BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)

Bull Temple Road, Basavanagudi, Bangalore – 560019

A project report on

"Event Hall Booking System"

Submitted in partial fulfilment of the requirements for the Self Study of

Mobile Application Development

(16IS7DCMAD)

By

H Anarghya - 1BM17IS028

Pannaga Sharma M L - 1BM17IS053

Submitted to

Dr. Sheela S V

Associate Professor

Department of Information Science and Engineering

2020-21

ABSTRACT

In recent years, there has been an increasing demand for online applications to ease the completion of tedious tasks. Mobile applications have been in demand more than other types of applications due to its reachability. Hence, an application is created that reduces the manual work of managing different reservations of various halls for different events. The main objective is to provide a user-friendly environment to both, the customer, who is interested to reserve a hall and the hall admin, who wants to effectively manage the bookings on various days.

In this application, the customer can view different categories of halls and book any hall in which they are interested in by sending an enquiry for the same. He also has the option to pay the advance amount online after his booking has been confirmed. The admin can view a list of all the enquiries sent by the users for the halls owned by them which then can be confirmed or rejected after checking all the details. Admin can edit the hall details that are displayed to the customers and can also view all the details of the confirmed bookings for their halls.

TABLE OF CONTENTS

1. INTRODUCTION

Event hall booking system is an application developed for Android whose aim is to override the problems present in the current manual system. It reduces and, in some cases, eliminates the hardships faced in the prevailing system. It enables the users of the application, both the customer and the admin, to carry out their operations in a smooth and effective manner.

The two main users of this application are – customer (also called the user) and admin. Customer is a person who is interested in viewing the different halls and wants to book a hall for an event for a specific date and time. When the person logs in as a user, he/she can view the different categories of halls available and under each category, the images and the corresponding names of the different halls available. Customers can also filter the list of halls displayed to them by filling a form which accepts their requirements such as the hall type, preferred location, required seating and their budget. The user can get further details of that hall by clicking on its image. If they are interested to book that hall, then they can fill out all the details in a form and send an enquiry for the same.

Admin is the person who is in charge of one or more halls and is interested in using the application to manage the bookings for those halls. It is assumed that the admin registers themselves with a person who is in charge of the application to obtain their credentials and to add the details regarding their hall(s) to the database. The admin has to login with the credentials (email and password) in order to use the application. After logging in, the admin can select any of the following three options – view enquiries, edit details and view bookings.

In the view enquiries option, the admin can see a list of all the enquiries that have been sent by different users. An appropriate mail is sent to that user when the admin confirms or rejects that enquiry. In the edit hall details option, the admin can edit any of the details for any hall that are stored in the database and which is displayed to the customers. On clicking the view bookings options, the admin can view a list of all the bookings that have been confirmed for those halls for which they are in charge.

The application includes a ‘Payment’ option, which enables the users to pay the advance amount after their booking has been confirmed. Razorpay payment gateway has been used. Razorpay aims to revolutionize money management for online businesses by providing clean, developer-friendly APIs and hassle-free integration. It provides different payment methods such as net banking, credit/debit card, UPI, etc.

The Firebase real-time database has been used to avail the database facilities for the application. It stores data in the form of JSON objects and provides real-time synchronisation with all the connected clients. It accelerates app development by providing fully managed backend infrastructure. Finally, Android Studio is the IDE (Integrated Development Environment) used to develop the application.

2. SCOPE AND OBJECTIVE

The main objectives of the application are:

- To reduce the manual task of the user
- To look into various types of venues in a short period
- To provide various categories of venues based on the occasion
- To provide a secure payment portal to ease the process of payment
- To provide the functionalities of booking and cancellation to the admin.
- To provide an easy form of communication via electronic mail.
- To provide the facilities for maintaining the records of all the bookings
- To reduce and to completely remove the broker cost.

The scope of the application is based on the functionalities provided by it. It includes:

1. Easy to use interface that allows the users to view the details of the required hall easily.
2. Reduce the manual labour involved in going around different halls in selecting a venue.
3. Reduce the manual work of the admins by listing all enquires and bookings in the same place.
4. Easy to use interface that enables any type of person residing in any place to use the application.

3. FEATURES

The features of the application are:

- Easy to use interface for both the customer interested in booking a hall and the admin who wants to view and maintain the records.
- Easy retrieval of user-specific information based on search filters.
- Reduces the manual work which was involved in the previous system.
- Integrated payment gateway that enables faster and secure payments.
- Dynamic segments that always display up-to-date information.
- Robust database backend.
- Easy and fast access to information from the online database at any place.
- Provide an option to reset the password if the admin has forgotten it using their registered email ID.
- Reduced loss of data.

4. REQUIREMENT ANALYSIS

4.1. Functional Requirements:

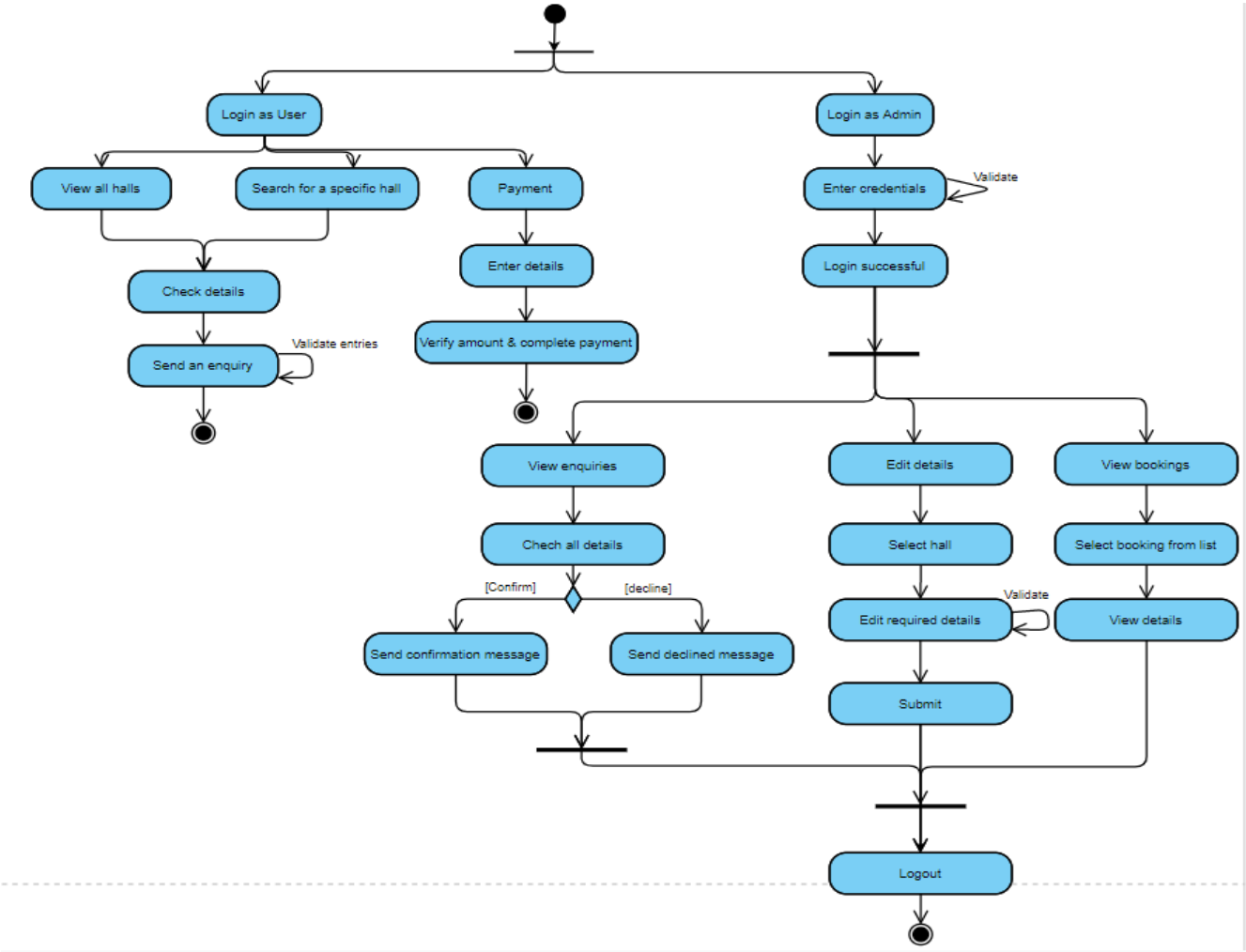
- The user is able to view different categories of halls and for each hall, the user can view all the details such as name, address, contact details of admin, seating capacity, and the rent per day.
- User is able to search for the required hall based on the location, budget and the capacity of the hall.
- User is able to select any hall and can send an enquiry for the same by filling their contact details and the date and time according to their requirements.
- User can pay the advance amount after their booking has been confirmed.
- Admin of a hall can view all the enquiries for their hall(s) and accept/reject the booking.
- Admin can edit the details of the halls and save the same into the database.
- Admin can view the details of all the bookings that have been confirmed for their hall(s).

4.2. Non-functional requirements:

- **Connectivity:** The application must be connected to the internet at all times.
- **Usability:** The application must be easy to use and easy to operate.
- **Performance:** The application is responsive to various user interactions and provides a good user experience.
- **Compatibility:** The application must be compatible with shared applications and with different versions of Android.
- **Reliability:** The application must work at all times without any failure of the software.

5. SYSTEM DESIGN

5.1. Activity Diagram:



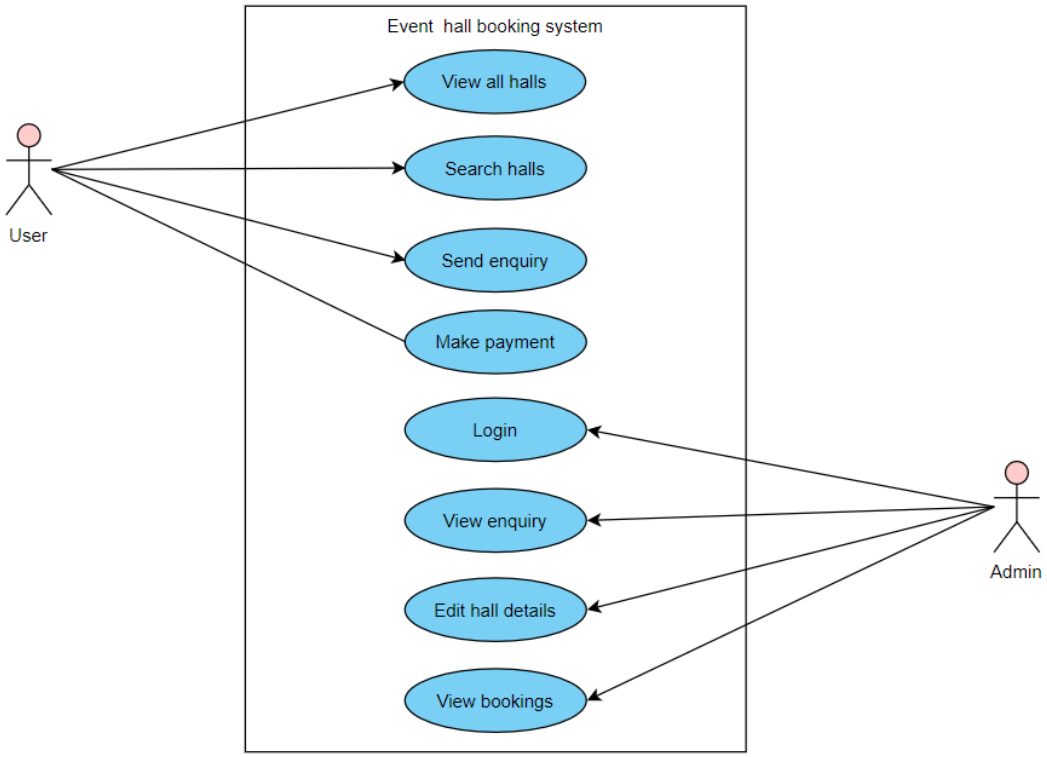
The activity diagram gives the flow of control in the application. On starting, it provides the option of either logging in as a user or as an admin. The admin has to enter a valid email and password to log in. After successful login, the admin can select one of the three options – View enquiries, edit hall details and view bookings. View enquiry allows the admin to view all the list of enquiries sent by the different users interested to book their hall(s). The admin will then have the options of confirming or rejecting that enquiry after looking at all the details. Edit hall details option provides a facility to edit the hall details that will be displayed to all the users using the application. View booking option allows the admin to view the list of all the bookings that have been confirmed.

