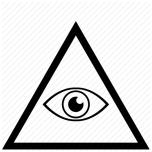
****

**EYE-HOMES**

**(Homes Everywhere)**

**Team Members:**

1. *Indrajeet Nayak* (BS17-509)
2. *Purabi Parida* (BS17-525)
3. *Pranatimayee jena* (BS17-528)

**College:** Buxi Jagabandhu Bidyadhar Autonomous College

**Teacher’s Signature :**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Acknowledgements**

First I would like to thank Mr. Rahul Garg (Chief Executive of Inspiredge IT Solutions) for helping us, with the necessary resources and platforms for testing and validation process. Then I would like to thank myself for not losing the motivation for creating this project and for working hard on this project .Also I would like to thank my team members for supporting me with necessary help, whenever required. Also I would like to thank my teachers for providing us the basic(to advanced) knowledge about the project.

The application is powered by NAS Pvt. Ltd. and will be further considered for financing by the known tech firm Inspiredge IT Solutions.

**Table Of Contents:**

1. **Introduction**
2. **Requirements for Application**

* **Functional requirements**
* **Non- Functional requirements**

1. **Terminologies Used**

* **Views used**
* **Toast**
* **AlertDialog**
* **ListView**
* **SearchView**
* **TextView**
* **EditText**
* **Button**
* **ImageView**
* **ProgressBar**
* **ScrollView**
* **RecyclerView**
* **Spinner**
* **CardView**
* **Layouts Used**
* **ConstraintLayout**
* **LinearLayout**
* **Adapter Used**
* **ArrayAdapter**
* **Database Library Used**
* **Volley**

1. **How it works ?**
2. **Testing result**
3. **Errors and Future Maintenance**
4. **Advantages**
5. **References**
6. **Bibliography**

This Page is intentionally left blank

**Introduction**

This is an android application, which deals with the real world problem of people. This application helps people to find the homes in their nearby location (city). This app is designed to finish the system of brokerage fees from its root level. Once the customer finds the home and location suitable, then he could request for the contact details of the owner (or seller). The application’s interface is made as simple as possible, both for the buyer and the seller for saving time. There is no need to create separate accounts for buyer and seller, because one buyer can be seller and vice-versa. The application is designed within 34 days, so there may be some bugs still available, but will be sorted with time.

**Requirements for Application**

**Function requirements**

Application must give the proper result to the user, when they search.

User should be provided with the real image which is posted by the respective seller.

No data mixing should be occurring from different users.

Each and every view of the application should work properly.

**Non-Functional Requirements**

User should have a proper Internet connection for Logging in to the application, getting search results etc.

For getting stability in the application and for proper functioning of the application, all it needs is an **Android device** with following specifications:

* RAM: More than 2 GB.
* Storage: 500 MB (min).
* OS: Android 4.4(or above).

Time complexity should be as minimum as possible.

**Terminologies**

1. **Views Used:**

All the terminologies were mentioned which were being used in the application for proper understanding of the views.

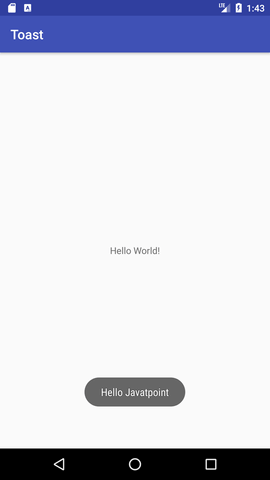
1. **Toast**

Andorid Toast can be used to display information for the short period of time. A toast contains message to be displayed quickly and disappears after sometime.

The android.widget.Toast class is the subclass of java.lang.Object class.

Toast toast = Toast.makeText(getApplicationContext(), “Simple Toast In Android”, Toast.LENGTH\_LONG); // initiate the Toast with context, message and duration for the Toast

toast.show(); // display the Toast



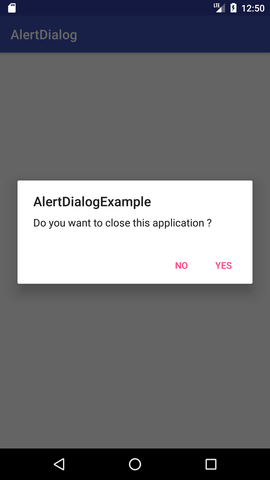
1. **AlertDialog**

It can be used to display the dialog message with OK and Cancel buttons. It can be used to interrupt and ask the user about his/her choice to continue or discontinue.

Android AlertDialog is composed of three regions: title, content area and action buttons.

Android AlertDialog is the subclass of Dialog class.

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);



1. **ListView**

Android **ListView** is a view which contains the group of items and displays in a scrollable list. ListView is implemented by importing android.widget.ListView class. ListView is a default scrollable which does not use other scroll view.

ListView uses Adapter classes which add the content from data source (such as string array, array, database etc) to ListView. Adapter bridges data between an AdapterViews and other Views (ListView, ScrollView etc).

**Here is Android ListView XML Code:**

<ListView xmlns:android=”http://schemas.android.com/apk/res/android”

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

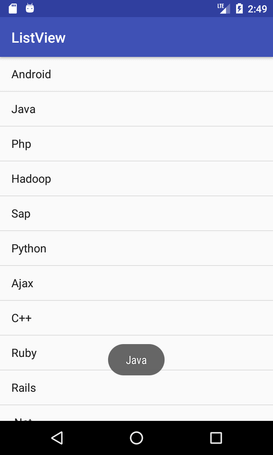
android:id=”@+id/simpleListView”

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_content”

tools:context=”abhiandroid.com.listexample.MainActivity”>

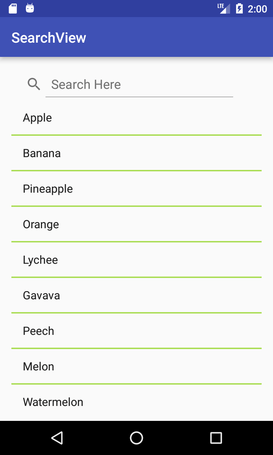
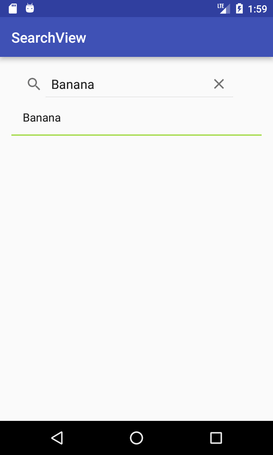
</ListView>



1. **SearchView**

Android **SearchView** provides user interface to search query submitted over search provider. SearchView widget can be implemented over ToolBar/ActionBar or inside a layout.

SearchView is by default collapsible and set to be iconified using setIconifiedByDefault(true) method of SearchView class. For making search field visible, SearchView uses setIconifiedByDefault(false) method.

1. **TextView**

In Android, [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self)displays text to the user and optionally allows them to edit it programmatically. [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self) is a complete text editor, however basic class is configured to not allow editing but we can edit it.

View is the parent class of [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self). Being a subclass of view the [text view](https://abhiandroid.com/ui/textview/) component can be used in your app’s GUI inside a ViewGroup, or as the content view of an activity.

We can create a TextView instance by declaring it inside a layout([XML](https://abhiandroid.com/ui/xml/" \o "XML in Android" \t "_self) file) or by instantiating it programmatically([Java](https://abhiandroid.com/java/) Class).

**TextView code in XML:**

<TextView android:id=”@+id/simpleTextView”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”AbhiAndroid” />

**TextView code in JAVA:**

TextView textView = (TextView) findViewById(R.id.textView);

textView.setText(“AbhiAndroid”); //set text for text view



1. **EditText**

In Android, [EditText](https://abhiandroid.com/ui/edittext/" \o "EditText" \t "_self) is a standard entry widget in android apps. It is an overlay over [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self) that configures itself to be editable. [EditText](https://abhiandroid.com/ui/edittext/" \o "EditText" \t "_self) is a subclass of [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self) with text editing operations. **We often use EditText in our applications in order to provide an input or text field, especially in forms.** The most simple example of [EditText](https://abhiandroid.com/ui/edittext/" \t "_self" \o "EditText) is Login or Sign-in form.

**EditText code in XML:**

<EditText

android:id=”@+id/simpleEditText”

android:layout\_height=”wrap\_content”

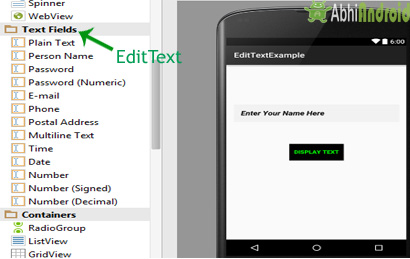
android:layout\_width=”match\_parent”/>

**Retrieving / Getting the Value From EditText In Java Class:**

Below is the example code of EditText in which we retrieve the value from a EditText in [Java](https://abhiandroid.com/java/) class. We have used this code in the example you will find at the end of this post.

