Ramakrishna Sayee Meka

rameka@ysr.edu

Hemanth Dahagam

hdahagam@gmail.com

PROJECT REPORT- Scout City

Abstract

This is a location based android application which provides the users with latest and up to date information about different cafes, restaurants, food joints and recommended places such as city attractions and theme park throughout the city around the world. The user can select any of the default cities provided while launching the application or use GPS to set the current location of the user. The app provides the user with GPS based navigation and Maps to the user, along with the entire location information like rating, website, contact info and location tips provided by others.

## Contents

[Contents 1](#_Toc500626272)

[1. Splash Screen: 2](#_Toc500626273)

[1.1 Layout File: 2](#_Toc500626274)

[1.2 Code Snippet for Splash Screen: 3](#_Toc500626275)

[2. Login Screen: 3](#_Toc500626276)

[2.1 Layout File: 4](#_Toc500626277)

[2.2 Code Snippets for Login Screen: 4](#_Toc500626278)

[3. SignUp Screen: 6](#_Toc500626279)

[3.1 Layout File: 6](#_Toc500626280)

[3.2 Code Snippet for SignUp Screen: 7](#_Toc500626281)

[4. Navigation Activity: 8](#_Toc500626282)

[4.1 Layout File: 8](#_Toc500626283)

[4.2 Code Snippet for Navigation Drawer: 9](#_Toc500626284)

[5. Recycler View: 9](#_Toc500626285)

[5.1 Layout file: 9](#_Toc500626286)

[5.2 Code Snippet for RecyclerView Fragment: 10](#_Toc500626287)

[5.3 Code Snippet for RecyclerView Adapter: 12](#_Toc500626288)

[5.4 Card View: 13](#_Toc500626289)

[5.3 RecyclerView Animations: 14](#_Toc500626290)

[6. ViewPager Activity: 15](#_Toc500626291)

[6.1 Layout File: 16](#_Toc500626292)

[6.2 Code Snippet for ViewPager: 17](#_Toc500626293)

[7. Location and GPS: 18](#_Toc500626294)

[7.1 Code Snippet for Displaying Location on Map: 19](#_Toc500626295)

[7.2 Code Snippet for Fetching Location through GPS: 19](#_Toc500626296)

[8. Profile Page: 21](#_Toc500626297)

[9. Advanced Animation: 24](#_Toc500626298)

[10. Retrofit Library and Real API Data: 25](#_Toc500626299)

[11. Multi-Window support and App Shortcuts: 27](#_Toc500626300)

[12. Menus: 28](#_Toc500626301)

[13. Coordinator Layout: 29](#_Toc500626302)

[14. Pop Up Activity: 30](#_Toc500626303)

[15. Custom Views: 32](#_Toc500626304)

# **Splash Screen:**

This is the initial splash screen of the application which is displayed every time the user logs in to the application. To implement this the, a splash screen activity must be run in a separate run under a thread, which is invoked by and Handler class postDelayed() method. is loaded for 4 seconds and then the next screen is loaded.

## 1.1 Layout File:

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout 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:orientation="vertical"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/splash"  
 tools:context="com.syracuse.rameka.scoutapp.SplashScreenActivity"**>  
 <**ImageView  
 android:layout\_width="200dp"  
 android:layout\_height="100dp"  
 android:src="@drawable/scoutcityimage"  
 android:layout\_gravity="center|bottom"**/>  
</**LinearLayout**>



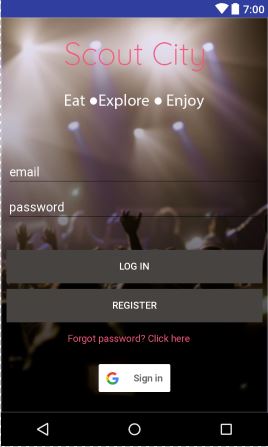
## 1.2 Code Snippet for Splash Screen:

**public class** SplashScreenActivity **extends** AppCompatActivity {  
 **private static int** *TIME\_INTERVAL* = 4000;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_splash\_screen***);  
 **new** Handler().postDelayed(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Intent intent = **new** Intent(SplashScreenActivity.**this**,MainActivity.**class**);  
 startActivity(intent);  
 finish();  
 }  
 },*TIME\_INTERVAL*);  
 }  
}

# **Login Screen:**

This screen is inflated after the splash screen. The user can login into the application, register for the app, can retrieve password if the user has forgotten password. The following functionalities implemented in this screen:

* Login
* Register
* Google login
* Forgot password



## 2.1 Layout File:

We have used constraint layout with three buttons, three text views, two image view and background image. We have used constraint Layout.

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.constraint.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:orientation="vertical" android:layout\_width="match\_parent"  
 android:background="@drawable/mainscreen"  
 android:backgroundTintMode="multiply"  
 android:backgroundTint="#EDE7F6"  
 android:layout\_height="match\_parent"**>

//code for buttons and textView

</**android.support.constraint.ConstraintLayout**>

We have used the **Google Signin** feature:

<**com.google.android.gms.common.SignInButton  
 android:id="@+id/sign\_in\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="8dp"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/register"  
 app:layout\_constraintVertical\_bias="0.733"  
 android:layout\_marginRight="8dp"** app:layout\_constraintRight\_toRightOf="parent"  
 **android:layout\_marginLeft="8dp"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginEnd="8dp"** />

## 2.2 Code Snippets for Login Screen:

* Enabling Google SignIn:

GoogleSignInOptions gso = **new** GoogleSignInOptions.Builder(GoogleSignInOptions.***DEFAULT\_SIGN\_IN***)  
 .requestIdToken(getString(R.string.***default\_web\_client\_id***))  
 .requestEmail()  
 .build();  
**mGoogleSignInClient** = GoogleSignIn.*getClient*(getActivity(), gso);  
view.findViewById(R.id.***sign\_in\_button***).setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 signIn();  
 }  
});  
**return** view;

* Code snippet for forgot password:

When the forgot password is clicked the user gets a mail to reset his/her password.

**forgotPassword**.setOnClickListener(**new** View.OnClickListener()  
{  
 @Override  
 **public void** onClick(View v)  
 {  
 String emailID = **inputEmail**.getText().toString();  
 **if** (TextUtils.*isEmpty*(emailID))  
 {  
 Toast.*makeText*(getContext(), **"Type your email address!"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 **auth**.sendPasswordResetEmail(emailID)  
 .addOnCompleteListener(**new** OnCompleteListener<Void>() {  
 @Override  
 **public void** onComplete(@NonNull Task<Void> task) {  
 **if** (task.isSuccessful()) {  
 Toast.*makeText*(getContext(), **"Mail has been sent regarding password!"**, Toast.***LENGTH\_SHORT***).show();  
 } **else** {  
 Toast.*makeText*(getContext(), **"Email sending failed!"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
 }  
});

* Code snippet for Firebase Authentication (i.e. log in with email and password):

