DESIGN PROJECT: MOBILE APPLICATION

A Report on

Implementation

By Nitya Kasturey

DOMAIN

Online Scheduling of appointments for sample collection for a Lab test either at person's home or going into the Lab. Procedure which goes on from collecting samples and accordingly producing the reports needs to be more efficient.

PROBLEM STATEMENT

Designing an application to perform scheduling of sample collection from user's location for a lab test and user can trace and track the reports from his/her account, get knowledge about the mentioned fields and will get the respective doctor's details. Developing an efficient way for the procedure which goes on from collecting samples and accordingly producing the reports.

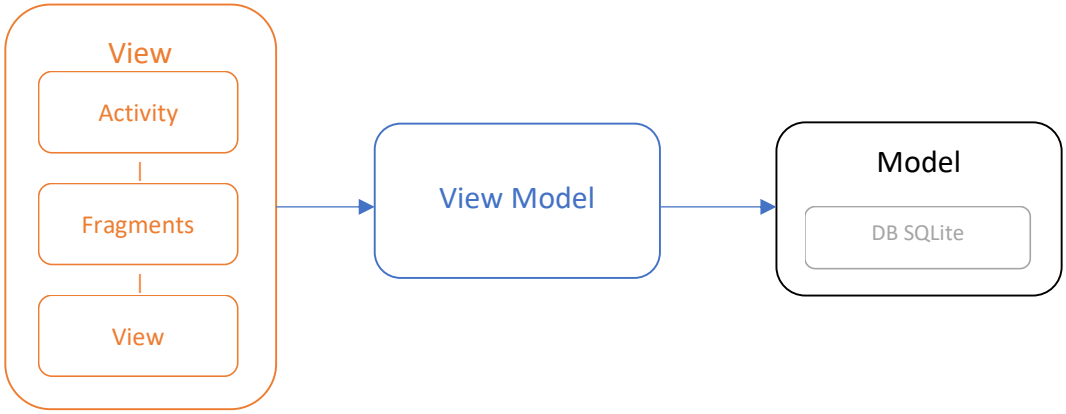
FUNCTIONAL REQUIREMENTS IMPLEMENTED

1. A platform to connect pathologies with the public for diagnosis purpose.
2. A person will register into the application with his/her credentials like username, email, contact, date of birth and password. With the same username and password user can login into the application.
3. A person can also register himself by registering for a pathology with required credentials.
4. User can see the list of pathologies according to the selected location of his/her home and can select anyone pathology from them.
5. User registered for taking services will see his profile with an option of edit profile where he can edit information like username, contact and profile image.
6. User from his profile can add members of the family with their age and gender so that at the time of registering for test these names would be visible and user can easily select the members from the list.
7. There would be list of lab test from which user needs to select required test by applying filters like number of parameters, home/lab collection, price etc.
8. While registering for lab test login user needs to calendarize a schedule for collection that is date and time and also needs to enter number of collections.
9. After booking of the test user can see list of his upcoming bookings and previous bookings with all the important information.
10. According to schedule user will receive a notification before 30 minutes from scheduled time.
11. Registered pathology would take samples according to selected test from user either by home collection or lab collection and would generate the reports accordingly.
12. User can track his journey from sample collection to doctor's visit through a tracker.
13. User can also track estimated time period for the reports to come and also the which parameter test is happening currently.

14. Labs while uploading the pdf documents of the result should also update some important field either positive or negative in a user-friendly manner. And in case more tests are required then intimate user about the same.
15. For contacting pathology, a chat option would be there for understanding the user needs for labs.

IMPLEMENTATION of DIFFERENT ANDROID and JAVA CONCEPTS

1. **Android Basics**
   a) **Architecture View**: Multiple activities are created with some fragments inside them and presented using view. The architecture flow is shown below:



   b) **Activities:** Different activities are created for each of the functioning. The activities are created in the project are:
      i) Splash Screen – Starting screen of the app which is visible for 3 seconds and then directs to the main activity.
      ii) Main Activity – Handles the whole project by providing direct links to navigate from home to other pages of the project. At the first activity opens up with a navigation left drawer, top and bottom navigation which directs to different fragments of main activity and home page fragment is opened by default.
      iii) Edit Profile Activity – Activity opens up when user select edit profile option from profile fragment. This activity provides user to edit user information such as username, contact and profile phot.
      iv) Book Appointment Activity – Opens a form in case user wants to book a test which takes information such as test name, date & time, members, contact etc.
      v) Add Members Pop Up Activity – This is a pop-up activity which allow user to add, delete and see the members added by the user linked with his email address.
      vi) Register Activity – For new User opens a registration form with all fields validation and register a new user into the application.
      vii) Login Activity – User can login in two ways that is either by his registered email or by using Gmail.

c) **Fragments:** Application Consist of bottom navigation which directs the user to different fragments linked with the main activity.

i) Home Page – Home fragment which is enabled when Main activity is opened and it shows details like about app and popular tests etc.

ii) Lab Test – Fragment shows the list of tests uploaded by all the pathologies. Filters such as search and checkboxes to choose for collection way that is home or lab.

iii) Profile – Fragment shows the profile details of user such as profile photo, name, contact, email and recent tests. Also, an option to add members to his account is available which directs him to Add Members Pop Up Activity.

