium’s details
- Provide package information
- Check auditorium bookings

c) The Auditorium Customer has following functions :

- Registration
- Login
- Search auditorium by capacity
- Search auditorium by location
- Check packages of different auditorium
- Check the availability of auditorium
- Select the date and time slot
- Confirm booking

4.3 Program specifications

OWNER REGISTRATION

Module	OWNER	
Program Name	Registration	
Purpose	Add owner's details, Auditorium details	
Input	**Constraints**	
User details	The Required field should not be null	**Description** Filling of required details.
Output	The Owner details are stored in the registration table and user gets the acknowledgment message of the record added.	User details get stored

VEHICLE REGISTRATION

Module	Customer	
Program Name	Customer Registration	
Purpose	Search and confirm booking	
Input	**Constraints**	
Vehicle details	The Required field should not be null	**Description** Filling of required details.
Output	The Customer details are stored in the customer table and admin gets the acknowledgment message of the record added.	customer details get stored

5. Drawbacks and limitations

Owner :

1.Owner cannot cancel booking for any date on the system

2.Customization in package is limited.

3.Access of one owner is limited to one auditorium.

4.At few instances response time is higher

5.

Customer :

1. Customer cannot compare more than one auditorium.

2. Customer cannot submit their review in the current system

3 Customization of package for customer is not available.

4.At few instances response time is higher

5.

Limitations of the system:

- System is based on SQLite database, so it does not have network connectivity.
- System does not have real time database.
- System does not provide a way to customize the auditorium package

6.Proposed Enhancement

- System will be based on real time database
- System will support network connectivity
- System will provide payment gateways
- System will allow ticket booking facility
- Owner can customize the package in a better way
- Owner can have access to more than one auditorium
- Owner can check reports in a better way
- Customer can submit their review.
- Customer can compare more than one auditorium.
- Customer can share the event details to his customers’

7.CONCLUSIONS

This project provided me the chance of working with technologies like android SQLite.

The development of any business application is part of our MCA curriculum. The project was expected to complete using **System Development Life Cycle (SDLC)** approach. The actual implementation of these aspects gave exposure to the problem in real life situations involved in project development and how to handle them in an efficient manner.

All the features in this application are developed as per customer specifications and as per they are specified in the documentation, thus satisfying all the requirements given by the agency.

8. Bibliography

1.Android: a. Learn android development i.e. Udemy Course

b. Android development i.e. android application

c. Stack overflow i.e. website

2. SQLite: a. Tutorials Point website

b. SQLite.org website

References :

Android: https://developer.android.com/docs

https://stackoverflow.com

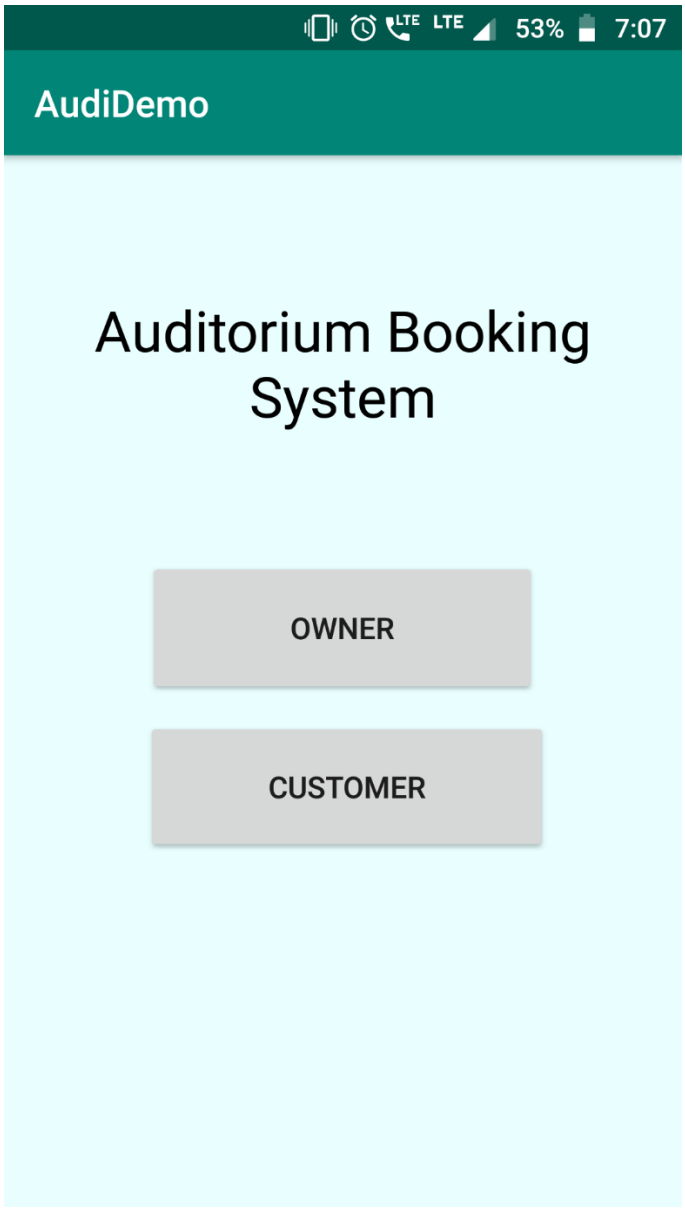
SQLite: https://sqlite.org/android/doc/trunk/www/index.wiki