EditText simpleEditText = (EditText) findViewById(R.id.simpleEditText);

String editTextValue = simpleEditText.getText().toString();



1. **Button**

In Android, **Button** represents a push [button](https://abhiandroid.com/ui/button/). A Push buttons can be clicked, or pressed by the user to perform an action. There are different types of buttons used in android such as CompoundButton, [ToggleButton](https://abhiandroid.com/ui/togglebutton/" \t "_self" \o "ToggleButton), [RadioButton](https://abhiandroid.com/ui/radiobutton/" \o "RadioButton" \t "_self).

[Button](https://abhiandroid.com/ui/button/) is a subclass of [TextView](https://abhiandroid.com/ui/textview) class and compound [button](https://abhiandroid.com/ui/button/) is the subclass of Button class. **On a button we can perform different actions or events like click event, pressed event, touch event etc.**

**Button code in XML:**

The below code will create Button and write “Abhi Android” text on it.

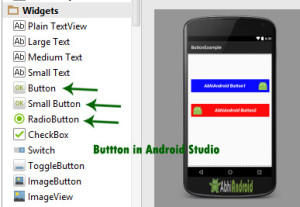
<Button

android:id=”@+id/simpleButton”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”Abhi Android”/>



1. **ImageView**

In Android, [ImageView](https://abhiandroid.com/ui/imageview/" \o "ImageView" \t "_self) class is used to display an image file in application. Image file is easy to use but hard to master in Android, because of the various screen sizes in Android devices. An android is enriched with some of the best UI design widgets that allows us to build good looking and attractive UI based application.

**Important Note:** [ImageView](https://abhiandroid.com/ui/imageview/" \o "ImageView" \t "_self) comes with different configuration options to support different scale types. Scale type options are used for scaling the bounds of an image to the bounds of the [imageview](https://abhiandroid.com/ui/imageview/" \o "ImageView" \t "_self). Some of them scaleTypes configuration properties are center, center\_crop, fit\_xy, fitStart etc. You can read our [ScaleType tutorial](https://abhiandroid.com/ui/scaletype-imageview-example.html) to learn all details on it.

**Below is an ImageView code in XML:**

Make sure to save lion image in drawable folder

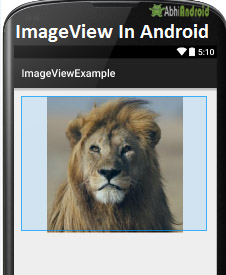
<ImageView

android:id=”@+id/simpleImageView”

android:layout\_width=”fill\_parent”

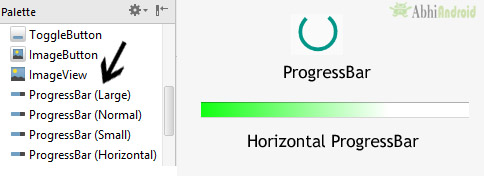
android:layout\_height=”wrap\_content”

android:src=”@drawable/lion” />



1. **ProgressBar**

In Android, [ProgressBar](https://abhiandroid.com/ui/progressbar/" \o "ProgressBar" \t "_self) is used to display the status of work being done like analyzing status of work or downloading a file etc. In Android, by default a [progress bar](https://abhiandroid.com/ui/progressbar/) will be displayed as a spinning wheel but If we want it to be displayed as a horizontal bar then we need to use style attribute as horizontal. It mainly use the**“android.widget.ProgressBar”**class.



**Important Note:**A [progress bar](https://abhiandroid.com/ui/progressbar/) can also be made indeterminate. In this mode a [progress bar](https://abhiandroid.com/ui/progressbar/) shows a cyclic [animation](https://abhiandroid.com/materialdesign/animation/) without an indication of progress. This mode is used in application when we don’t know the amount of work to be done.

To add a progress bar to a layout ([xml](https://abhiandroid.com/ui/xml/)) file, you can use the <[ProgressBar](https://abhiandroid.com/ui/progressbar/" \o "ProgressBar" \t "_self)> element. By default, a progress bar is a spinning wheel (an indeterminate indicator). To change to a horizontal progress bar, apply the progress bar’s horizontal style.

**ProgressBar code:**

<ProgressBar

android:id=”@+id/simpleProgressBar”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content” />

**Horizontal ProgressBar code:**

<ProgressBar

android:id=”@+id/simpleProgressBar”

android:layout\_width=”fill\_parent”

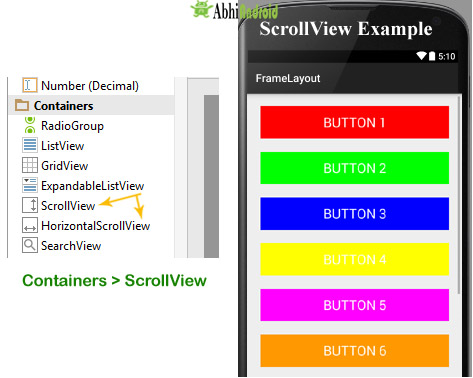
android:layout\_height=”wrap\_content”

style=”@style/Widget.AppCompat.ProgressBar.Horizontal”/>

1. **ScrollView**

In android [ScrollView](https://abhiandroid.com/ui/scrollview/" \o "ScrollView And Horizontal ScrollView" \t "_self) can hold only one direct child. This means that, if you have complex layout with more views(Buttons, TextViews or any other view) then you must enclose them inside another standard layout like [Table Layout](https://abhiandroid.com/ui/tablelayout/), [Relative Layout](https://abhiandroid.com/ui/relative-layout/) or [Linear Layout](https://abhiandroid.com/ui/linear-layout/). You can specify layout\_width and layout\_height to adjust width and height of screen. You can specify height and width in dp(density pixel) or px(pixel). Then after enclosing them in a standard layout, enclose the whole layout in [ScrollView](https://abhiandroid.com/ui/scrollview/" \o "ScrollView And Horizontal ScrollView" \t "_self) to make all the element or views scrollable.

**ScrollView in Android Studio Design:** It is present inside Containers >> [ScrollView](https://abhiandroid.com/ui/scrollview/" \o "ScrollView And Horizontal ScrollView" \t "_self) or HorizontalScrollView



**Important Note 1:**We never use a Scroll View with a [ListView](https://abhiandroid.com/ui/listview/" \t "_self" \o "ListView) because [List View](https://abhiandroid.com/ui/listview/) is default scrollable(i.e. vertical scrollable). More importantly, doing this affects all of the important optimizations in a [List View](https://abhiandroid.com/ui/listview/) for dealing with large lists(list items). Just because it effectively forces the [List View](https://abhiandroid.com/ui/listview/) to display its entire list of items to fill up the infinite container supplied by a ScrollView so we don’t use it with List View.

**Important Note 2:** In android default ScrollView is used to scroll the items in vertical direction and if you want to scroll the items horizontally then you need to implement [horizontal ScrollView](https://abhiandroid.com/ui/scrollview/).

**ScrollView Syntax:**

<ScrollView

android:id=”@+id/scrollView”

android:layout\_width=”fill\_parent”

android:layout\_height=”fill\_parent”>

<!—add child view’s here 🡪

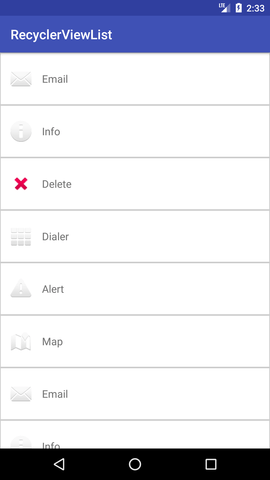
</ScrollView>

1. **RecyclerView**

The **RecyclerView** class extends the ViewGroup class and implements ScrollingView interface. It is introduced in Marshmallow. It is an advanced version of the ListView with improved performance and other benefits. RecyclerView is mostly used to design the user interface with the fine-grain control over the lists and grids of android application.

In this tutorial, we will create a list of items with ImageView (for the icon) and TextView (for description) using **RecyclerView** and performs click listener on the item of its list.

Create an Android project, and add the RecyclerView support library **com.android.support:recyclerview-v7:23.1.0** or above this version in build.gradle file.