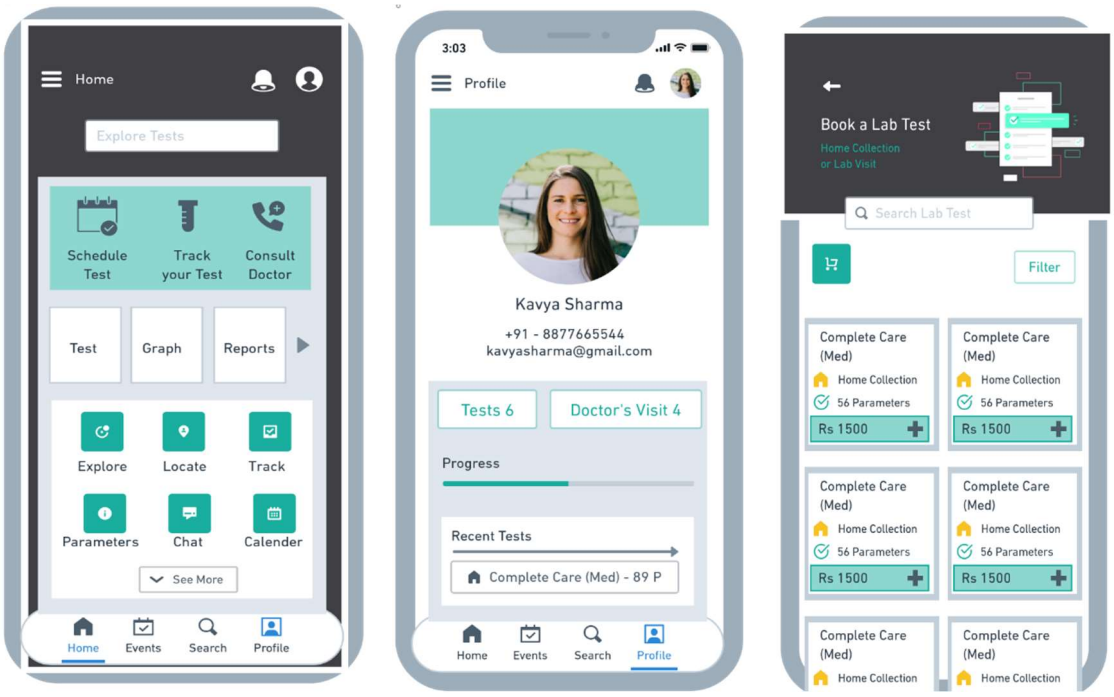
iv) Lab Test Details – Fragments open when user clicks on a lab test from the list shown and this shows details of test like pre requisite, price, time taken by the report etc.

v) My Bookings – Fragment shows user's upcoming test sample booking and previous test bookings.

d) **Activity Life Cycle** – The java code is implemented by following the activity life cycle that implementations and workings needs to get started when th application creates, starts and resumes and keeping these implementations are kept in onCreate and onResume and whenever the app restarts then also the activity runs in the same way.

2. User Interaction

a) **Application Wireframe** – For developing the user's interaction with the application wireframes were made while designing by implementing the concepts of User Interface and User Experience Design. Some of the application wireframes are:



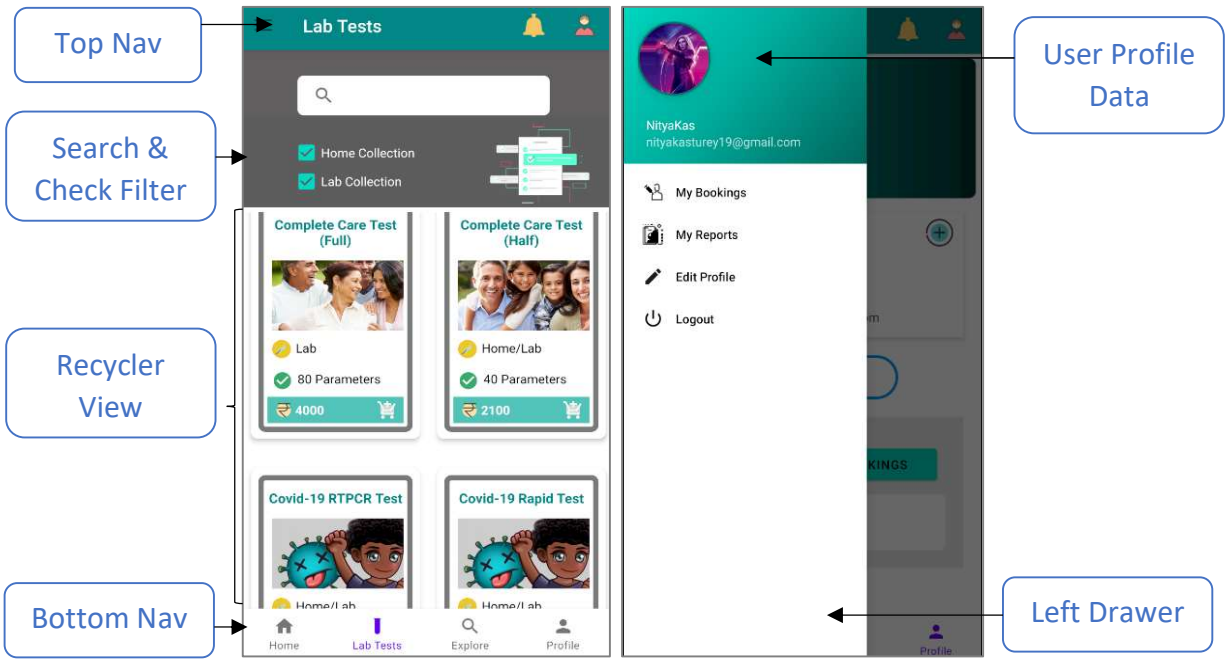
b) **Implementing UI and UX**: For the better User interaction these wireframes were created and implemented in an organised way by applying different layout like Linear Layout, Relative Layout etc. Below are different effects implemented in XMLs:

i) Layouts such as Linear Layout and Relative Layout are used to organise different widgets.

ii) Recycler View, Circular Image View, Bottom Navigation are used to show data is a structured way with effects like ripple, colours etc.

iii) Slide bars with effects like fade duration, alignment etc.

iv) Form fields such as edit text, spinner, check box etc and Search bar with filter.



c) **Testing UI:** All the pages created are designed keeping in mind the concepts of UI and most of the design patterns include structured layouts for the application. The points which are included while designing the pages are:

- Functionality
- Visual Design
- Performance
- Usability
- Compliance

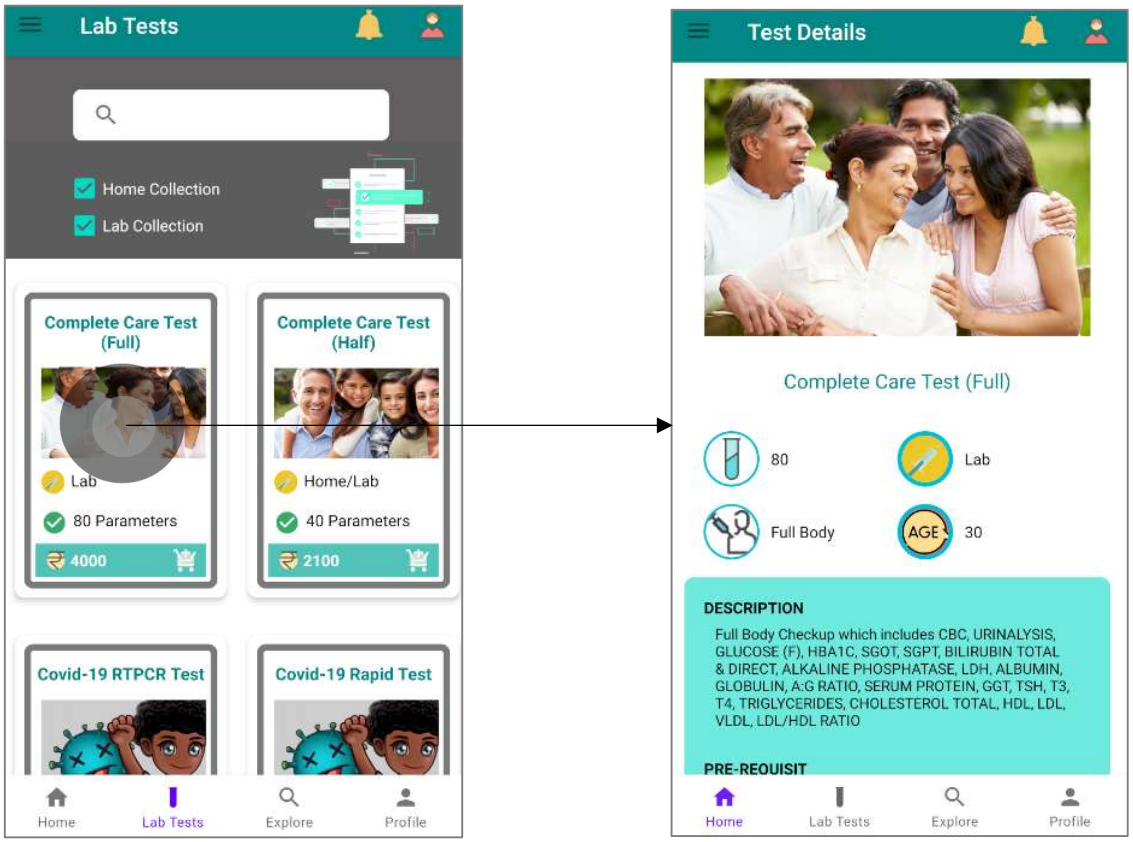
3. Background Services

a) **Intent**: Intent is used to send data from activity to activity with the use of startActivity and with intent we can send any amount of data using bundle through Extras. Below is the code snippet showing the working of intent. Bundle will save the data by assigning keys to them and through intent whole bundle would be passed to next activity.

```
bookTest.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(), BookTestAct.class);
        intent.putExtras(bundle);
        startActivity(intent);
    }
});
```

b) Transaction: Transaction occurs from fragment to fragment. In the below example Current fragment would be replaced by LabTestDetails fragment. Bundle will save the data by assigning keys to them and through set Arguments whole bundle would be passed to next fragment.

```
bundle = new Bundle();
bundle.putString("image", jObj.getString( name: "image"));
bundle.putString("testname", jObj.getString( name: "name"));
bundle.putString("parameter", jObj.getString( name: "parameter"));

FragmentTransaction transaction = getActivity().
        getSupportFragmentManager().beginTransaction();
transaction.addToBackStack(null);
transaction.setTransitionStyle(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
Fragment fragment = new LabTestDetails();
fragment.setArguments(bundle);
transaction.replace(getId(), fragment).commit();
```



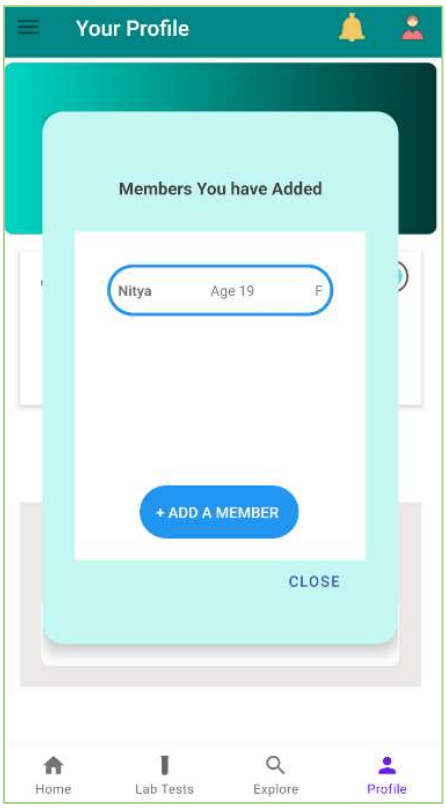
c) List and Recycler View:

- List are used to store model objects such as Bookings, LabTests etc so that it can be retrieve and provided to the model when required. Also lists are used to store the objects data while user is updating or inserting data into database.
- Recycler View will display the list of data by repeating the views and replacing its data.

```
JSONObject jsonObject = new JSONObject(response);
JSONArray jsonArray = jsonObject.getJSONArray( name: "data");
for (int i=0; i<jsonArray.length(); i++)
{
    JSONObject jObj = jsonArray.getJSONObject(i);
    Member memberData = new Member(jObj.getString( name: "name"),
            jObj.getString( name: "age"), jObj.getString( name: "gender"));
    memberArrayList.add(memberData);
}
```

```
memberAdapter= new MyMemberAdapter(getApplicationContext(), memberArrayList);
recyclerViewMembers.setAdapter(memberAdapter);
```

In the below code a Member object is created and member array list stores member type of objects. This list will be added to the adapter to take the values from list and print it into the correct spot of the card. Recycler view will generate the card again and again according to the list size.



