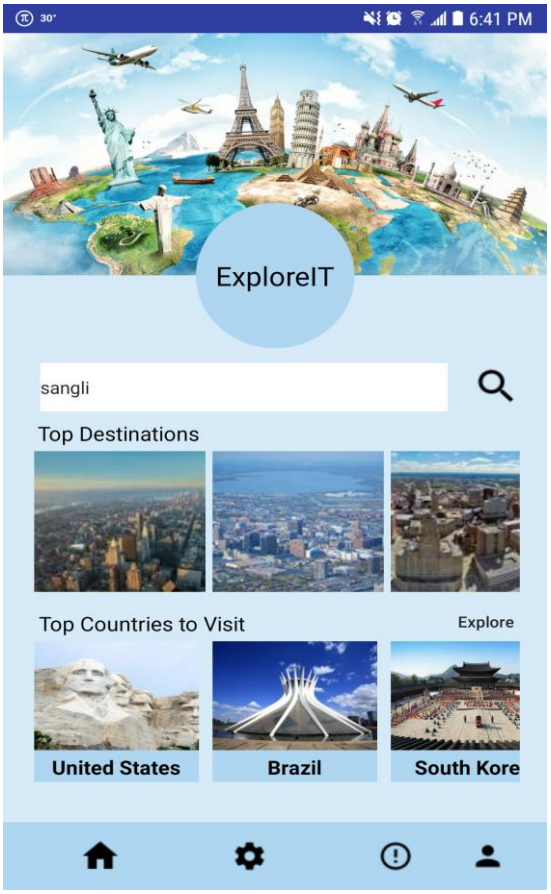


ExploreIt

Android Final Project Report

Team Members

1. Sonal Patil
2. Chetanraj Kadam

Instructor – Prof. Mina Jung

CIS 600 Android Programming

Table of contents

1. Introduction

1.1. App Idea –

Are you an Explorer/ Backpacker/ planning your next vacation?
Welcome to the world of exploration! You are at the right place! This app will take you to any corner of the world you want to explore!
The basic idea of this app is to give our app user all the data he/she can get about any country/state/city/place by searching it on google and then following the connected links on a single screen.

1.2. Features –

1. Travel guide generator
2. Location of country/state/city/place on Map useful for GPS guide nearby restaurants & hotels
3. Weather report
4. Wikipedia, Instagram, Kayak & BBC links
5. Best Clicks
6. Best Youtube videos
7. Top destinations for country/state/city

1.3. Users –

This app gets used by all android users. From people who are totally into travelling to people who plan at least one holiday trip each year.

In short, it helps 2 kinds of people –

- First, those are interested to travel to particular city and wants to know more. [Trip Planning]
- Others who want to plan a trip and are yet to finalize the city/area/state/country and open for many options. [Travel Guide]

1.4. Uses –

1. Users can use this app to plan their upcoming trips with no clue about the destination
2. Users can use this app to get data in well displayed format about any place in the world.
3. Using weather details, user can decide best time to visit according to their preferences.
4. Users can visualize that place by going through best clicks and youtube videos.
5. Users can use this app even when they are travelling. Suppose, an user of this app is visiting India and he/she don't know the best places he can visit in that particular city. In this case, he/she can go inside this app and search for that place and that's it! He can get all the info about that city. How easy life will become by using this!

1.5. Functionality –

Few functionalities to make your way through this app :-

1. You have to create your user account for this app. [Do not worry your data is in right hand]
2. You can also login with your Google/Gmail Account.
3. You have a facility to search for any country/state/city/place in the world and fetch all data together on single screen to decide whether you want to visit them or not.
4. You will get well formatted and extremely useful info as a search result gathered from different resources [e.g. Wikipedia, YouTube, Kayak, Instagram]
5. The flow of our app that takes you on an explore tour -
   a. Select a country from top countries available
   b. Select a state from top states of selected country
   c. Select a city from top cities of selected state
   d. Select a place from top places for selected city.
6. Few other things to remember -
   a. Do not turn off your mobiles' internet connection.
   b. Keep your mobiles' battery level high.

2. Technical details

▪ Navigation Drawer

The navigation drawer is a panel that displays the app's main navigation options and user profile photo, full name, location. We have implemented it from on the right edge of the screen. It is hidden most of the time, but is revealed when the user swipes a finger from the right edge of the screen or, while at the bottom level of the app, the user touches the user icon on bottom toolbar.

Menu item on navigation drawer are home, edit profile, app info, help and log out menu.

```
<group android:checkableBehavior="single">
    <item
        android:id = "@+id/nav_home"
        android:checked="false"
        android:icon="@drawable/ic_about"
        android:title="Home"/>

    <item
        android:id = "@+id/nav_editprofile"
        android:checked="false"
        android:icon="@drawable/ic_editprofile"
        android:title="Edit Profile"/>

    <item
        android:id = "@+id/nav_info"
        android:checked="false"
        android:icon="@drawable/ic_info"
        android:title="App Info"/>

    <item
        android:id = "@+id/nav_help"
        android:checked="false"
        android:icon="@drawable/ic_help"
        android:title="Help"/>
</group>
```

```
drawer = (DrawerLayout)findViewById(R.id.drawer_layout);
NavigationView navigationView = (NavigationView) findViewById(R.id.navView);
if(currentuser != null){
    View hView =  navigationView.getHeaderView(0);
    TextView nav_user = (TextView)hView.findViewById(R.id.user_name);
    nav_user.setText(currentuser.getFname()+ " " + currentuser.getLname());
    TextView nav_location = (TextView)hView.findViewById(R.id.user_location);
    nav_location.setText(currentuser.getLocation());
    ImageView user_image = (ImageView) hView.findViewById(R.id.profileImg);
    user_image.setImageBitmap(currentuser.getImage1());
}

navigationView.setNavigationItemSelectedListener(this);

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,drawer,bottomToolbar,"Open Drawer" , "Close Drawer");
toggle.setDrawerIndicatorEnabled(false);
toggle.setHomeAsUpIndicator(R.drawable.ic_person);
toggle.syncState();

bottomToolbar.setNavigationOnClickListener((OnClickListener) (v) -> {
        if (drawer.isDrawerOpen(Gravity.RIGHT)) {
            drawer.closeDrawer(Gravity.RIGHT);
        } else {
            drawer.openDrawer(Gravity.RIGHT);
        }
});
```

Here first drawer and navigation view objects is attached to drawer layout and nav view layout respectively. Navigation view object will be attached to its listener to handle on item selected listener.