<?xml version=”1.0” encoding=”utf-8”?>

<RelativeLayout xmlns:android=”http://schemas.android.com/apk/res/android”

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

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:context=”abhiandroid.com.recyclerviewexample.MainActivity”>

<android.support.v7.widget.RecyclerView

android:id=”@+id/recyclerView”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent” />

</RelativeLayout>

**Gradle Dependency to use RecyclerView:**

The RecyclerView widget is a part of separate library valid for API 7 level or higher. Add the following dependency in your Gradle build file to use recyclerview.

**Gradle Scripts > build.gradle and inside dependencies**

dependencies {

...

compile “com.android.support:recyclerview-v7:23.0.1”

}

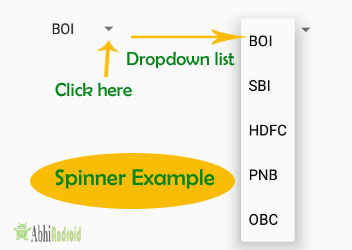
#### ****Need of RecyclerView In Android****

RecyclerView uses a ViewHolder for storing the reference of the view for one entry in the RecyclerView. When we use ListView or GridView for displaying custom items then we create a custom xml file and then use it inside our Adapter. For this we create a CustomAdapter class and then extends our Base or any other Adapter in it. In getView() method of our Adapter we inflate the item layout xml file and then give the reference of every view by using the unique id’s we provide in our xml file . Once finished we pass that view to the ListView, ready to be drawn, but the truth is that ListView and GridView do only half the job of achieving true memory efficiency.

ListView/GridView recycle the item layout but don’t keep the reference to the layout children, forcing us to call findViewById() for every child of our item layout for every time we call getView(). This issue causes the scrolling or non responsive problem as it frantically tries to grab references to the view’s we needed.

1. **Spinner**

In Android, [Spinner](https://abhiandroid.com/ui/spinner/) provides a quick way to select one value from a set of values. Android spinners are nothing but the drop down-list seen in other programming languages. In a default state, a [spinner](https://abhiandroid.com/ui/spinner/) shows its currently selected value. It provides a easy way to select a value from a list of values.



In Simple Words we can say that a [spinner](https://abhiandroid.com/ui/spinner/) is like a combo box of AWT or swing where we can select a particular item from a list of items. Spinner is a sub class of AsbSpinner class.

**Important Note:**Spinner is associated with [Adapter](https://abhiandroid.com/ui/adapter/) view so to fill the data in spinner we need to use one of the [Adapter](https://abhiandroid.com/ui/adapter/) class.

**Here is the XML basic code for Spinner:**

<Spinner

android:id=”@+id/simpleSpinner “

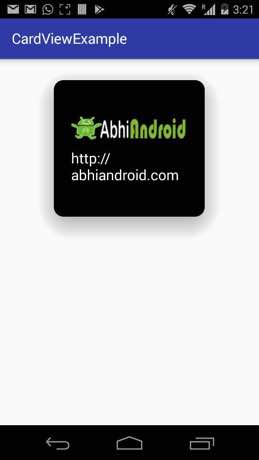
android:layout\_width=”fill\_parent”

android:layout\_height=”wrap\_content” />

**Important Note:** To fill the data in a spinner we need to implement an [adapter](https://abhiandroid.com/ui/adapter/) class. A spinner is mainly used to display only text field so we can implement Array Adapter for that. We can also use [Base Adapter](https://abhiandroid.com/ui/baseadapter-tutorial-example.html) and other custom adapters to display a spinner with more customize list. Suppose if we need to display a [textview](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self) and a [imageview](https://abhiandroid.com/ui/imageview/" \o "ImageView" \t "_self) in spinner item list then [array adapter](https://abhiandroid.com/ui/arrayadapter-tutorial-example.html) is not enough for that. Here we have to implement custom adapter in our class. Below image of Spinner and Custom Spinner will make it more clear.

1. **CardView**

In Android, [CardView](https://abhiandroid.com/materialdesign/cardview" \t "_self) is another main element that can represent the information in a card manner with a drop shadow called elevation and corner radius which looks consistent across the platform. [CardView](https://abhiandroid.com/materialdesign/cardview" \t "_self) was introduced in Material Design in API level 21 (Android 5.0 i.e Lollipop).



CardView uses elevation property on Lollipop for shadows and falls back to a custom emulated shadow implementation on older platforms.

This new widget is a big step for displaying data/information inside cards. We can easily design good looking UI when we combined CardView with [RecyclerView](https://abhiandroid.com/materialdesign/recyclerview-as-listview.html" \t "_self). A CardView is a ViewGroup that can be added in our Activity or Fragment using a layout XML file.

#### ****Basic CardView XML code In Android Studio:****

<android.support.v7.widget.CardView

xmlns:card\_view=”http://schemas.android.com/apk/res-auto”

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_content”>

</android.support.v7.widget.CardView>

#### ****Gradle Dependency to use CardView:****

The CardView widget is a part of separate library valid for API 7 level or higher. Add the following dependency in your Gradle build file to use CardView.  
**Add inside Gradle Scripts > build.gradle (Module: app) and inside dependencies**

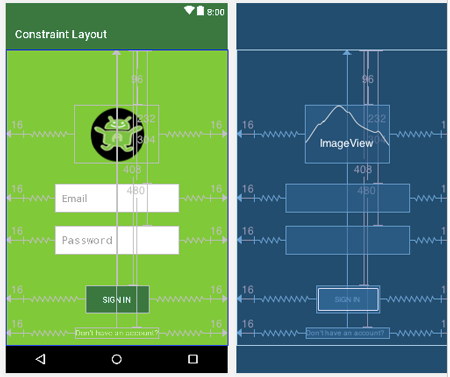
dependencies {

compile ‘com.android.support:cardview-v7:23.0.1’

}

1. **Layouts Used:**
2. **ConstraintLayout**

Constraint Layout is a ViewGroup (i.e. a view that holds other views) which allows you to create large and complex layouts with a flat view hierarchy, and also allows you to position and size widgets in a very flexible way. It was created to help reduce the nesting of views and also improve the performance of layout files.



ConstraintLayout is very similar to [RelativeLayout](https://abhiandroid.com/ui/relative-layout/" \o "RelativeLayout" \t "_self) in such a way because, views are laid out according to relationships between sibling views and the parent layout yet it’s a lot more flexible and works better with the Layout Editor of the Android Studio’s. It was released at Google I/O 2016. Since it came into existence (i.e. as at [Android studio](https://abhiandroid.com/androidstudio/)2.3), it has become a wildly used viewgroup and supports Android 2.3 or higher.

**Using Constraint Layout In Android Studio:**

It is not bundled as part of Android SDK and is available as a support library. Due to this, any update in the future would be compatible with all versions of Android.

To use Constraint Layout make sure you have declared below repository in build.gradle file

repositories {

maven {

url ‘https://maven.google.com’

}

}

Now to use ConstraintLayout features in our android project, we will need to add the library to our build.gradle app module dependencies section.  
Open your build.gradle (Module app) and add the code below:

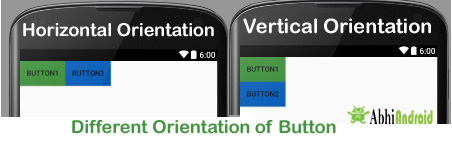
dependencies {

compile ‘com.android.support.constraint:constraint-layout:1.1.0-beta3’

}

1. **LinearLayout**

[Linear layout](https://abhiandroid.com/ui/linear-layout/) is a simple layout used in android for layout designing. In the [Linear layout](https://abhiandroid.com/ui/linear-layout/) all the elements are displayed in linear fashion means all the childs/elements of a [linear layout](https://abhiandroid.com/ui/linear-layout/) are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.



**Types Of Linear Layout Orientation**

There are two types of linear layout orientation:

1. Vertical
2. Horizontal

As the name specified these two orientations are used to arrange there child one after the other, in a line, either vertically or horizontally. Let’s we describe these in detail.

**1.Vertical:**

In this all the child are arranged vertically in a line one after the other. In below code snippets we have specified orientation “vertical” so the childs/views of this layout are displayed vertically.

<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”

android:layout\_width=”fill\_parent”

android:layout\_height=”wrap\_content”

android:orientation=”vertical”> <!—Vertical Orientation set 🡪

<!—Child Views(In this case 2 Button) are here 🡪

<Button

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”Button1”

android:id=”@+id/button”

android:background=”#358a32” />

<Button

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”Button2”

android:id=”@+id/button2”

android:background=”#0058b6” />

</LinearLayout>



1. **Horizontal:**

In this all the child are arranged horizontally in a line one after the other. In below code snippets we have specified orientation “horizontal” so the childs/views of this layout are displayed horizontally.