https://www.tutorialspoint.com/android/android_sqlite_database.htm

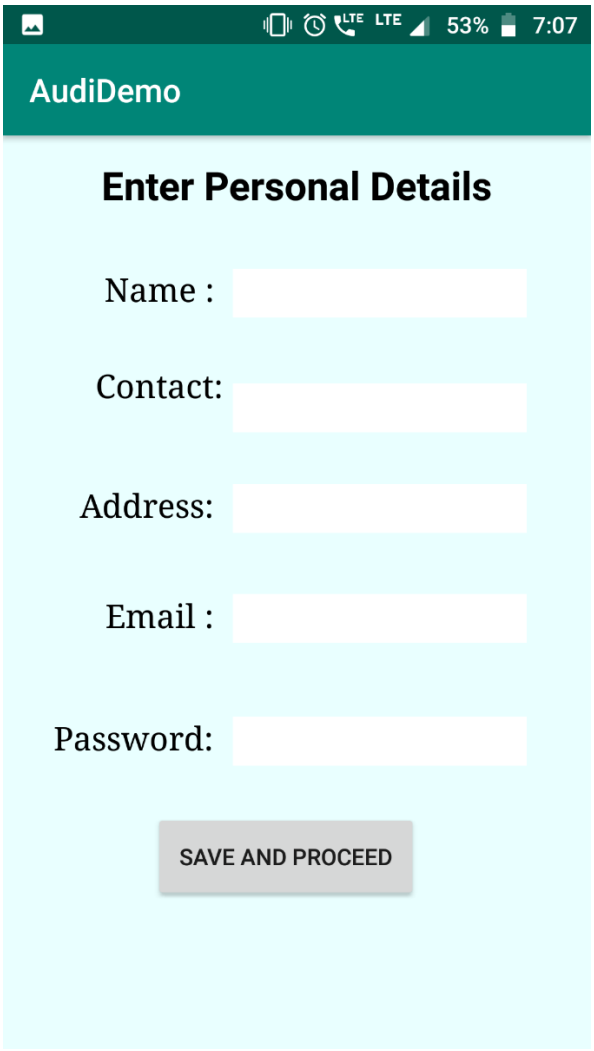
ANNEXRUES

Annexure 1- annexure for user interface

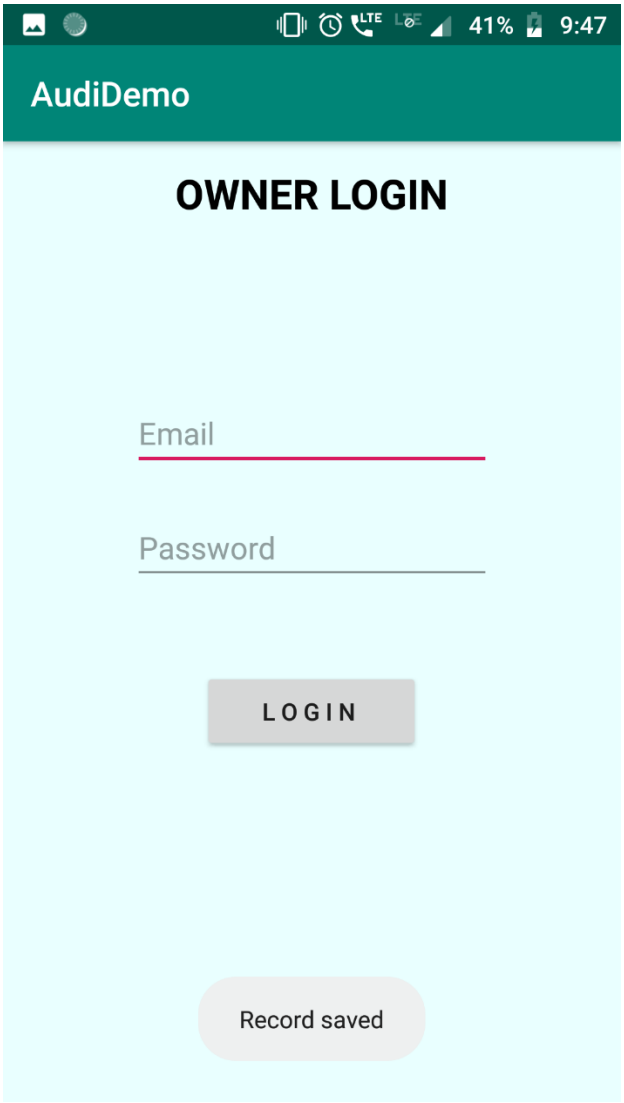
- Main Screen



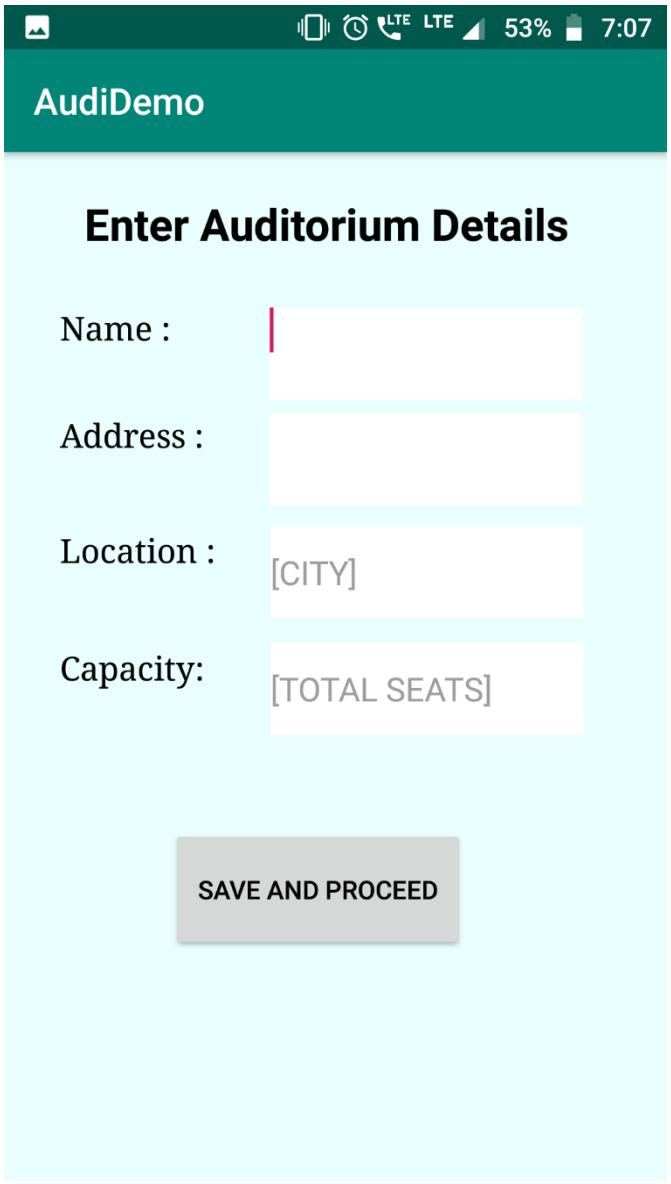
- **Owner registration screen :**



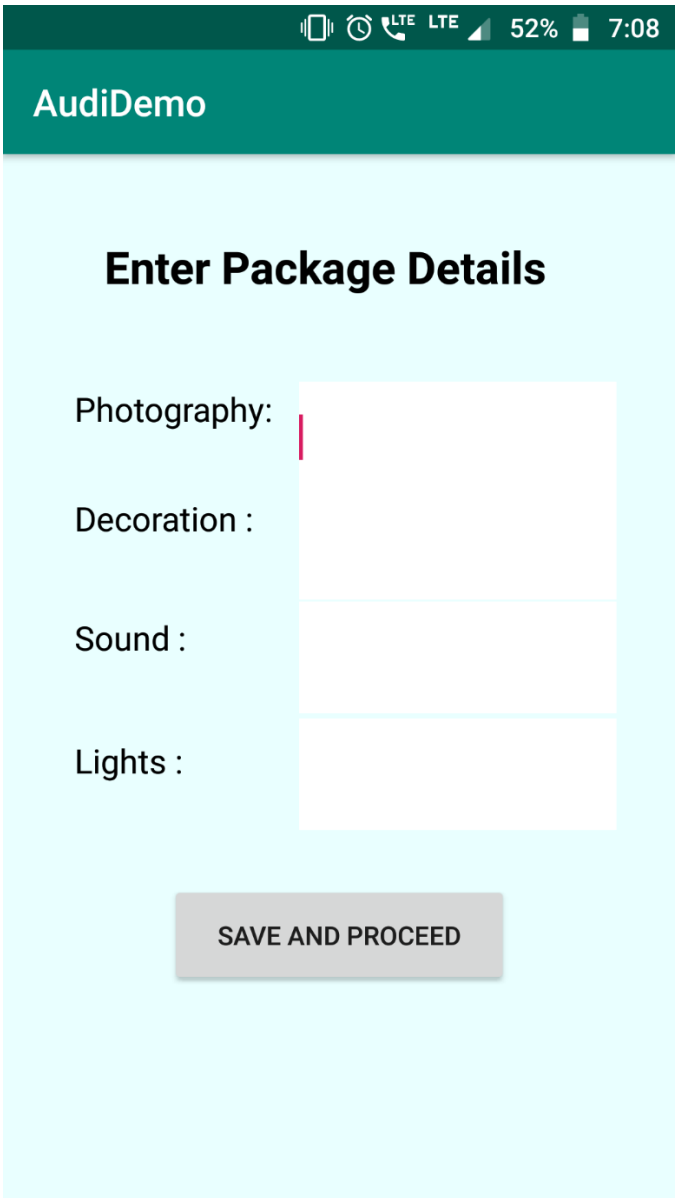
- **Owner login screen :**



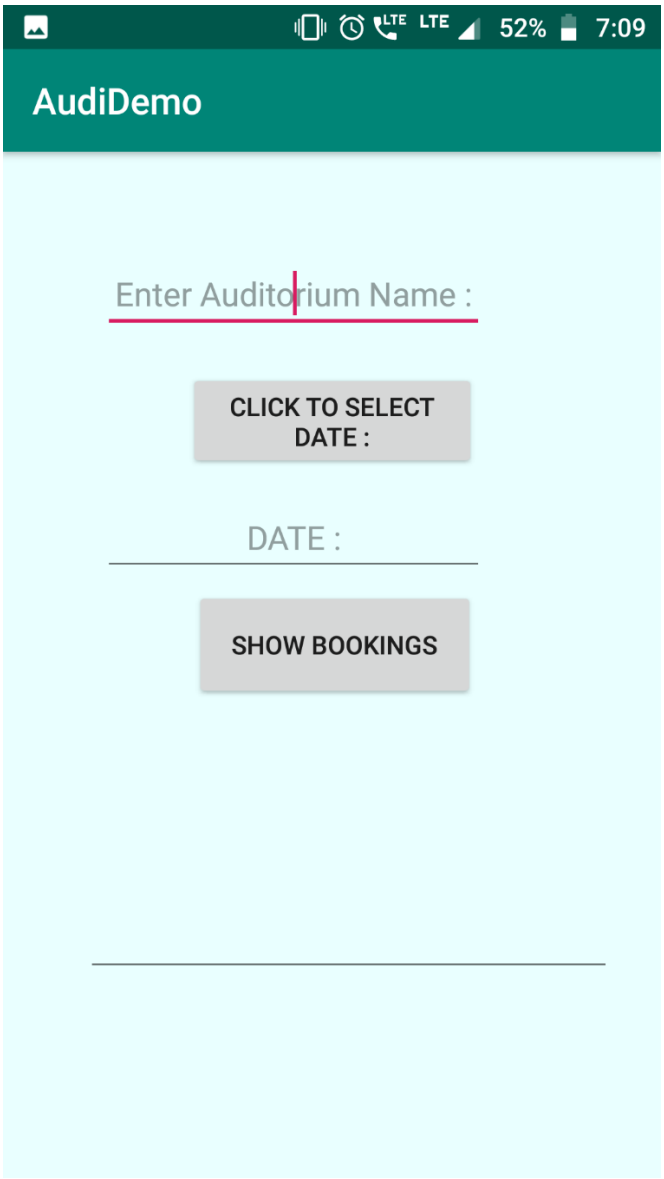
- **Auditorium registration screen :**



- **Package information screen :**

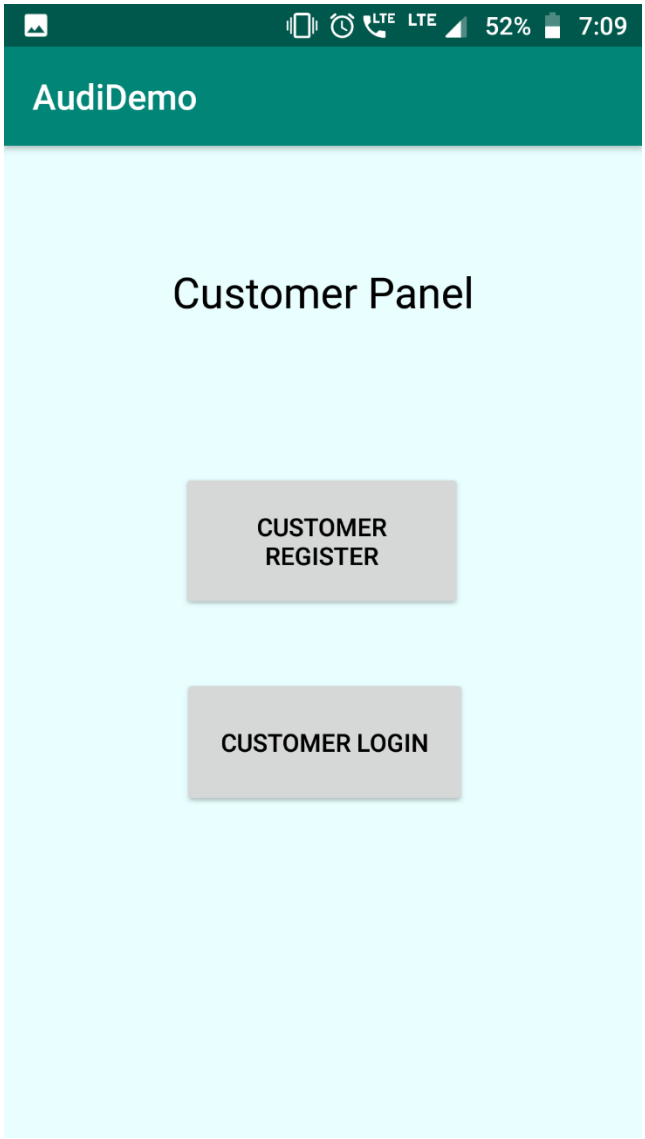