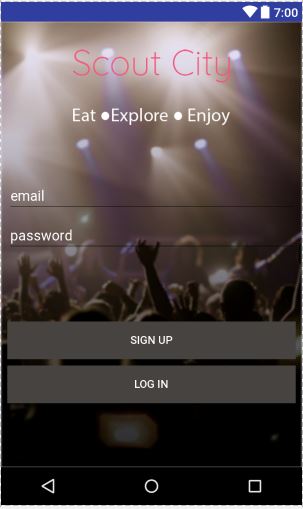
*//Get Firebase instance***auth** = FirebaseAuth.*getInstance*();

*//Get Firebase auth instance***auth** = FirebaseAuth.*getInstance*();

*//authenticate user***auth**.signInWithEmailAndPassword(email , password).addOnCompleteListener(getActivity(), **new** OnCompleteListener<AuthResult >()  
{  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult> task)  
 {  
  
 **if** (!task.isSuccessful())  
 {  
 *// there was an error* **if** (password.length() < 6)  
 {  
 **inputPassword**.setError(**"Minumum password"**);  
 }  
 **else** {  
 Toast.*makeText*(getActivity(), **"Authentication Failed"**, Toast.***LENGTH\_LONG***).show();  
 }  
 }  
 **else** {  
 Intent intent = **new** Intent(getActivity(), Navigation.**class**);  
 startActivity(intent); getActivity().finish();  
 }  
  
 }  
});

# **SignUp Screen:**

This screen is inflated when the user wants to register for the app. The user can register by giving a valid email id and password. The password should me minimum of 6 characters. Once the user has created the data will be stored in the firebase. Then the user can login with his new credentials.



## 3.1 Layout File:

In this layout we have used Constraint Layout. It contains login button, signup button, a field to enter the email id, a field to enter password. Once the user enters his valid details he/she can click on the signup and create their account. After that the user can click log in button and go to the login fragment and use the app.

Constraint Layout:

<**android.support.constraint.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:orientation="vertical" android:layout\_width="match\_parent"  
 android:background="@drawable/mainscreen"  
 android:backgroundTintMode="multiply"  
 android:backgroundTint="#EDE7F6"  
 android:layout\_height="match\_parent"**>

//code for buttons and textView

</**android.support.constraint.ConstraintLayout**>

## 3.2 Code Snippet for SignUp Screen:

* Code snippet for onButtonSignUp:

**btnSignUp**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 String email = **inputEmail**.getText().toString().trim();  
 String password = **inputPassword**.getText().toString().trim();  
 **if** (TextUtils.*isEmpty*(email)) {  
 Toast.*makeText*(getContext(), **"Enter email address!"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 **if** (TextUtils.*isEmpty*(password)) {  
 Toast.*makeText*(getContext(), **"Enter password!"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }  
 **if** (password.length() < 6) {  
 Toast.*makeText*(getContext(), **"Password too short , enter minimum 6 characters!"**, Toast.***LENGTH\_SHORT***).show();  
 **return**;  
 }

* Code snippet creating user:

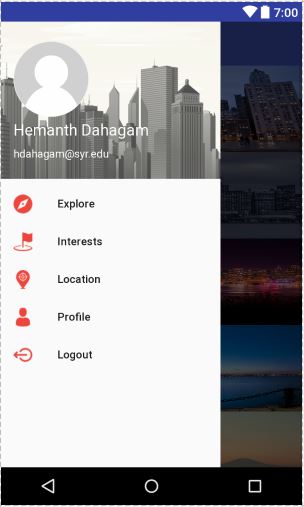
The fire base is authenticated, and the user is created.

*//create user* **auth**.createUserWithEmailAndPassword(email , password).addOnCompleteListener(getActivity(),  
 **new** OnCompleteListener<AuthResult>()  
 {  
 @Override  
 **public void** onComplete(@NonNull Task<AuthResult > task) {  
 Log.*d*(**"createUserWithEmail: "**, **"onComplete:"** + task.isSuccessful());  
 Toast.*makeText*(getContext(), **"createUserWithEmail:onComplete:"** + task.isSuccessful(), Toast.***LENGTH\_LONG***).show();  
  
 **if** (!task.isSuccessful())  
 {  
 Toast.*makeText*(getContext(), **"Authentication failed."** + task.getException(), Toast.***LENGTH\_LONG***).show();  
 }  
 **else** {  
 *// sign up success* **mListener**.onSigninRoutine();  
 }  
 }  
 });  
 }  
});

# **Navigation Activity:**

After the user has successfully logged in then, the user is displayed with navigation drawer and its activity. In this we have options such as explorer, report bugs, location, profile, and logout. The user can directly navigate from this activity to other activity by clicking on any of the options.

It also contains the image view, name, and email id of the user.



## 4.1 Layout File:

We have used Drawer Layout to design the navigation drawer. Inside the navigation drawer we have included app bar navigation and a navigation view.

<**android.support.v4.widget.DrawerLayout 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/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:openDrawer="start"**>  
  
 <**include  
 layout="@layout/app\_bar\_navigation"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"** />  
  
 <**android.support.design.widget.NavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:fitsSystemWindows="true"  
 app:itemIconTint="#ed453b"  
 app:headerLayout="@layout/nav\_header\_navigation"  
 app:menu="@menu/activity\_navigation\_drawer"** />  
  
</**android.support.v4.widget.DrawerLayout**>

## 4.2 Code Snippet for Navigation Drawer:

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.***drawer\_layout***);  
ActionBarDrawerToggle toggle = **new** ActionBarDrawerToggle(  
 **this**, drawer, toolbar, R.string.***navigation\_drawer\_open***, R.string.***navigation\_drawer\_close***);  
drawer.setDrawerListener(toggle);  
toggle.syncState();  
  
NavigationView navigationView = (NavigationView) findViewById(R.id.***nav\_view***);  
navigationView.setNavigationItemSelectedListener(**this**);  
**headerView** = navigationView.getHeaderView(0);

# **Recycler View:**

We are using three recycler views for displaying the recommended, cafes and restaurants data. We are using three different card views to populate it with different JSON data received for each model from the server on the screen. The three recycler views are implemented inside a view pager and tab bar layout for more appealing look. Each recycler view has its own search button, and each card view has a pop up menu which has two options share and interested.

## 5.1 Layout file:

We have implemented three RecyclerViews in Frame Layouts and Linear layout. Inside the layout we are implementing the recyclerview.

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/recyclerbackground"  
 android:id="@+id/cafe\_recycler"**>  
 <**android.support.v7.widget.RecyclerView  
 android:id="@+id/recyclerviewrow2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**/>  
</**LinearLayout**>

## 5.2 Code Snippet for RecyclerView Fragment:

In the recycler fragment the data is received from the Four Square API which is stored in a. The data which is received is given to the recycler adapter. Similar code is written for the remaining two Recycler View fragments, but each fragment uses different API’s to get the data and populate the field in the recycler card view.

* Data given to the POJO (Plain Old Java Object) and is set to the adapter.