<LinearLayout xmlns:android=”http://schemas.android.com/apk/res/android”

android:layout\_width=”fill\_parent”

android:layout\_height=”wrap\_content”

android:orientation=”horizontal”> <!—Horizontal Orientation set 🡪

<!—Child Views(In this case 2 Button) are here 🡪

<Button

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”Button1”

android:id=”@+id/button”

android:background=”#358a32” />

<Button

android:layout\_width=”wrap\_content”

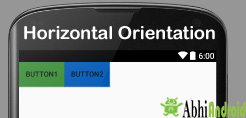
android:layout\_height=”wrap\_content”

android:text=”Button2”

android:id=”@+id/button2”

android:background=”#0058b6” />

</LinearLayout>



**Important Note:** All of the layout managers can be nested. This means that you can put a [Relative Layout](https://abhiandroid.com/ui/relative-layout/) or [Frame Layout](https://abhiandroid.com/ui/framelayout/) as a child to Linear Layout.

1. **Adapters Used:**
2. **ArrayAdapter**

In android, An [adapter](https://abhiandroid.com/ui/adapter/) is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to [adapter](https://abhiandroid.com/ui/adapter/) view then view can takes the data from the [adapter](https://abhiandroid.com/ui/adapter/) view and shows the data on different views like [listview](https://abhiandroid.com/ui/listview/" \o "ListView" \t "_self), [gridview](https://abhiandroid.com/ui/gridview/" \o "GridView" \t "_self), [spinner](https://abhiandroid.com/ui/spinner/) etc. ArrayAdapter is more simple and commonly used Adapter in android.

* Whenever you have a list of single type of items which is backed by an array, you can use ArrayAdapter. For instance, list of phone contacts, countries or names.
* By default, ArrayAdapter expects a Layout with a single [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self), If you want to use more complex views means more customization in grid items or list items, please avoid ArrayAdapter and use custom adapters.

**Important Note:** ArrayAdapter is an implementation of BaseAdapter so if we need to create a custom [list view](https://abhiandroid.com/ui/listview/) or  a [grid view](https://abhiandroid.com/ui/gridview/)  then we have to create our own custom adapter and extend ArrayAdapter in that custom class. By doing this we can override  all the function’s of BaseAdapter in our custom adapter.

**Here is code of ArrayAdapter in Android:**

ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

In the above code snippet is the implementation of a ArrayAdapter. Below is the description of all the parameters used for the implementation of a ArrayAdapter to show a list of elements in a [list view](https://abhiandroid.com/ui/listview/) or in a [spinner](https://abhiandroid.com/ui/spinner/).

**Parameters used in ArrayAdapter:**

Lets discuss parameter in ArrayAdapter class:

* **context:**

The first parameter is used to pass the context means the reference of current class. Here this is a keyword used to show the current class reference. We can also use getApplicationContext(), getActivity() in the place of this keyword. getApplicationContext() is used in a Activity and getActivity() is used in  a [Fragment](https://abhiandroid.com/ui/fragment/).

Below is the example code in which we set the current class reference in a adapter.

ArrayAdapter arrayAdapter = new ArrayAdapter(this, int resource, int textViewResourceId, T[] objects);

* **resource:**

The second parameter is resource id used to set the layout([xml](https://abhiandroid.com/ui/xml/" \o "XML in Android" \t "_self) file) for list items in which you have a [text view](https://abhiandroid.com/ui/textview/).

Below is the example code in which we set the layout.

ArrayAdapter arrayAdapter = new ArrayAdapter(this, R.layout.list\_view\_items, int textViewResourceId, T[] objects);

* **textViewResourceId:**

The third parameter is textViewResourceId which is used to set the id of [TextView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self) where you want to display the actual text.

Below is the example code in which we set the id(identity) of a [text view](https://abhiandroid.com/ui/textview/).

ArrayAdapter arrayAdapter = new ArrayAdapter(this, R.layout.list\_view\_items, R.id.textView, T[] objects);

* **objects:**

The fourth parameter is an array of objects, used to set the array of elements in the [textView](https://abhiandroid.com/ui/textview/" \o "TextView" \t "_self). We can set the object of array or array list here.

Below is the example code in which we set the Animal array in adapter to display the Animal name’s list.

String nimalist[] = {“Lion”,”Tiger”,”Monkey”,”Elephant”,”Dog”,”Cat”,”Camel”};

ArrayAdapter arrayAdapter = new ArrayAdapter(this, R.layout.list\_view\_items, R.id.textView, nimalist);

1. **Database Library Used:**

**Volley**

[Volley](https://abhiandroid.com/programming/volley/) is a HTTP library developed by Google and was first introduced during Google I/O 2013. This library is used to transmit data over the network. It actually makes networking faster and easier for Apps. It is available through AOSP(Android Open Source Project) repository.

The [volley](https://abhiandroid.com/programming/volley/) library has the features like automatic scheduling of network request, multiple concurrent connections, request prioritization, cancel/block a request, easier management of UI with data fetched asynchronously from the network and also offers easier customization.

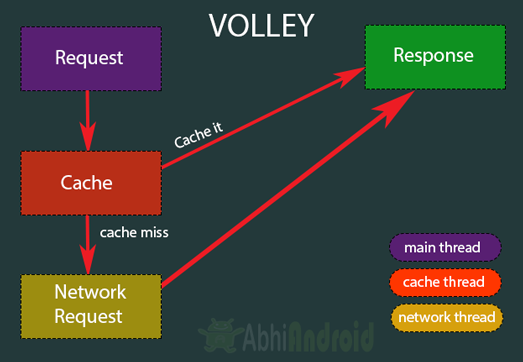
**Important Note:** [Volley](https://abhiandroid.com/programming/volley/) uses cache to improve the App performance by saving memory and bandwidth of remote server.

**Volley Uses Caches To Improve Performance In Android**:

Volley uses caches concept to improve the performance of App. For example, lets say you are using [Asynctask](https://abhiandroid.com/programming/asynctask/" \o "AsyncTask" \t "_self) to fetch image and description from a [JSON](https://abhiandroid.com/programming/json/) array created in server API. The content is fetched in the portrait mode and now user rotate screen to change it to landscape mode. The activity is destroyed and so the content will be fetched again. Since the user is calling the same resource again to server so it is a waste of resource and ultimately a poor user experience.

Volley provide solution to this problem as it caches the data. When user request the same data, instead of calling from server Volley directly shows it from cache saving resource and thus improving user experience.

**Below is the basic diagram of Volley:**



**Understanding RequestQueue & Working With Volley In Android:**

Volley is a networking library managed by the RequestQueue and mainly used for smaller Networking purposes in Android. To use it, first you need to instantiate the RequestQueue and later on you can start or stop request, add or cancel request and access the response cache(s).

RequestQueue queue = Volley.newRequestQueue(this);

After instantiating RequestQueue, a request must be created. The default request classes already included in Volley library are String request, [JSON](https://abhiandroid.com/programming/json/) request, and image request. You can also create custom request by extending Volley’s request class.

**Request Constructors used in Volley takes 4 parameter:**

JsonObjectRequest request = JsonObjectRequest(Request.Method.GET, url, new ResponseListener(), new ErrorListener();

**First Parameter: Request.Method.GET –** The GET is used to read. You can also use POST (to create), PUT (To update/replace), DELETE (to delete), PATCH (to update/modify) and more.

**Second Parameter: URL –** The url that will response to the HTTP request.

**Third Parameter: Successful Response Listener –** Where your data will go after the request is successfully complete.

Private class ResponseListener implements Response.Listener{

@Override

public void onResponse(JSONObject response){

}

}

**Fourth Parameter: Error Listener –** What will be told if there was a problem with your request. For example, you can display it in Log to see the error.

Private class ErrorListener implements Response.ErrorListener{

@Override

public void onErrorResponse(VolleyError error){

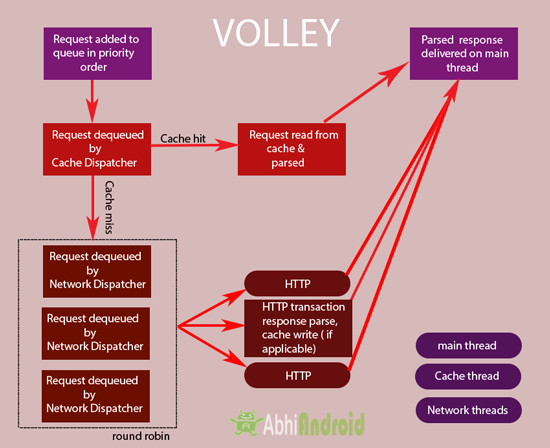
}

}

Now the last step is to add your request to Request queue and rest volley will handle for you.