- **Owner auditorium availability screen :**

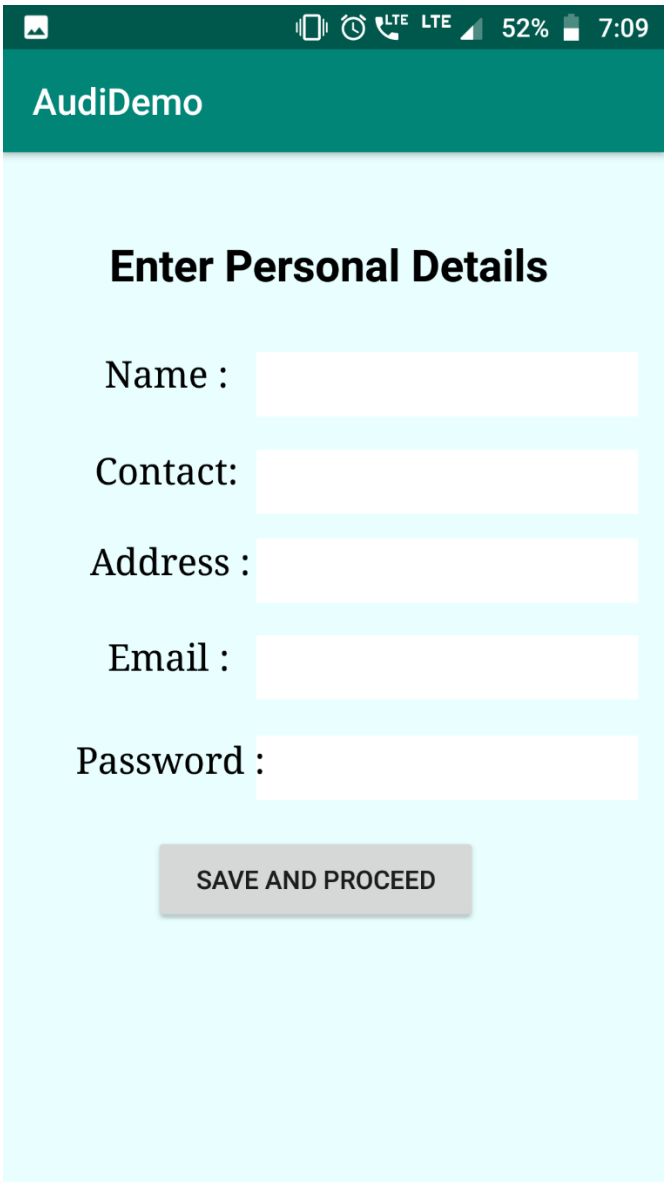


C. Auditorium Customer :

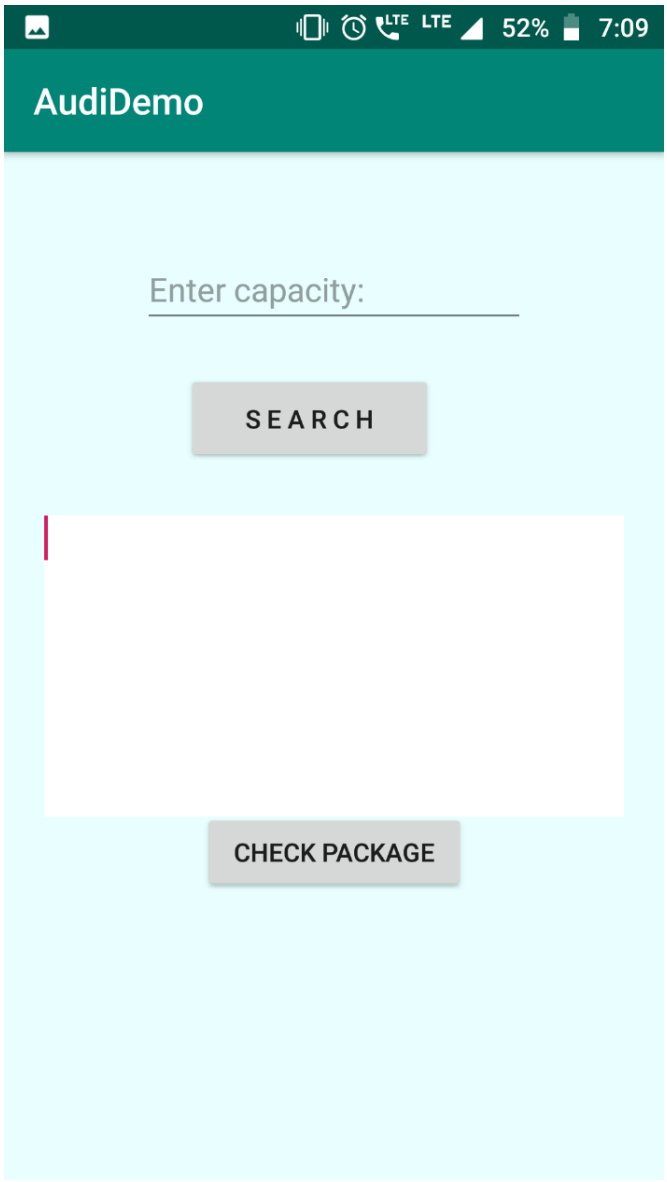
- **Customer search panel :**



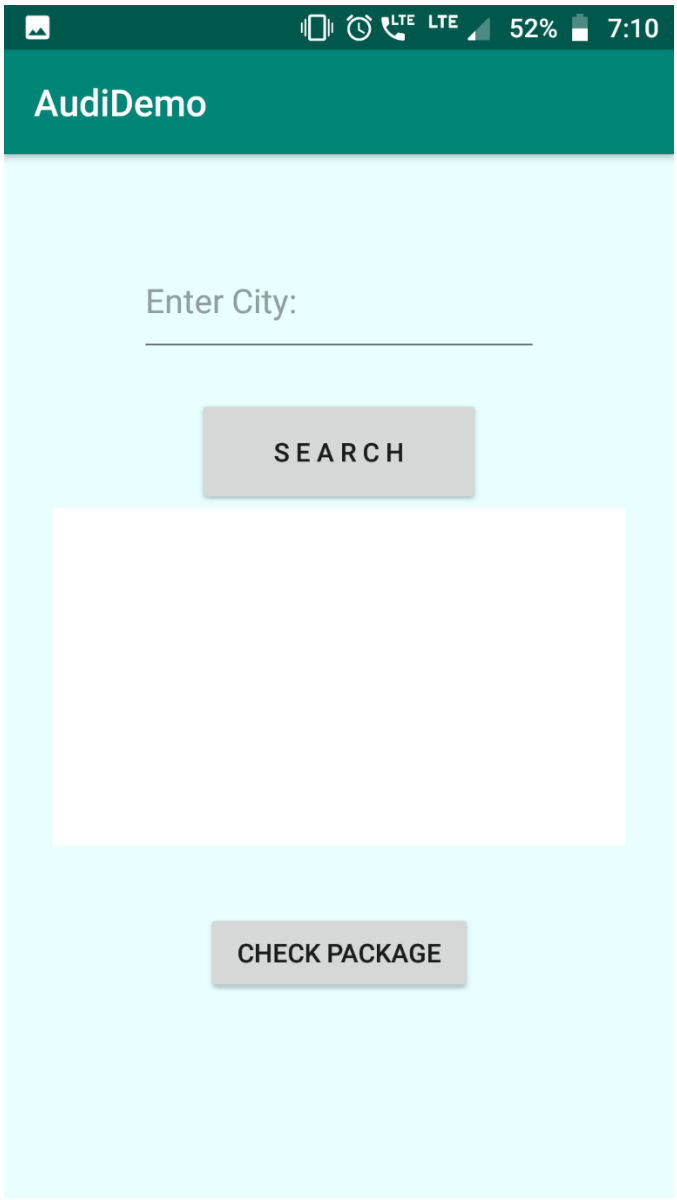
- **Customer registration screen :**



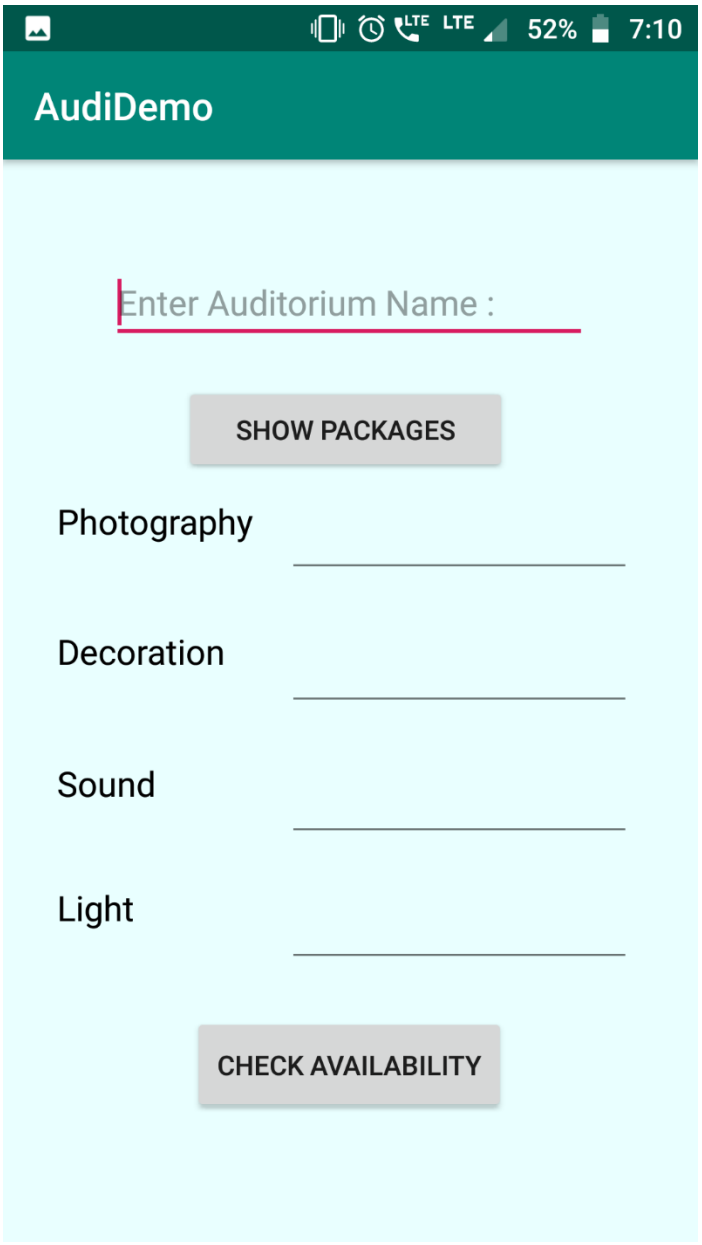
- **Search by capacity screen :**



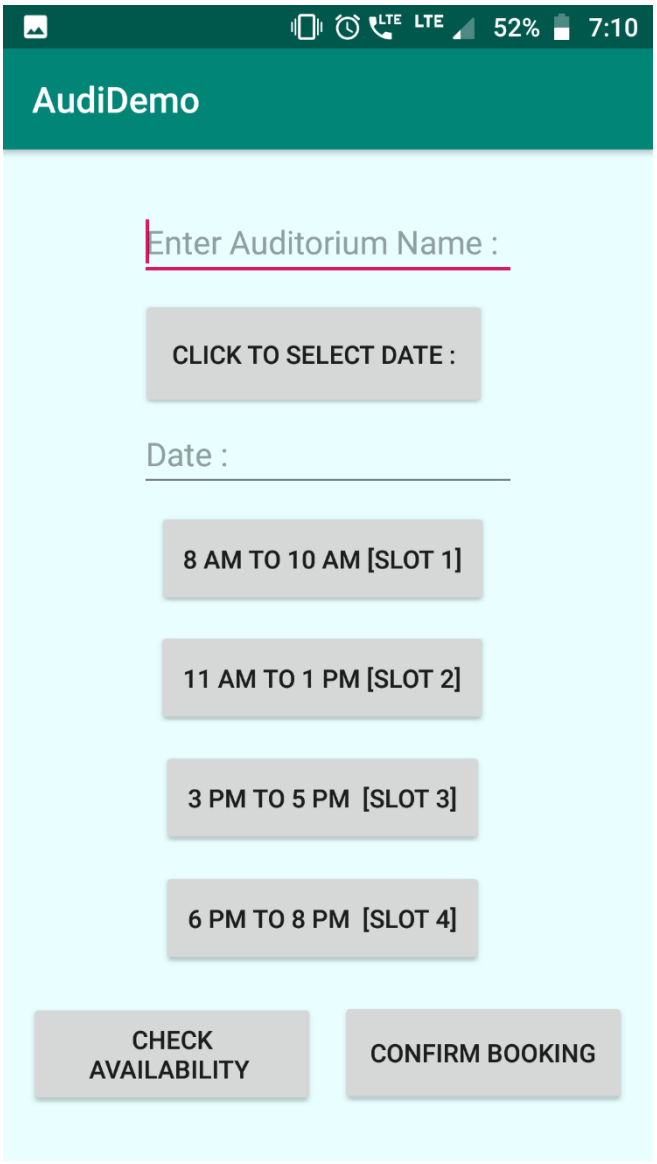
- **Search by Location :**



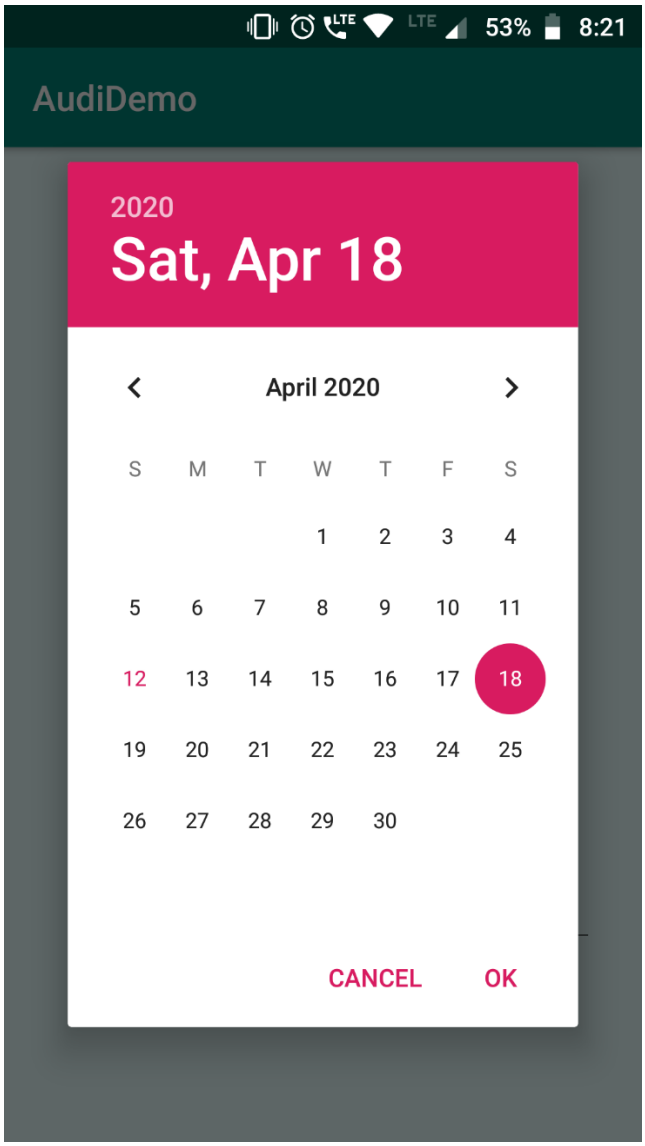
- **Check package information screen :**