Toggle is used to add listener to the drawer and to add the hamburger menu icon. We change gravity to RIGHT to open it from right side.

```
public boolean onNavigationItemSelected(@NonNull MenuItem item) {

    int id = item.getItemId();
    Intent intent;

    switch(id)
    {
        case R.id.nav_editprofile:
            intent = new Intent(HomeScreen.this,EditProfile.class);
            startActivity(intent);
            break;
        case R.id.nav_home:
            intent = new Intent(HomeScreen.this,HomeScreen.class);
            startActivity(intent);
            break;
        case R.id.nav_info:
            intent = new Intent(HomeScreen.this,InfoActivity.class);
            startActivity(intent);
            break;
        case R.id.nav_help:
            intent = new Intent(HomeScreen.this,HelpActivity.class);
            startActivity(intent);
            break;
        case R.id.nav_signout:
            mAuth.signOut();
            finish();
            intent = new Intent(HomeScreen.this,MainActivity.class);
            startActivity(intent);
            Toast.makeText(HomeScreen.this,"User Successfully Loged out!",Toast.LENGTH_SHORT).show();
            break;
```

Then navigation item selected listener is used to handle item click events.

▪ Recycler View

We have implemented mainly 2 types of recycler views in our project.

First one is to display top destinations, top countries/states/cities and best clicks. This we implemented by using horizontal feature of recycler views and card view.

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/recyclerview2"
    android:layout_width="370dp"
    android:layout_height="120dp"
    android:layout_marginEnd="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginStart="8dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView4">
```

First we created the RecyclerView and RecyclerView given HORIZONTAL attribute to display it horizontally.

```
recyclerView1 = (RecyclerView) findViewById(R.id.recyclerview1);
recyclerView1.setHasFixedSize(true);

layoutManager = new LinearLayoutManager(this, LinearLayoutManager.HORIZONTAL, false);
recyclerView1.setLayoutManager(layoutManager);
```

Then we attached an adapter extending RecyclerView.Adapter class

```
recyclerViewAdapter = new DestinationRecyclerViewAdapter(CountryActivity.this, destinations);
    recyclerView1.setAdapter(recyclerViewAdapter);
```

Adapter class' definition using its view holder

```
DestinationRecyclerViewAdapter.MyViewHolder holder;
public DestinationRecyclerViewAdapter(Context myContext,ArrayList<TopDestinations> myDataset)
{
    this.context = myContext;
    this.destinations_names = myDataset;
}

@Override
public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

    View v = LayoutInflater.from(parent.getContext())
            .inflate(R.layout.country_cardview, parent, false);

    MyViewHolder vh = new MyViewHolder(v);
    holder = vh;
    return vh;
}
```

We provide animation to adapter and each item.

```
public void adapterAnimation()    {

    AlphaInAnimationAdapter alphaAdapter1 = new AlphaInAnimationAdapter(rva);
    ScaleInAnimationAdapter scaleAdapter1 = new ScaleInAnimationAdapter(alphaAdapter1);

    recyclerView2.setAdapter(scaleAdapter1);
}

private void itemAnimation()    {
    SlideInLeftAnimator animator = new SlideInLeftAnimator();
    animator.setInterpolator(new OvershootInterpolator());

    animator.setAddDuration(500);
    animator.setRemoveDuration(500);

    //recyclerView1.setItemAnimator(animator);
    recyclerView2.setItemAnimator(animator);
}
```

Second type of recycler view is using grid layout to display more countries after clicking explore button.

```
recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
GridLayoutManager gridLayoutManager = new GridLayoutManager(getApplicationContext(),2);
recyclerView.setLayoutManager(gridLayoutManager);
ca = new CustomAdapter(ViewmoreCountries.this, c_info);
recyclerView.setAdapter(ca); // set the Adapter to RecyclerView
```

- ## View Pager -

The ViewPager is the widget that allows the user to swipe left or right to see an entirely new screen. ViewPager is nothing but combination of Fragments and PageAdapters. In this case, Fragments are pages. Each screen that the ViewPager allows the user to scroll to is really a Fragment and Page Adapter's job is to supply Fragments (instead of views) to UI for drawing.

```
myViewPagerAdapter = new MyViewPagerAdapter();
viewPager.setAdapter(myViewPagerAdapter);
viewPager.addOnPageChangeListener(viewPagerPageChangeListener);

btnSkip.setOnClickListener((v) → { launchHomeScreen(); });

btnNext.setOnClickListener((v) → {
        // checking for last page
        // if last page home screen will be launched
        int current = getItem( i +1);
        if (current < layouts.length) {
            // move to next screen
            viewPager.setCurrentItem(current);
        } else {
            launchHomeScreen();
        }
});
```

The above code snippet shows the declaration and initialization of the viewpager and viewpageadapter. Here, the swipe left and right occurs on button click of skip and next.

```
public class MyViewPagerAdapter extends PagerAdapter {
    private LayoutInflater layoutInflater;

    public MyViewPagerAdapter() {
    }

    @Override
    public Object instantiateItem(ViewGroup container, int position) {
        layoutInflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View view = layoutInflater.inflate(layouts[position], container,  attachToRoot: false);
        container.addView(view);

        return view;
    }

    @Override
    public int getCount() { return layouts.length; }

    @Override
    public boolean isViewFromObject(View view, Object obj) { return view == obj; }


    @Override
    public void destroyItem(ViewGroup container, int position, Object object) {
        View view = (View) object;
        container.removeView(view);
    }
}
```

The above code snippets is for viewpageadapter class where we have are inflating the layout on object initialization of the viewpageadapter just like doing some process on the constructor of the class.

- ## Toolbar and Menu

We implemented toolbar at bottom to navigate through all pages with 4 menus for home, app info, app help and user.

```
bottomToolbar = (Toolbar) findViewById(R.id.toolbar_bottom);
bottomToolbar.inflateMenu(R.menu.bottombar_menu);
```

We have attached onClicklistener to handle on click events from toolbar menus.

```
bottomToolbar.setOnMenuItemClickListener((item) → {

        int id = item.getItemId();
        Intent intent;

        switch (id)
        {
            case R.id.home:
                intent = new Intent(CountryActivity.this,HomeScreen.class);
                startActivity(intent);
                return true;

            case R.id.settings:
                intent = new Intent(CountryActivity.this,HelpActivity.class);
                startActivity(intent);
                return true;

            case R.id.help:
                intent = new Intent(CountryActivity.this,InfoActivity.class);
                startActivity(intent);
                return true;
```

- ## Coordinator Layout –

  It is super powered frame layout. We used coordinator layout as top level décor in our signup page. The below code is backend side java code where we are replacing the activity with the signup fragment. Here, we used coordinator layout with collapsing AppBar.

```
public class CoordinatorActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_coordinator);
        if(savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.container, SignupFragment.newInstance(0)).commit();
        }
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayShowTitleEnabled(false);

        TextView mTitle = (TextView) toolbar.findViewById(R.id.toolbar_title);
        mTitle.setText("Create User Account");

    }
```

The below code is XML layout resource file which gives us an idea of how the coordinator layout is designed and how it'll look in the app.

When the user goes to signup page for creating his/her account, the Appbar will get collapsed as scroll down for fill more fields. To see the collapsing behavior of the appbar you has to have this property in your coordinator layout file.