Call<FSVenueCafeModel> venueSearch = FSVenueCafeAPI.getService().searchCafe(latlng);

venueSearch.enqueue(new Callback<FSVenueCafeModel>() {

@Override

public void onResponse(Call<FSVenueCafeModel> call, Response<FSVenueCafeModel> response) {

final FSVenueCafeModel list = response.body();

//Log.d("Data",list.getResponse().getVenues().toString());

listData = list.getResponse().getGroups().get(0).getItems();

myAdapter = new CafeRecyclerAdapter(getActivity(),listData);

myRecyclerView.setAdapter(myAdapter);

myAdapter.setOnClickListner(new CafeRecyclerAdapter.OnItemClickListener() {

@Override

public void onItemClick(View v, int position) {

Toast.makeText(getActivity(),"Chicked:"+position,Toast.LENGTH\_SHORT).show();

customOnClickRvListener.onCafeRecycleViewItemClicked(v, list.getResponse().getGroups().get(0).getItems().get(position));

}

* When the onCreate is called in the RecyclerView class the layout manager sets the layout of the recycler view, but the adapter for recycler view is set while fetching data from API.

@Override  
**public** View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstance){  
 **final** View rootView = inflater.inflate(R.layout.***fragment\_cafe\_recycler***,container,**false**);  
  
  
 **customOnClickRvListener** = (CafeRecycleViewListener) rootView.getContext();  
  
 **myRecyclerView** = (RecyclerView) rootView.findViewById(R.id.***recyclerviewrow2***);  
 *//Implement myRecyclerView* **myLayoutManager** = **new** LinearLayoutManager(getActivity());  
 *//Implement myLayoutManager* **myRecyclerView**.setLayoutManager(**myLayoutManager**);  
  
  
 getData();  
  
 **return** rootView;  
}

* Pop up menu for each card view has a share button, following is the code snippet for sharing to various platform and pop up menu.

@Override

public void onOverflowMenuClick(View v, final int position) {

final PopupMenu popup = new PopupMenu(getActivity(), v);

popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {

@Override

public boolean onMenuItemClick(MenuItem item) {

switch (item.getItemId()) {

case R.id.item\_interested:

HashMap map = new HashMap();

//put this map in firebasewith ID

return true;

case R.id.item\_share:

Intent sendIntent = new Intent();

sendIntent.setAction(Intent.ACTION\_SEND);

String textTosend = "Want to go out to this place with me! - "+list.getResponse().getGroups().get(0).getItems().get(position).getVenue().getName();

sendIntent.putExtra(Intent.EXTRA\_TEXT, textTosend);

sendIntent.setType("text/plain");

startActivity(Intent.createChooser(sendIntent, "Share with:"));

return true;

default:

return false;

}

}

});

MenuInflater inflater1 = popup.getMenuInflater();

inflater1.inflate(R.menu.card\_menu, popup.getMenu());

popup.show();

}

});

* Following is the code snippet for implementing search feature for each recycler view inside a view pager. This snippet should be present in each recycler view onCreateOptionsMenu() for having its individual search facility.

@Override

public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {

if (menu.findItem(R.id.icon\_searchButton) == null) {

inflater.inflate(R.menu.menu\_recycler, menu);

}

SearchView search = (SearchView) menu.findItem(R.id.icon\_searchButton).getActionView();

if (search != null) {

search.setOnQueryTextListener(new SearchView.OnQueryTextListener() {

@Override

public boolean onQueryTextSubmit(String query) {

int pos = findFirst(query);

if (pos >= 0)

myRecyclerView.scrollToPosition(pos);

return true;

}

@Override

public boolean onQueryTextChange(String query) {

return true;

}

});

}

super.onCreateOptionsMenu(menu, inflater);

}

## 5.3 Code Snippet for RecyclerView Adapter:

The data which is taken from the POJO class is given to the fragment and the fragment in turns invokes the adapter and the adapter sets the data into the card view. Similar code below is also implemented for the remaining two recyclerviews.

* The data is populated onto the card view.

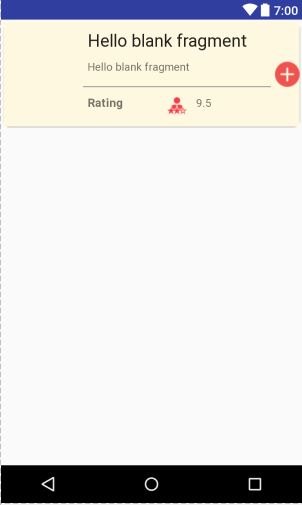
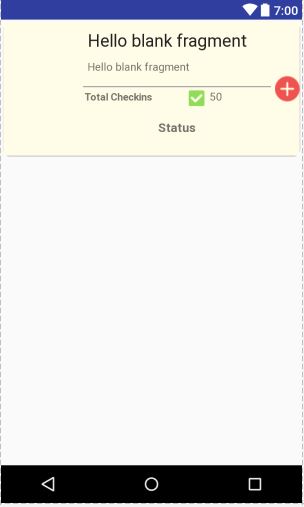
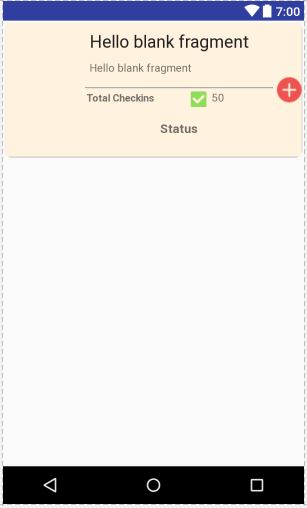
**public class** CafeListViewHolder **extends** RecyclerView.ViewHolder {  
  
 **public** ImageView **place\_image**;  
 **public** TextView **place\_title**;  
 **public** TextView **place\_location**;  
 **public** TextView **place\_rating**;  
 **public** TextView **place\_status**;  
 **public** ImageView **options**;  
  
 CafeListViewHolder(View view) {  
 **super**(view);  
 **place\_image** = (ImageView) view.findViewById(R.id.***rp\_imageView2***);  
 **place\_title** = (TextView) view.findViewById(R.id.***recommendedTitle2***);  
 **place\_location** = (TextView) view.findViewById(R.id.***location2***);  
 **place\_rating** = (TextView) view.findViewById(R.id.***cardrating2***);  
 **options** = (ImageView) view.findViewById(R.id.***rp\_moreOptions2***);  
 **place\_status** = (TextView) view.findViewById(R.id.***statusloc***);  
 view.setOnClickListener(**new** View.OnClickListener(){  
  
 @Override  
 **public void** onClick(View v) {  
 **if**(**myItemClickListener** != **null**)  
 {  
 **myItemClickListener**.onItemClick(v, getAdapterPosition());  
 }  
 }  
 });  
  
  
 **if**(**options** != **null**)  
 {  
 **options**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **if**(**myItemClickListener** != **null**) {  
 **myItemClickListener**.onOverflowMenuClick(v,getAdapterPosition());  
 }  
 }  
 });  
 }  
 }  
  
}

* The image is downloaded in the adapter class from the corresponding API using the POJO class.

**private void** getImage(String venueID, **final** CafeListViewHolder holder){  
 Call<FSVenuePhotoModel> venueSearch = FSVenueImageAPI.*getService*().getImages(venueID);  
  
 venueSearch.enqueue(**new** Callback<FSVenuePhotoModel>(){  
  
 @Override  
 **public void** onResponse(Call<FSVenuePhotoModel> call, Response<FSVenuePhotoModel> response) {  
  
 FSVenuePhotoModel list = response.body();  
 String url = list.getResponse().getPhotos().getItems().get(0).getPrefix().toString()+**"600x800"**+list.getResponse().getPhotos().getItems().get(0).getSuffix().toString();  
 Picasso.*with*(**myContext**).load(url).into(holder.**place\_image**);  
  
 }  
  
 @Override  
 **public void** onFailure(Call<FSVenuePhotoModel> call, Throwable t) {  
  
  
 }  
 });  
  
}

## 5.4 Card View:

We have designed 3 card views for recommended, cafes and restaurants. Each of the card view is loaded into three recycler views.