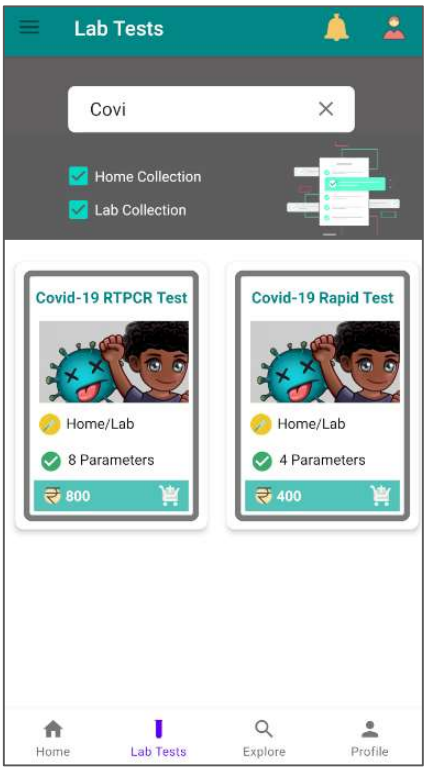
d) **Search Filter**: This is used to filter searching for lab tests. Filter is applied on the test name and all those tests having similar test name would be displayed in the result. When the list item is similar to the input field then only it would get added to the filtered list and would be displayed on the screen.

```
@Override
public Filter getFilter() { return exampleFilter; }

private final Filter exampleFilter = new Filter() {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        ArrayList<LabTestData> filteredList = new ArrayList<>();
        if (constraint==null|| constraint.length()==0){
            filteredList.addAll(list);
        }else {
            String filterPattern = constraint.toString().toLowerCase().trim();
            for (LabTestData item : labsArrayList){
                if (item.getName().toLowerCase().contains(filterPattern)){
                    filteredList.add(item);
                }
            }
        }
        FilterResults results = new FilterResults();
        results.values = filteredList;
        Log.i( tag: "result", results.values.toString());
        return results;
    }

    @Override
    protected void publishResults(CharSequence constraint, FilterResults results) {
        labsArrayList.clear();
        labsArrayList.addAll((List) results.values);
        notifyDataSetChanged();
    }
```

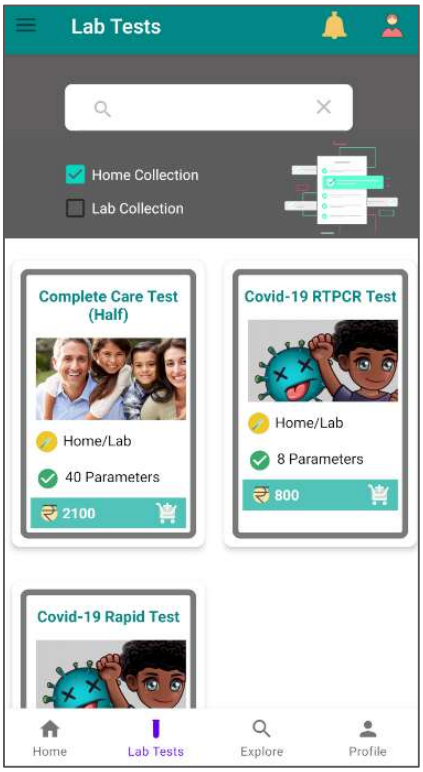
e) **Check Box Filter:** Check boxes for home and lab is made and initially both the boxes are checked. When user tries to uncheck any of them then only those tests would be shown which is having similar collection type.

```
public void collectionFilter(String coll){
    ArrayList<LabTestData> filteredList = new ArrayList<>();
    String filterPattern = coll.trim();
    for (LabTestData item : labsArrayList){
        if (item.getCollection_type().contains(filterPattern)){
            filteredList.add(item);
        }
    }
    labsArrayList.clear();
    labsArrayList = filteredList;
}
```

Search Filter



Check Box Filter



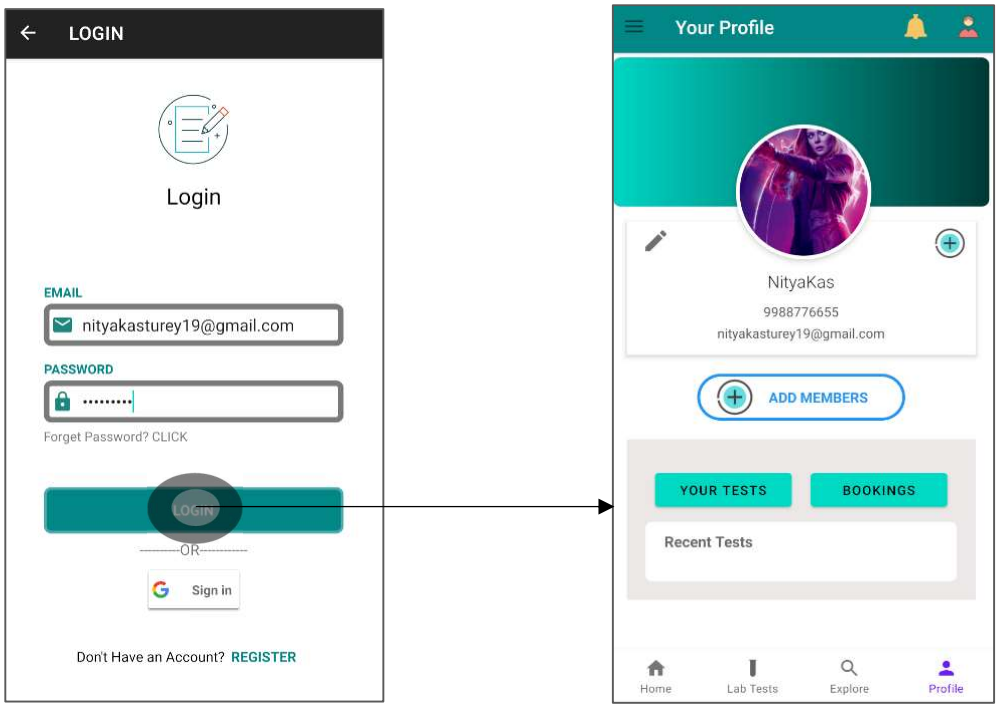
4. **Android Storage**
   a) **Shared Preferences**: Shared Preferences are used to main the session of the used who has login into the application. It would store the data into the created preferences and on each wherever required these preferences would be called and data of user can be accessed from them.

   - Editing the Preferences

```
SharedPreferences.Editor editor = preferences.edit();
editor.putString("username", obj.getString( name: "username"));
editor.putString("password", obj.getString( name: "password"));
editor.putString("email", obj.getString( name: "email"));
editor.putString("contact", obj.getString( name: "contact"));
editor.putString("image", obj.getString( name: "image"));
editor.apply();
editor.commit();
```

   - Accessing the preferences and printing values

```
SharedPreferences preferences = getSharedPreferences( name: "mypref", MODE_PRIVATE);
profileUsername.setText(preferences.getString( key: "username", defValue: ""));
profileEmail.setText(preferences.getString( key: "email", defValue: ""));
```



a) **Storing Data into the Database**: Cloud server is used to store user's information through php. For each type of information different tables are used and to store them php file link is stored in a constant string from which the data can be inserted into the system.

b) **Implementing CRUD operations**

Database operations requires a Volley Singleton class that sends request for any operation into the database. For each of the below operation string request is created by calling the respective url for that operation

i) Insert Data – Inserting data into the database, includes creating a string request with insert url and passing values using HashMap into the php file. If the data is inserted then php would echo success and same response can be seen in android java file from which user can understand about the successful data insert.