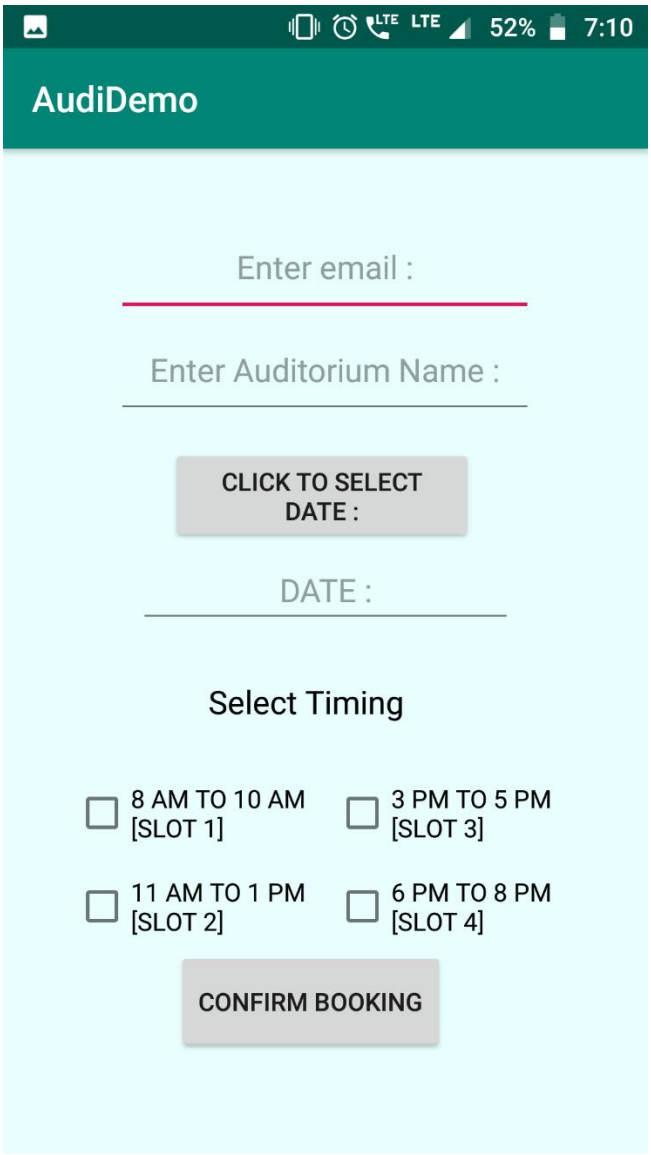
- **Check auditorium availability screen :**



- **Customer select date screen :**

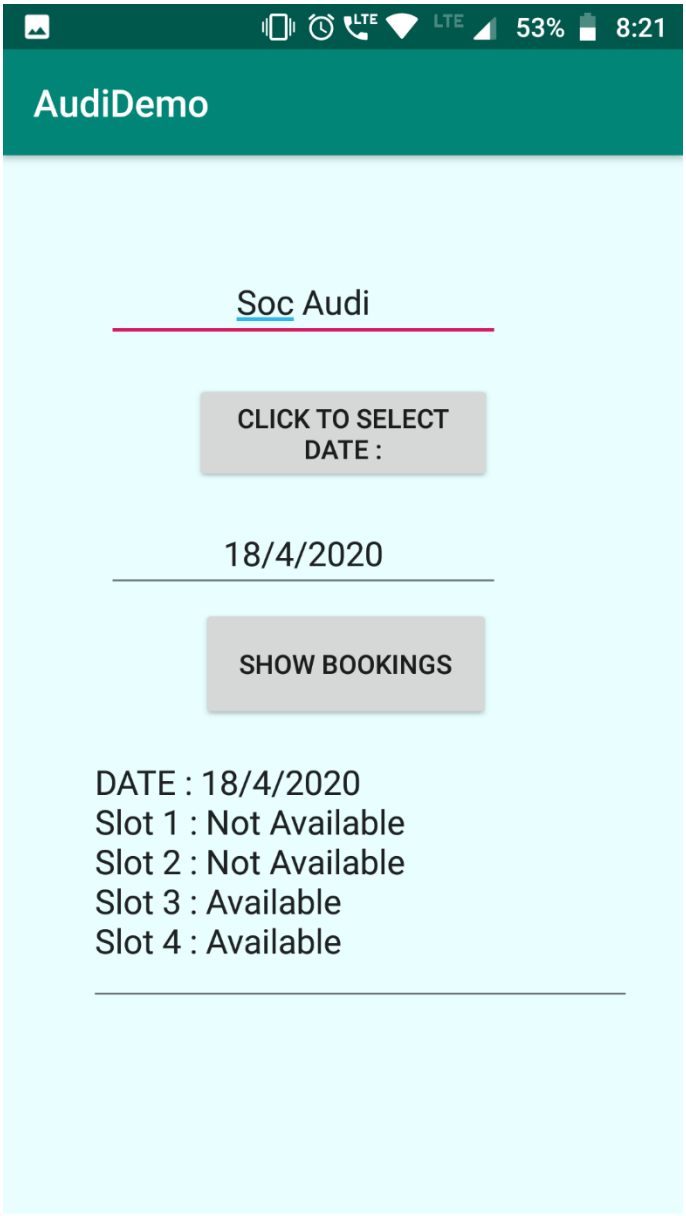


- **Confirm booking screen:**

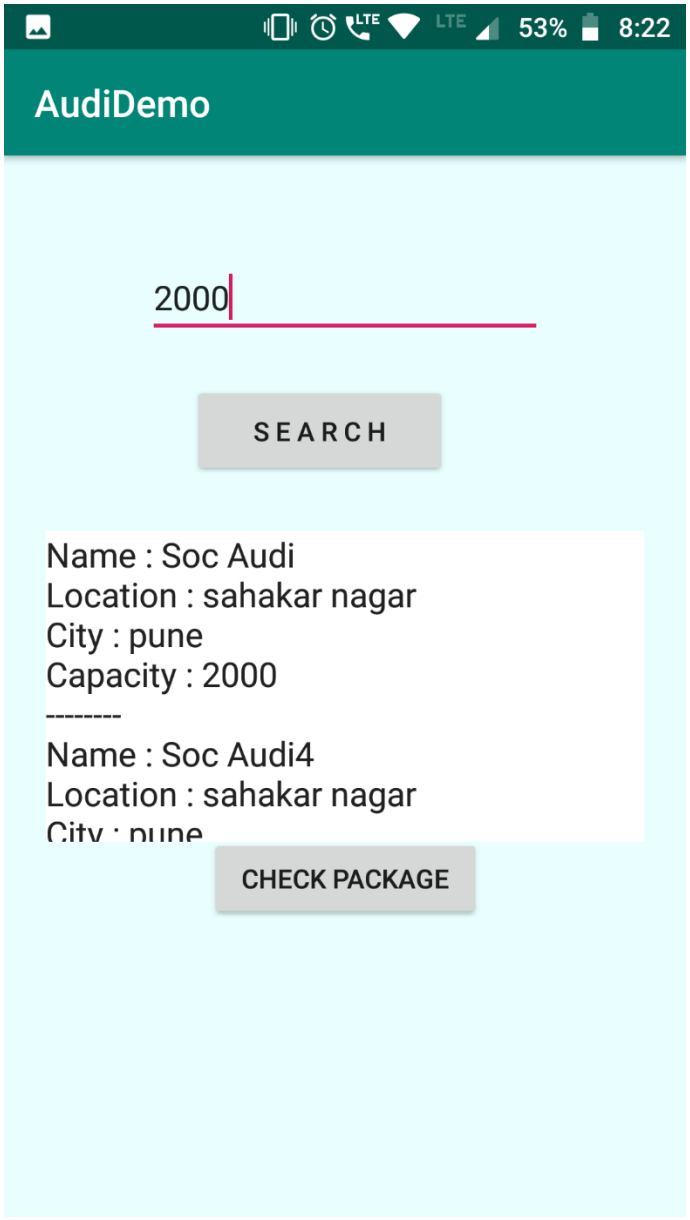


Annexure2- User interface with data

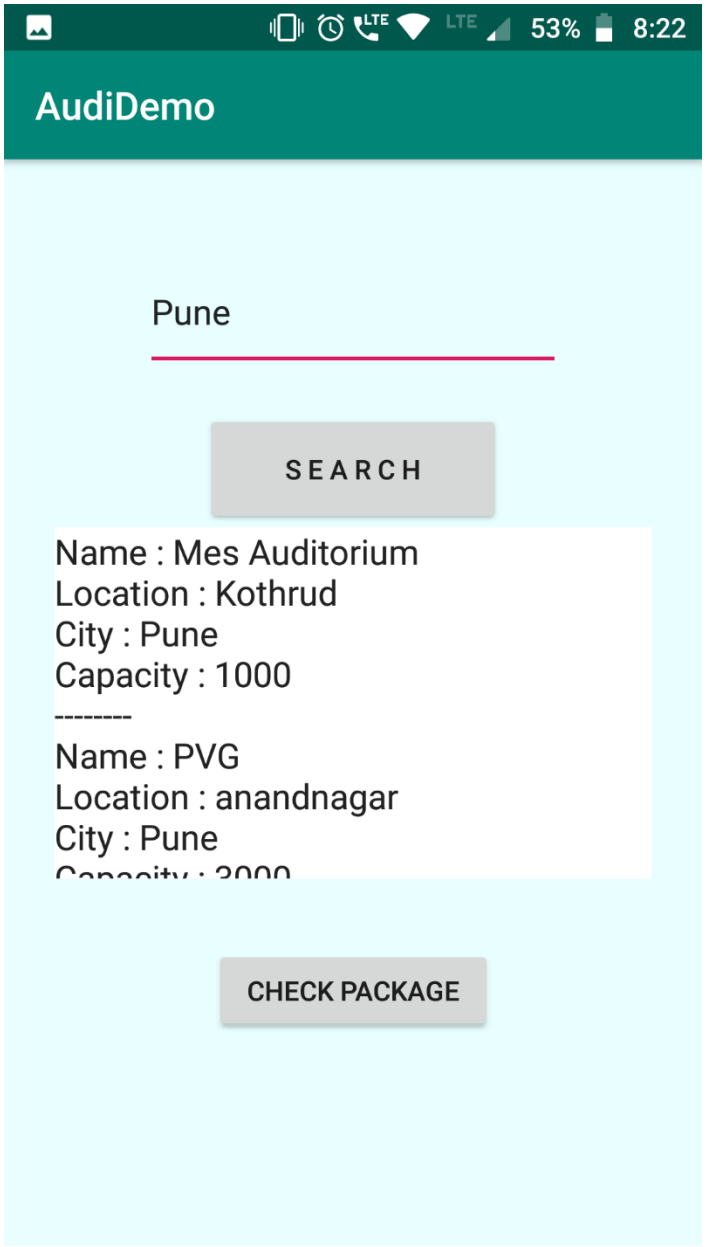
1. **Owner Checking Bookings of particular date and time :**