## 5.3 RecyclerView Animations:

We have implemented the fall down animation in our three recycler views. We need to create two XML files for the animations.

**Layout File:**

<**set xmlns:android="http://schemas.android.com/apk/res/android"  
 android:duration="@integer/anim\_duration\_long"**>  
  
 <**translate  
 android:fromYDelta="-20%"  
 android:toYDelta="0"  
 android:interpolator="@android:anim/decelerate\_interpolator"** />  
  
 <**alpha  
 android:fromAlpha="0"  
 android:toAlpha="1"  
 android:interpolator="@android:anim/decelerate\_interpolator"** />  
  
 <**scale  
 android:fromXScale="105%"  
 android:fromYScale="105%"  
 android:toXScale="100%"  
 android:toYScale="100%"  
 android:pivotX="50%"  
 android:pivotY="50%"  
 android:interpolator="@android:anim/decelerate\_interpolator"** />  
  
</**set**>

*<?***xml version="1.0" encoding="utf-8"***?>*

<**layoutAnimation**

**xmlns:android="http://schemas.android.com/apk/res/android"**

**android:animation="@anim/right"**

**android:delay="15%"**

**android:animationOrder="random"**

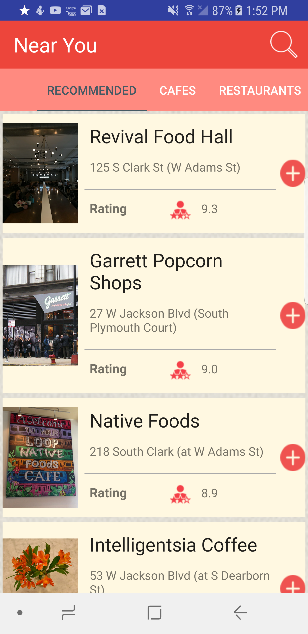
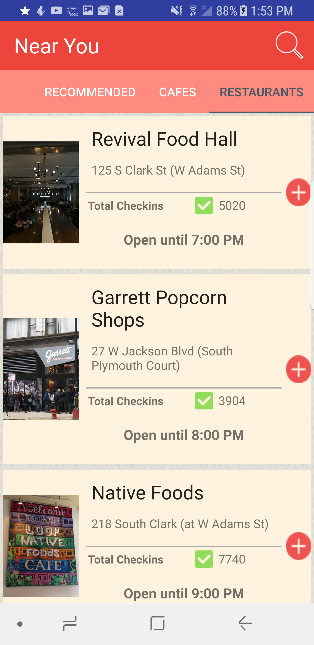
/>

We need to add a single line in the XML code of the recycler view.

**android:layoutAnimation="@anim/item\_animation\_right"**

# **ViewPager Activity:**

The View Pager is a widget for Android that makes possible displaying different views (or fragments) side by side allowing the user to swipe between them. In this application we have implemented three recycler views and this view pager helps us to project these three different views. In the view pager we have tab layout attached which displays the name of the recycler view. There is a tool bar on top of the view pager which is being inflated with search from the corresponding fragments.

## 6.1 Layout File:

We have used linear layout and inside linear layout we have given a frame layout and we also implemented tab layout which is used to display the names of the data present in the recycler view as shown above.

<**LinearLayout 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:orientation="vertical"  
 tools:context="com.syracuse.rameka.scoutapp.ViewPagerRecyclerActivity"**>  
 <**FrameLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/locationDetail"**>  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"** >  
 <**android.support.v7.widget.Toolbar  
 android:id="@+id/recyclerToolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="56dp"  
 android:background="#ed453b"  
 android:elevation="4dp"  
 android:theme="@style/ThemeOverlay.AppCompat.ActionBar"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"**>  
 <**TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_toolbarTitle"  
 android:gravity="center"  
 android:textSize="24sp"  
 android:layout\_marginLeft="10dp"  
 android:textColor="#FFFFFF"  
 android:textAppearance="@style/Base.TextAppearance.AppCompat.Large"** />  
 </**android.support.v7.widget.Toolbar**>  
 <**android.support.design.widget.TabLayout  
 android:layout\_height="50dp"  
 android:layout\_width="match\_parent"  
 android:id="@+id/tabBar"  
 android:layout\_gravity="center"  
 app:tabIndicatorColor="#455A64"  
 app:tabSelectedTextColor="#455A64"  
 app:tabTextColor="#ffffff"  
 app:tabMode="scrollable"  
 app:tabGravity="center"  
 app:paddingEnd="-1dp"  
 app:paddingStart="-1dp"  
 app:tabContentStart="55dp"  
 android:background="#FF8A80"** />  
 <**android.support.v4.view.ViewPager  
 android:layout\_height="match\_parent"  
 android:layout\_width="match\_parent"  
 android:id="@+id/viewpager"  
 tools.context = ".activity\_task2"**>  
 </**android.support.v4.view.ViewPager**>  
 </**LinearLayout**>  
 </**FrameLayout**>  
</**LinearLayout**>

## 6.2 Code Snippet for ViewPager:

* Code Snippet for setting up view pager and tab bar layout and it also includes the setting of flipbook animation for view pager.

FragmentAdapter = new MyFragmentPageAdapter(getSupportFragmentManager(),ViewPagerRecyclerActivity.this);

viewPage = (ViewPager) findViewById(R.id.viewpager);

viewPage.setAdapter(FragmentAdapter);

viewPage.setPageTransformer(false, new FlipPageViewTransformer());

TabLayout tabLayout = (TabLayout)findViewById(R.id.tabBar);

tabLayout.setupWithViewPager(viewPage);

tabLayout.setTabMode(TabLayout.MODE\_SCROLLABLE);

* Code snippet for implementing the tab layout adapter in the view pager.

**public class** MyFragmentPageAdapter **extends** FragmentPagerAdapter {  
 **final int PAGE\_COUNT** = 3;  
 **private** Context **context**;  
 String[] **tabElements** = **new** String[]{**"RECOMMENDED"**,**"CAFES"**,**"RESTAURANTS"**};  
  
 **public** MyFragmentPageAdapter(FragmentManager fm, Context context){  
 **super**(fm);  
 **this**.**context** = context;  
 }  
  
 @Override  
 **public** Fragment getItem(**int** position){  
 Fragment returningFragment;  
 **switch**(position)  
 {  
 **case** 0:  
 returningFragment= RecommendedRecyclerFragment.*newInstance*(**lat**,**lng**,**lname**);  
 **break**;  
 **case** 1:  
 returningFragment =CafeRecyclerFragment.*newInstance*(**lat**,**lng**,**lname**);  
 **break**;  
 **case** 2:  
 returningFragment =RestaurantRecyclerFragment.*newInstance*(**lat**,**lng**,**lname**);  
 **break**;  
 **default**:  
 returningFragment =RecommendedRecyclerFragment.*newInstance*(**lat**,**lng**,**lname**);  
 }  
  
 **return** returningFragment;  
 }  
  
 @Override  
 **public int** getCount() {  
 **return tabElements**.**length**;  
 }  
 @Override  
 **public** CharSequence getPageTitle(**int** position){  
 **return tabElements**[position];  
 }  
}

* Code snippet for set the custom toolbar, which will be updated by the corresponding recycler fragment.