```
app:layout_scrollFlags="scroll|enterAlways"
```

```
<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">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light">
            <TextView
                android:id="@+id/toolbar_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Toolbar Title"
                android:textColor="@android:color/white"
                style="@style/TextAppearance.AppCompat.Widget.ActionBar.Title"
                android:layout_gravity="center"
                />
        </android.support.v7.widget.Toolbar>
    </android.support.design.widget.AppBarLayout>
```

▪ Multiple Fragments and Activities –

A **Fragment** is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity.**

Following are important points about fragment –

- A fragment has its own layout and its own behavior with its own life cycle callbacks.
- You can add or remove fragments in an activity while the activity is running.
- You can combine multiple fragments in a single activity to build a multi-pane UI.
- A fragment can be used in multiple activities.
- Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
- A fragment can implement a behavior that has no user interface component.
- Fragments were added to the Android API in Honeycomb version of Android which API version 11.
- Fragment creation –

  We created our fragments by extending fragment class. Sample one fragment creation code is give below. It shows hoe we created the fragments and the inflation of that fragment on it's onCreateView() method which will return us the view of fragment and using which we can access the buttons or textviews we have included on the fragment.

```
public static CoverPageFragment newInstance(int position) {
    CoverPageFragment fragment = new CoverPageFragment();
    return fragment;
}
public CoverPageFragment() {
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    final View rootView = inflater.inflate(R.layout.fragment_coverpage, container,  attachToRoot: false);
    userName = (EditText)rootView.findViewById(R.id.email);
```

Fragment replacement with the activity –

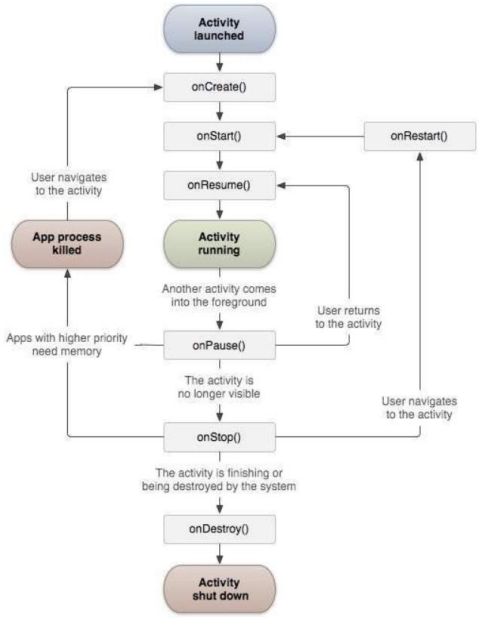
After creating fragment, other thing is to load that fragment in the activity. For that we just have get the supported fragment from fragment manager and then replace the container of that activity with the new instance created foe a specific fragment. This will get you a fragment inside an activity.

```java
public class MainActivity extends AppCompatActivity {

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

        getSupportFragmentManager().beginTransaction()
                .replace(R.id.container, CoverPageFragment.newInstance(0))
                .commit();
    }
```

Activities –

An activity represents a single screen with a user interface just like window or frame of Java.Android activity is the subclass of ContextThemeWrapper class.

Any activity in android works in the following way by going through the cycle.



An application can have one or more activities without any restrictions. Every activity you define for your application must be declared in your *AndroidManifest.xml* file and the main activity for your app must be declared in the manifest with an <intent-filter> that includes the MAIN action and LAUNCHER category

Activity Creation –

We create the activity by extending AppComatActivity and onCreate() method of the activity we have set the content view for that activity by connecting it to a layout XML file.

```java
public class InfoActivity extends AppCompatActivity {

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

Shared element transition between two activities –

When a user clicks on a button or any view on the layout the next activity will have an appealing animation effect with that element shared for that transition.

We have achieved this animation by giving the shared element same transition name in the layout files like as shown below –

1. Transition name given for a button in one activity -

```
<Button
    android:id="@+id/explore"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="2dp"
    android:layout_marginLeft="10dp"
    android:textAllCaps="false"
    android:layout_marginRight="8dp"
    android:layout_marginTop="0dp"
    android:background="@android:color/transparent"
    android:text="    Explore"
    android:transitionName="transitionAnimation"
    app:layout_constraintBottom_toTopOf="@+id/textView3"
    app:layout_constraintHorizontal_bias="1"
    app:layout_constraintLeft_toRightOf="@+id/textView4"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/recyclerview1"
    app:layout_constraintVertical_bias="0.02" />
```

2. Transition name given for a layout in another activity -

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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.exploreit8.sonalpatil.welcomeexploreit.ViewmoreCountries"
    android:background="#D6EAF8"
    android:transitionName="activityAnimation">


    <android.support.v7.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#D6EAF8"
        android:layout_marginTop="25dp"/>

</RelativeLayout>
```

3. For this transition we have to define a style element in styles/xml

```
<item name="android:windowContentTransitions">true</item>
```

4. Now applying the transition animation on button click –

```
explore = (Button)findViewById(R.id.explore);
explore.setOnClickListener((v) → {
    Intent intent = new Intent( packageContext: CountryActivity.this, ViewmoreStates.class);
    intent.putExtra( name: "id",c_id);
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        ActivityOptionsCompat options = ActivityOptionsCompat.makeSceneTransitionAnimation( activity: CountryActivity.this, v,
                sharedElementName: "transitionAnimation");
        startActivity(intent, options.toBundle());
    }
    else{
        startActivity(intent);
    }
});
```

User/App State (Handling Runtime Changes) –

If your user is doing the login process and he/she is typing something in the edittext view and suddenly the orientation of the phone changes, this change must not change the behavior of your app that is what user/app state is. This means if the orientation changes from portrait to horizontal and there is some text there in the text box then user/app state will keep the state of the activity/fragment and return user the exactly same view as of portrait with same text in text box and this can be achieved by using the savedInstanceState and onSaveInstanceState() callback method as implemented below.

```
public class MainActivity extends AppCompatActivity {

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

        getSupportFragmentManager().beginTransaction()
                .replace(R.id.container, CoverPageFragment.newInstance(0))
                .commit();
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
    }
```

Gallery, Floating button and Host user data

We first ask for permissions to from user to access gallery and storage.

```
<uses-permission android:name="android.permission.MANAGE_DOCUMENTS" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(this, new String[] { Manifest.permission.READ_EXTERNAL_STORAGE, Manifest.permission.WRITE_EXTERNAL_STORAGE }, 0);
}
```

When permission granted we will perform further action to select image from gallery on click event of button. Here we have used floating action button to perform this action.

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|end"
    android:layout_marginRight="108dp"
    android:layout_marginTop="232dp"
    android:src="@android:drawable/ic_menu_gallery"
    app:layout_behavior="com.example.sonalpatil.welcomeexploreit.ShrinkBehavior"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Intent is created and accessing activity will be started.