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, Constants.INSERT_MEMBER_URL,
        new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
        Log.i( tag: "INFO", response);
        if(name!=null && age!=null && gender!=null) {
            if (response.equals("success")) {
                Toast.makeText(getApplicationContext(), text: "Member Added Successfully",
                        Toast.LENGTH_SHORT).show();
                recyclerViewMembers.removeAllViews();
                loadData();
            } else {
                Toast.makeText(getApplicationContext(), response, Toast.LENGTH_SHORT).show();
            }
        } }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
    }
}) {
    @Nullable
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        HashMap<String, String> hashMap = new HashMap<>();
        hashMap.put(KEY_EMAIL, preferences.getString( key: "email", defValue: ""));
        hashMap.put(KEY_NAME, name);
        hashMap.put(KEY_AGE, age);
        hashMap.put(KEY_GENDER, gender);
        return hashMap;
    }
};
singleton = VolleySingleton.getInstance(this);
singleton.addToRequestQueue(stringRequest);
```

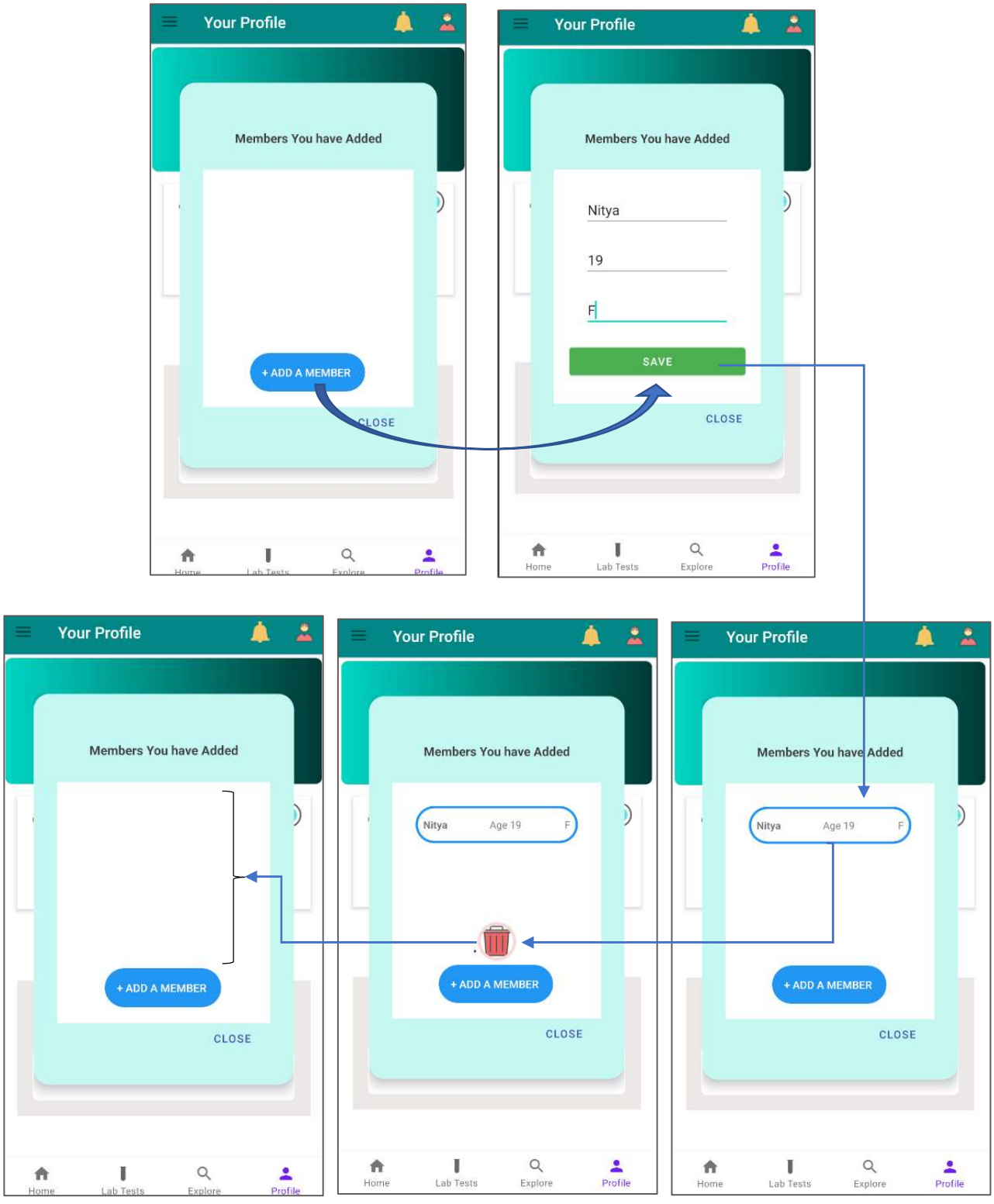
ii) Delete Data – Deleting data from the database, includes creating a string request with insert url and passing values using HashMap into the php file for which we need to delete the data. If the data is deleted then php would echo success and same response can be seen in android java file from which user can understand about the successful data deletion.