Queue.add(request);

Here you can also add more requests to the queue that you would like at one time and the response will be send to their respective response/error classes.



**How it works?**

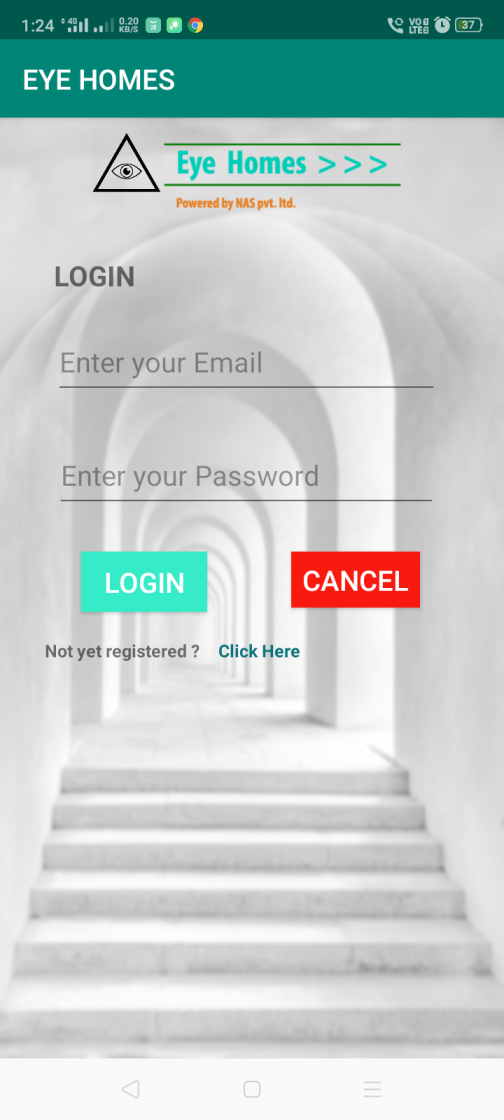
1. So the app first starts with and animated slideshow of the logo. The animation look place using the scaling and fading attributes.



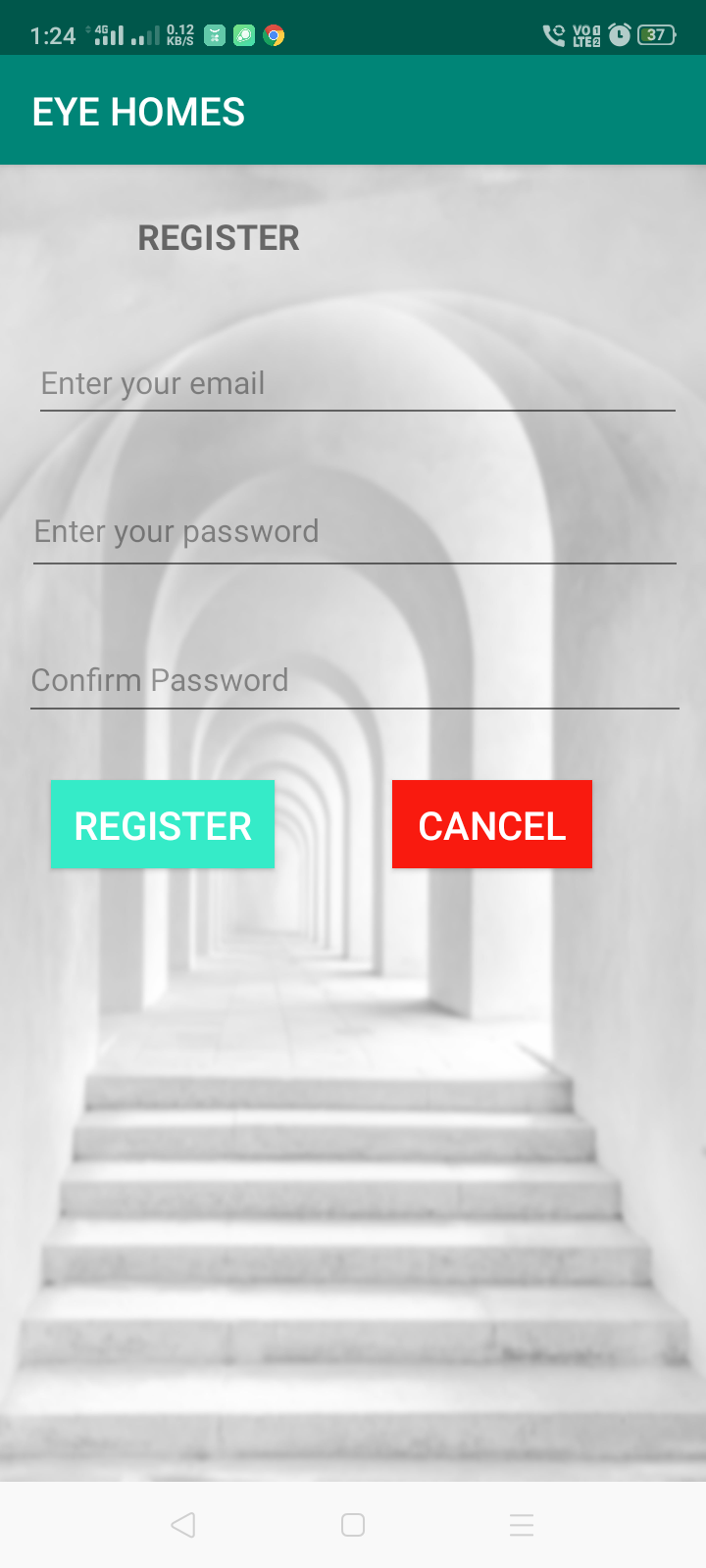
For loading the animation, we used **loadAnimation** function of **AnimationUtils** class which takes two parameters:

* The calling class object
* Anim file which consist of the processing of animation.

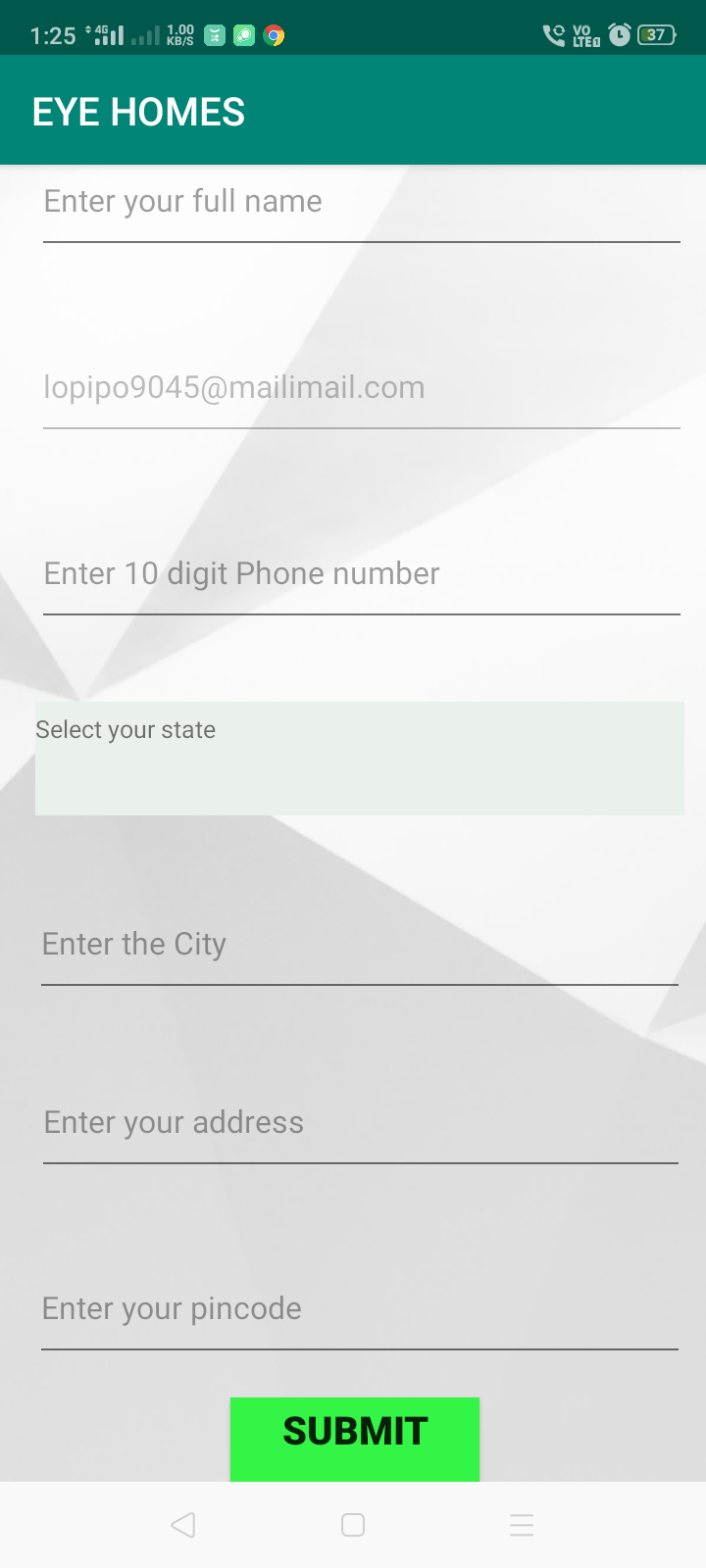
1. The next is the login activity which lets the user to log in to his/her account if s/he is a registered user. Here we used the **Backendless** server for handling user account. This server provides several features of handling validation and verification processes.