```
editProfileImgBtn.setOnClickListener(new Button.OnClickListener(){
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(Intent.ACTION_PICK,
                android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        startActivityForResult(intent, 0);
    }
});
```

On success of this activity we will get bitmap image of user photo which we will set to imageview and after update profile click we will store this as user image.

```
if (resultCode == RESULT_OK){
    Uri targetUri = data.getData();

    try {
        bitmap = BitmapFactory.decodeStream(getContentResolver().openInputStream(targetUri));
        profileImage.setImageBitmap(bitmap);
    }
    catch (FileNotFoundException e) {

        e.printStackTrace();
```

After user click update button we will save user info on firebase

```
ref.child(uid).child("id").setValue(uid);
ref.child(uid).child("fname").setValue(user.getFname());
ref.child(uid).child("lname").setValue(user.getLname());
ref.child(uid).child("gender").setValue(user.getGender());
ref.child(uid).child("location").setValue(user.getLocation());

globalUser.setCurrentuser(user);
```

Multiple Devices/Screens and Orientation changes

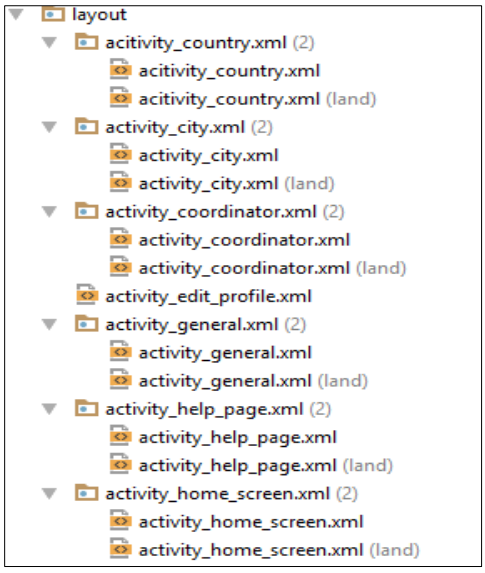
For most of the screens in our app we have used constraint layout.

```
<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:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#D6EAF8"
android:descendantFocusability="beforeDescendants"
android:focusableInTouchMode="true">
```

We have used configChanges property of activity to handle multiple devices, orientations and screen sizes in android manifest.

```
android:theme="@style/AppTheme.NoActionBar"
android:configChanges="keyboardHidden|orientation|screenSize"/>
```

Then we have created landscape variations of each screen layout to handle orientation changes.



Host your data in the cloud –

Hosting data in the cloud means saving the data you are taking from user and keeping it safe and secure for future use. In this app, we are creating a profile for a user. In this we are asking

user about some basic details – name, gender, location, photo. In future, this info can be used to create notifications for user depending upon their interest as male or female or location.
The hosting of this data is done using firebase. We have created a database for this app, to write and retrieve data from firebase.

Firebase Login Authentication

In this app we have provided user to either to create new account and sign in or sign in with Google options. When user created new account we will provide the user access to our app through created username and password.
For this sign in purpose we will first create firebase authentication listener and attach it to handle firebases method on authentication state changed.

```
mAuth = FirebaseAuth.getInstance();
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();
        if(user!=null){
            Intent myIntent = new Intent(SigninActivity.this, HomeScreen.class);
            SigninActivity.this.startActivity(myIntent);
        }else{

        }
    }
};
```

Then we take username and password from user and call firebase method sign In with Email and password. If both are correct it will trigger task is successfully completed and we will navigate user to homescreen.

```
//authenticate user
mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(SigninActivity.this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (!task.isSuccessful()) {
                    // there was an error
                    if (password.length() < 6) {
                        userNameET.setError(getString(R.string.minimum_password));
                    } else {
                        Toast.makeText(SigninActivity.this, getString(R.string.auth_failed), Toast.LENGTH_LONG).show();
                    }
                } else {
                    Intent intent = new Intent(SigninActivity.this, HomeScreen.class);
                    startActivity(intent);
                    finish();
                }
            }
        });
```

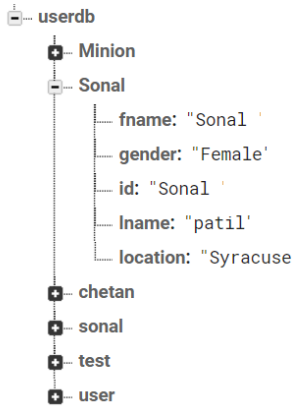
1. Write data on firebase [saving users' data in user database] –

   Retrieve an instance of your database using getInstance() and reference the location you want to write to.

   You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.

```
DatabaseReference ref;
ref = FirebaseDatabase.getInstance().getReference().child("userdb").getRef();
ref.child(uid).child("id").setValue(uid);
ref.child(uid).child("fname").setValue(user.getFname());
ref.child(uid).child("lname").setValue(user.getLname());
ref.child(uid).child("gender").setValue(user.getGender());
ref.child(uid).child("location").setValue(user.getLocation());
```

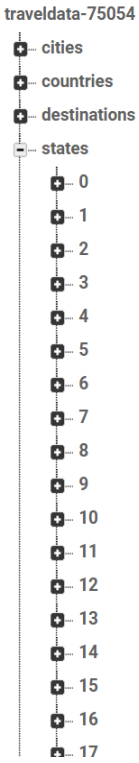
   Our Firebase Database -



2. Retrieving data from firebase [getting top countries, states, cities] –

   To make your app data update in realtime, you should add a ValueEventListener to the reference you just created.

   The onDataChange() method in this class is triggered once when the listener is attached and again every time the data changes, including the children.




```
public void getDataFromServer()     {
    showProgressDialog();
    databaseReference.child("states").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                int i = 0;
                for(DataSnapshot postSnapShot:dataSnapshot.getChildren())
                {
                    String id = postSnapShot.child("country_id").getValue(String.class);
                    if(!id.equals(c_id))
                       continue;
                    String name = postSnapShot.child("name").getValue(String.class);
                    Log.d( tag: "state_names", msg: "***********"+name+"**********");
                    String image = postSnapShot.child("image").getValue(String.class);
                    Log.d( tag: "state_image", msg: "***********"+image+"**********");
                    country_names[i] = name;
                    State country=postSnapShot.getValue(State.class);
                    c_info.add(country);
                    rva.notifyDataSetChanged();
                }
                names = country_names;
            }
            hideProgressDialog();
        }
        @Override
        public void onCancelled(DatabaseError databaseError) { hideProgressDialog(); }
    });
}
```

▪ Google Knowledge Graph API -

To get the information about each country/state/city/place, we have used this API.

The concept behind calling an API is getting the data from the internet. But if we call an API from the main activity i.e. from main UI thread it will take a lot of load and time to perform it.

To avoid this, all the API calls made for the app should be executed using AsyncTask i.e. on the other thread so that main UI won't get affected. The execution of the call is done in doInbackground method of the Async task and returned back the result using PostExecte method of the same API.