```
private void deleteMemberData(int position) {
    Member member = memberArrayList.get(position);
    StringRequest stringRequest = new StringRequest(Request.Method.POST, Constants.DELETE_MEMBER_URL,
            new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            Log.i( tag: "INFO", response);
                if (response.equals("success")) {
                    Toast.makeText(getApplicationContext(), text: "Member Data Deleted Successfully",
                            Toast.LENGTH_SHORT).show();
                    recyclerViewMembers.removeAllViews();
                    loadData();
                } else {
                    Toast.makeText(getApplicationContext(), response, Toast.LENGTH_SHORT).show();
                }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
        }
    }) {
        @Nullable
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            HashMap<String, String> hashMap = new HashMap<>();
            hashMap.put(KEY_EMAIL, preferences.getString( key: "email", defValue: ""));
            hashMap.put(KEY_NAME, member.getName());
            return hashMap;
        }
    };
    singleton = VolleySingleton.getInstance(this);
    singleton.addToRequestQueue(stringRequest);
}
```

iii) Display Data – Deleting data from the database, includes creating a string request with insert url and passing values using HashMap into the php file for which we need to delete the data. If the data is deleted then php would echo success and same response can be seen in android java file from which user can understand about the successful data deletion.

On Item Click – When click on any of the item that is any member data then delete icon would be shown and by clicking on that icon selected item data would be deleted from the database.

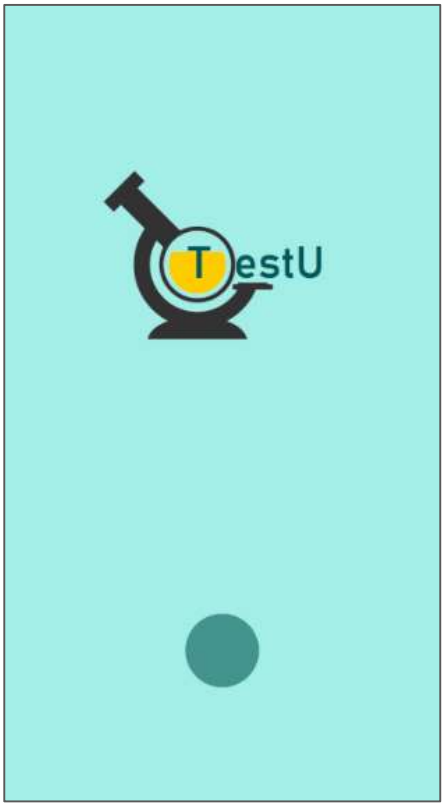
```
private void loadData() {
    StringRequest stringRequest = new StringRequest(Request.Method.POST, Constants.MEMBERS_DATA_URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Log.i( tag: "INFO", response);
                    try {