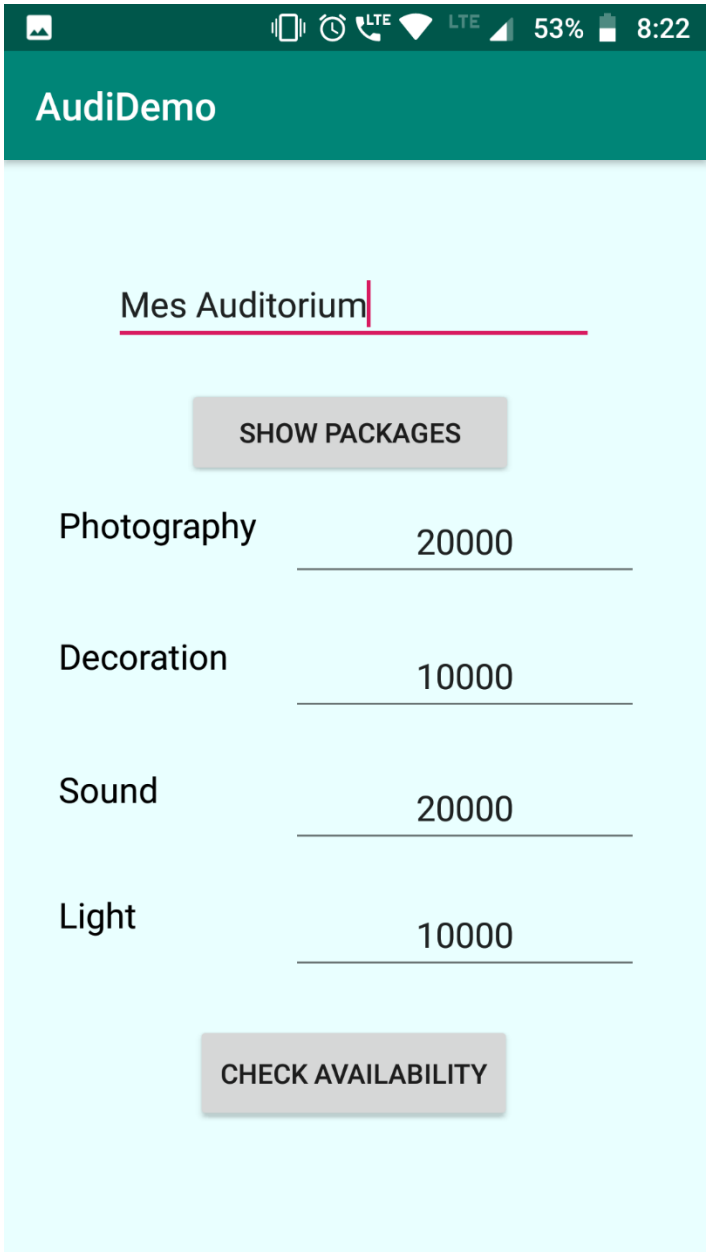
2.Customer search by capacity report :



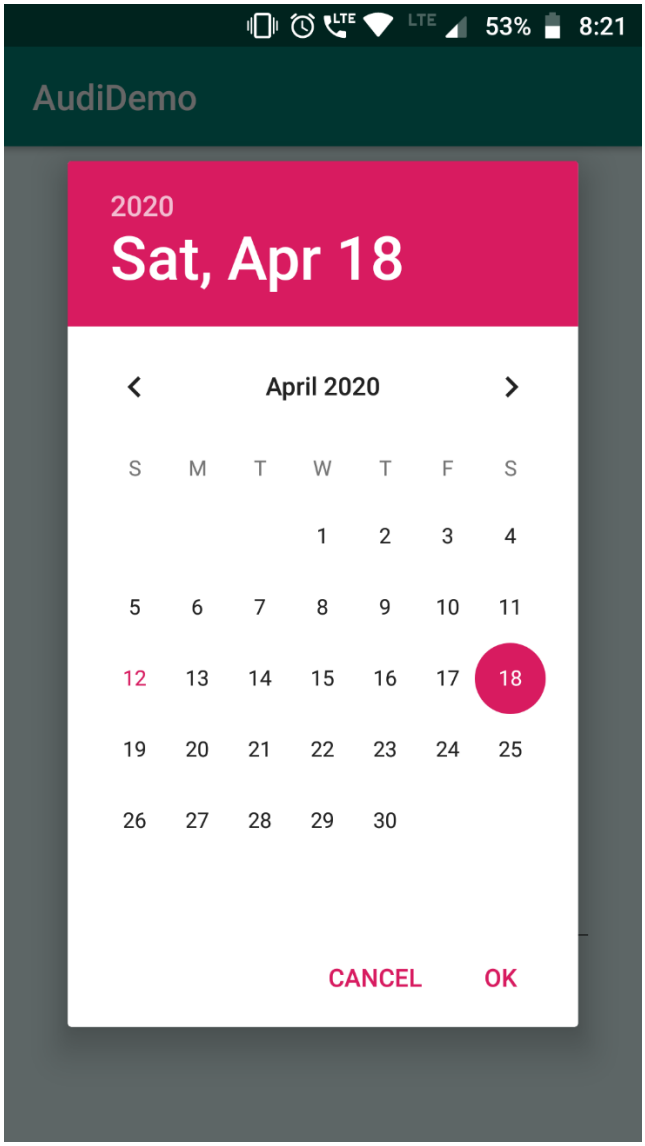
2. Customer search by city report :



3. **Customer check package report :**

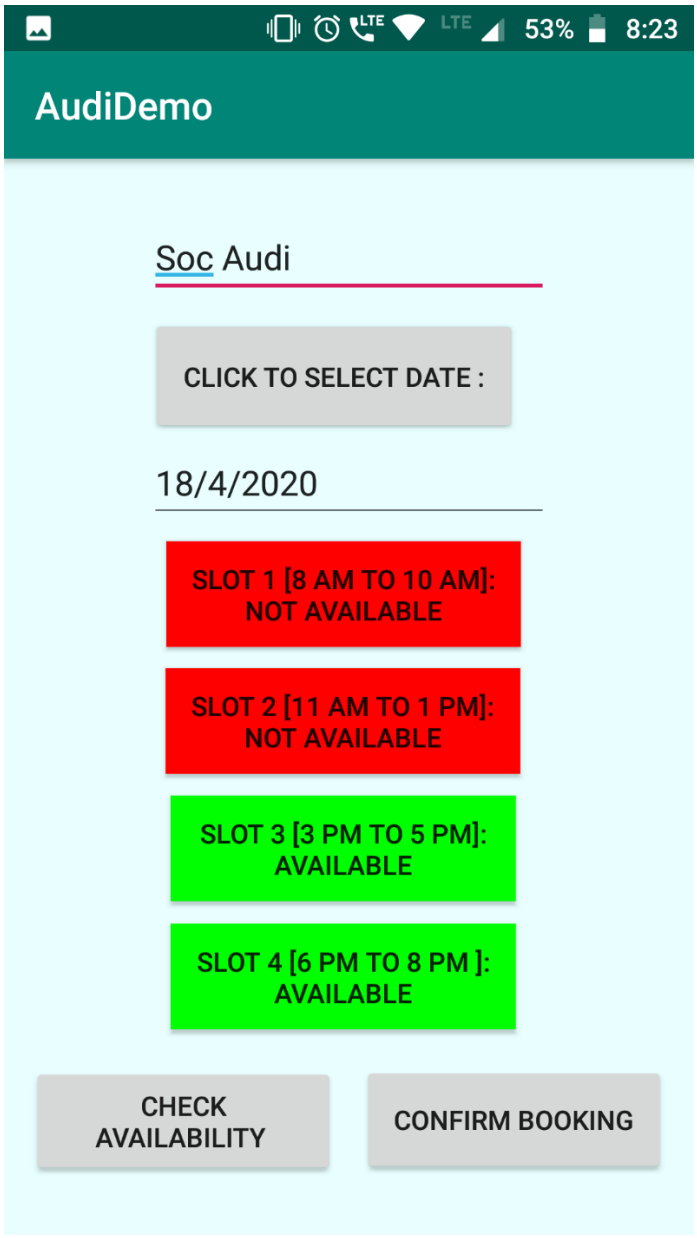


4. Customer select date screen :

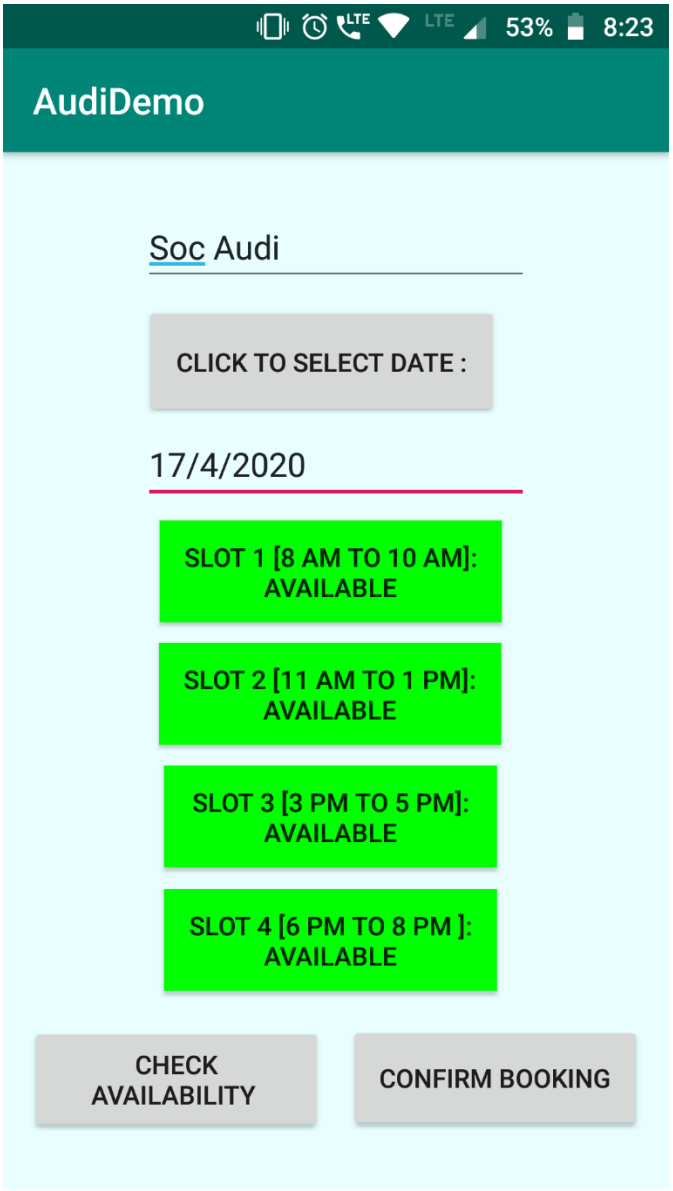


5. Customer check auditorium availability :

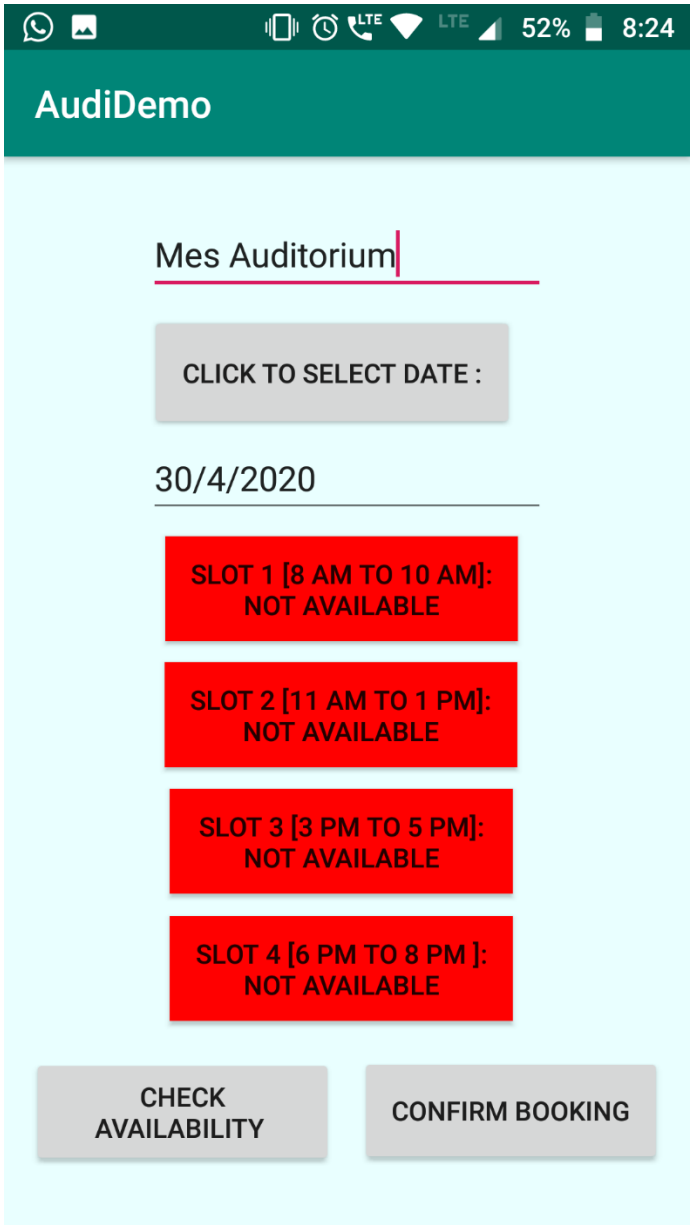
A. When few slots are available and few are not available.