```
private class places extends AsyncTask<String, Void, String> {
    private Exception exception;
    protected String doInBackground(String... urls) {
        try {
            HttpTransport httpTransport = new NetHttpTransport();
            HttpRequestFactory requestFactory = httpTransport.createRequestFactory();
            //JSONParser parser = new JSONParser();
            GenericUrl url = new GenericUrl( encodedUrl: "https://kgsearch.googleapis.com/v1/entities:search");
            url.put( fieldName: "query", p_name);
            url.put( fieldName: "limit",  value: "10");
            url.put( fieldName: "indent",  value: "true");
            url.put( fieldName: "key", API_Key);
            HttpRequest request = requestFactory.buildGetRequest(url);
            HttpResponse httpResponse = request.execute();
            venuesList = (ArrayList<PlaceKnwData>) parseknowledgeParse(httpResponse.parseAsString());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return "";
    }
    protected void onPostExecute(String feed) {
        try {
            description.setText(venuesList.get(0).getoverview());
            wiki_link = venuesList.get(0).getwiki();
            LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams( width: 1500,  height: 1000);
            image.setScaleType(ImageView.ScaleType.FIT_XY);//setLayoutParams(layoutParams);
            Picasso.with(context).load(venuesList.get(0).getimage()).into(image);
        }catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

The data you get from the internet is not directly can get used because when we make an API call, we get a HTTP response as a string in return which is unreadable for us.

So, the other thing we need for completing the API call and get the required data is JsonParsing. Jsonpasing technique takes an HTTP response string as a input and then it iterated again an again until matching it done and it';; return back the desired data for the user and set it into views' or do something else.

```
private static ArrayList<PlaceKnwData> parseknowledgeParse(final String response) {

    ArrayList<PlaceKnwData> temp = new ArrayList<>();
    try {

        // make an jsonObject in order to parse the response
        JSONObject jsonObject = new JSONObject(response);

        // make an jsonObject in order to parse the response
        if (jsonObject.has( name: "itemListElement")) {

            JSONArray jsonArray = jsonObject.getJSONArray( name: "itemListElement");
            PlaceKnwData poi = new PlaceKnwData();
            if (jsonArray.getJSONObject( index: 0).getJSONObject("result").has( name: "name")) {
                poi.setname(jsonArray.getJSONObject( index: 0).getJSONObject("result").optString( name: "name"));

                if (jsonArray.getJSONObject( index: 0).getJSONObject("result").has( name: "image")) {
                    if (jsonArray.getJSONObject( index: 0).getJSONObject("result").getJSONObject("image").has( name: "contentUrl")) {
                        poi.setimage(jsonArray.getJSONObject( index: 0).getJSONObject("result").getJSONObject("image").getString( name: "contentUrl"));
                    }
                } else {
                    poi.setimage(" ");
                }
                if (jsonArray.getJSONObject( index: 0).getJSONObject("result").has( name: "detailedDescription")) {
                    //JSONArray typesArray = jsonArray.getJSONObject(i).getJSONObject("result").getJSONArray("detailedDescription");
                    poi.setoverview(jsonArray.getJSONObject( index: 0).getJSONObject("result").getJSONObject("detailedDescription").getString( name: "ar
                    poi.setwiki(jsonArray.getJSONObject( index: 0).getJSONObject("result").getJSONObject("detailedDescription").getString( name: "url"))
                }
            }
            temp.add(poi);
        }
    } catch (Exception e) {
        e.printStackTrace();
        return new ArrayList<PlaceKnwData>();
    }
    return temp;
```

Also, the calling for async task from on create of activity or fragment is done using

new asynctask_name.execute( )

▪ Google Places API data

We have used google places API to get data about top destinations to visit in particular country/state/city. By calling this API it will return array of fixed length Jason data which we parse to get destination name and photo reference of that particular destination. This photo reference is then used to download photo of that destination.

We have used AsyncTask to download all this data in background without conflicting with main thread.

API is called by using HTTP request and returned data is stored in array list which is then send for parsing.

```
ArrayList<TopDestinations> getDestinations = new ArrayList<~>();
protected String doInBackground(String... urls) {
    try {
        //HttpTransport httpTransport = new NetHttpTransport();
        //HttpRequestFactory requestFactory = httpTransport.createRequestFactory();

        HttpTransport HTTP_TRANSPORT = new NetHttpTransport();
        HttpRequestFactory httpRequestFactory = createRequestFactory(HTTP_TRANSPORT);

        GenericUrl url = new GenericUrl("https://maps.googleapis.com/maps/api/place/textsearch/json?");
        url.put("query", "top destinations in "+c_name);
        url.put("dest_mid", "/m/02_286");
        url.put("key", API_Key);
        HttpRequest request = httpRequestFactory.buildGetRequest(url);
        HttpResponse httpResponse = request.execute();
        getDestinations = (ArrayList<TopDestinations>) googleparse(httpResponse.parseAsString());
```

```
rivate static ArrayList<TopDestinations> googleparse(final String response) {

  ArrayList<TopDestinations> temp = new ArrayList<~>();
  try {

      // make an jsonObject in order to parse the response
      JSONObject jsonObject = new JSONObject(response);

      // make an jsonObject in order to parse the response
      if (jsonObject.has("results")) {

          JSONArray jsonArray = jsonObject.getJSONArray("results");

          for (int i = 0; i < 6; i++) {
              TopDestinations poi = new TopDestinations();
              String ph_url = "https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&key="+API_Key;
              if (jsonArray.getJSONObject(i).has("name")) {

                  poi.setName(jsonArray.getJSONObject(i).optString("name"));

                  if (jsonArray.getJSONObject(i).has("photos") && jsonArray.getJSONObject(i).getJSONArray("photos").length() > 0) {

                      JSONArray photos = jsonArray.getJSONObject(i).getJSONArray("photos");
                      JSONObject p = photos.getJSONObject(0);
                      String photoreference = p.getString("photo_reference");
                      ph_url = ph_url + "&photoreference=" + photoreference;
                      poi.setImageurl(ph_url);
```

Once parsing is done the data will be saved in array list of destinations object model class and then this array list is attached to horizontal recyclerview to display top destinations.