                        JSONObject jsonObject = new JSONObject(response);
                        JSONArray jsonArray = jsonObject.getJSONArray( name: "data");
                        for (int i=0; i<jsonArray.length(); i++)
                        {
                            JSONObject jObj = jsonArray.getJSONObject(i);
                            Member memberData = new Member(jObj.getString( name: "name"),
                                    jObj.getString( name: "age"), jObj.getString( name: "gender"));
                            memberArrayList.add(memberData);
                        }
                        memberAdapter= new MyMemberAdapter(getApplicationContext(), memberArrayList);
                        recyclerViewMembers.setAdapter(memberAdapter);
                        memberAdapter.setOnItemClickListener(new MyMemberAdapter.OnItemClickListener() {
                            @Override
                            public void onItemClick(int position) {
                                deleteMember.setVisibility(View.VISIBLE);
                                deleteMember.setEnabled(true);
                                deleteMember.setOnClickListener(new View.OnClickListener() {
                                    @Override
                                    public void onClick(View v) {
                                        deleteMemberData(position);
                                    }
                                });
                            }
                        });
                    }catch (JSONException e){
                        Log.e( tag: "ERROR", e.getMessage());
                    }
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "ERROR", error.getMessage());
        }
    }){
        @Nullable
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            HashMap<String, String> hashMap = new HashMap<>();
            hashMap.put(KEY_EMAIL, preferences.getString( key: "email",  defValue: ""));
            return hashMap;
        }
    };
    singleton = VolleySingleton.getInstance(this);
    singleton.addToRequestQueue(stringRequest);
```



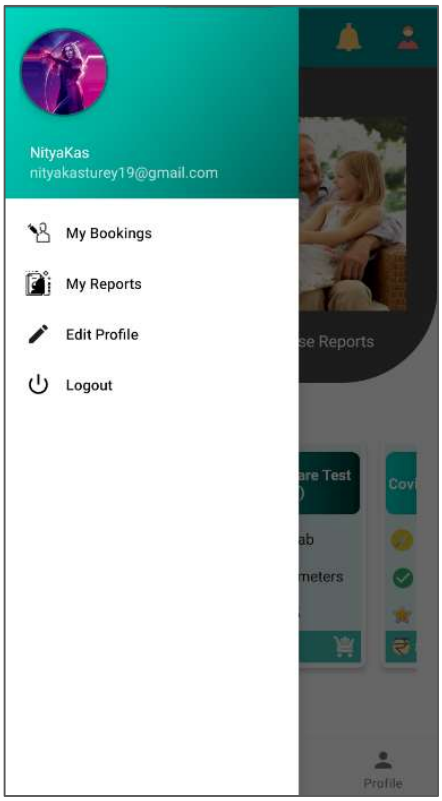
5. **Permissions**
   a) **Permissions:** Permissions such as INTERNET and ACCESS NETWORK STATE are used for accessing ay data from the internet. For example, some images are directly displayed from internet so for that internet permission was needed.

IMPLEMENTATION RESULTS

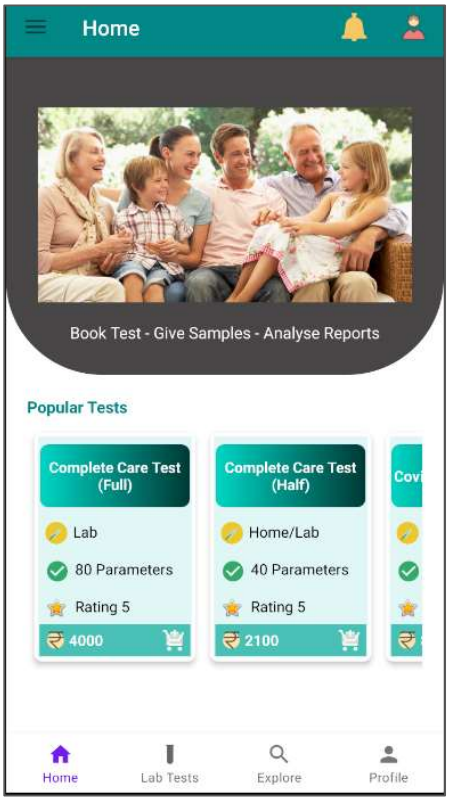
- Implementation results of all activities and fragments.



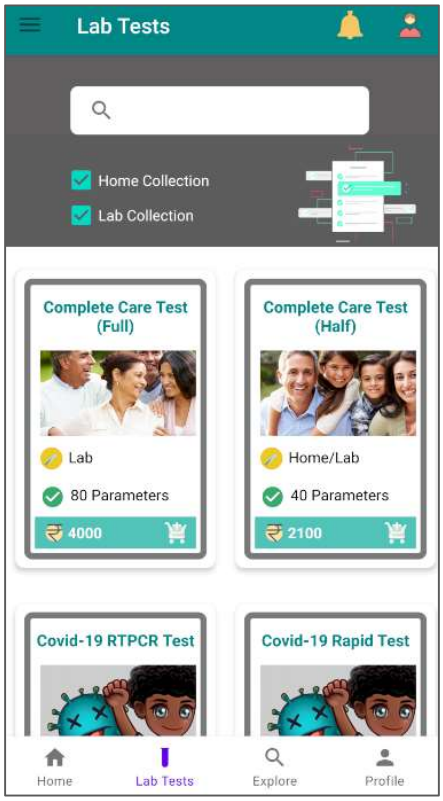
Splash Screen



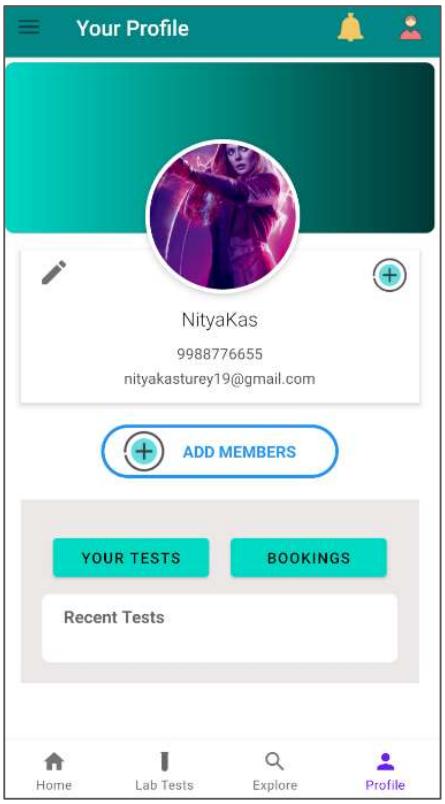
Main Activity Drawer



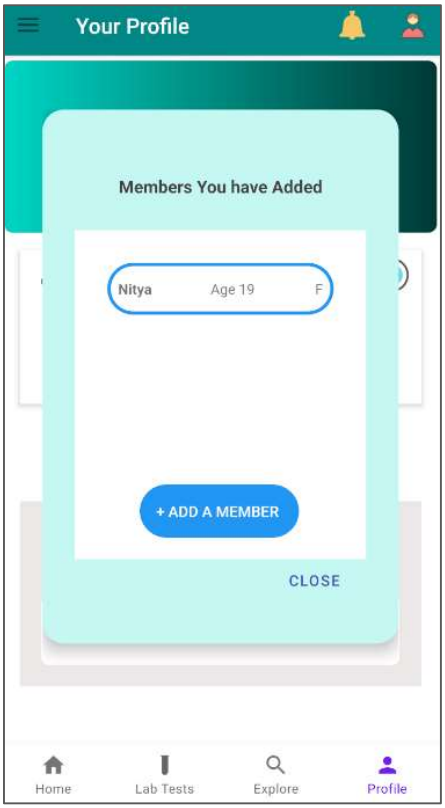
Home Page



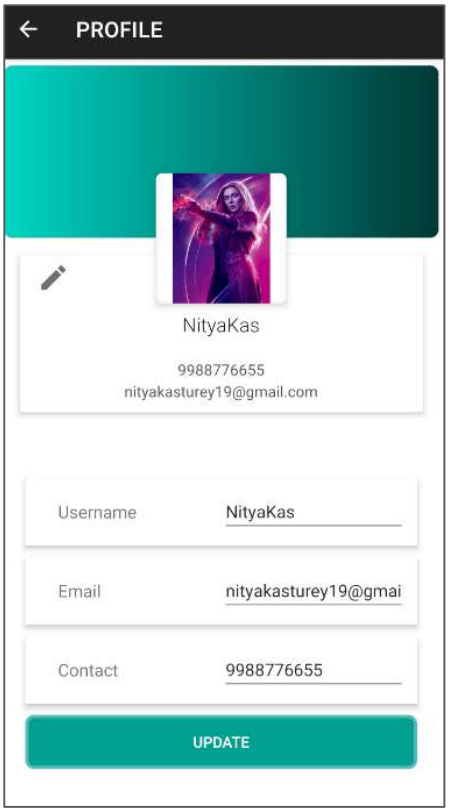
Lab Tests Page



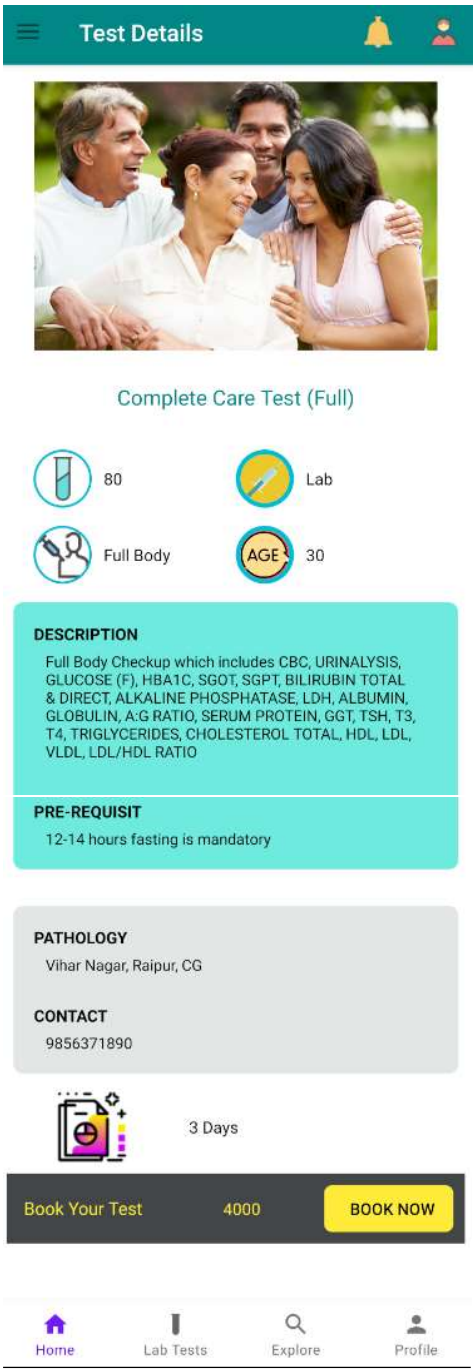
User Profile Page



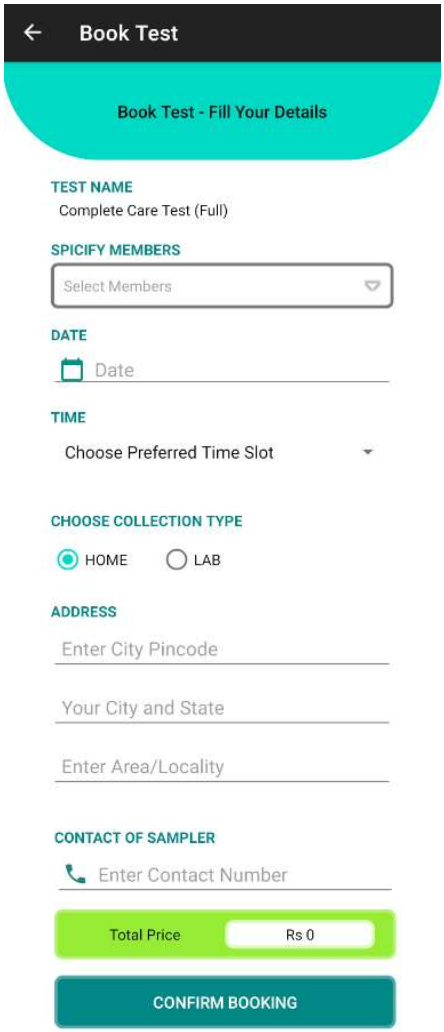
Adding Members



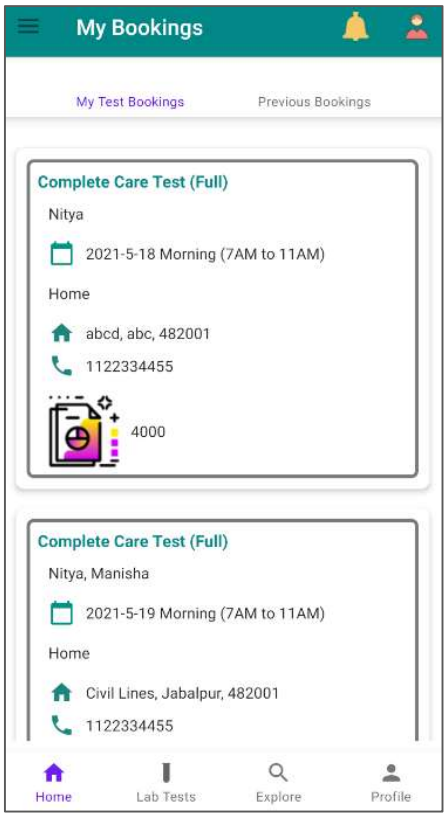
Edit Profile Activity



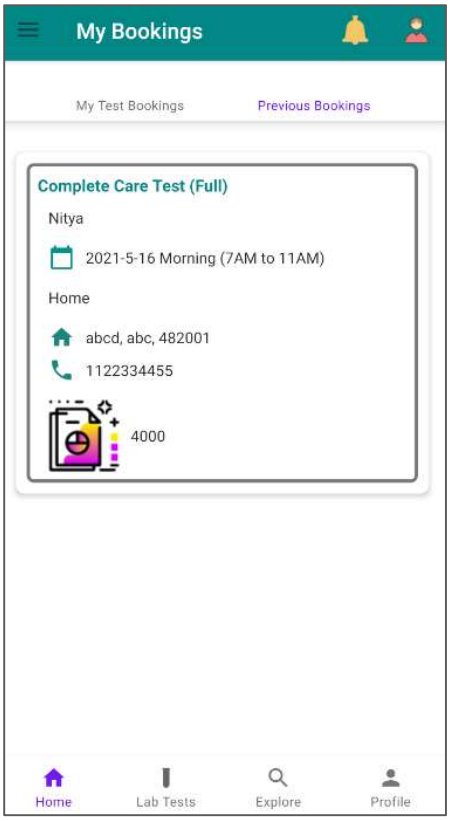
Lab Test Detail



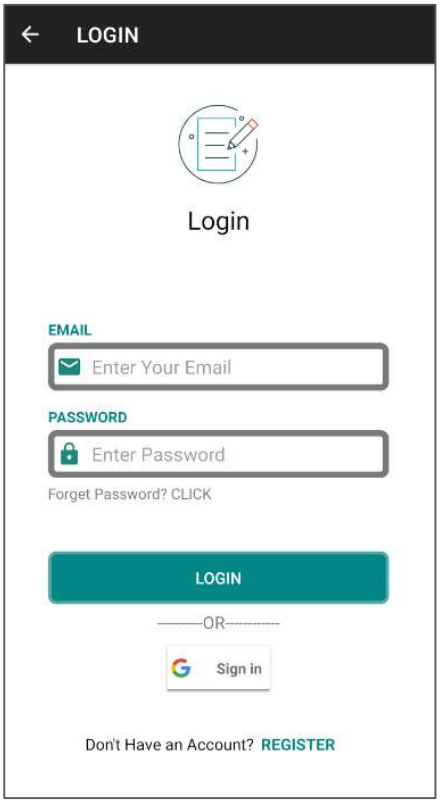
Book Test Form



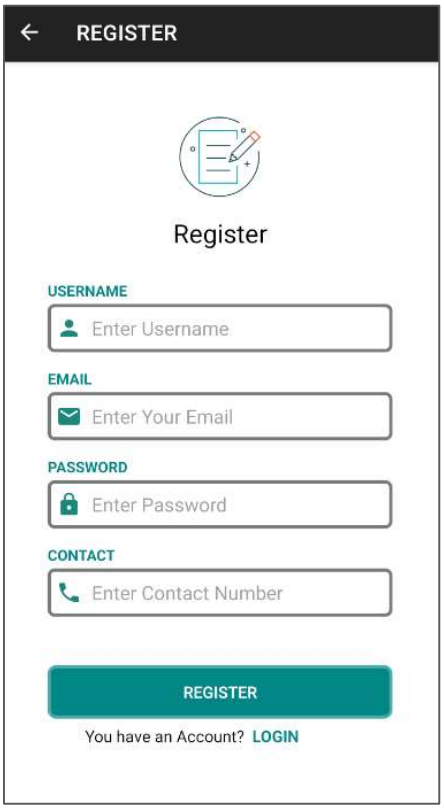
My Upcoming Booking



My Previous Bookings



Login Page



Register Page