B. When all slots are available :



C. When all slots are booked :



Annexure 3

1. ActivityMain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBackground"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView32"
        android:layout_width="316dp"
        android:layout_height="102dp"
        android:text="Auditorium Booking System"
        android:textAlignment="center"
        android:textColor="@android:color/black"
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.159" />

    <Button
        android:id="@+id/btnGoToOwner"
        android:layout_width="208dp"
        android:layout_height="74dp"
        android:width="120dp"
        android:height="40dp"
        android:onClick="btnClick"
        android:text="OWNER"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.442" />
```

```
    <Button
        android:id="@+id/btnGoToCutomer"
        android:layout_width="215dp"
        android:layout_height="73dp"
        android:width="120dp"
        android:height="40dp"
        android:onClick="btnClick"
        android:text="CUSTOMER"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.515"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.615" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. OwnerLogin.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".AudiOwnerLogin"
   android:background="@color/colorBackground">

   <TextView
      android:id="@+id/textView24"
      android:layout_width="314dp"
      android:layout_height="54dp"
      android:layout_marginStart="8dp"
      android:layout_marginTop="8dp"
      android:layout_marginEnd="8dp"
      android:layout_marginBottom="8dp"
      android:text="OWNER LOGIN"
      android:textAlignment="center"
      android:textColor="@android:color/background_dark"
      android:textSize="24sp"
      android:textStyle="bold"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.493"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.012" />
   <EditText
      android:id="@+id/etOwnerEmailLog"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="8dp"
      android:layout_marginTop="8dp"
      android:layout_marginEnd="8dp"
      android:layout_marginBottom="8dp"
      android:ems="10"
      android:hint="Email "
      android:inputType="textPersonName"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"

      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.28" />

   <Button
```

```
        android:id="@+id/btnOwnerLogin"
        android:layout_width="128dp"
        android:layout_height="49dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="L O G I N "
        android:onClick="click"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView24"
        app:layout_constraintVertical_bias="0.551" />

    <EditText
        android:id="@+id/etOwnerPassLog"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView24"
        app:layout_constraintVertical_bias="0.4" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.OwnerRegistration.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nestedScrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorBackground">

    <TextView
        android:id="@+id/textView15"
        android:layout_width="314dp"
        android:layout_height="54dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:text="Enter Personal Details"
        android:textAlignment="center"
        android:textColor="@android:color/background_dark"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.012" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="110dp"
        android:layout_height="30dp"
        android:layout_marginStart="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:fontFamily="serif"
        android:text="Name :"
        android:textAlignment="center"
        android:textColor="@android:color/background_dark"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.14"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.155" />

<TextView
    android:id="@+id/textView"
    android:layout_width="110dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:fontFamily="serif"
    android:text="Contact:"
    android:textAlignment="center"
    android:textColor="@android:color/background_dark"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.14"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.268" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="110dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:fontFamily="serif"
    android:text="Email :"
    android:textAlignment="center"
    android:textColor="@android:color/background_dark"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.538" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="131dp"
```

```
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:fontFamily="serif"
    android:text="Password:"
    android:textAlignment="center"
    android:textColor="@android:color/background_dark"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.03"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.682" />

<TextView
    android:id="@+id/textView21"
    android:layout_width="118dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:fontFamily="serif"
    android:text="Address:"
    android:textAlignment="center"
    android:textColor="@android:color/background_dark"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.093"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.409" />

<EditText
    android:id="@+id/etName"
    android:layout_width="180dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorText"
    android:ems="10"
    android:inputType="textPersonName"
```

```
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.809"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.143" />

<EditText
    android:id="@+id/etPassword"
    android:layout_width="180dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorText"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.809"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.678" />

<EditText
    android:id="@+id/etContact"
    android:layout_width="180dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorText"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.809"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.28" />

<EditText
    android:id="@+id/etAddr"
    android:layout_width="180dp"
```

```
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorText"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.809"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.4" />

<EditText
    android:id="@+id/etEmail"
    android:layout_width="180dp"
    android:layout_height="30dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:background="@color/colorText"
    android:ems="10"
    android:inputType="textPersonName"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.809"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.531" />

<Button
    android:id="@+id/btnSaveOwnerInfo"
    android:layout_width="163dp"
    android:layout_height="56dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:onClick="btnClick"
    android:text="Save and Proceed"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.463"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.832" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

7.OwnerRegistration.java

```
package com.example.audidemo;

import androidx.appcompat.app.AppCompatActivity;
import android.content.ContentValues;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class RegistrationForm extends AppCompatActivity {
    Button btnSaveInfo;
    EditText name,contact,address,email,password;
    DatabaseController controller;
    AuditoriumOwnerDAO ownerDAO;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registration_form);
        controller= new DatabaseController(this);
        btnSaveInfo = findViewById(R.id.btnSaveOwnerInfo);
        name = findViewById(R.id.etName);
        contact = findViewById(R.id.etContact);
        address = findViewById(R.id.etAddr);
        email = findViewById(R.id.etEmail);
        password=findViewById(R.id.etPassword);
    }
```

```
    public void btnClick(View view){
        if(view.getId()==R.id.btnSaveOwnerInfo)
        {
            String NAME,CONTACT,ADDRESS,EMAIL,PASSWORD;
            NAME = name.getText().toString();
            CONTACT = contact.getText().toString();
            ADDRESS = address.getText().toString();
            EMAIL = email.getText().toString();
            PASSWORD =password.getText().toString();

            ownerDAO = new
AuditoriumOwnerDAO(NAME,CONTACT,ADDRESS,EMAIL,PASSWO
RD);

            SQLiteDatabase db = controller.getWritableDatabase();
            ContentValues contentValues = new ContentValues();
            contentValues.put("NAME", ownerDAO.getNAME());
            contentValues.put("CONTACT", ownerDAO.getADDRESS());
            contentValues.put("ADDRESS", ownerDAO.getCONTACT());
            contentValues.put("EMAIL", ownerDAO.getEMAIL());
            contentValues.put("PASSWORD", ownerDAO.getPASSWORD());


            long result = db.insert("AudiOwner", "", contentValues);
            if (result > 0) {
                Toast.makeText(this, "Record saved",
Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(this , AudiOwnerLogin.class);
                startActivity(intent);

            } else {
                Toast.makeText(this, "Error"+result,
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

8. AddAudiInfo.java

```
package com.example.audidemo;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddAudiInfo extends AppCompatActivity {
    Button addPackage,saveAudiInfo;
    EditText
etAudiName,etAudiAddr,etAudiCity,etAudiCapacity,etFetchOwnerID;
    DatabaseController controller;
    AuditoriumInfoDAO infoDAO;
    int Owner_ID_val=0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_audi_info);

        controller = new DatabaseController(this);
        Owner_ID_val = controller.getMaxOwnerID();

        saveAudiInfo = findViewById(R.id.btnSaveAudiInfo);

        etAudiName = findViewById(R.id.etGetAudiName);
        etAudiAddr = findViewById(R.id.etAudiAddr);
        etAudiCity = findViewById(R.id.etAudiCity);
        etAudiCapacity = findViewById(R.id.etAudiCapacity);

    }
```

```
    public void click(View view)
    {
        if(view.getId()==R.id.btnSaveAudiInfo){
                String AudiName,AudiAddr,AudiCity,AudiCapacity;
                 AudiName = etAudiName.getText().toString();
                 AudiAddr = etAudiAddr.getText().toString();
                 AudiCity = etAudiCity.getText().toString();
                 AudiCapacity = etAudiCapacity.getText().toString();
                 infoDAO = new
AuditoriumInfoDAO(AudiName,AudiAddr,AudiCity,AudiCapacity);

                SQLiteDatabase db1 = controller.getWritableDatabase();
                ContentValues values = new ContentValues();

                values.put("AUDI_NAME",infoDAO.getAudiName());
                values.put("AUDI_ADDRESS",infoDAO.getAudiAddr());
                values.put("AUDI_CITY",infoDAO.getAudiCity());
                values.put("AUDI_CAPACITY",infoDAO.getAudiCapacity());
                values.put("OWNER_ID",Owner_ID_val);

                long result = db1.insert("AudiInfo", "", values);
                if (result > 0) {
                   Toast.makeText(this, "Record saved ",
Toast.LENGTH_SHORT).show();
                   Intent i = new Intent(this, AddPackageInfo.class);
                   startActivity(i);
                } else {
                   Toast.makeText(this, "Error" + result,
Toast.LENGTH_SHORT).show();
                }
       }
    }
```

9. AddPackageInfo

```
package com.example.audidemo;

import androidx.appcompat.app.AppCompatActivity;

import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class AddPackageInfo extends AppCompatActivity {
    DatabaseController controller;
    EditText etPhotography,etDecoration,etSound,etLight;
    Button btnSavePackageInfo;
    AuditoriumPackageDAO packageDAO;
    int AudiMaxValue = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_package_info);

        etPhotography=findViewById(R.id.etPhotography);
        etDecoration=findViewById(R.id.etDecoration);
        etSound=findViewById(R.id.etSound);
        etLight=findViewById(R.id.etLight);
        btnSavePackageInfo=findViewById(R.id.btnSavePackageInfo);
        controller =new DatabaseController(this);
        AudiMaxValue = controller.getMaxAuditoriumID();
    }
```

```
    public void savePackage(View view){
        if (view.getId()==R.id.btnSavePackageInfo){
            String PHOTOGRAPHY,DECORATION,SOUND,LIGHT;

            PHOTOGRAPHY = etPhotography.getText().toString();
            DECORATION = etDecoration.getText().toString();
            SOUND = etSound.getText().toString();
            LIGHT = etLight.getText().toString();
            packageDAO = new
AuditoriumPackageDAO(PHOTOGRAPHY,DECORATION,SOUND,LIG
HT);

            SQLiteDatabase db =controller.getWritableDatabase();
            ContentValues values = new ContentValues();


values.put("PHOTOGRAPHY",packageDAO.getPHOTOGRAPHY());
            values.put("DECORATION",packageDAO.getDECORATION());
            values.put("SOUND",packageDAO.getSOUND());
            values.put("LIGHT",packageDAO.getLIGHT());
            values.put("AUDI_ID",AudiMaxValue);
            Toast.makeText(this,"Check 1",Toast.LENGTH_SHORT).show();
            long result = db.insert("PackageInfo", "", values);

            if (result > 0) {
                Toast.makeText(this,"Check 2",Toast.LENGTH_SHORT).show();
                Toast.makeText(this,"Inforamtion
Saved",Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(this , AudiPackagePanel.class);
                startActivity(intent);
            } else {
                Toast.makeText(this, "Error",Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