- **YouTube –**

  For loading the youtube videos for specific place related to tourism we have used, Google YouTube Data API v3.

```
public static final String key = "AIzaSyDbguOHCXo7WF5jMeZufd44Qw3uCNxaBmI";

public PlaceVideoData(Context context){
    youtube = new YouTube.Builder(new NetHttpTransport(), new JacksonFactory(), new HttpRequestInitializer() {
        @Override
        public void initialize(HttpRequest hr) throws IOException {}
    }).setApplicationName("ExploreIt").build();
    try{
        query = youtube.search().list( part: "id,snippet");
        query.setKey(key);
        query.setType("video");
        query.setFields("items(id/videoId,snippet/title,snippet/description,snippet/thumbnails/default/url)");
    }catch(IOException e){
        Log.d( tag: "YC",  msg: "Could not initialize: "+e);
    }
}
public List<VideoData> search(String keywords){
    query.setQ(keywords);
    try{
        SearchListResponse response = query.execute();
        Log.d( tag: "response", msg: "*****" + response.toString());
        List<SearchResult> results = response.getItems();
        List<VideoData> items = new ArrayList<~>();
        for(SearchResult result:results){
            VideoData item = new VideoData();
            item.setTitle(result.getSnippet().getTitle());
            Log.d( tag: "title", msg: "**********"+result.getSnippet().getTitle());
            item.setDescription(result.getSnippet().getDescription());
            item.setThumbnailURL(result.getSnippet().getThumbnails().getDefault().getUrl());
            item.setId(result.getId().getVideoId());
            items.add(item);
        }
        return items;
    }catch(IOException e){
        Log.d( tag: "YC",  msg: "Could not search: "+e);
```

For loading the youtube player we need to create a activity with youtube player view. By doing this the youtube video will get loaded into that view.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#000">

    <com.google.android.youtube.player.YouTubePlayerView
        android:id="@+id/player_view"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_marginTop="150dp"
        />

</LinearLayout>
```

After creating the activity for youtube player, we need to do some backend tickling to make the youtube API work. This youtube player activity we created must extend youtubebaseactivity instead of what we normally do for an activity [i.e. by extending appcompatactivity]

```
public class PlayerActivity extends YouTubeBaseActivity implements YouTubePlayer.OnInitializedListener {

    private YouTubePlayerView playerView;

    @Override
    protected void onCreate(Bundle bundle) {
        super.onCreate(bundle);

        setContentView(R.layout.activity_player);

        playerView = (YouTubePlayerView)findViewById(R.id.player_view);
        playerView.initialize(PlaceVideoData.key, onInitializedListener: this);
    }

    @Override
    public void onInitializationFailure(YouTubePlayer.Provider provider,
                                        YouTubeInitializationResult result) {
        Toast.makeText( context: this, text: "failed", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onInitializationSuccess(YouTubePlayer.Provider provider, YouTubePlayer player,
                                        boolean restored) {
        if(!restored){
            player.cueVideo(getIntent().getStringExtra( name: "VIDEO_ID"));
        }
    }
}
```

- ## Weather API –

The API used to get the weather data is OpenWeatherMap. The concept behind calling an API is getting the data from the internet. But if we call an API from the main activity i.e. from main UI thread it will take a lot of load and time to perform it. To avoid this, all the API calls made for the app should be executed using AsyncTask i.e. on the other thread so that main UI won't get affected. The execution of the call is done in doInbackground method of the Async task and returned back the result using PostExecte method of the same API.

We used HttpURLConnection class to make the remote request. The OpenWeatherMap API expects the API key in an HTTP header named x-api-key. This is specified in our request using the setRequestProperty method.
We use a BufferedReader to read the API's response into a StringBuffer. When we have the complete response, we convert it to a JSONObject object.

```
private static final String OPEN_WEATHER_MAP_URL
="http://api.openweathermap.org/data/2.5/weather?q=%s&units=metric";
```

This is the API url which gets loaded or get called when we ask asynctask to do and the data gets stored in some views or arraylist.

```
public static class placeIdTask extends AsyncTask<String, Void, JSONObject> {
    public AsyncResponse delegate = null;//Call back interface
    public placeIdTask(AsyncResponse asyncResponse) {
        delegate = asyncResponse;//Assigning call back interfacethrough construotor
    }
    @Override
    protected JSONObject doInBackground(String... params) {
        JSONObject jsonWeather = null;
        try {
            jsonWeather = getWeatherJSON(params[0]);//(params[0], params[1]);
        } catch (Exception e) {
            Log.d( tag: "Error",  msg: "Cannot process JSON results", e);
        }
        return jsonWeather;
    }
    @Override
    protected void onPostExecute(JSONObject json) {
        try {
            if(json != null){
                JSONObject details = json.getJSONArray( name: "weather").getJSONObject( index: 0);
                JSONObject main = json.getJSONObject("main");
                DateFormat df = DateFormat.getDateTimeInstance();
                String city = json.getString( name: "name").toUpperCase(Locale.US) + ", " + json.getJSONObject("sys").getString( name: "country");
                String description = details.getString( name: "description");
                String temperature = String.format("%.2f", main.getDouble( name: "temp"))+ "°C";
                String humidity = main.getString( name: "humidity") + "%";
                String pressure = main.getString( name: "pressure") + " hPa";
                String updatedOn = df.format(new Date(json.getLong( name: "dt")*1000));
                String iconText = setWeatherIcon(details.getInt( name: "id"),
                        sunrise: json.getJSONObject("sys").getLong( name: "sunrise") * 1000,
                        sunset: json.getJSONObject("sys").getLong( name: "sunset") * 1000);
                delegate.processFinish(city, description, temperature, humidity, pressure, updatedOn, iconText,  output8: ""+ (json.getJSONObject("s

            }
        } catch (JSONException e) {      }
    }
```

- ## Maps –

We have implemented the map in a such a way that when user clicks on it, it will take you to map view where you can zoom in/zoom out search for nearby places, hotels, restaurants. As everybody already knows how to use Google maps it becomes easy to use this feature and make it useful. The implementation is done in following way –

```
map.setOnClickListener((v) → {
        new Handler().postDelayed(() → {
                String uri = "http://maps.google.com/?q="+p_name;
                Intent mapIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(uri));
                mapIntent.setPackage("com.google.android.apps.maps");
                startActivity(mapIntent);
        },  delayMillis: 1000);
});
```

- ## Flipbook Style –

Animation makes app look good. We have already implemented basic animations in the recycler view while loading the cardviews with their own data in it.
The flipbook style animation gives you a very different effect. The effect is closely similar to how a book page gets flipped.

We have used '`com.wajahatkarim3.easyflipview.EasyFlipView`' this library to implement this animation. The changes must be done in the layout file and the java class.

Here, we have implemented the flipbook for 2 images.

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000">

    <com.wajahatkarim3.easyflipview.EasyFlipView
        android:id="@+id/easyFlipView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:flipOnTouch = "true"
        app:flipDuration = "500"
        app:flipEnabled = "true">

        <ImageView...>

        <ImageView...>

    </com.wajahatkarim3.easyflipview.EasyFlipView>

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

And then give the image flipbook animation as back and frond page.

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

    final EasyFlipView easyFlipView = (EasyFlipView) findViewById(R.id.easyFlipView);
    easyFlipView.setFlipDuration(1000);
    easyFlipView.setFlipEnabled(true);

    findViewById(R.id.imageView3).setOnClickListener((view) -> {
            easyFlipView.flipTheView();
    });

    findViewById(R.id.imageView4).setOnClickListener((view) -> {
            easyFlipView.flipTheView();
    });
}
```

▪ Broadcast receivers -

We have used two broadcast receivers to implement connectivity change event and battery low event.