On logging in as a user, he/she can view the different categories of the halls and the images and its corresponding names of the different halls present in each category. The user can also search for a specific hall by entering the constraints in the filter form, such as the type of hall, preferred location, seating capacity and budget. After selecting the required hall, it shows all the relevant details such as the address, contact details,

seating capacity and the rent. After taking a look at those details, the user can send an enquiry to book that hall for a specific date and time.

If the enquiry has been confirmed by the admin and the hall has been booked, then the user has the payment option to pay the advance amount as part of the booking online. The user has to confirm the hall and the email used to book that hall in order to complete the payment.

5.2. Use case diagram:



The two main actors in the use case diagram are:

User: The person who is interested to view and book a hall for a date and time.

Admin: The person who is responsible to manage enquiries sent by the user for their hall(s).

The use cases of the user are:

View halls: The user can view all the categories of halls available and view various halls available in each category.

Search halls: The user can search specific hall(s) base on some constraints such as type, preferred location, seating capacity and budget.

Send enquiry: The user can send an enquiry for a particular date and time for a hall if he is interested to book it.

Make Payment: The user can pay the advance amount after the booking has been confirmed through the Razorpay gateway.

The use cases of the admin are:

Login: The admin can use the application only after logging in with valid credentials.

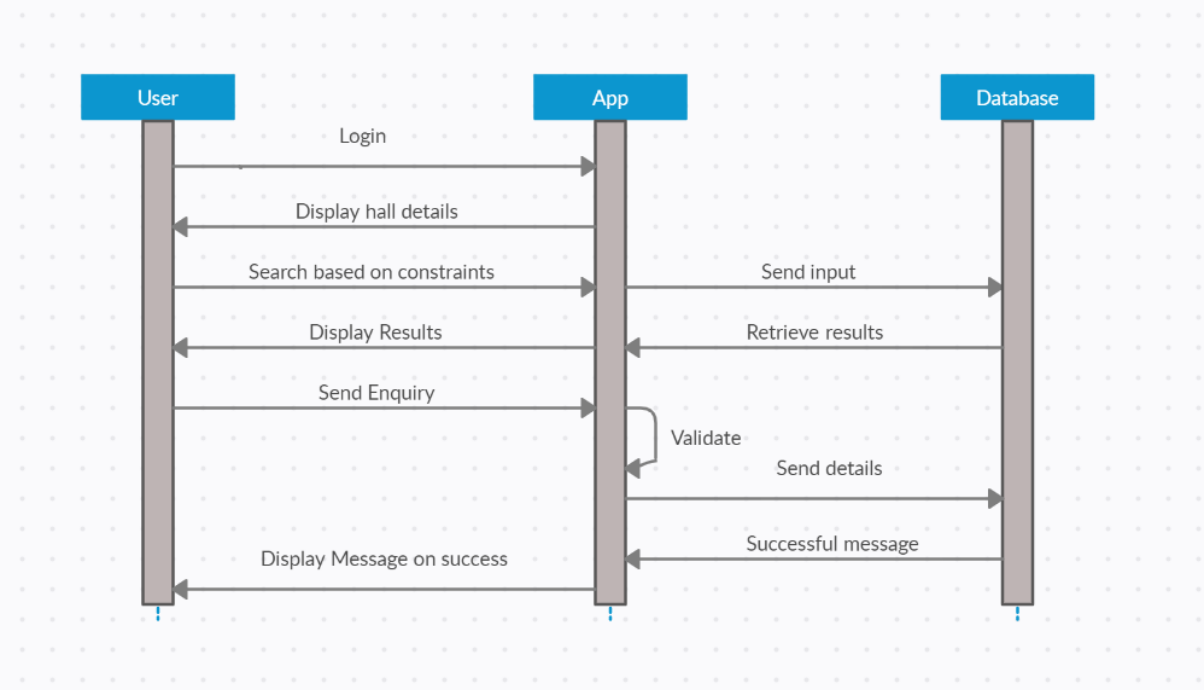
View Enquiries: Admin can view the list of enquiries along with the details, sent by different users to book their hall(s).

Edit hall details: Admin can modify hall details such as the name, address, rent, seating capacity and contact details for the hall(s) owned by them.

View Bookings: The admin can view the list of all the bookings for different events that have been confirmed by them.

5.3. Sequence diagrams:

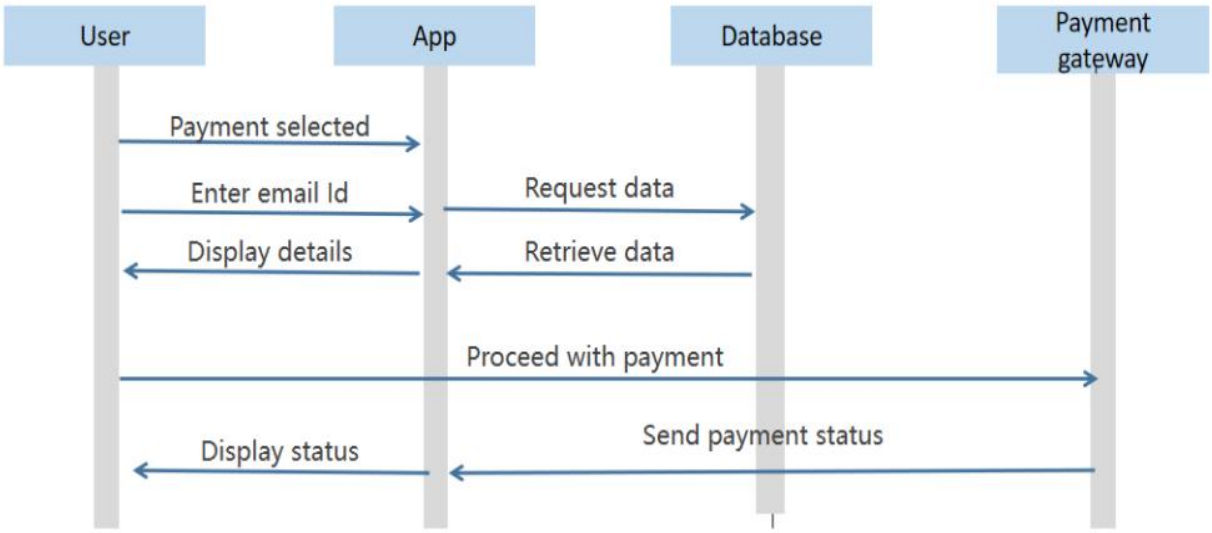
Sequence diagram for the user:



On opening the app and logging in as a user, the app will display the different categories of the halls and in each category, it will display all the different halls present. The user can also filter the results based on the constraints such as the hall type, preferred location, budget and seating capacity. After that, the user can select whichever hall he wants and send an enquiry for the same. After validating the input,

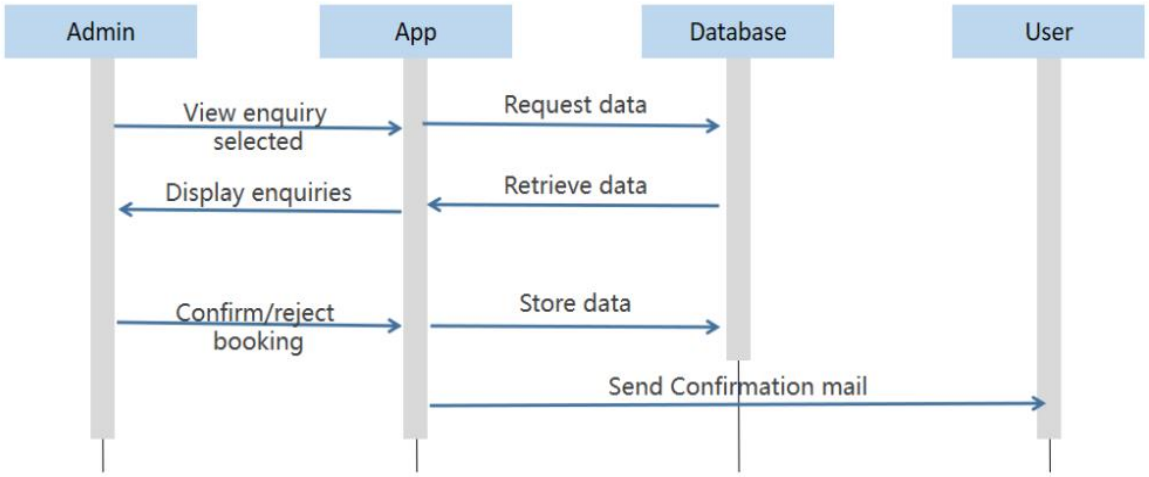
the details will be stored in the database and on success, a toast message will be displayed.

Sequence diagram for user payment:



The user selects the hall and enters his email using which the booking was done. If the record exists and the booking has been confirmed, then we retrieve those details from the database along with the advance amount to be paid. The user is then redirected to the payment gateway and will be able to complete the payment. The payment status is returned back to the app and the same is displayed to the user.

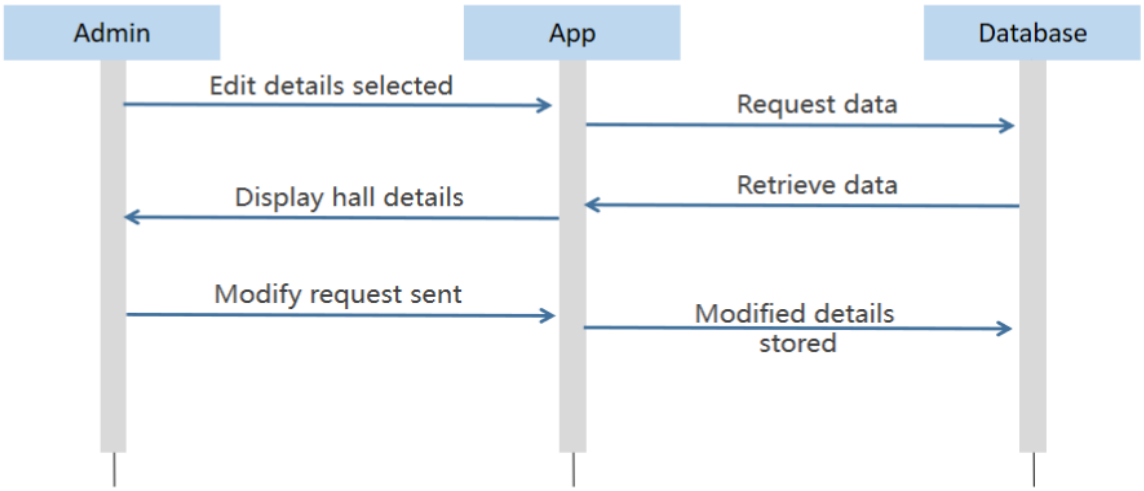
Sequence diagram for Viewing Enquiries:



When the admin selects view enquiry option, the application requests data from the database. Then the enquiries sent to the halls owned by that admin is retrieved and is displayed on the app interface. After viewing the enquiries, the admin can confirm or

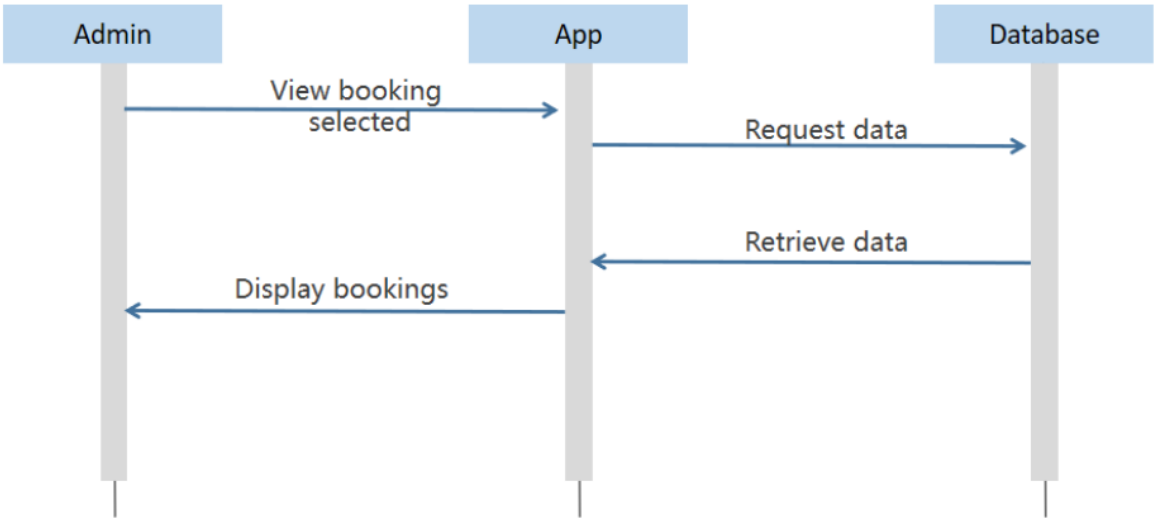
reject the booking. Once this is done the data is updated and stored in the database and a mail is sent to the user.