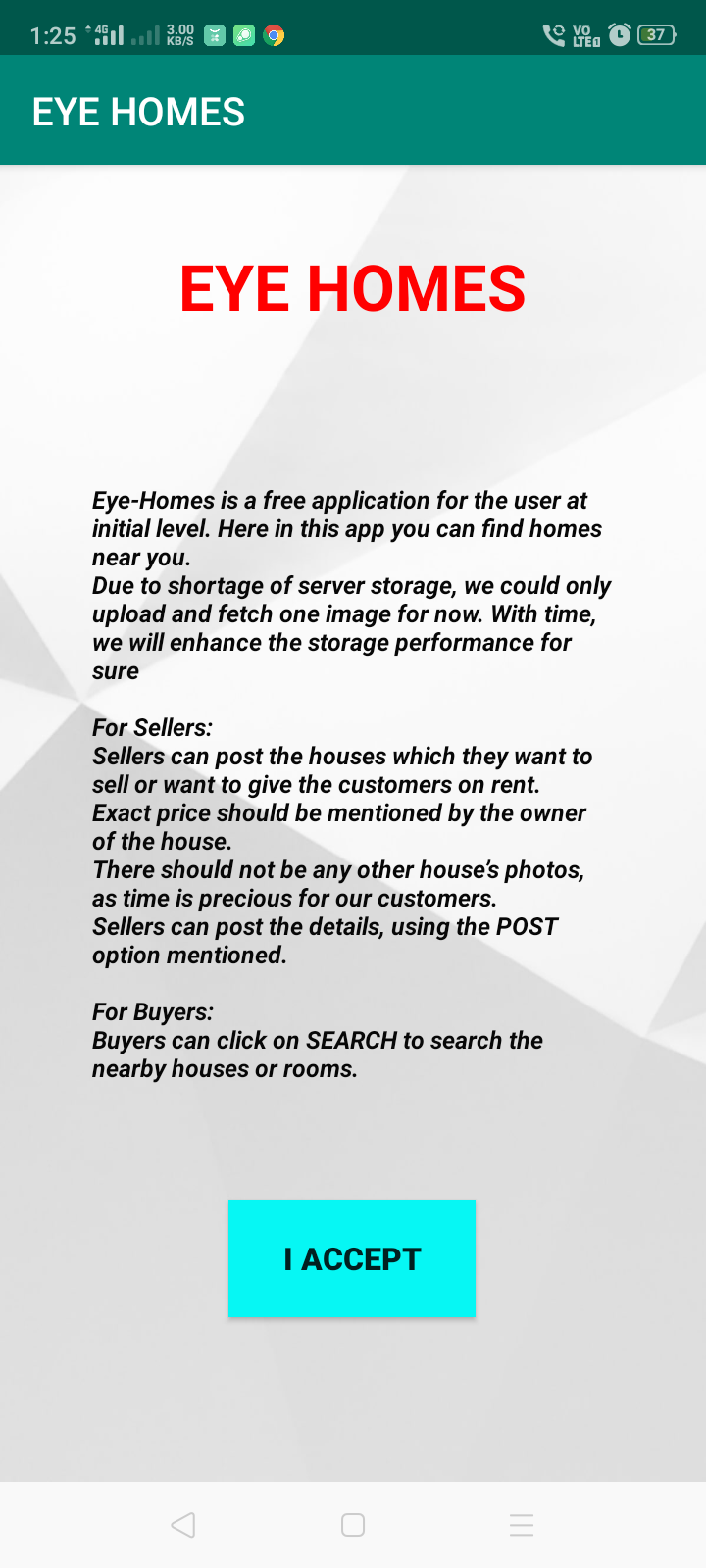
1. The Same server is used for the registration process too. It just requires and initialization process to start and then required **login()** for log in service and **register()** for registering the user which is inside **UserService**.



1. After registration it will need to verify the email id using the service provided by the Backendless which is in the admin area only.
2. After successfully log in to the account, the user is required to provide his details like name, mobile number, state, address, city, pincode only for once until and unless they uninstall their application. This one time data entry for getting user details is accomplished using the **SharedPreference** variable, which is stored locally in the user’s device. For submitting the details, the **volley** library is used.



1. Then the user is then bound to accept the terms and condition section for further approval of the user, which is also once in the application’s lifetime. This one time activity is also occurred using the **SharedPreference**.



1. Then the user is provided with the two options:

* **POST**- This option is used for the sellers for posting details about the house they were selling.
* **SEARCH**- This option will help users to search the homes in their appropriate searched cities.

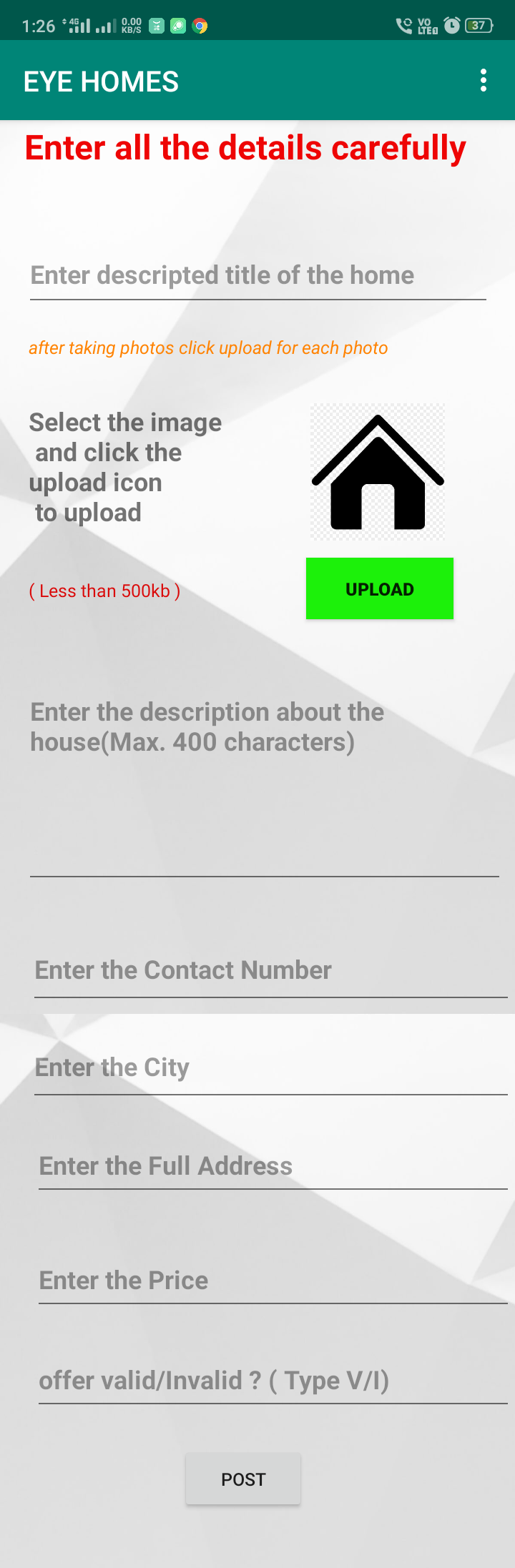
The application’s interface is tried to make as simple as possible so that, the users will not found any difficulties while using the application. This will save their time and work too.



1. In the POST section the user is required to provide the complete details as mentioned in the section. As there is the shortage of the storage, the user is bound to provide the photo of max. 500kb. Only after the successful uploading of the photo, the user can post his ad or details in the server.

We use the **volley** library here for storing the details of the house.

But for storing images we use, **000webhost** hosting, which is provided with the 1 GB data for the developers. The image is stored in 000webhost and its path is stored in XAMPP web server which is provided by 000webhosting itself.