10. AudiOwnerCheckBookings.java

```
package com.example.audidemo;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;

import java.util.Calendar;

public class AudiOwnerCheckBooking extends AppCompatActivity implements
DatePickerDialog.OnDateSetListener{

    EditText etOwCBGetAudiName,etOwCBGetDate,etOwResult;
    DatabaseController controller;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audi_owner_check_booking);

        controller = new DatabaseController(this);

        etOwCBGetAudiName = findViewById(R.id.etOwCBGetAudiName);
        etOwCBGetDate = findViewById(R.id.etOwCBGetDate);
        etOwResult = findViewById(R.id.etOwCBResult);
    }

    public void OwnerCheckBooking(View view){
        if (view.getId()==R.id.btnOwCBDATE){
            showDatePickerDialog();
        }
        else if(view.getId()==R.id.btnOwShowB){
            String getAudiName,getDATE;
            int AudiID;
            getAudiName = etOwCBGetAudiName.getText().toString();
            getDATE = etOwCBGetDate.getText().toString();
            AudiID = getAudiID(getAudiName);

            Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT *
FROM AudiBooking WHERE DATE='"+getDATE+"' AND
AUDI_ID='"+AudiID+"'",null);
```

```
        etOwResult.append( "DATE : "+getDATE);
        if(cursor.getCount()==0){
            etOwResult.append("\nAll slots are available");
        }
        else{
            while (cursor.moveToNext()){

                if(cursor.getInt(4)==0){
                    etOwResult.append("\nSlot 1 : Available");
                }
                else if (cursor.getInt(4)==1){
                    etOwResult.append("\nSlot 1 : Not Available");
                }

                if(cursor.getInt(5)==0){
                    etOwResult.append("\nSlot 2 : Available");
                }
                else if (cursor.getInt(5)==1){
                    etOwResult.append("\nSlot 2 : Not Available");
                }

                if(cursor.getInt(6)==0){
                    etOwResult.append("\nSlot 3 : Available");
                }
                else if (cursor.getInt(6)==1){
                    etOwResult.append("\nSlot 3 : Not Available");
                }

                if(cursor.getInt(7)==0){
                    etOwResult.append("\nSlot 4 : Available");
                }
                else if (cursor.getInt(7)==1){
                    etOwResult.append("\nSlot 4 : Not Available");
                }
            }
        }
    }
}
```

```
    public void showDatePickerDialog(){
        DatePickerDialog datePickerDialog = new DatePickerDialog(
                this,
                this,
                Calendar.getInstance().get(Calendar.YEAR),
                Calendar.getInstance().get(Calendar.MONTH),
                Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
        );

        datePickerDialog.show();
    }

    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        String date = i2 +"/"+ (i1 + 1) +"/"+i;
        etOwCBGetDate.setText(date);
    }

    public int getAudiID(String AudiName){
        int AudiID=0;
        String thisAudiName = AudiName;
        Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT
AUDI_ID FROM AudiInfo WHERE AUDI_NAME='"+thisAudiName+"'",null);
        if (cursor!=null){
            cursor.moveToFirst();
            AudiID = cursor.getInt(0);
        }
        else{
            //Toast.makeText(this, "Incorrect Auditorium
Name",Toast.LENGTH_SHORT).show();
            AudiID=0;
        }
        return AudiID;
    }
}
```

11. CustomerRegistration.java

```
import android.content.ContentValues;
```

```
public class AddCustInfo extends AppCompatActivity {
    EditText custName,custContact,custEmail,custAddress,custPassword;
    Button btnCustSaveAndProceed;
    DatabaseController controller;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_cust_info);
        controller = new DatabaseController(this);
        custName = findViewById(R.id.etCustName);
        custContact = findViewById(R.id.etCustContact);
        custAddress = findViewById(R.id.etCustAddress);
        custEmail = findViewById(R.id.etCustEmail);
        custPassword = findViewById(R.id.etCustPassword);
        btnCustSaveAndProceed = findViewById(R.id.btnGoCustLog);
    }
    public void custSaveClick(View view){
        if (view.getId()==R.id.btnGoCustLog){
            SQLiteDatabase db = controller.getWritableDatabase();
            ContentValues contentValues1 = new ContentValues();
            contentValues1.put("CustNAME", custName.getText().toString());
            contentValues1.put("CustCONTACT",
custContact.getText().toString());
            contentValues1.put("CustADDRESS",
custAddress.getText().toString());
            contentValues1.put("CustEMAIL", custEmail.getText().toString());
            contentValues1.put("CustPASSWORD",
custPassword.getText().toString());
            long result = db.insert("AudiCust", "", contentValues1);
            if (result > 0) {
                Toast.makeText(this,"Login Form",Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(this , AudiCustLogin.class);
                startActivity(intent);
            } else {
                Toast.makeText(this, "Error"+result,
Toast.LENGTH_SHORT).show();
            }       }    }    }
```

12..SearchByCapacity.java

```
import android.widget.Toast;

public class SearchByCapacity extends AppCompatActivity {
    DatabaseController controller;
    EditText etCapacity,etCapacityResult;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_by_capacity);
        etCapacity=findViewById(R.id.etCapacity);
        etCapacityResult=findViewById(R.id.etCapacityResult);
        controller = new DatabaseController(this);
    }
    public void capacitySearch(View view)    {
        if(view.getId()==R.id.btnCapacityResult) {
            String capacity = etCapacity.getText().toString();
            Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT *
FROM AudiInfo WHERE AUDI_CAPACITY='"+capacity.trim()+"'",null);
            if(cursor.getCount()==0) {
                etCapacityResult.setText("");
                if (capacity.equals("") || capacity.equals(null)) {
                    Toast.makeText(this, "Please enter capacity",
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(this, "No Records Found",
Toast.LENGTH_SHORT).show();
                }
            }
            else {etCapacityResult.setText("");
                while (cursor.moveToNext()) {
                    etCapacityResult.append("Name : "+cursor.getString(1) + "\nLocation :
"+cursor.getString(2) +"\nCity : "+ cursor.getString(3)+"\nCapacity : "+
cursor.getString(4)+"\n--------\n");
                }
            }
        }
        else if(view.getId()==R.id.btnCheckPackages1){
            Intent intent = new Intent(this,CheckPackageInfo.class);
            startActivity(intent);
        }
    }
}
```

13.SearchByLocation.java

```
import android.widget.Toast;
public class SearchByLocation extends AppCompatActivity {
   EditText etLocation,etLocationResult;
   DatabaseController controller;
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_search_by_location);

      controller = new DatabaseController(this);
      etLocation = (EditText)findViewById(R.id.etLocation);
      etLocationResult =(EditText) findViewById(R.id.etLocationResult);
   }

   public void locationSearch(View view){
      if(view.getId()==R.id.btnLocationResult)
      {   String city = etLocation.getText().toString();
         Cursor cursor= controller.getReadableDatabase().rawQuery("SELECT
* FROM AudiInfo WHERE AUDI_CITY='"+city.trim()+"'",null);
         if(cursor.getCount()==0) {
            etLocationResult.setText("");
            if (city.equals("") || city.equals(null)) {
               Toast.makeText(this, "Please enter city",
Toast.LENGTH_SHORT).show();
            } else {
               Toast.makeText(this, "No Records Found",
Toast.LENGTH_SHORT).show();
            }
         }
         else {
            etLocationResult.setText("");
            while (cursor.moveToNext()) {
               etLocationResult.append("Name : "+cursor.getString(1) +
"\nLocation : "+cursor.getString(2) +"\nCity : "+
cursor.getString(3)+"\nCapacity : "+ cursor.getString(4)+"\n--------\n");
            }     }       }
      else if(view.getId()==R.id.btnCheckPackages2){
         Intent intent = new Intent(this, CheckPackageInfo.class);
         startActivity(intent);
      }
   }
}
```

14.CheckPackageInfo.java

```
import android.widget.Toast;
public class CheckPackageInfo extends AppCompatActivity {
   DatabaseController controller;
EditTextetGetAudiName,etGetPhotograpgy,etGetDecoration,etGetSound,e
tGetLight;
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_check_package_info);
      controller = new DatabaseController(this);
      etGetAudiName = findViewById(R.id.etGetAudiName);
      etGetPhotograpgy= findViewById(R.id.etGetPhotography);
      etGetDecoration= findViewById(R.id.etGetDecoration);
      etGetSound= findViewById(R.id.etGetSound);
      etGetLight= findViewById(R.id.etGetLight);
   }
   public void showPackages(View view) {
      if(view.getId()==R.id.btnShowPackage){
         String AuditoriumName = etGetAudiName.getText().toString().trim();
         Cursor cursor =
controller.getReadableDatabase().rawQuery("SELECT * FROM
PackageInfo WHERE AUDI_ID = (SELECT AUDI_ID FROM AudiInfo
WHERE AUDI_NAME ='"+AuditoriumName+"')",null);
         if (cursor.getCount()==0){
            Toast.makeText(this, "No Records Found",
Toast.LENGTH_SHORT).show();
         }
         else{
            while(cursor.moveToNext()) {
                  etGetPhotograpgy.setText(cursor.getString(1));
                  etGetDecoration.setText(cursor.getString(2));
                  etGetSound.setText(cursor.getString(3));
                  etGetLight.setText(cursor.getString(4));
            }        }      }
      else if(view.getId()==R.id.btnCheckAvailability){
         Intent intent = new Intent(this,AudiCustomerCheckBooking.class);
         startActivity(intent);
      }
   }
}
```

15..CheckAudiAvailability.java

```
import android.widget.Toast;

import java.util.Calendar;

public class AudiCustomerCheckBooking extends AppCompatActivity implements
DatePickerDialog.OnDateSetListener {

    EditText etChkbookDate,etCheckbookName,etTemp;
    DatabaseController controller;
    Button btnChkB1,btnChkB2,btnChkB3,btnChkB4;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audi_customer_check_booking);
         controller = new DatabaseController(this);
        etChkbookDate = findViewById(R.id.etChkbookDate);
        etCheckbookName = findViewById(R.id.etChkbookAudiName);
        btnChkB1 = findViewById(R.id.btnChkbookSlot1);
        btnChkB2 = findViewById(R.id.btnChkbookSlot2);
        btnChkB3 = findViewById(R.id.btnChkbookSlot3);
        btnChkB4 = findViewById(R.id.btnChkbookSlot4);
    }

    public void chkbookAvailability(View view){
        if(view.getId()==R.id.btnChkbookDate){
            shoDateTimePicker();
        }
        else if(view.getId()==R.id.btnChkbookAvailability){
            String getAudiName,getDATE;
            getAudiName = etCheckbookName.getText().toString();
            getDATE = etChkbookDate.getText().toString();
            Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT *
FROM AudiBooking WHERE AUDI_NAME='"+getAudiName+"' AND
DATE='"+getDATE+"'",null);

            if(cursor.getCount()==0) {
                btnChkB1.setText("Slot 1 [8 AM TO 10 AM]:\nAvailable");
                btnChkB1.setBackgroundColor(Color.GREEN);

                btnChkB2.setText("Slot 2 [11 AM TO 1 PM]:\nAvailable");
                btnChkB2.setBackgroundColor(Color.GREEN);

                btnChkB3.setText("Slot 3 [3 PM TO 5 PM]:\nAvailable");
                btnChkB3.setBackgroundColor(Color.GREEN);
```

```
            btnChkB4.setText("Slot 4 [6 PM TO 8 PM ]:\nAvailable");
            btnChkB4.setBackgroundColor(Color.GREEN);

            Toast.makeText(this, "All slots available",Toast.LENGTH_SHORT).show();
        }
        else {
            while (cursor.moveToNext()) {
                //Button 1
                if (cursor.getInt(4)== 0){
                    btnChkB1.setText("Slot 1 [8 AM TO 10 AM]:\nAvailable");
                    btnChkB1.setBackgroundColor(Color.GREEN);
                }else if (cursor.getInt(4)== 1){
                    btnChkB1.setText("Slot 1 [8 AM TO 10 AM]:\nNot Available");
                    btnChkB1.setBackgroundColor(Color.RED);
                }
                //Button 2
                if (cursor.getInt(5)== 0){
                    btnChkB2.setText("Slot 2 [11 AM TO 1 PM]:\nAvailable");
                    btnChkB2.setBackgroundColor(Color.GREEN);
                }else if (cursor.getInt(5)== 1){
                    btnChkB2.setText("Slot 2 [11 AM TO 1 PM]:\nNot Available");
                    btnChkB2.setBackgroundColor(Color.RED);
                }
                //Button 3
                if (cursor.getInt(6)== 0){
                    btnChkB3.setText("Slot 3 [3 PM TO 5 PM]:\nAvailable");
                    btnChkB3.setBackgroundColor(Color.GREEN);
                }else if (cursor.getInt(6)== 1){
                    btnChkB3.setText("Slot 3 [3 PM TO 5 PM]:\nNot Available");
                    btnChkB3.setBackgroundColor(Color.RED);
                }
                //Button 4
                if (cursor.getInt(7)== 0){
                    btnChkB4.setText("Slot 4 [6 PM TO 8 PM ]:\nAvailable");
                    btnChkB4.setBackgroundColor(Color.GREEN);
                }else if (cursor.getInt(7)== 1){
                    btnChkB4.setText("Slot 4 [6 PM TO 8 PM ]:\nNot Available");

                    btnChkB4.setBackgroundColor(Color.RED);
                }
            }
        }
    }
    else if(view.getId()==R.id.btnGotoConfirmBooking){
        Intent intent = new Intent(this, AuditoriumCustomerConfirmBooking.class);
        startActivity(intent);
```

```
        }
    }

    public void shoDateTimePicker(){
        DatePickerDialog datePickerDialog = new DatePickerDialog(
          this,
          this,
          Calendar.getInstance().get(Calendar.YEAR),
          Calendar.getInstance().get(Calendar.MONTH),
          Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
        );
        datePickerDialog.show();
    }
    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        String date = i2+ "/" + (i1 + 1)+ "/" + i;
        etChkbookDate.setText(date);

    }
}
```