Toolbar myToolbar = (Toolbar) findViewById(R.id.recyclerToolbar);

setSupportActionBar(myToolbar);

mActionBar = getSupportActionBar();

// mActionBar.setDisplayHomeAsUpEnabled(true);

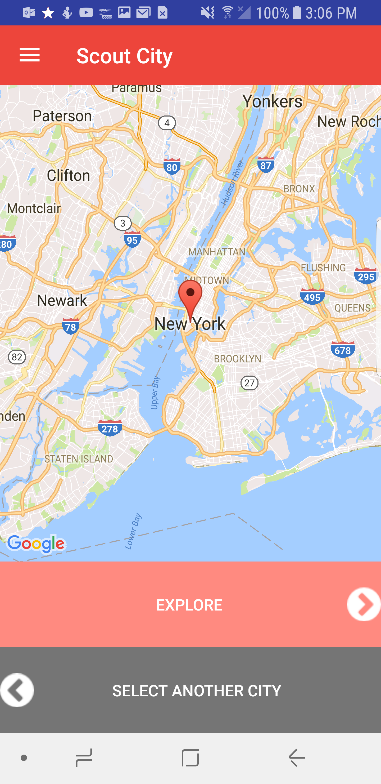
TextView tv\_toolbarTitle = (TextView)myToolbar.findViewById(R.id.tv\_toolbarTitle);

tv\_toolbarTitle.setText("Near You");

mActionBar.setTitle(null);

# **Location and GPS:**

In our application we have used location to recommend the places to user based on his latitudes and longitudes. We have used google API to get the latitude and longitude information and calculated the user location approximately. We have given default options for each city once the user clicks on the place, the latitudes and longitudes of the data is taken and the places which are present near to it are displayed in the recycler view. Once the corresponding place is clicked the location of the user is determined.

## 7.1 Code Snippet for Displaying Location on Map:

We are using the google Places APIs and Maps for calculation of the location. Once the user clicks a location the latitudes and longitudes of that area are taken, and the location is displayed.

SupportMapFragment mapFragment = (SupportMapFragment) getChildFragmentManager()  
 .findFragmentById(R.id.***map4***);  
 mapFragment.getMapAsync(**new** OnMapReadyCallback(){  
  
 @Override  
 **public void** onMapReady(GoogleMap googleMap) {  
 LatLng loc = **new** LatLng(*latitude*, *longitude*);  
 googleMap.addMarker(**new** MarkerOptions().position(loc)  
 .title(*locName*));  
 googleMap.moveCamera(CameraUpdateFactory.*newLatLng*(loc));  
 googleMap.animateCamera( CameraUpdateFactory.*zoomTo*( 10.0f));  
 }  
 });

## 7.2 Code Snippet for Fetching Location through GPS:

User has the privilege to set their current location as a way to search different cafes, recommended places and restaurants. Following the code for enabling GPS if not active and fetching current latitudes and longitudes.

if (ActivityCompat.checkSelfPermission(Navigation.this, Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED && ActivityCompat.checkSelfPermission(Navigation.this, Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

requestPermissions(new String[]{

Manifest.permission.ACCESS\_FINE\_LOCATION, Manifest.permission.ACCESS\_COARSE\_LOCATION, Manifest.permission.INTERNET}, 10);

return;

}

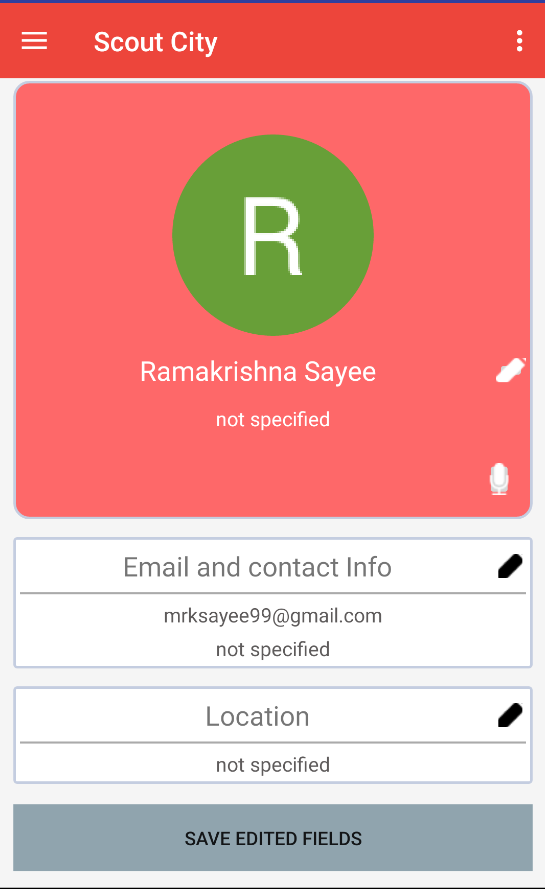
locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, 10000, 30, locationListener);

**Location Manager** fetches the required latitude and longitude when a location is changed, we can set the refresh rate of the location manage to retrieve latitude and longitude periodically. Following is the code for location manager.

**locationManager** = (LocationManager) getSystemService(***LOCATION\_SERVICE***);  
**locationListener** = **new** LocationListener() {  
 @Override  
 **public void** onLocationChanged(Location location) {  
 **latitude** = location.getLatitude();  
 **longitude** = location.getLongitude();  
 Toast.*makeText*(Navigation.**this**,**"Your current location set!"**+location.getLatitude() + **" "** + location.getLongitude(),Toast.***LENGTH\_LONG***).show();  
 inflateFragment(**latitude**,**longitude**,**"Your Location"**);  
  
 **if**(**firebBaseLatLngRequestUpdate** == **true**){  
 FirebaseDatabase.*getInstance*().getReference().child(**profileRef**.*getInstance*().***PROFILE\_PATH***).child(**profileRef**.*getInstance*().**profileInfo**.get(**"email"**).toString().replace(**"."**,**"="**))  
 .child(**"lat"**).setValue(**latitude**);  
 FirebaseDatabase.*getInstance*().getReference().child(**profileRef**.*getInstance*().***PROFILE\_PATH***).child(**profileRef**.*getInstance*().**profileInfo**.get(**"email"**).toString().replace(**"."**,**"="**))  
 .child(**"lng"**).setValue(**longitude**);  
 **firebBaseLatLngRequestUpdate** = **false**;  
 }  
  
 }  
  
 @Override  
 **public void** onStatusChanged(String s, **int** i, Bundle bundle) {  
  
 }  
  
 @Override  
 **public void** onProviderEnabled(String s) {  
  
 }  
  
 @Override  
 **public void** onProviderDisabled(String s) {  
  
 Intent intent = **new** Intent(Settings.***ACTION\_LOCATION\_SOURCE\_SETTINGS***);  
 startActivity(intent);  
  
 }  
};

# **Profile Page:**

Profile page gives the details of the user, user is given the option to update the fields of the profile. The User can update the data only when all the data are specified including image. We have used **Galley** selection when a user clicks the profile image. And **voice to text** for description about the user. User cannot change the mail id. Once save edited fields is clicked the data is uploaded into firebase database.