Sequence diagram for editing details:



When the admin selects the edit details option in the application, it displays the list of halls that are owned by that admin. On selecting any hall from the list, all the details of that hall are retrieved from the database and the same can be edited in a form. When admin modifies the details and confirms it, the modified details are stored in the database.

Sequence diagram for viewing details:

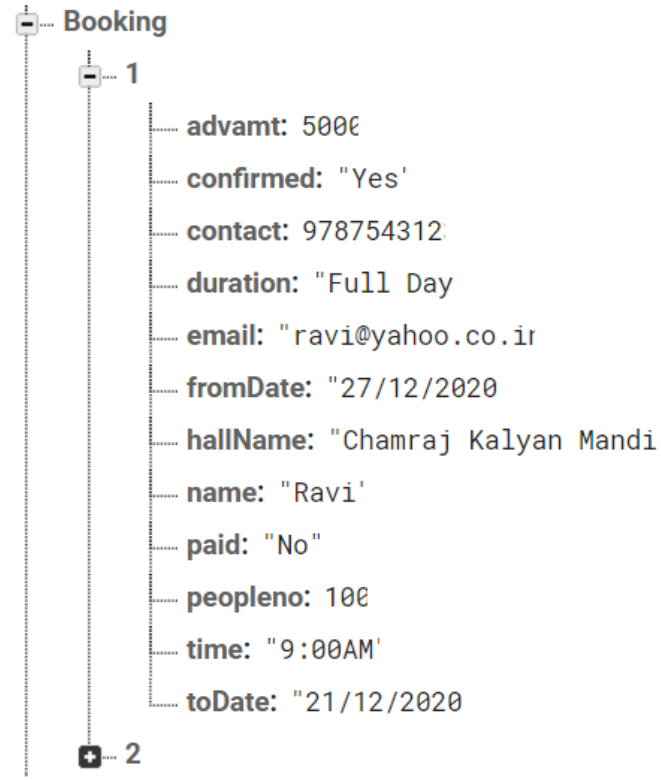


When the admin selects the view booking option in the application, it requests data from the database. Then the requested data is retrieved from the database. Then the application displays the list of all the bookings to the admin, which then can be viewed in detail on selecting any item in the list.

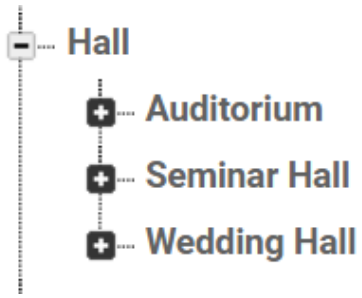
6. DATABASE

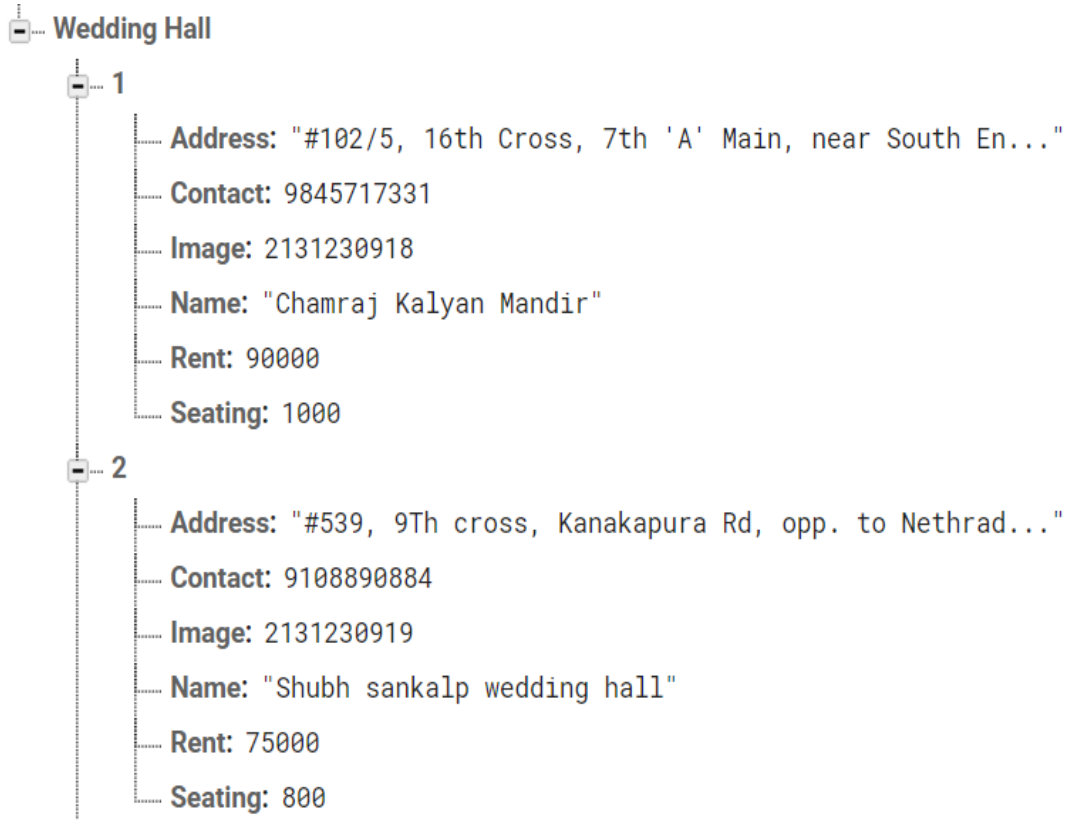
The Firebase Realtime Database is a cloud-hosted database. Data is stored as a JSON tree structure and synchronized in real-time to every connected client. The three main nodes in the database are Booking, Hall and Owner.

The Booking node stores each booking as a child node. Each child node consists of children in the form of key-value pairs that store all the relevant details such as user details, date and time of booking, advance amount to be paid, etc.

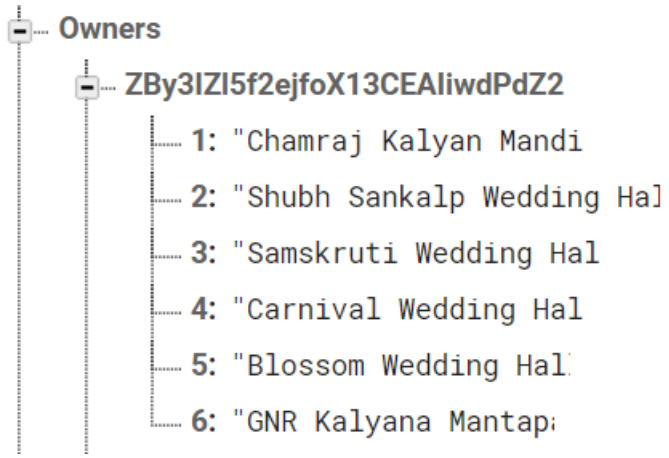


The Hall node stores all the details of the halls that are available. It has 3 main child nodes – Auditorium, Seminar Hall and Wedding Hall. Under each of these child nodes, the corresponding hall details are stored as key-value pairs.





The Owner node stores the details as to which admin manages which set of halls. The admin is identified by a unique User ID that would be created when the admin credentials were created.



7. IMPLEMENTATION

User Interface:

```
SectionsPagerAdapter sectionsPagerAdapter = new SectionsPagerAdapter(getApplicationContext(), getSupportFragmentManager());
ViewPager viewPager = findViewById(R.id.view_pager);
viewPager.setAdapter(sectionsPagerAdapter);
TabLayout tabs = findViewById(R.id.tabs);
tabs.setupWithViewPager(viewPager);
```

```
@Override
public Fragment getItem(int position) {
    Fragment fragment = null;
    int[] images;
    String[] names;
    switch (position) {
        case 0:

            fragment = new HallViewFragment(MainActivity.weddingHallDetails);
            break;
        case 1:
            fragment = new HallViewFragment(MainActivity.seminarHallDetails);
            break;
        case 2:
            fragment = new HallViewFragment(MainActivity.auditoriumDetails);
            break;
    }
    return fragment;
}
```

```
DatabaseReference dbref1 = dr.child("Wedding Hall");
dbref1.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        int i=0;
        for(DataSnapshot d: snapshot.getChildren()){
            weddingHallDetails[i] = d.getValue(Hall.class);
            i++;
        }

        DatabaseReference dbref2 = dr.child("Seminar Hall");
        dbref2.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                int i=0;
                for(DataSnapshot d: snapshot.getChildren()) {
                    seminarHallDetails[i] = d.getValue(Hall.class);
                    i++;
                }
```

For the main Activity of the user which displays all the halls with the category, Tab Layout has been used which loads different fragments for each tab. It is implemented using **ViewPager** and **SectionsPagerAdapter**.

Section Pager Adapter has a method that determines the fragment to be displayed for each tab. ViewPager is a layout manager that allows the user to scroll left and right to navigate through multiple fragments. So, the Tab layout is set up with ViewPager.

Data is retrieved from the database using the **addValueEventListener** function which reads and listens for changes to the entire contents of a path.

Retrieving search results:

```
t = type.getSelectedItem().toString();
location = autoCompleteTextView.getText().toString();
if(seating.getText().toString().isEmpty())
    seats = 0;
else
    seats = Integer.parseInt(seating.getText().toString());
if(start.getText().toString().isEmpty())
    begin = 0.0;
else
    begin = Double.parseDouble(start.getText().toString());
if(end.getText().toString().isEmpty())
    e = Double.POSITIVE_INFINITY;
else
    e = Double.parseDouble(start.getText().toString());
```

```
dbRef = FirebaseDatabase.getInstance().getReference( path: "Hall/" + t);
dbRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        results.clear();
        for(DataSnapshot d : snapshot.getChildren()){
            if(d.getValue(Hall.class).Address.contains(location) || d.getValue(Hall.class).Seating >= seats || (d.getValue(Hall.class).Rent >= begin
                    && d.getValue(Hall.class).Rent <= e))
                results.add(d.getValue(Hall.class));
        }
        startActivity(new Intent( packageContext: FilterFormActivity.this, SearchResultsActivity.class).putExtra( name: "results", results));

    }
```

When the user searches for a specific hall, the results are retrieved by comparing the values of the hall details stored in the database with the user input. The input values will be set to default if the user leaves any field empty.