16..ConfirmBooking.java

```
import java.util.Calendar;

public class AuditoriumCustomerConfirmBooking extends AppCompatActivity
implements DatePickerDialog.OnDateSetListener {
    EditText etConfirmGetDate,etGetCBEmail,etGetCBAudiName;
    CheckBox Slot_ONE,Slot_TWO,Slot_THREE,Slot_FOUR;
    AuditoriumBookingDAO bookingDAO;
    DatabaseController controller;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_customer_confirm_booking);
        controller = new DatabaseController(this);
        etGetCBEmail = findViewById(R.id.etCBEmail);
        etGetCBAudiName=findViewById(R.id.etCBAudiName);
        etConfirmGetDate = findViewById(R.id.etCBGetDate);
        Slot_ONE = findViewById(R.id.chkCBSlot1);
        Slot_TWO = findViewById(R.id.chkCBSlot2);
        Slot_THREE = findViewById(R.id.chkCBSlot3);
        Slot_FOUR = findViewById(R.id.chkCBSlot4);
    }

    public void confirmBooking(View view){
        if(view.getId()==R.id.btnConfirmDate){
            showDatePickerDialog();
        }
        else if(view.getId()==R.id.btnFinalBooking){
            Toast.makeText(this,"Check 1",Toast.LENGTH_SHORT).show();
            String getEmail,getAudiName,getDATE;
            int slot1,slot2,slot3,slot4,AudiID,CustID;

            getEmail = etGetCBEmail.getText().toString();
            getAudiName = etGetCBAudiName.getText().toString();
            getDATE = etConfirmGetDate.getText().toString();
            if(Slot_ONE.isChecked()){slot1=1;}else{slot1=0;}
            if(Slot_TWO.isChecked()){slot2=1;}else{slot2=0;}
            if(Slot_THREE.isChecked()){slot3=1;}else{slot3=0;}
            if(Slot_FOUR.isChecked()){slot4=1;}else{slot4=0;}
            CustID = getCustID(getEmail);
            AudiID = getAudiID(getAudiName);
            bookingDAO = new
```

```
AuditoriumBookingDAO(getEmail,getAudiName,getDATE,slot1,slot2,slot3,slot4,Cu
stID,AudiID);
            SQLiteDatabase db = controller.getWritableDatabase();
            ContentValues val =new ContentValues();
            val.put("CustEMAIL",bookingDAO.getCustEMAIL());
            val.put("AUDI_NAME",bookingDAO.getAudiNAME());
            val.put("DATE",bookingDAO.getDATE());
            val.put("Slot_ONE",bookingDAO.getSlot_ONE());
            val.put("Slot_TWO",bookingDAO.getSlot_TWO());
            val.put("Slot_THREE",bookingDAO.getSlot_THREE());
            val.put("Slot_FOUR",bookingDAO.getSlot_FOUR());
            val.put("Cust_ID",bookingDAO.getCust_ID());
            val.put("AUDI_ID",bookingDAO.getAUDI_ID());
            long result = db.insert("AudiBooking", "", val);
            if (result > 0) {
                Toast.makeText(this,"Check 2",Toast.LENGTH_SHORT).show();
                Toast.makeText(this,"Booking
Confirmed",Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(this, AuditoriumTemporary.class);
                startActivity(intent);
            } else {
                Toast.makeText(this, "Error",Toast.LENGTH_SHORT).show();
            }
        }
    }

    public void showDatePickerDialog(){
        DatePickerDialog datePickerDialog = new DatePickerDialog(
          this,
          this,
          Calendar.getInstance().get(Calendar.YEAR),
                Calendar.getInstance().get(Calendar.MONTH),
                Calendar.getInstance().get(Calendar.DAY_OF_MONTH)
        );
        datePickerDialog.show();
    }
    public void onDateSet(DatePicker datePickerDialog, int i, int i1, int i2) {
        String date = i2 +"/"+ (i1 + 1) +"/"+i;
        etConfirmGetDate.setText(date);
    }
```

```
    public int getCustID(String CustEmail){
        int CustID=0;
        String thisCustEMAIL = CustEmail;
        //controller = new DatabaseController(this);
        Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT Cust_ID
FROM AudiCust WHERE CustEMAIL='"+thisCustEMAIL+"'",null);
        if (cursor!=null){
            cursor.moveToFirst();
            CustID = cursor.getInt(0);
        }else{
            Toast.makeText(this, "Incorrect Email",Toast.LENGTH_SHORT).show();
        }
        return CustID;
    }
    public int getAudiID(String AudiName){
        int AudiID=0;
        String thisAudiName = AudiName;
        Cursor cursor = controller.getReadableDatabase().rawQuery("SELECT
AUDI_ID FROM AudiInfo WHERE AUDI_NAME='"+thisAudiName+"'",null);
        if (cursor!=null){
            cursor.moveToFirst();
            AudiID = cursor.getInt(0);
        }
        else{AudiID=0;
        }
        return AudiID;
    }
}
```

1. DatabaseController Code

```
package com.example.audidemo;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;


public class DatabaseController extends SQLiteOpenHelper {
    public DatabaseController(Context context) {
        super(context, "AudiDemo.db", null,1);
    }

    public void onOpen(SQLiteDatabase db){
        super.onOpen(db);
        if (!db.isReadOnly()){
            db.execSQL("PRAGMA foreign_keys = True");
            db.execSQL("PRAGMA case_sensitive_like = True;");
        }
    }
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL("CREATE TABLE AudiOwner(OWNER_ID
INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,NAME TEXT,
CONTACT TEXT, ADDRESS TEXT, EMAIL TEXT, PASSWORD TEXT )");
        sqLiteDatabase.execSQL("CREATE TABLE AudiInfo(AUDI_ID INTEGER
PRIMARY KEY AUTOINCREMENT,AUDI_NAME TEXT,AUDI_ADDRESS
TEXT,AUDI_CITY TEXT,AUDI_CAPACITY TEXT,OWNER_ID
INTEGER,FOREIGN KEY (OWNER_ID) REFERENCES
AudiOwner(OWNER_ID))");
        sqLiteDatabase.execSQL("CREATE TABLE PackageInfo(PACKAGE_ID
INTEGER PRIMARY KEY AUTOINCREMENT, PHOTOGRAPHY
TEXT,DECORATION TEXT,SOUND TEXT, LIGHT TEXT,AUDI_ID
INTEGER,FOREIGN KEY (AUDI_ID) REFERENCES AudiInfo(AUDI_ID))");
        sqLiteDatabase.execSQL("CREATE TABLE AudiCust(Cust_ID integer NOT
NULL PRIMARY KEY AUTOINCREMENT,CustNAME TEXT, CustCONTACT
TEXT, CustADDRESS TEXT, CustEMAIL TEXT, CustPASSWORD TEXT )");
        sqLiteDatabase.execSQL("CREATE TABLE AudiBooking(Booking_ID
INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, CustEMAIL TEXT,
```

```
AUDI_NAME TEXT,DATE TEXT,Slot_ONE INTEGER, Slot_TWO
INTEGER,Slot_THREE INTEGER, Slot_FOUR INTEGER, Cust_ID INTEGER,
AUDI_ID INTEGER, FOREIGN KEY (Cust_ID) REFERENCES
AudiCust(Cust_ID),FOREIGN KEY (AUDI_ID) REFERENCES
AudiInfo(AUDI_ID))");
    }

    public void onConfigure(SQLiteDatabase db) {
        db.setForeignKeyConstraintsEnabled(true);
        super.onConfigure(db);
    }


    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS AudiOwner");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS AudiInfo");
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS AudiCust");
        onCreate(sqLiteDatabase);
    }

    public int getMaxOwnerID(){
        int maxID= 0;
        Log.i("LOG ID","In GetMaxID");
        Cursor cursor = getReadableDatabase().rawQuery("SELECT
MAX(OWNER_ID) FROM AudiOwner",null);

        if (cursor!=null) {
            cursor.moveToFirst();
            maxID = cursor.getInt(0);
        }
        return maxID;
    }
    public int getMaxAuditoriumID(){
        int maxID= 0;
        Cursor cursor = getReadableDatabase().rawQuery("SELECT MAX(AUDI_ID)
FROM AudiInfo",null);
        if (cursor!=null) {
            cursor.moveToFirst();
            maxID = cursor.getInt(0);
```

```
        }
        return maxID;
    }
}
```

1.ActivityMain.java

```
package com.example.audidemo;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

 public class MainActivity extends AppCompatActivity {
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

    public void btnClick(View view)
    {  if(view.getId()==R.id.btnGoToOwner) {
          Intent intent = new Intent(this,ActivityMainOwner.class);
          startActivity(intent);
       }
       else if(view.getId()==R.id.btnGoToCutomer){
          Intent intent = new Intent(this,ActivityMainCutomer.class);
          startActivity(intent);
       }    }   }
```

OwnerLogin.java

```
package com.example.audidemo;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.widget.EditText;
import android.widget.Toast;

public class AudiOwnerLogin extends AppCompatActivity {
    EditText etOwEmailLog,etOwPassLog;
    Button btnOwLog;
    DatabaseController controller;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audi_owner_login);
        controller=new DatabaseController(this);
        etOwEmailLog = findViewById(R.id.etOwnerEmailLog);
        etOwPassLog = findViewById(R.id.etOwnerPassLog);
        btnOwLog = findViewById(R.id.btnOwnerLogin);
    }
    public void click(View view)
    {
        if(view.getId()==R.id.btnOwnerLogin)
        { String OwEmail = etOwEmailLog.getText().toString();
            String OwPass = etOwPassLog.getText().toString();
            SQLiteDatabase db= controller.getReadableDatabase();
            Cursor cursor = db.rawQuery("select * from AudiOwner where EMAIL=? and
PASSWORD=?",new String[]{OwEmail,OwPass});
            if(cursor.getCount()>0){
                Toast.makeText(getApplicationContext(),"Log In Sucessful
",Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(this, AudiPackagePanel.class);
                startActivity(intent);
            }
            else {
                Toast.makeText(getApplicationContext(),"Incorrect Email or Password
",Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```