* Code Snippet for selecting image from **gallery** when a user clicks the image, we have used **Content Provider** for fetching the file URI from the device with **ContentResolver** class (File sharing content provider).

**public** String GetFileExtension(Uri uri) {  
  
 ContentResolver contentResolver = getActivity().getContentResolver();  
  
 MimeTypeMap mimeTypeMap = MimeTypeMap.*getSingleton*();  
  
 *// Returning the file Extension.* **return** mimeTypeMap.getExtensionFromMimeType(contentResolver.getType(uri)) ;  
  
}  
  
*// Creating UploadImageFileToFirebaseStorage method to upload image on storage.***public void** UploadImageFileToFirebaseStorage() {  
  
 *// Checking whether FilePathUri Is empty or not.* **if** (**FilePathUri** != **null**) {  
  
  
 *// Creating second StorageReference.* StorageReference storageReference2nd = **storageReference**.child(**Storage\_Path** + System.*currentTimeMillis*() + **"."** + GetFileExtension(**FilePathUri**));  
  
 *// Adding addOnSuccessListener to second StorageReference.* storageReference2nd.putFile(**FilePathUri**)  
 .addOnSuccessListener(**new** OnSuccessListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  
  
 *// Getting image name from EditText and store into string variable.* String TempImageName = **"Test Image Name"**;  
  
  
 *// Showing toast message after done uploading.* Toast.*makeText*(getContext(), **"Image Uploaded Successfully "**, Toast.***LENGTH\_LONG***).show();  
  
 @SuppressWarnings(**"VisibleForTests"**)  
 ImageUploadInfo imageUploadInfo = **new** ImageUploadInfo(TempImageName, taskSnapshot.getDownloadUrl().toString());  
 *imageURL* = taskSnapshot.getDownloadUrl().toString();  
 *// Getting image upload ID.  
  
 // Adding image upload id s child element into databaseReference.* FirebaseDatabase.*getInstance*().getReference().child(**profile**.*getInstance*().***PROFILE\_PATH***).child(**keyID**).child(**"imagePath"**).setValue(*imageURL*);  
 }  
 })  
 *// If something goes wrong .* .addOnFailureListener(**new** OnFailureListener() {  
 @Override  
 **public void** onFailure(@NonNull Exception exception) {  
  
  
 *// Showing exception erro message.* Toast.*makeText*(getActivity(), exception.getMessage(), Toast.***LENGTH\_LONG***).show();  
 }  
 })  
  
 *// On progress change upload time.* .addOnProgressListener(**new** OnProgressListener<UploadTask.TaskSnapshot>() {  
 @Override  
 **public void** onProgress(UploadTask.TaskSnapshot taskSnapshot) {  
  
  
 }  
 });  
 }  
 **else** {  
  
 Toast.*makeText*(getActivity(), **"Please Select Image or Add Image Name"**, Toast.***LENGTH\_LONG***).show();  
  
 }  
}

Also we have to implement the following code in onActivityResult() of the fragment calling the image change when clicked.

**if** (requestCode == **Image\_Request\_Code** && resultCode == ***RESULT\_OK*** && data != **null** && data.getData() != **null**) {  
  
 **FilePathUri** = data.getData();  
  
 **try** {  
  
 *// Getting selected image into Bitmap.* Bitmap bitmap = MediaStore.Images.Media.*getBitmap*(getActivity().getContentResolver(), **FilePathUri**);  
  
 *// Setting up bitmap selected image into ImageView.* **profileImage**.setImageBitmap(bitmap);  
  
  
 }  
 **catch** (IOException e) {  
  
 e.printStackTrace();  
 }  
}

* For the user description we have used the **voice to text** conversion provided by the GoogleAPI. Following is the code snippet for achieving the result.

Intent intent = **new** Intent(RecognizerIntent.***ACTION\_RECOGNIZE\_SPEECH***);  
intent.putExtra(RecognizerIntent.***EXTRA\_LANGUAGE\_MODEL***,  
 RecognizerIntent.***LANGUAGE\_MODEL\_FREE\_FORM***);  
intent.putExtra(RecognizerIntent.***EXTRA\_LANGUAGE***, Locale.*getDefault*());  
intent.putExtra(RecognizerIntent.***EXTRA\_PROMPT***,  
 **"Describe yourself!"**);  
**try** {  
 startActivityForResult(intent, **REQ\_CODE\_SPEECH\_INPUT**);  
} **catch** (ActivityNotFoundException a) {  
 Toast.*makeText*(getContext(),  
 **"Your device doesnt support voice Recognition!"**,  
 Toast.***LENGTH\_SHORT***).show();  
}

We should also add certain code in the onActivityResult() of the fragment which is as follows,

*//Google voice to text***switch** (requestCode) {  
 **case REQ\_CODE\_SPEECH\_INPUT**: {  
 **if** (resultCode == ***RESULT\_OK*** && **null** != data) {  
  
 ArrayList<String> result = data  
 .getStringArrayListExtra(RecognizerIntent.***EXTRA\_RESULTS***);  
 **descriptionProfile**.setText(result.get(0));  
 }  
 **break**;  
 }  
  
}

# **Advanced Animation:**

We have used animations for Recycler Views, View Pagers and transition between activities. We have achieved animation for View Pager (i.e. Flipbook Style Animation) by defining a class and setting the animation for the view pager in its activity.

* Code snippet for **Flip Book Style Animation** is specified as follows**.** We have defined a Transformer class which implements **ViewPager.PageTransformer.**

**public class** FlipPageViewTransformer **implements** ViewPager.PageTransformer {  
 @Override  
 **public void** transformPage(View page, **float** position) {  
 **float** percentage = 1 - Math.*abs*(position);  
 page.setCameraDistance(12000);  
 setVisibility(page, position);  
 setTranslation(page);  
 setSize(page, position, percentage);  
 setRotation(page, position, percentage);  
 }  
  
 **private void** setVisibility(View page, **float** position) {  
 **if** (position < 0.5 && position > -0.5) {  
 page.setVisibility(View.***VISIBLE***);  
 } **else** {  
 page.setVisibility(View.***INVISIBLE***);  
 }  
 }  
  
 **private void** setTranslation(View page) {  
 ViewPager viewPager = (ViewPager) page.getParent();  
 **int** scroll = viewPager.getScrollX() - page.getLeft();  
 page.setTranslationX(scroll);  
 }  
  
 **private void** setSize(View page, **float** position, **float** percentage) {  
 page.setScaleX((position != 0 && position != 1) ? percentage : 1);  
 page.setScaleY((position != 0 && position != 1) ? percentage : 1);  
 }  
  
 **private void** setRotation(View page, **float** position, **float** percentage) {  
 **if** (position > 0) {  
 page.setRotationY(-180 \* (percentage + 1));  
 } **else** {  
 page.setRotationY(180 \* (percentage + 1));  
 }  
 }  
}

Finally we can set this animation for view pager by adding the following line in onCreate(), “viewPage.setPageTransformer(false, new FlipPageViewTransformer())”.