Validating form entries:

The values entered by the user while sending an enquiry are validated before storing in the database. The name and email are validated using regular expressions and the dates of the booking are validated so that they are not less than today's date. After validation, it is stored in the database using the **setValue** function.

```
String regex1 = "^[A-Z][a-zA-Z\\s]+";
if(!n.matches(regex1)){
    Toast.makeText(getApplicationContext(), text: "Please enter a valid name",Toast.LENGTH_LONG).show();
    name.setTextColor(Color.RED);
    return false;
}
else
    name.setTextColor(Color.BLACK);

if(c.length() != 10) {
    Toast.makeText(getApplicationContext(), text: "Please check your contact number", Toast.LENGTH_LONG).show();
    contact.setTextColor(Color.RED);
    return false;
}
else
    contact.setTextColor(Color.BLACK);

String regex2 = "^[\\w-.+]*[\\w-.]@([\\w]+\\.)+[\\w]+[\\w]$";
if(!em.matches(regex2)){
    Toast.makeText(getApplicationContext(), text: "Please enter a valid email address",Toast.LENGTH_LONG).show();
    email.setTextColor(Color.RED);
    return false;
}
else
    email.setTextColor(Color.BLACK);
```

```
if (d1.after(d2) || d1.equals(d2)) {
    Toast.makeText(getApplicationContext(), text: "Please enter a valid From date",Toast.LENGTH_LONG).show();
    fromDate.setTextColor(Color.RED);
    return false;
}
else
    fromDate.setTextColor(Color.BLACK);

if(d1.after(d3) || d1.equals(d3)){
    Toast.makeText(getApplicationContext(), text: "Please enter a valid To date",Toast.LENGTH_LONG).show();
    toDate.setTextColor(Color.RED);
    return false;
}
else
    toDate.setTextColor(Color.BLACK);

if(d2.after(d3)){
    Toast.makeText(getApplicationContext(), text: "Please enter a valid From and To date",Toast.LENGTH_LONG).show();
    fromDate.setTextColor(Color.RED);
    toDate.setTextColor(Color.RED);
    return false;
}
else {
    fromDate.setTextColor(Color.BLACK);
    toDate.setTextColor(Color.BLACK);
}
```

```
Boolean valid  = validate();
if(valid) {
    Booking b = new Booking(name.getText().toString(), email.getText().toString(), Long.parseLong(contact.getText().toString()),
            selectedHall.getText().toString(), fromDate.getText().toString(), toDate.getText().toString(), chooseTime.getText().toString(),
            rbChoice, Integer.parseInt(noOfPeople.getText().toString()));
    String no = String.valueOf(++count);
    ref.child(no).setValue(b, new DatabaseReference.CompletionListener() {
        @Override
        public void onComplete(@Nullable DatabaseError error, @NonNull DatabaseReference ref) {
            Toast.makeText(getApplicationContext(), text: "Enquiry sent successfully", Toast.LENGTH_LONG).show();
            finish();
        }
    });
}
```

Admin Login:

```
signIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String adminEmail = emailText.getText().toString();
        String pass = password.getText().toString();
        progressDialog.setMessage("Signing In");
        progressDialog.show();
        firebaseAuth.signInWithEmailAndPassword(adminEmail, pass).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){

                    progressDialog.dismiss();
                    finish();
                    startActivity(new Intent( packageContext: AdminLoginActivity.this, AdminNavigation.class).putExtra( name: "adminid", adminEmail));
                }
                else {
                    error.setVisibility(View.VISIBLE);
                    progressDialog.dismiss();
                }
            }
        });
    }
});
```

Admin login is done using the **FirebaseAuthentication** class.

The **signInWithEmailAndPassword** function takes the email and password entered by the admin, and if it's valid, only then admin will be signed it else it will display an error message.

Confirming/Deleting Bookings:

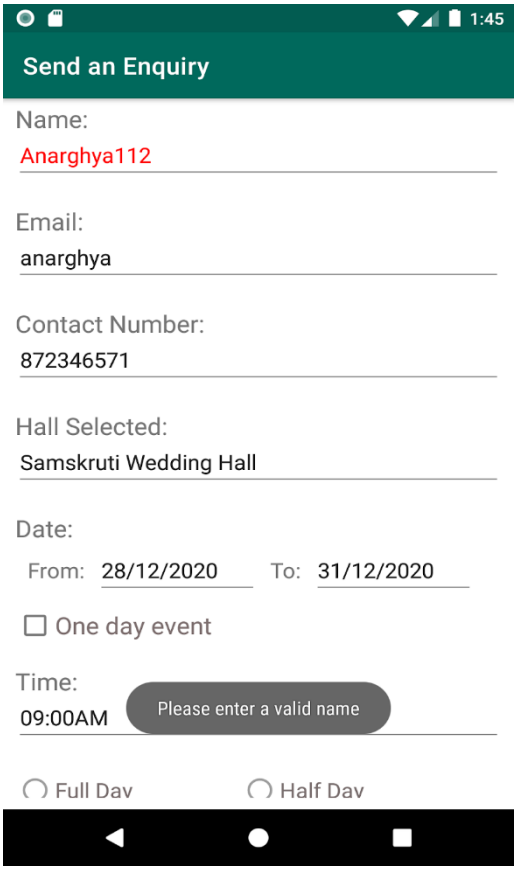
```
b.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        LayoutInflater li = LayoutInflater.from(getApplicationContext());
        View view = li.inflate(R.layout.prompt_user_input,  root: null);
        AlertDialog.Builder  builder = new AlertDialog.Builder( context: ViewEnquiryDetailsActivity.this);
        builder.setTitle("Enter the advance amount");
        builder.setView(view);
        builder.setPositiveButton( text: "OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                EditText amt = view.findViewById(R.id.forgotPasswordMail);
                String adv = amt.getText().toString().trim();
                editBooking.confirmed = "Yes";
                editBooking.advamt = Double.parseDouble(adv);
                dbRef.child(id).setValue(editBooking);
                SendMail s = new SendMail(editBooking);
                s.execute();
                Toast.makeText(getApplicationContext(),  text: "Booking confirmed", Toast.LENGTH_LONG).show();
                finish();
            }
        }).setCancelable(false).setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) { dialog.dismiss(); }
        }).create().show();
    }
});
```

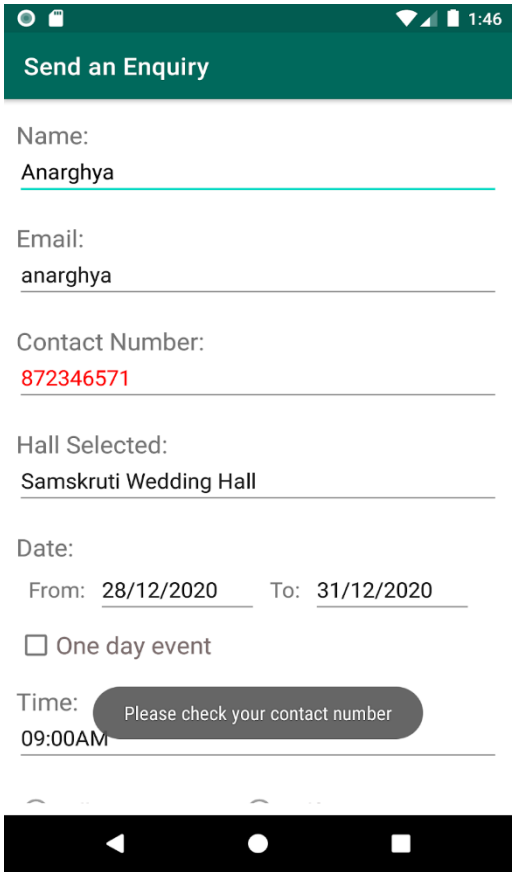

```
Button r = findViewById(R.id.reject);
r.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        editBooking.confirmed = "Rejected";
        dbRef.child(id).setValue(editBooking);
        SendMail s = new SendMail(editBooking);
        s.execute();
        Toast.makeText(getApplicationContext(), text: "Booking cancelled", Toast.LENGTH_LONG).show();
        finish();
    }
});
```

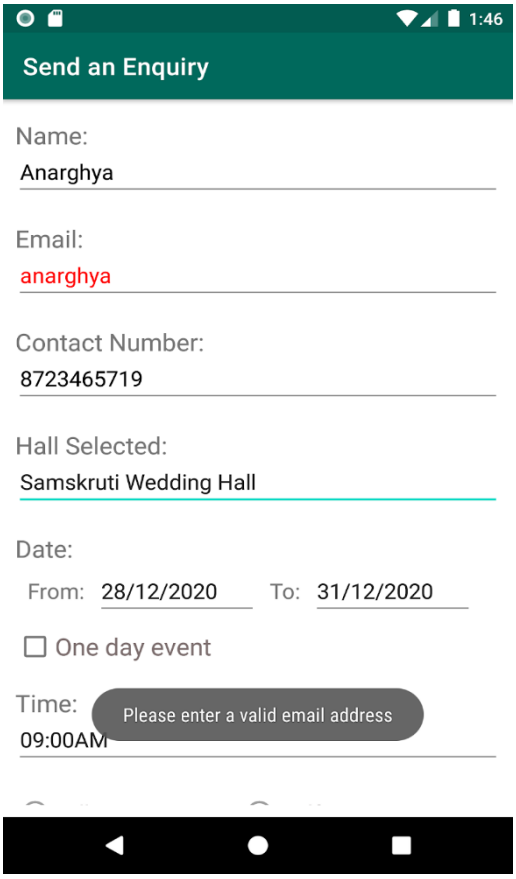
This is the part where the admin confirms or rejects the booking and the corresponding mail is sent for the same.

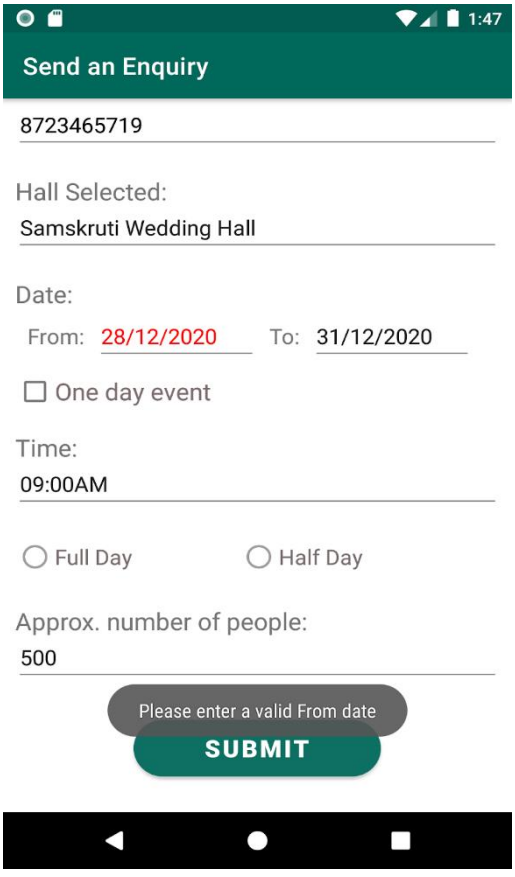
If the admin accepts the enquiry, the confirmed string is set to 'yes' and the advance amount to be paid is taken as input from the admin. These details will be updated in the database and the mail will also be sent to the user with the same details.