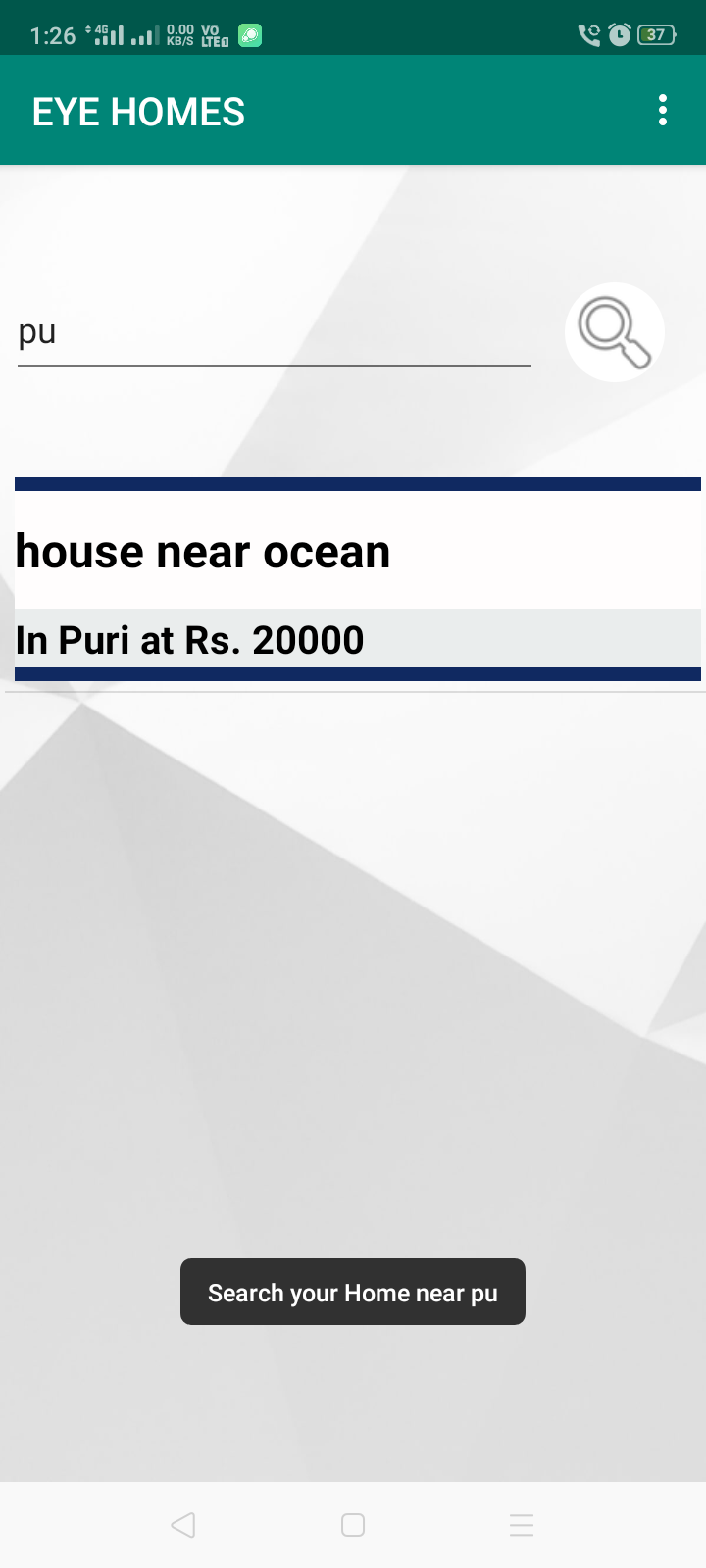


The user should be available with the constant and fast internet speed.

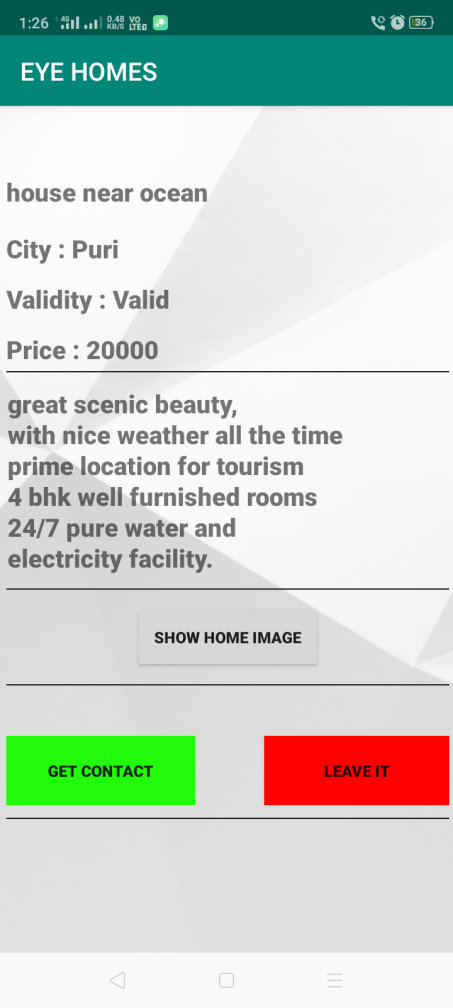
1. The SEARCH section has a **SearchView** which is used to search the houses available in the respective searched city. Also database connectivity is created using the **volley** library again.

The result is shown in the **ListView** and the rest of the data is passed to the next activity for getting the more details using the **intent** feature.

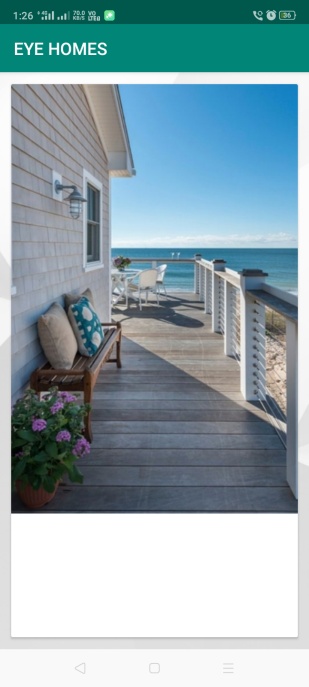




1. The details were then shown using normal TextView and the user is provided with some buttons for retrieving image of the house, for getting contact details, and if not interested then also have a button to reject that offer.



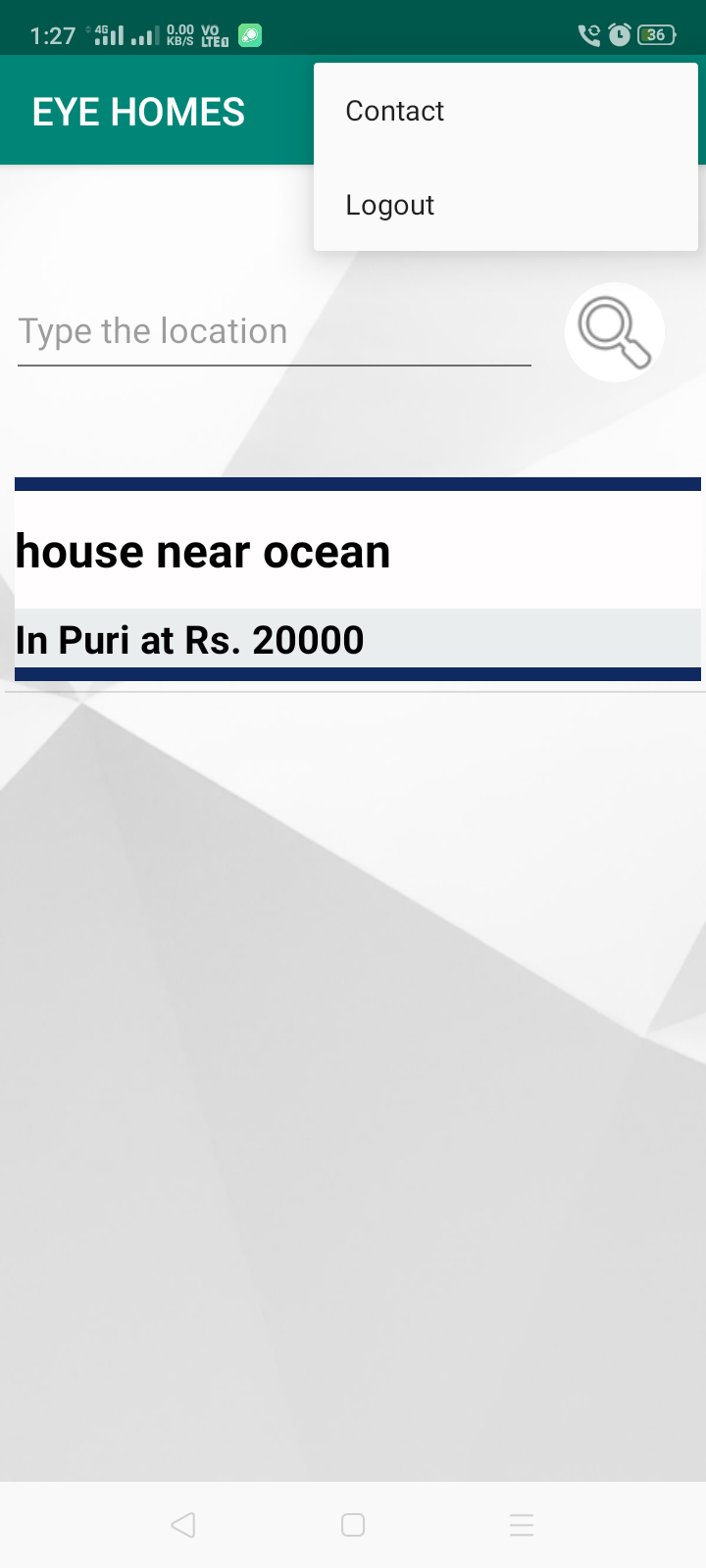
1. The image is shown in **RecyclerView** using the attribute **CardView** for giving the professional look to the images.