# **Retrofit Library and Real API Data:**

This is the heart of our entire project. We are using Four Square API to fetch the required data for near-by recommended, restaurants and cafes. We have made model class (POJO) for each type of data we require. In this document the example for one type of data fetched from retrofit API is explained. One of the JSON extracted using retrofit is “Recommended Places” which is explained further in this section.

1. Firstly, we have to define a model class for the type of JSON we are receiving the call from the server. Following is the model class for Recommended Places,

**public class** FSVenueRecommendedModel {  
  
 @SerializedName(**"meta"**)  
 @Expose  
 **private** Meta **meta**;  
 @SerializedName(**"response"**)  
 @Expose  
 **private** Response **response**;  
  
 **public** Meta getMeta() {  
 **return meta**;  
 }  
  
 **public void** setMeta(Meta meta) {  
 **this**.**meta** = meta;  
 }  
  
 **public** Response getResponse() {  
 **return response**;  
 }  
  
 **public void** setResponse(Response response) {  
 **this**.**response** = response;  
 }  
  
}

1. Next, we have to define an API, from which we require to download the JSON in to model objects which can be further used to fetch data. We have used Asynchronous retrofit callbacks to achieve this. Following is the API class with interface to connect to the API for “Recommended Places”

**public class** FSVenueRecommendedAPI {  
  
 **public static final** String ***CLIENT\_ID*** = **"OEP4H5DPW3R43JVQ0NVBU2BGKJEHKLHDB3I4X0TBIH1GYERY"**;  
 **public static final** String ***CLIENT\_SECRECT*** = **"42O1PNTOG1LXDNMTQ3COTWH1A3GBJOKD4LZ4WZSD23PROG5R"**;  
 **public static final** String ***url*** = **"https://api.foursquare.com/v2/"**;  
 *//public static final String ll = "44.3,37.2";* **public static** VenueRecommendedService *venueRecommendedService* = **null**;  
 **public static** VenueRecommendedService getService(){  
 **if**(*venueRecommendedService* ==**null**){  
 Retrofit retrofit = **new** Retrofit.Builder().baseUrl(***url***).addConverterFactory(GsonConverterFactory.*create*())  
 .build();  
 *venueRecommendedService* = retrofit.create(VenueRecommendedService.**class**);  
 }  
 **return** *venueRecommendedService*;  
 }  
  
 **public interface** VenueRecommendedService{  
  
 @GET(**"venues/explore?client\_id=JO0RTJORSERKBRXCG4OC1JUWXMDGHEUPDWRU33ILZLRJQW1Y&client\_secret=B1W5DDTRZB5LPKU0DEFHYLZFSCNB4R44224X2W0PIX1T22O1&v=20161101"**)  
 Call<FSVenueRecommendedModel> searchRecommended(@Query(**"ll"**) String latlng);  
  
 }  
  
  
}

1. Finally, we can get the data using retrofit callback functionality, in the recycler view. “Recommended data” get function inside recycler fragment. The data is extracte om response parameter of onResponse() call back. The code for the following is already explained in Recycler section.

String latlng = *lat*.toString()+**","**+*lng*.toString();  
Call<FSVenueRecommendedModel> venueSearch = FSVenueRecommendedAPI.*getService*().searchRecommended(latlng);  
venueSearch.enqueue(**new** Callback<FSVenueRecommendedModel>() {  
 @Override  
 **public void** onResponse(Call<FSVenueRecommendedModel> call, Response<FSVenueRecommendedModel> response) {  
   
//Do stuff  
 }  
  
 @Override  
 **public void** onFailure(Call<FSVenueRecommendedModel> call, Throwable t) {  
  
 }  
 }  
  
);

}

# **Multi-Window support and App Shortcuts:**

To support multi-window for application we have altered the manifest file for the entire application. The application will be fully functional in multi-window environment. The application parameters in manifest.xml is as follows, we have to set android:resizeableActivity = “true”, android:supportsPictureInPicture="true"

* <**application  
   android:allowBackup="true"  
   android:hardwareAccelerated="false"  
   android:icon="@drawable/appicon1"  
   android:label="@string/app\_name"  
   android:largeHeap="true"  
   android:resizeableActivity="true"  
   android:roundIcon="@drawable/appicon1  
   android:supportsRtl="true"  
   android:theme="@style/AppTheme"**>

// All activities tag and meta information.

</ **application** >

* We have implemented shortcuts in our app by declaring **shortcuts.xml** in **xml resource file.** The xml file is as follows, **shortcuts wont work if android version is less than 7.1.1**.

Have to set the meta-data in manifest.xml

<**meta-data  
 android:name="android.app.shortcuts"  
 android:resource="@xml/shortcuts"** />

<**shortcuts xmlns:android="http://schemas.android.com/apk/res/android"**>  
 <**shortcut  
 android:shortcutId="Recommended"  
 android:icon="@drawable/recommendedshortcut"  
 android:enabled="true"  
 android:shortcutShortLabel="@string/short\_label1"  
 android:shortcutLongLabel="@string/long\_label1"** >  
 <**intent  
 android:action="android.intent.action.VIEW"  
 android:targetPackage="com.syracuse.rameka.scoutapp"  
 android:targetClass="com.syracuse.rameka.scoutapp.Navigation"** />  
 *<!-- If your shortcut is associated with multiple intents, include them  
 here. The last intent in the list determines what the user sees when  
 they launch this shortcut. -->* <**categories android:name="android.shortcut.conversation"** />  
 </**shortcut**>  
  
</**shortcuts**>

# **Menus:**

We have implemented menus in the Navigation Activity, to implement menus, first create and **menu.xml** file in **menus** resource directory. Then inflate the menu in the navigation activity.

* + - * Following is the **menu.xml** file which defines the items to be present in the menu.