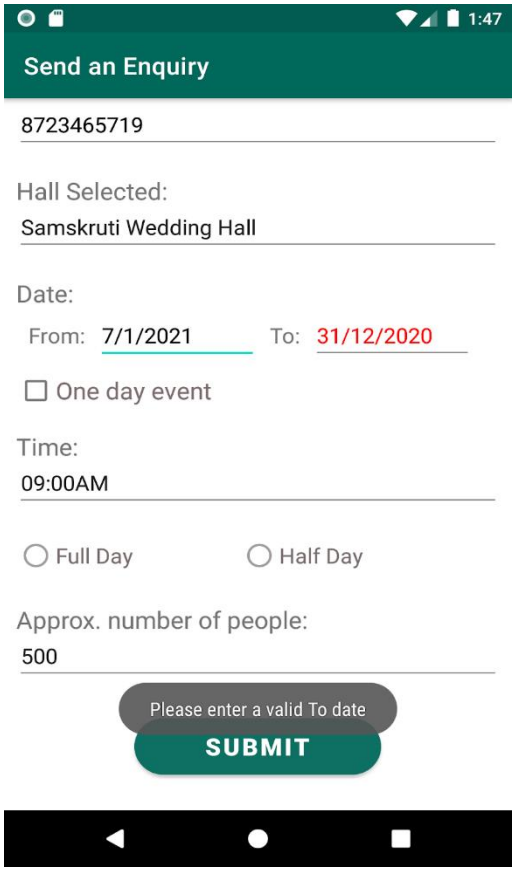
8. TESTING

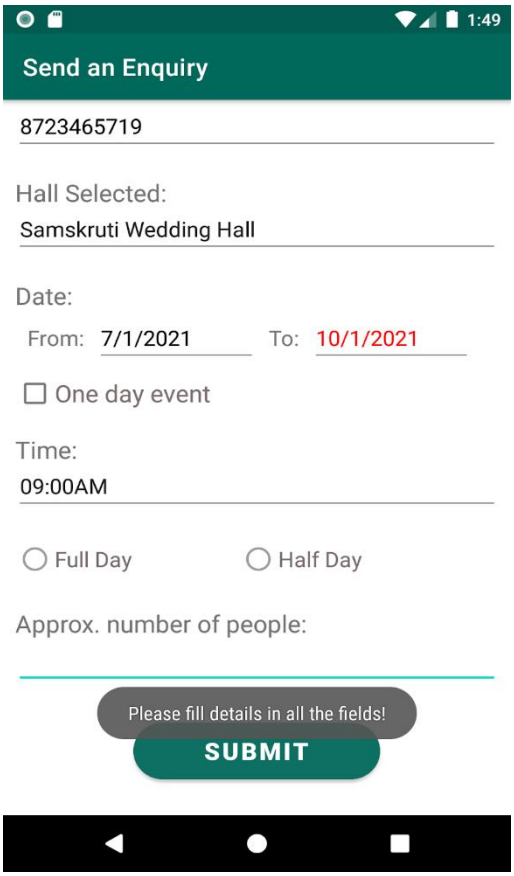

1:45
Send an Enquiry
Name:
Anarghya112
Email:
anarghya
Contact Number:
872346571
Hall Selected:
Samskruti Wedding Hall
Date:
From: 28/12/2020
To: 31/12/2020
One day event
Time:
09:00AM
Please enter a valid name
Full Dav
Half Dav


1:46
Send an Enquiry
Name:
Anarghya
Email:
anarghya
Contact Number:
872346571
Hall Selected:
Samskruti Wedding Hall
Date:
From: 28/12/2020
To: 31/12/2020
One day event
Time:
09:00AM
Please check your contact number

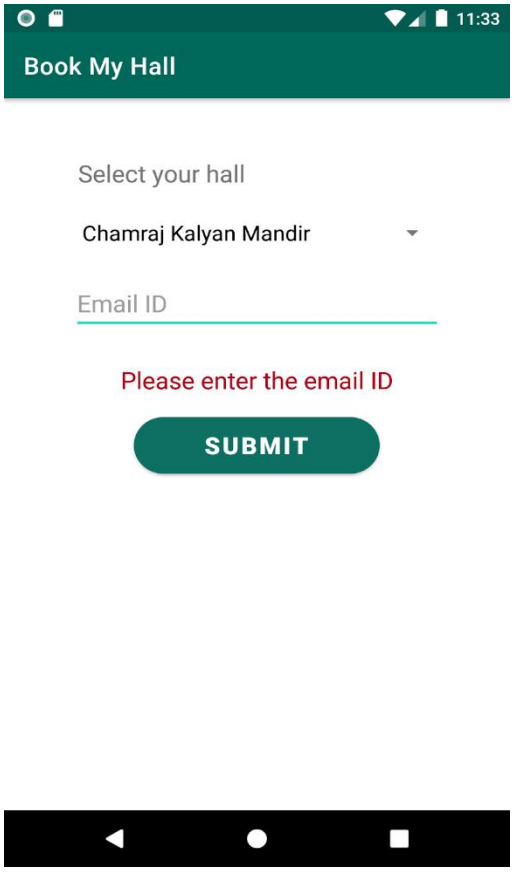

1:46
Send an Enquiry
Name:
Anarghya
Email:
anarghya
Contact Number:
8723465719
Hall Selected:
Samskruti Wedding Hall
Date:
From: 28/12/2020
To: 31/12/2020
One day event
Time:
09:00AM
Please enter a valid email address


1:47
Send an Enquiry
8723465719
Hall Selected:
Samskruti Wedding Hall
Date:
From: 28/12/2020
To: 31/12/2020
One day event
Time:
09:00AM
Full Day
Half Day
Approx. number of people:
500
Please enter a valid From date
SUBMIT

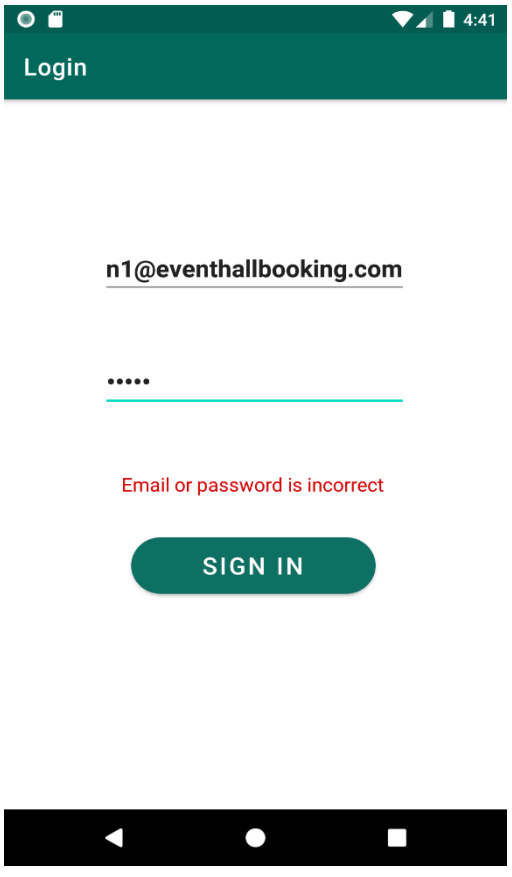




The validations implemented for the enquiry form filled by the user is implemented here. It gives a toast message when any of the fields have an invalid entry. When the From or To date is smaller than the current date, it gives an error. It also displays a toast message if any field is left empty.

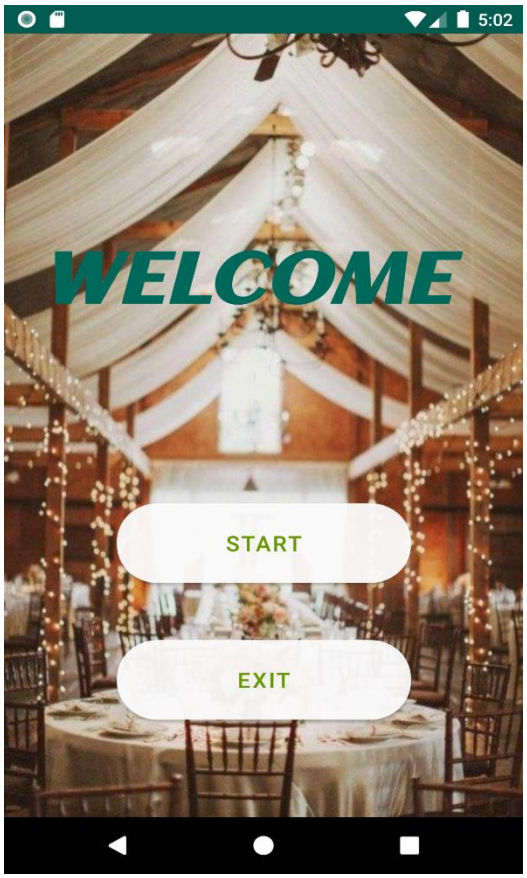


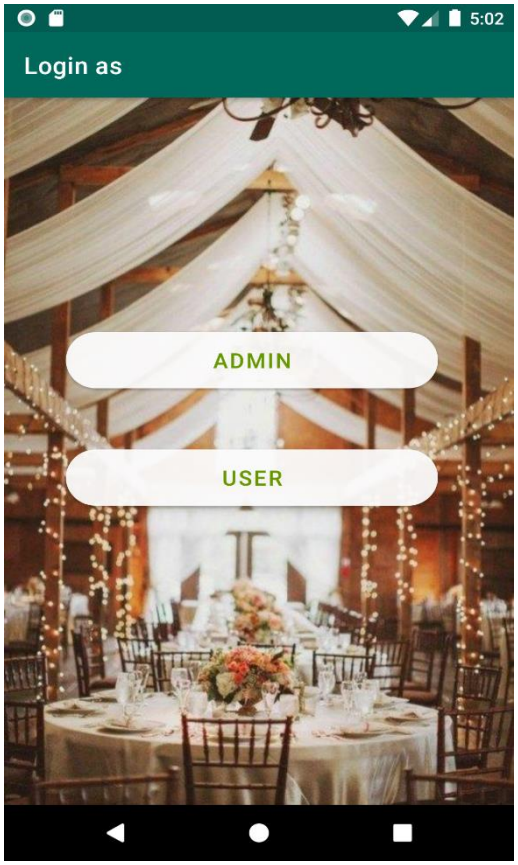
The validation given while the user enters their email ID while trying to pay the advance amount is tested here. It shows an error message if the filed is left empty.



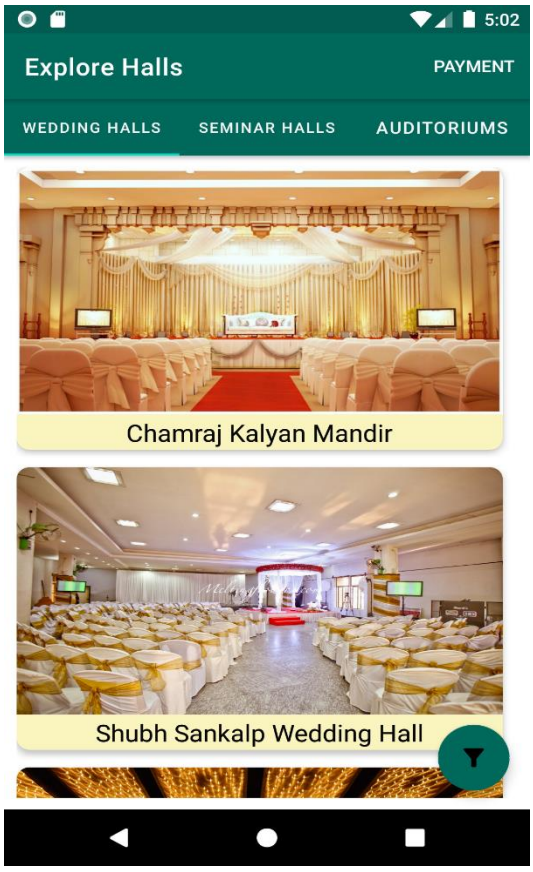
The validation implemented when the admin tries to sign in is tested. If either the username or password is incorrect, then an error message is displayed.