1. And finally if the user like the deal, then he/she will be provided with the contact details of the owner(like phone number, verified email address, address).



1. Apart from all these things, there is one menu option also included in the top right corner, which we created using the **ContextMenu** option, and is used for features like **logout** and **contact**.



In this way the app works and helps user to find homes near

their location with as efficient way as possible.

**Testing Results**

The test results were done by Mohit Mishra, certified tester at Inspiredge IT Solutions in the guidance of Rahul Garg. The test performed- is Unit testing.

All the test cases were provided in the report mentioned in the next page. The test was conducted on 21st April, 2020 showing the successful reports of the project.

The unit testing helps to find each and every bug in unit level and helps for proper functioning of the application after re-corrected.

Please check out the test reports mentioned in the next page. The negative false reports were not mentioned.

|  |  |
| --- | --- |
| **Test Scenario Id**  7809A45 | **Project Name**  EYE-HOMES |
| **Test Case Description**  Login and Registration | **Pre-requisite**  A valid user account |
| **Test Designed By:**  Mohit Mishra | **Test Designed Date:**  21-April-2020 |
| **Current Version:**  *1.01* | **Test Result:**  *Passed* |
| Inspiredge IT Solutions | NAS Pvt. Ltd. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No.** | **Action** | **Inputs** | **Expected**  **outputs** | **Actual Output** | **Test result** |
|  | **Launch Application** | Open the app | Visit the login activity by completing the animation series | Visit the login screen after completing the animation series | Pass |
|  | **Register** new account | Email:  [Lopipo9045@mailimail.com](mailto:Lopipo9045@mailimail.com)  Password:  abc\*\*\*\* | Complete the registration by email verification | Complete the registration by email verification | Pass |
|  | **Log in** to the account | Email:  [Lopipo9045@mailimail.com](mailto:Lopipo9045@mailimail.com)  Password:  abc\*\*\*\* | Successfully log in to the account and get the home page | Successfully log in to the account and get the home page | Pass |
| 4) | Enter details about the user and store it in **XAMPP** server | Name: Ajay  Email: [Lopipo9045@mailimail.com](mailto:Lopipo9045@mailimail.com)  Phone: 902301xxx0  State: Assam  City: Guwahati  Address: Plot-27, near Raj chowk  Pincode: 781001 | Successfully stored in the database, and transfer the control to Terms\_and\_Conditions activity. | Successfully stored in the database, and transfer the control to Terms\_and\_Conditions activity. | Pass |
|  | Click the **Accept** button | Accept the terms and condition by clicking accept button | Visit the Main screen of POST and SEARCH | Visit the Main screen of POST and SEARCH | Pass |
|  | Visit the **POST** section | Click the POST button to visit the POST section | Visit the POST section | Visit the POST section | Pass |
|  | Visit the **SEARCH** section | Click the SEARCH button to visit the SEARCH section | Visit the SEARCH section | Visit the SEARCH section | Pass |
|  | Post the data to the **XAMPP** server and upload image to **000webhost** | **Title:** 2BHK room in Central Market  **Description:** Nice house to live in, with amazing features and well furnished furnitures and utensils.  **Phone:** 963xxxxxx0  **City:** Dibrugarh  **Address:** Near dibrugarh  **Price:** 4500  **Validity:** v  Photo: Uploaded from gallery(size 439 KB) | Upload the image and post the data successfully and returned to the main screen of POST and SEARCH section. | Upload the image and post the data successfully and returned to the main screen of POST and SEARCH section. | Pass |
|  | Search the Home by typing the city | Typed Dibrugarh  And wait for the result | House offer should prompt up  With  title as “2BHK room in Central Market”,  city as “Dibrugarh”, and price as”Rs. 4500”  along with the Toast message as “Search your home near Dibrugarh” . | House offer should prompt up  With  title as “2BHK room in Central Market”,  city as “Dibrugarh”, and price as”Rs. 4500”  along with the Toast message as “Search your home near Dibrugarh”. | Pass |
|  | Retrieve the List of offers in ListView | Wait for sometime after searching for the city | List of offers related to the searched term | List of offers related to the searched term | Pass |
|  | Click the offer and get the details of the house | Click the offer | Get the complete details about the offer with 3 buttons embedded in that activity Show Image, Get Contact, Leave it. | Get the complete details about the offer with 3 buttons embedded in that activity Show Image, Get Contact, Leave it. | Pass |
|  | Click the Show image Button to get the image | Click the Show image Button to get the image | The Image shown in CardView of RecyclerView followed by the Toast message conveying the message to wait. | The Image shown in CardView of RecyclerView followed by the Toast message conveying the message to wait. | Pass |
|  | Get the contact details of owner | Click the Get Contact button | Activity shows the contact details of the owner | Activity shows the contact details of the owner | Pass |
| 14) | Not like the offer | Click the “Leave it” button to reject the offer. | Transfer the control to previous activity where the list of offers is shown. | Transfer the control to previous activity where the list of offers is shown. | Pass |
| 15) | Log out from the account | Clicking the logout option in context menu in the top right corner | Log out from the account and move to the log in activity. | Log out from the account and move to the log in activity. | Pass |
| 16) | Contact the support | Clicking the contact option in context menu in the top right corner | A toast will generate showing the contact details of the support team. | A toast will generate showing the contact details of the support team. | Pass |

**Errors and Future Maintenance**

**Errors:**

While testing the application at unit (atomic) level, we find many errors and bugs in this version. But the errors will be sorted with time. Some of the errors pointed out during testing were as follows :

1. The app returns back to the last activity even after getting logout from the application.
2. The application is running in the main thread, so lagging and skipping of some frames may obviously occur.
3. The TextView shifted while typing, Views do not fit into ScrollView.
4. If mistakenly the user presses back button after login and gets out of the application, then he/she have to re-log in to the application.

Similarly, there were some small errors which nevertheless will affect the user’s data searching.

**Future Maintenance:**

Current version of the Application: 1.01.

The app requires further maintenance for the smooth evaluation of the application. The application requires modifying the following the set of features in the upcoming update:

1. The feature of location will be provided, so that it will be easier for the user to find the homes, directly from their near locations.
2. The views will be made compatible for the users, in order to avoid lagging and invisibility of some views.
3. Some extra features will also be added in the upcoming update like Forgot password option, Delete Post option etc.
4. With upcoming updates, storage property will be surely upgrade.

Hope the up-gradation will not take much time.

**Advantages**

Some of the main advantages of this application were mentioned as follows:

1. Security of the user account is mentioned as we were using the two servers at a time.
2. Genuine users will be there all over the app, as email verification took place before making people sign in to their account.
3. Application interface is made as simple as possible, i.e., simplicity is maintained.
4. Time will be saved, both for the buyers as well as sellers.

And similarly there were many advantages which the user could experience only after using it.

**References**

Android Books:

The busy coder’s guide to advanced android development by Mark L. Murphy

Android Sites:

<https://www.javatpoint.com/android-tutorial>

<https://www.abhiandroid.com>

<https://www.backendless.com/docs/android/>

**Bibliography**

The references mentioned above were the proper source for this application to be developed.

Some other persons helped us in completing this project were- Rahul Garg, Mohit Mishra, Tarun Kumar yadav sir.

I would like to thank all of them who were included in this project.

The End…