As our app is heavily based on usage of internet, user should be always connected to internet. So whenever connectivity is lost we are giving toast notification to user that internet connection is lost and you should connect it.

To implement this we need permissions in manifest file

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```

Then we registered network receiver to receive connectivity changed events.

```
private BroadcastReceiver networkReceiver = (context, intent) -> {
        if (intent.getAction().equals(ConnectivityManager.CONNECTIVITY_ACTION)) {
            NetworkInfo networkInfo = intent.getParcelableExtra(ConnectivityManager.EXTRA_NETWORK_INFO);
            if (networkInfo != null && networkInfo.getDetailedState() == NetworkInfo.DetailedState.CONNECTED) {
                Toast.makeText(context, "Connected to Internet", Toast.LENGTH_SHORT).show();
            } else if (networkInfo != null && networkInfo.getDetailedState() == NetworkInfo.DetailedState.DISCONNECTED) {
                Toast.makeText(context, "Disconnected from Internet", Toast.LENGTH_LONG).show();
            }
        }
};
```

Second broadcast receiver is for battery low event which is triggered when users phone battery is very low. So this will notify user that battery is currently low and need to charge battery soon.

For this we have included intent filter in manifest file to get broadcasted events about battery low from system.

```
<receiver android:name=".MyReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BATTERY_LOW"/>
    </intent-filter>
</receiver>
```

We have included My Receiver class to handle that intent when received.

```
public class MyReceiver extends BroadcastReceiver {

    public MyReceiver() {
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "Battery is getting Low", Toast.LENGTH_LONG).show();
    }
}
```

▪ Service -

We have used AsyncTask to handle background tasks such as calling Google API, retrieve results, parse them and attach it to recycler view on post execute.This is helped to minimize load on main thread and do such tasks in background.

We have called this Async Task from onCreate() method of activity. This will start this async task.

```
new destination().execute();
new country().execute("hi");
```

For this we create class destination which extends Async Task and in doInbackground we are call Google API.

```
private class destination extends AsyncTask<String, Void, String> {

    private Exception exception;
    ArrayList<TopDestinations> getDestinations = new ArrayList<~>();
    protected String doInBackground(String... urls) {
        try {
            //HttpTransport httpTransport = new NetHttpTransport();
            //HttpRequestFactory requestFactory = httpTransport.createRequestFactory();

            HttpTransport HTTP_TRANSPORT = new NetHttpTransport();
            HttpRequestFactory httpRequestFactory = createRequestFactory(HTTP_TRANSPORT);

            GenericUrl url = new GenericUrl("https://maps.googleapis.com/maps/api/place/textsearch/json?");
            url.put("query", "top destinations in "+c_name);
            url.put("dest_mid", "/m/02_286");
            url.put("key", API_Key);
            HttpRequest request = httpRequestFactory.buildGetRequest(url);
            HttpResponse httpResponse = request.execute();
            getDestinations = (ArrayList<TopDestinations>) googleparse(httpResponse.parseAsString());
```

And on post execute we are attaching the returned and parsed data to recyclerview.

```
protected void onPostExecute(String feed) {
    destinations = getDestinations;
    recyclerViewAdapter = new DestinationRecyclerViewAdapter(StateActivity.this, destinations);
    recyclerView1.setAdapter(recyclerViewAdapter);

    AlphaInAnimationAdapter alphaAdapter = new AlphaInAnimationAdapter(recyclerViewAdapter);
    ScaleInAnimationAdapter scaleAdapter = new ScaleInAnimationAdapter(alphaAdapter);
    recyclerView1.setAdapter(scaleAdapter);
    SlideInLeftAnimator animator = new SlideInLeftAnimator();
    animator.setInterpolator(new OvershootInterpolator());

    animator.setAddDuration(500);
    animator.setRemoveDuration(500);

    recyclerView1.setItemAnimator(animator);
```

- ## Shortcuts –

  For creating shortcuts, all you have to do is some changes in manifest file and creating new resource directory under res named as xml and creating a new resource file named 'shortcuts.xml'

  The manifest file changes will allow the app know that it has to include shortcuts.xml file, the file where we have actually defined the shortcuts.

```
<activity
    android:name="com.exploreit8.sonalpatil.welcomeexploreit.WelcomeActivity"
    android:theme="@style/welcome"
    android:configChanges="keyboardHidden|orientation|screenSize">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>

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

After doing this, let's go to sortcuts.xml following is the format of the xml file which defines all the shortcuts for your app.