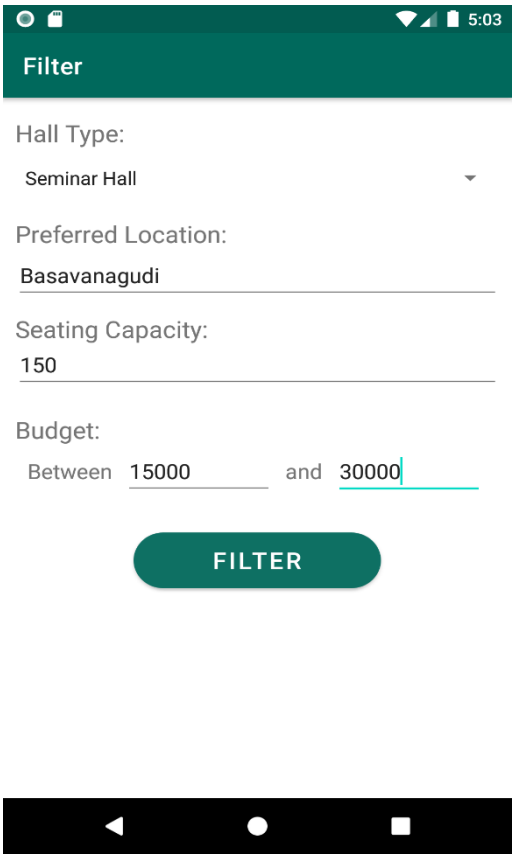
9. RESULTS AND SNAPSHOTS



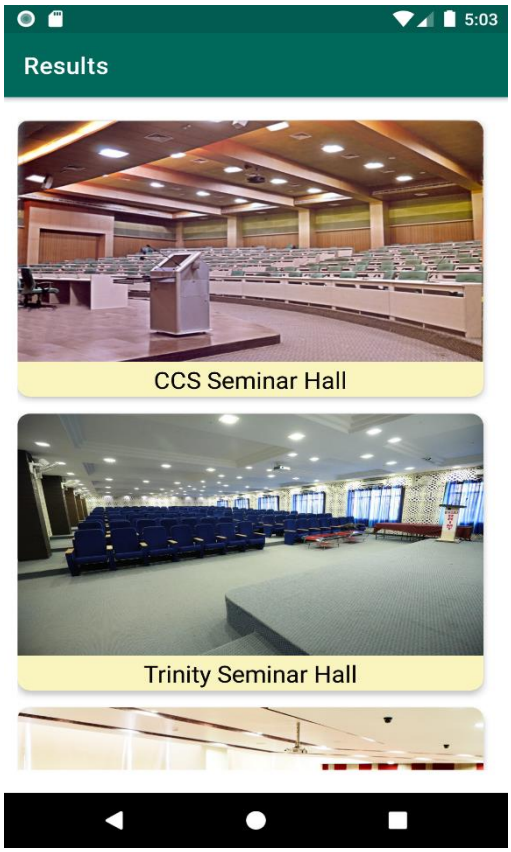


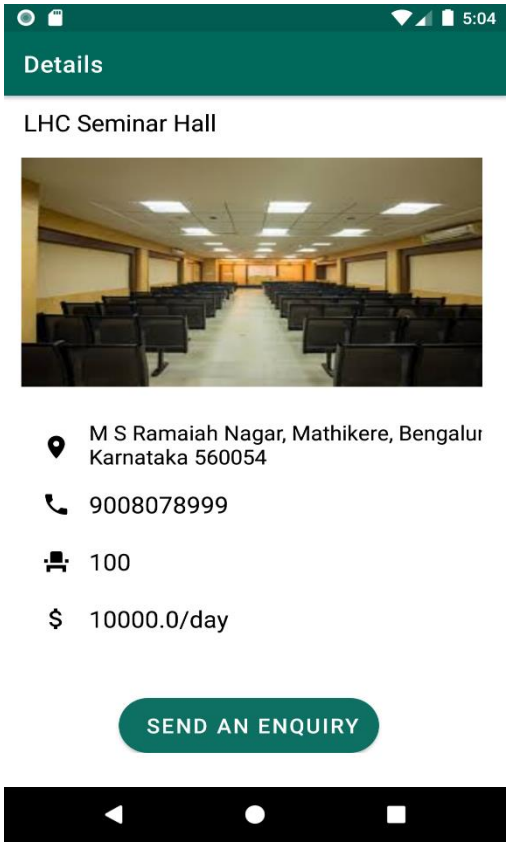
User Interface for customer:

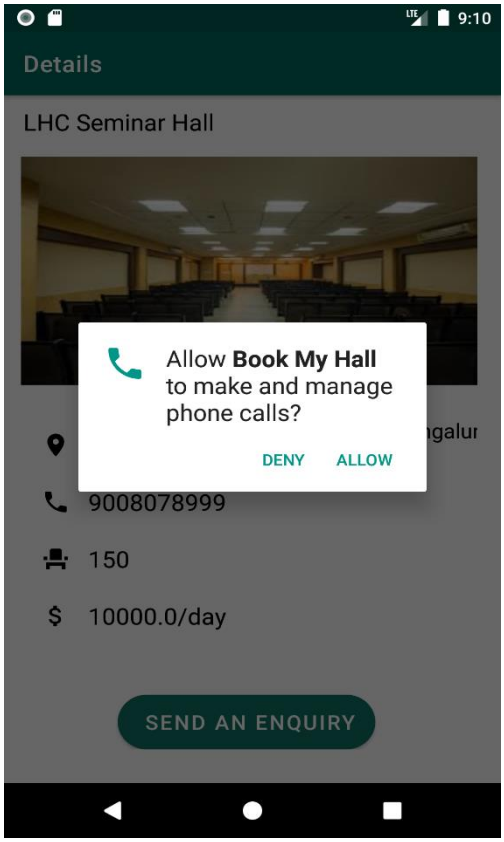


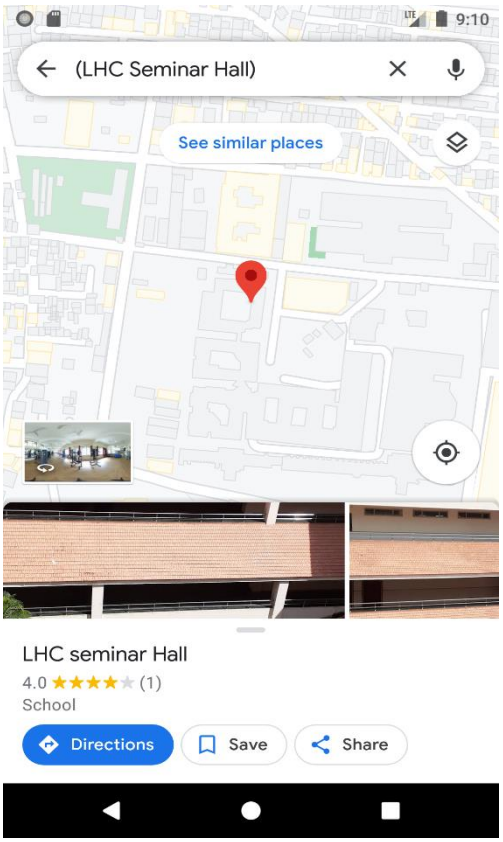


Filtering the halls by specifying preferences:

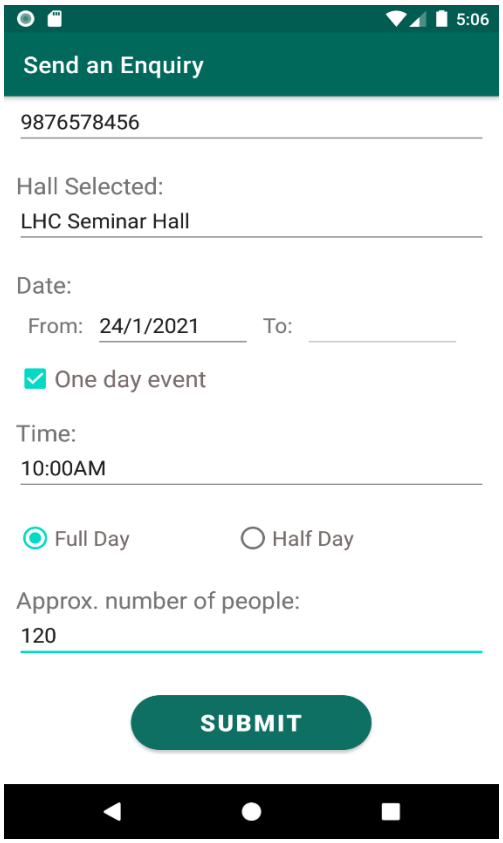


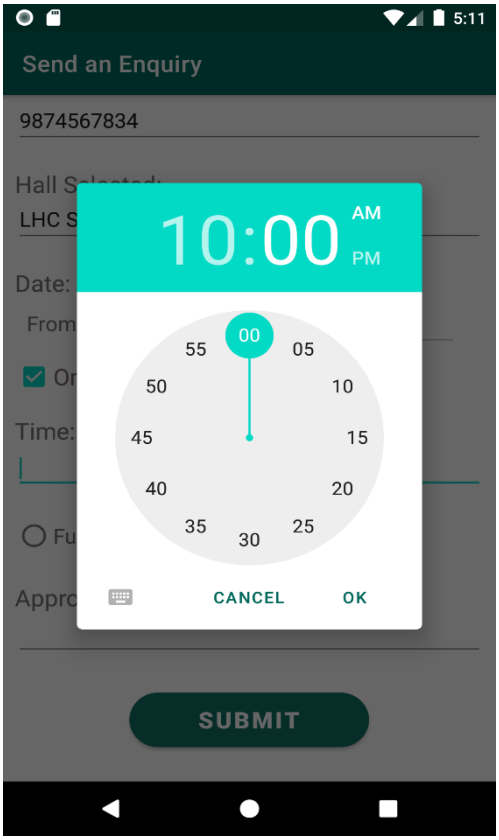




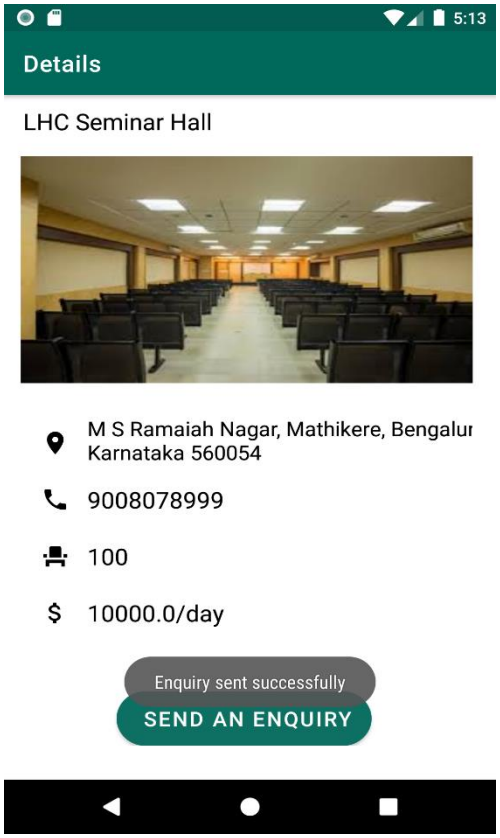


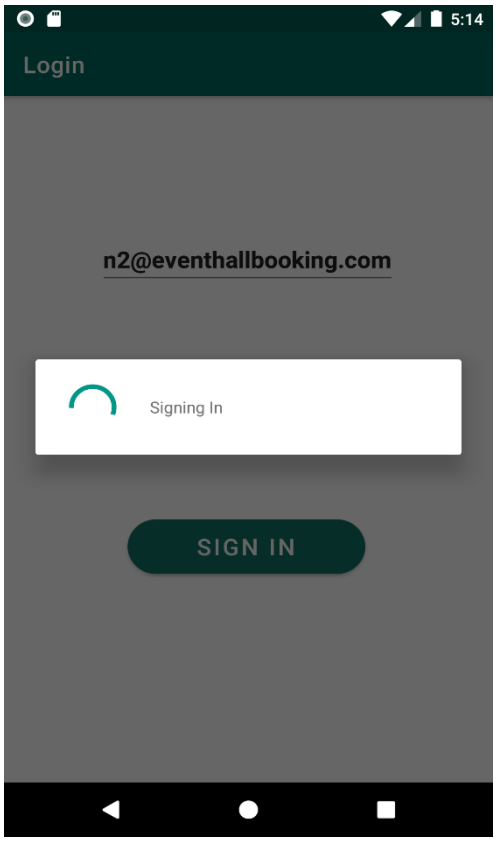
Filling the form to send an enquiry for the hall:

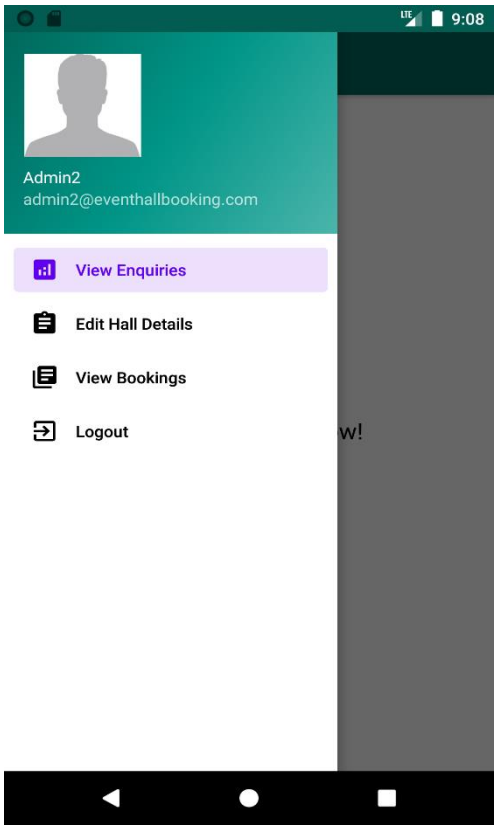


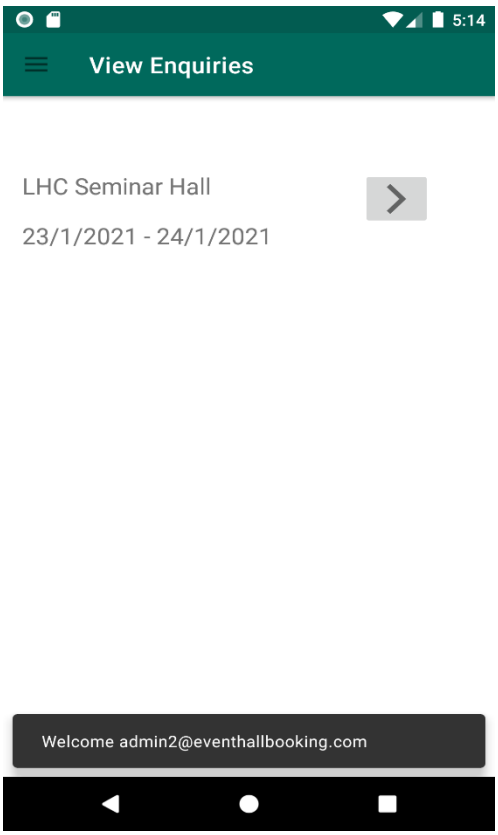


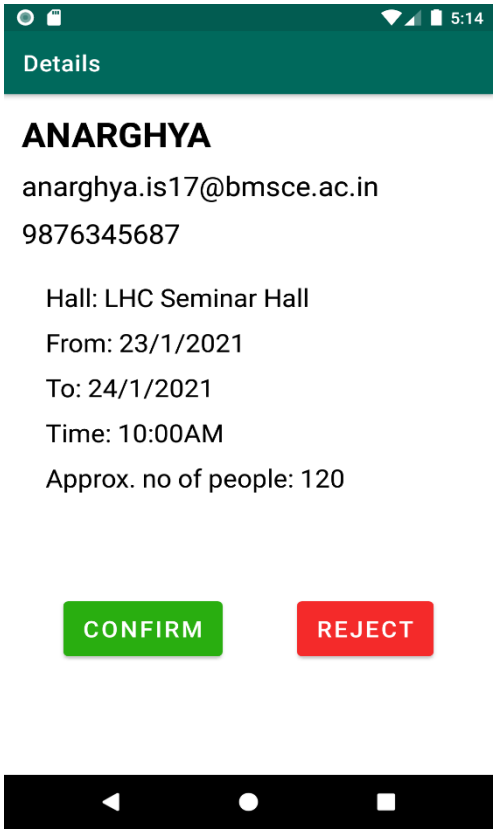
User interface for admin:

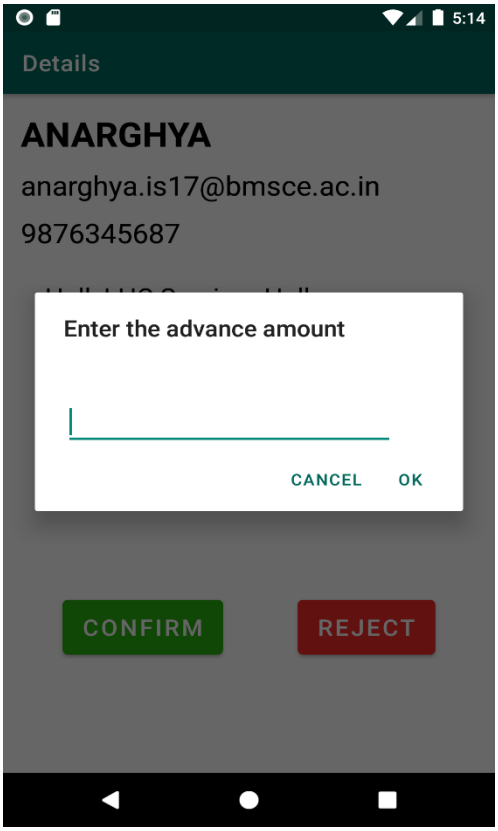




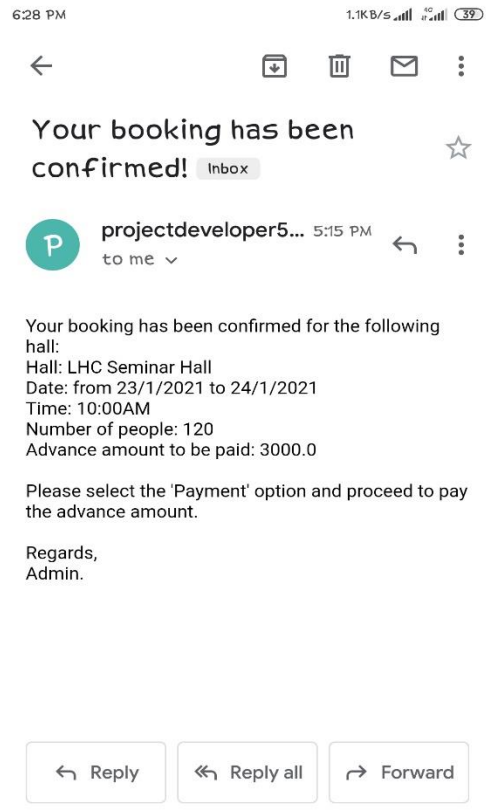




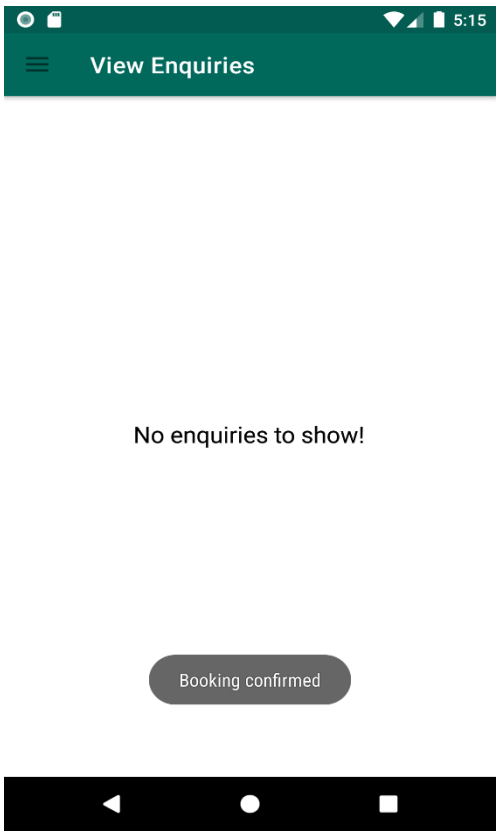


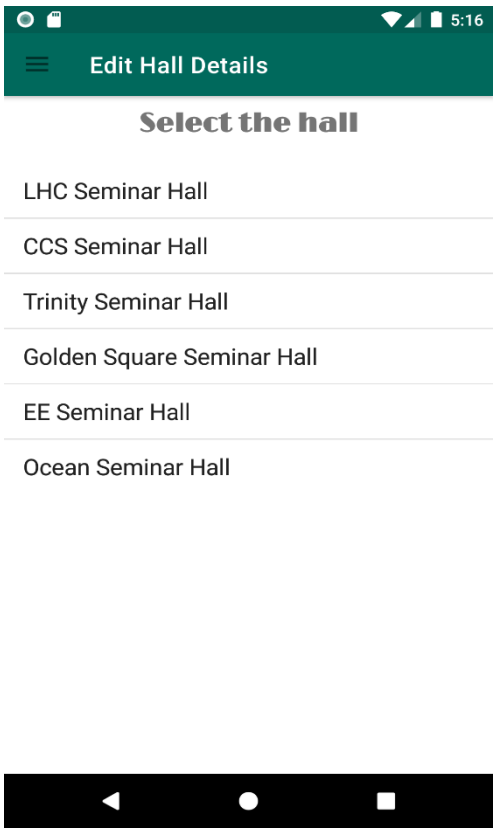


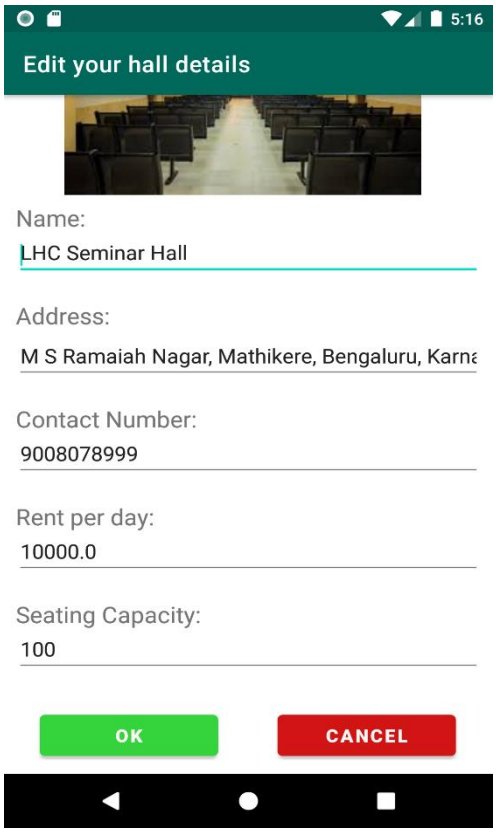
Confirmation/rejection mail sent to the user:

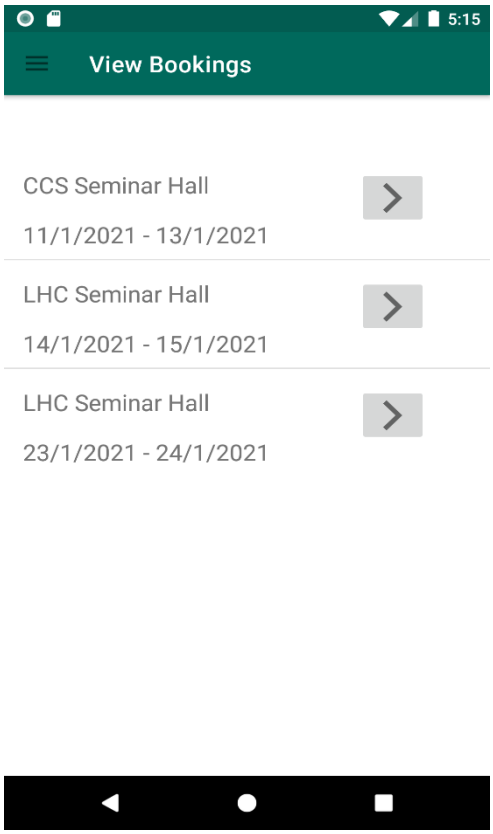


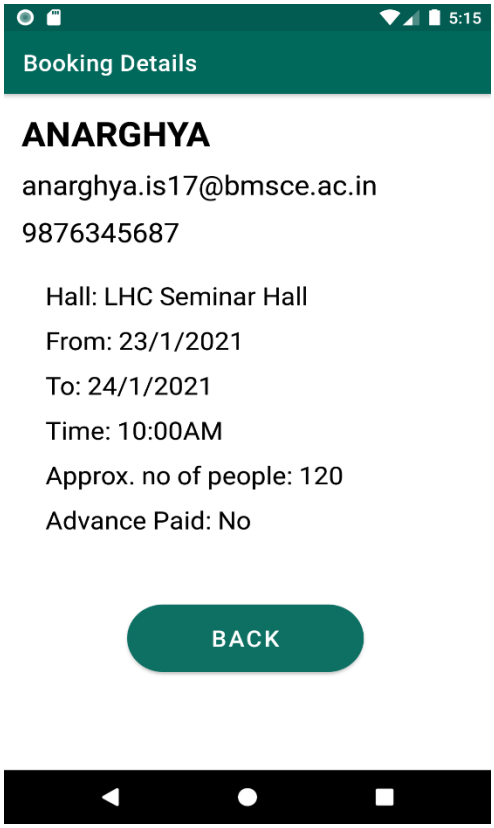
Edit Hall Details and View Bookings options:

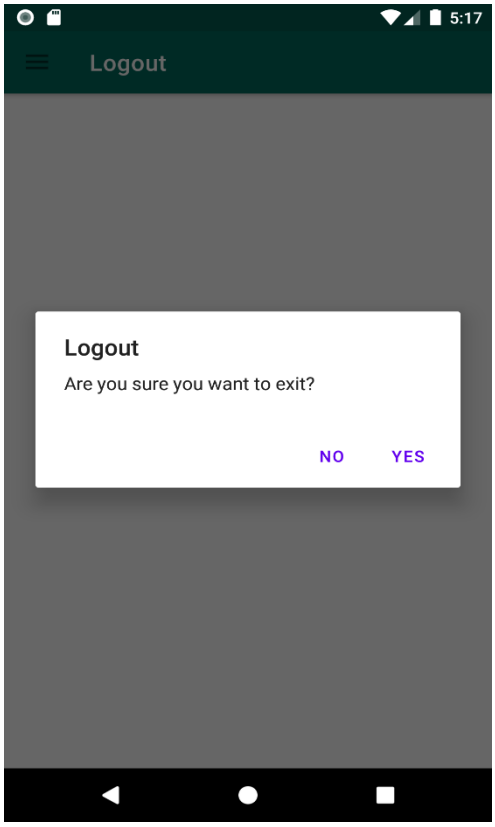




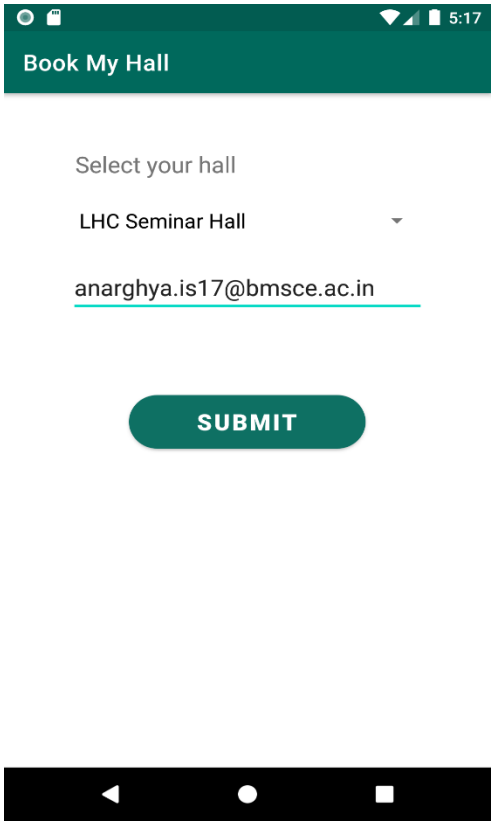


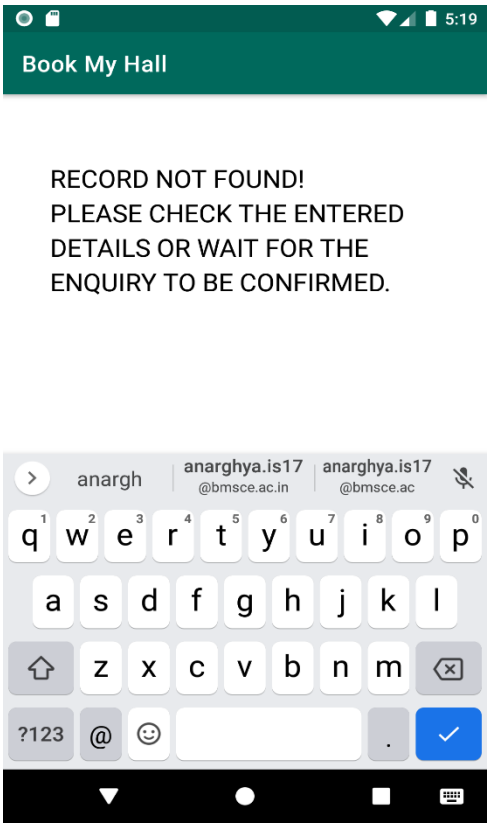


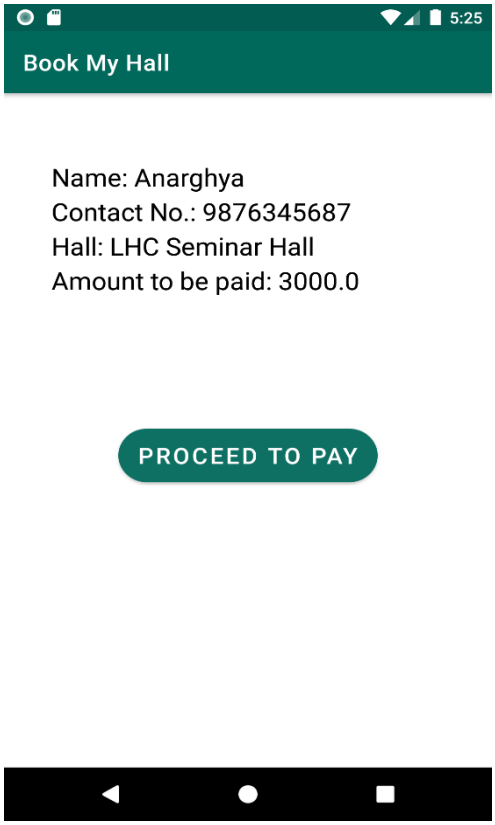


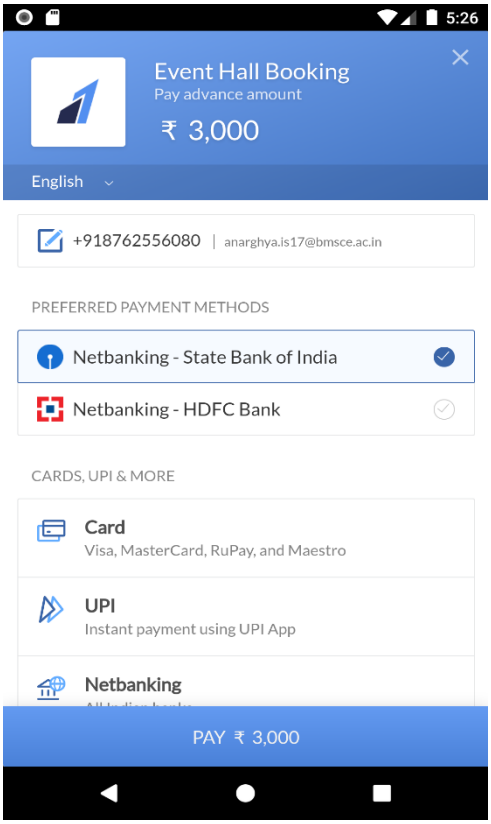


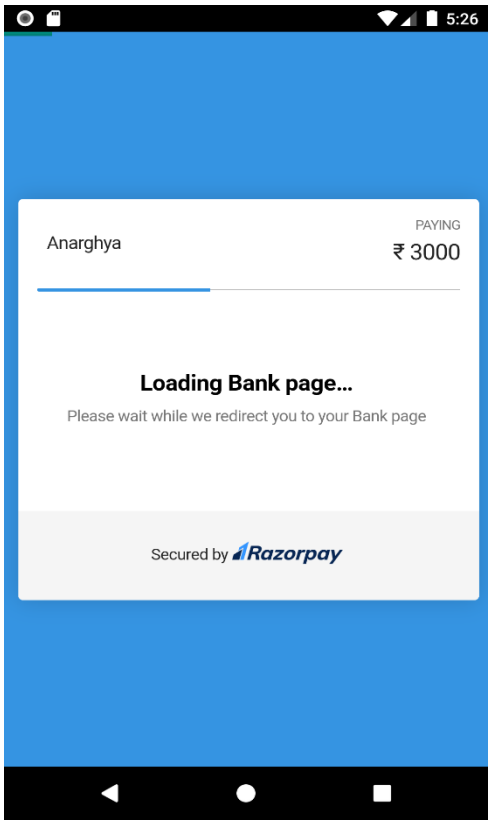
Payment activity used after booking has been confirmed:

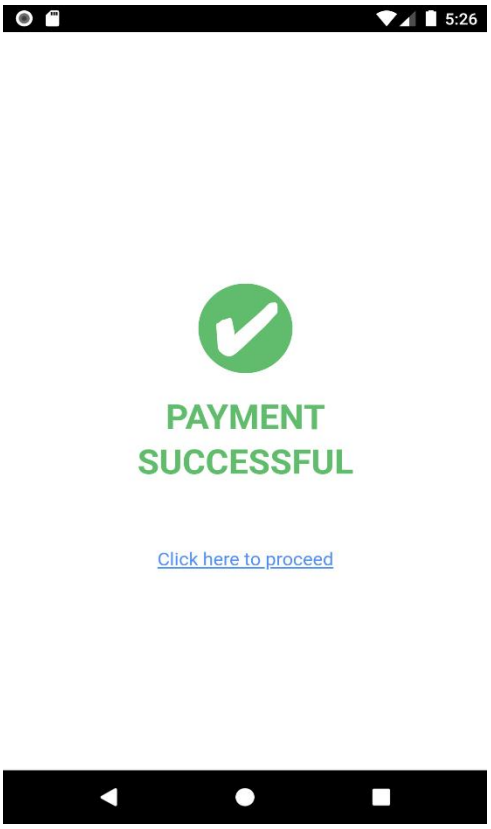





5:25
Book My Hall
Name: Anarghya
Contact No.: 9876345687
Hall: LHC Seminar Hall
Amount to be paid: 3000.0
PROCEED TO PAY


5:26
Event Hall Booking
Pay advance amount
₹ 3,000
English
+918762556080 | anarghya.is17@bmsce.ac.in
PREFERRED PAYMENT METHODS
Netbanking - State Bank of India
Netbanking - HDFC Bank
CARDS, UPI & MORE
Card
Visa, MasterCard, RuPay, and Maestro
UPI
Instant payment using UPI App
Netbanking
PAY ₹ 3,000


5:26
Anarghya
PAYING
₹ 3000
Loading Bank page...
Please wait while we redirect you to your Bank page
Secured by Razorpay


5:26
PAYMENT SUCCESSFUL
Click here to proceed