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto">

<item

android:id="@+id/previousDefault"

android:title="@string/previous\_Default"

app:showAsAction="never" />

<item

android:id="@+id/cityDefault"

android:title="@string/city\_Default"

app:showAsAction="never" />

</menu>

Then inflate the menu in the activity, in Navigation Activity,

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main\_menu, menu);

return true;

}

Then implement the corresponding methods in onOptionsItemSelected(),

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if(id == R.id.previousDefault){

if(longitude==0.0||latitude==0.0) {

Toast.makeText(Navigation.this,"No previous default location found!",Toast.LENGTH\_LONG).show();

}

else{

FirebaseDatabase.getInstance().getReference().child(profileRef.getInstance().PROFILE\_PATH).child(profileRef.getInstance().profileInfo.get("email").toString().replace(".", "="))

.child("lat").setValue(latitude.toString());

FirebaseDatabase.getInstance().getReference().child(profileRef.getInstance().PROFILE\_PATH).child(profileRef.getInstance().profileInfo.get("email").toString().replace(".", "="))

.child("lng").setValue(longitude.toString());

Toast.makeText(Navigation.this,"success",Toast.LENGTH\_SHORT).show();

}

}

else if(id == R.id.cityDefault){

startActivity(new Intent(Navigation.this,PopUpActivity.class));

}

return super.onOptionsItemSelected(item);

}

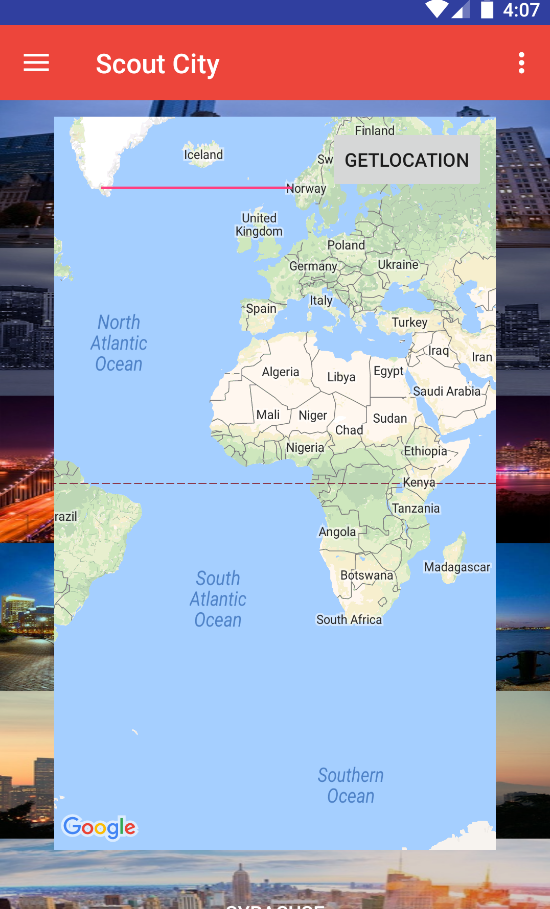
# **Coordinator Layout:**

We have used coordinator layout as a part of navigation drawer. The layout for coordinator layout is as follows,

*<?***xml version="1.0" encoding="utf-8"***?>*<**android.support.design.widget.CoordinatorLayout 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"  
 tools:context="com.syracuse.rameka.scoutapp.ViewPagerRecyclerActivity"**>  
  
 <**android.support.design.widget.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/AppTheme.AppBarOverlay"**>  
  
 <**android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:popupTheme="@style/AppTheme.PopupOverlay"** />  
  
 </**android.support.design.widget.AppBarLayout**>  
  
 <**include layout="@layout/content\_navigation"** />  
  
</**android.support.design.widget.CoordinatorLayout**>

# **Pop Up Activity:**

**Note: We have implemented this feature in our app but forgot to mention in features implemented sheet and demo video. But this functionality is present in the submitted apk.**



* + In our pop up activity, we have we have allowed the user to set any city as default location which will be updated in firebase. The image for the pop up activity is as follows,
  + The layout file for this activity is as follows,

<android.support.constraint.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="#f79a8a"

tools:context="com.syracuse.rameka.scoutapp.PopUpActivity">

<fragment xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:id="@+id/mapPopup"

android:name="com.google.android.gms.maps.SupportMapFragment"

android:layout\_width="0dp"

android:layout\_height="0dp"

tools:context="com.example.mapwithmarker.MapsMarkerActivity"

android:layout\_alignParentBottom="true"

android:layout\_alignParentStart="true"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintHorizontal\_bias="1.0"

app:layout\_constraintVertical\_bias="0.0" />

<EditText

android:id="@+id/cityEdit"

android:layout\_width="150dp"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="8dp"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

android:layout\_marginTop="16dp"

android:layout\_marginStart="8dp"

app:layout\_constraintRight\_toLeftOf="@+id/showInMap"

android:layout\_marginRight="8dp" />

<Button

android:id="@+id/showInMap"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="GetLocation"

android:layout\_marginRight="8dp"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

android:layout\_marginTop="8dp" />

</android.support.constraint.ConstraintLayout>

* + We have created class, called PopUpActivity and write the following code in that activity.

// write this in onCreate()

DisplayMetrics dm = new DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(dm);

int width = dm.widthPixels;

int height = dm.heightPixels;

getWindow().setLayout((int) (width\*.8),(int)(height\*.8));

getLoc = (Button) findViewById(R.id.showInMap);

geteditText = (EditText) findViewById(R.id.cityEdit);

getLoc.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if(geteditText.getText().toString()!=""){

try {

getLLfromCity(geteditText.getText().toString());

} catch (IOException e) {

e.printStackTrace();

}

}

}

});

# **Custom Views:**

**Views** are also responsible for saving their UI state and handling touch events. Developers can also create **custom views** and use them in their application. We have created custom view in the report bugs. Following is the code to add in layout file.

<**com.syracuse.rameka.scoutapp.DoubleTextView**

**android:layout\_width="match\_parent"**

**android:layout\_height="match\_parent"**

**app:leftText="Name:Hemanth Dahagam"**

**app:rightText="Name:Ramakrishna Sayee Meka"**/>

**Code snippets for custom view:**

**public class** DoubleTextView **extends** LinearLayout {  
 LinearLayout **layout** = **null**;

TextView **leftTextView** = **null**;

TextView **rightTextView** = **null**;

Context **mContext** = **null**;

**public** DoubleTextView(Context context) {

**super**(context);  
 **mContext** = context;  
 }

**public** DoubleTextView(Context context, AttributeSet attrs) {

**super**(context, attrs);  
 **mContext** = context;  
 TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.***DoubleText***);  
 String leftText = a.getString(R.styleable.***DoubleText\_leftText***);  
 String rightText = a.getString(R.styleable.***DoubleText\_rightText***);  
 leftText = leftText == **null** ? **""** : leftText;  
 rightText = rightText == **null** ? **""** : rightText;  
 String service = Context.***LAYOUT\_INFLATER\_SERVICE***;  
 LayoutInflater li = (LayoutInflater) getContext().getSystemService(service);  
 **layout** = (LinearLayout) li.inflate(R.layout.***double\_text***, **this**, **true**);  
 **leftTextView** = (TextView) **layout**.findViewById(R.id.***left\_text***);  
 **rightTextView** = (TextView) **layout**.findViewById(R.id.***right\_text***);  
 **leftTextView**.setText(leftText);  
 **rightTextView**.setText(rightText);  
 a.recycle();  
 }

**public** DoubleTextView(Context context, AttributeSet attrs, **int** defStyle) {  
 **super**(context, attrs, defStyle);  
 **mContext** = context;

}

@SuppressWarnings(**"unused"**)  
 **public void** setLeftText(String text) {  
 **leftTextView**.setText(text);  
 }  
 @SuppressWarnings(**"unused"**)  
 **public void** setRightText(String text) {  
 **rightTextView**.setText(text);

}

@SuppressWarnings(**"unused"**)  
 **public** String getLeftText() {  
 **return leftTextView**.getText().toString();  
 }

@SuppressWarnings(**"unused"**)  
 **public** String getRightText() {  
 **return rightTextView**.getText().toString();  
 }  
}