```
<shortcuts xmlns:android="http://schemas.android.com/apk/res/android">
    <shortcut
        android:shortcutId="login"
        android:enabled="true"
        android:icon="@drawable/acuser"
        android:shortcutShortLabel="Login"
        android:shortcutLongLabel="Login"
        android:shortcutDisabledMessage="Login">
        <intent
            android:action="android.intent.action.VIEW"
            android:targetPackage="com.exploreit8.sonalpatil.welcomeexploreit"
            android:targetClass="com.exploreit8.sonalpatil.welcomeexploreit.MainActivity" />
        <!-- 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>
```

By adding multiple child shortcut elements you can add as many shortcuts you want to add for you app.

- ## MultiWindow -

To get the multiwindow effect, Set this attribute in your manifest's <activity> or <application> element to enable or disable multi-window display:

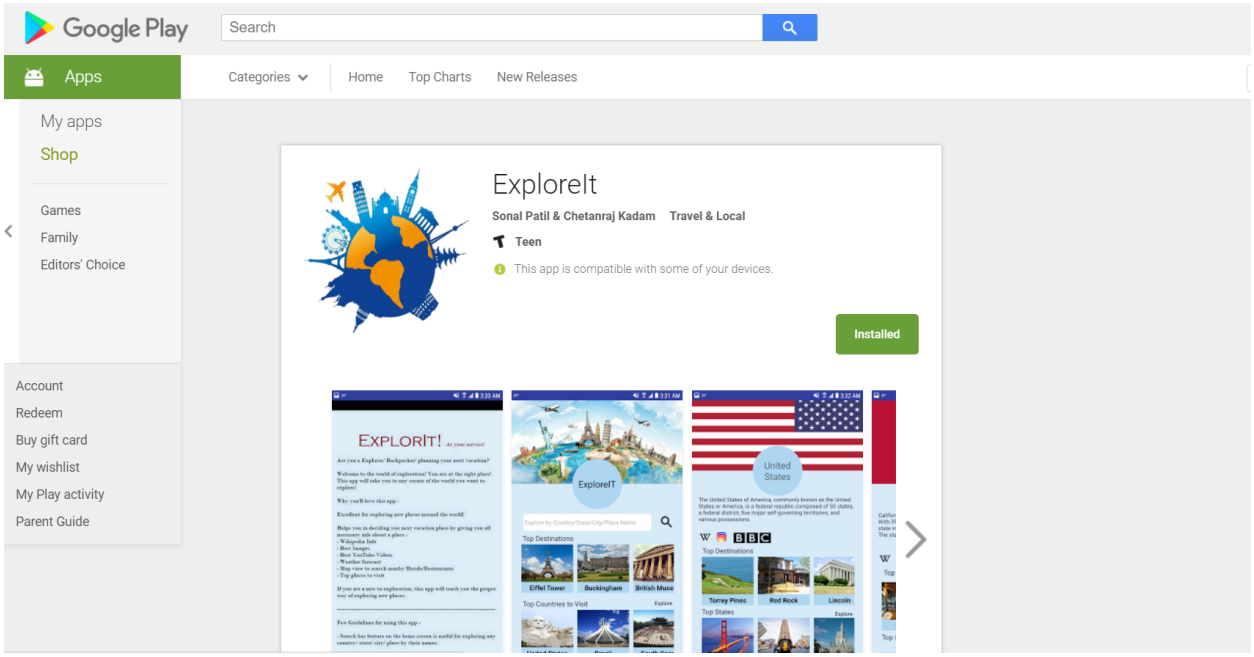
android:resizeableActivity=["true" | "false"]

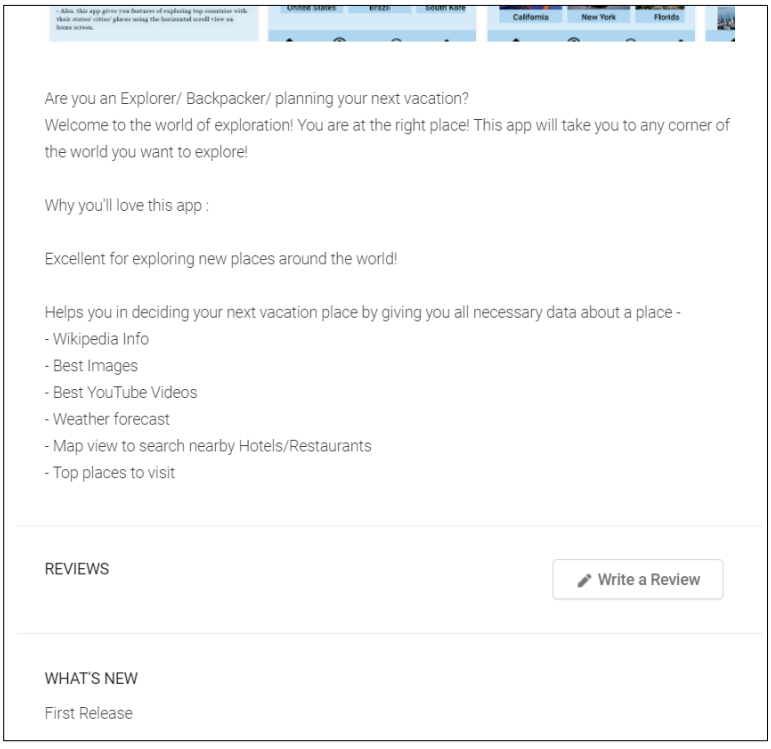
If this attribute is set to true, the activity can be launched in split-screen and freeform modes. If the attribute is set to false, the activity does not support multi-window mode. If this value is false, and the user attempts to launch the activity in multi-window mode, the activity takes over the full screen.

If your app targets API level 24, but you do not specify a value for this attribute, the attribute's value defaults to true.

- **Playstore –**

We have published our app on playstore.





ADDITIONAL INFORMATION

Updated
December 9, 2017

Current Version
1.0

Requires Android
7.0 and up

Content Rating
Teen
Learn more

Interactive Elements
Users Interact

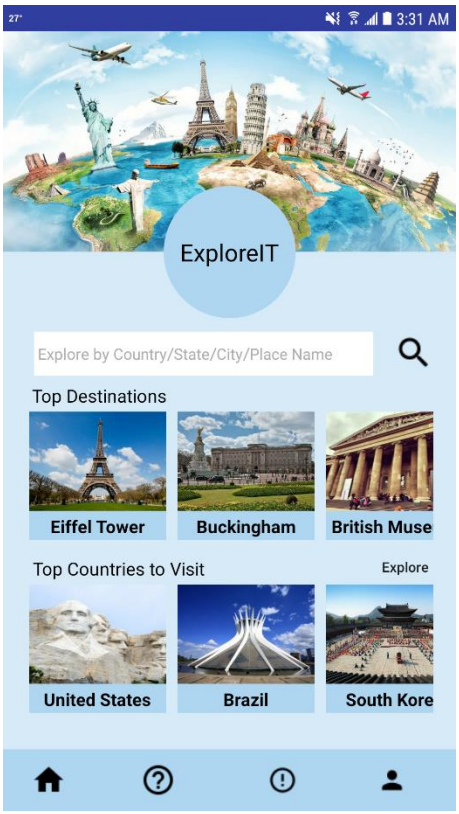
Permissions
View details

Report
Flag as inappropriate

Offered By
Sonal Patil & Chetanraj Kadam

Developer
Email sonalpatil9402@gmail.com

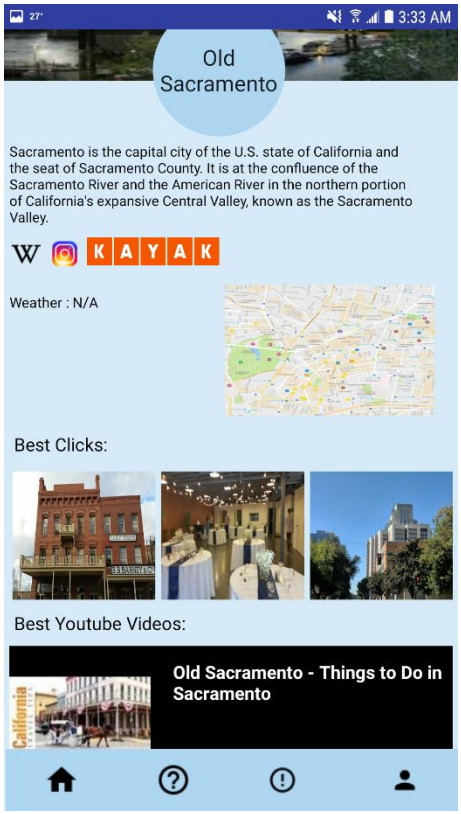
3. UI



Home Screen

Country Screen

State Screen



City Screen

Explore Screen

App Info Screen

4. Conclusion

This app is something that will widen users' views in all ways. It will give user the opportunity to explore something new, something different, something that you never thought of before. It's full of surprises. You come, you explore, you decide, you travel. Your happiness makes us feel proud.

5. References

- https://dzone.com/articles/android-tutorial-using
- http://openweathermap.org/api
- https://www.tutorialspoint.com/android/android_acitivities.htm
- https://www.androstock.com/tutorials/create-a-weather-app-on-android-android-studio.html
- https://developers.google.com/youtube/android/player/sample-applications