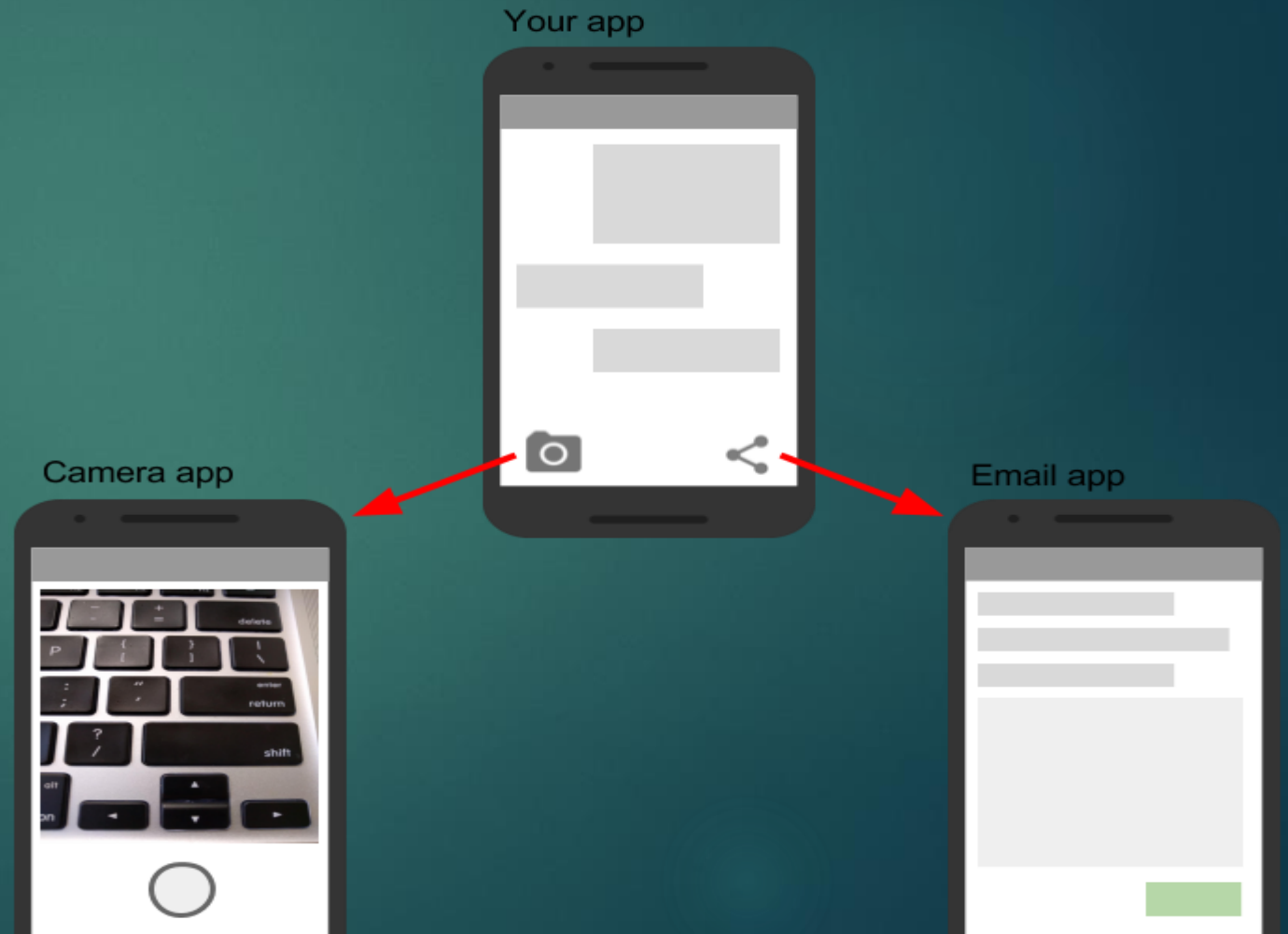
Intent

- Apps can contain more than one Activity
- For example, if your app is simple it can display a list of recipes.

But most of time, you will need to use multiple activities one for displaying the list of recipes and another for adding a single recipe.

- You start an activity by creating an intent and using it in the startActivity() method.
- Android Intent is the message that is passed between components such as activities, content providers, broadcast receivers, services etc. The dictionary meaning of intent is intention or purpose. So, it can be described as the intention to do action.

- A messaging app could start an activity in a camera app to take a picture, then start an activity in an email app to let the user share the picture in email.



Intent Applications

- Generally using intents we can achieve:
  - Sending the User to Another App
  - Getting a Result from an Activity
  - Allowing Other Apps to Start Your Activity
  - Start App Components (Activities, Services, send broadcast).

Types of Intent

- Two types of intent

  i) Implicit intent

  ii) Explicit intent

Implicit intent: Implicit Intent doesn't specify the component(). In such case, intent provides information of available components provided by the system that is to be invoked.

- When you use an implicit intent, Android uses the information in the intent to figure out which components are able to receive it. So you leave the details of which activity performs it to Android. Android does this by checking the **intent filters** in every app's copy of AndroidManifest.xml.

Cont'd

- Intent Filters: specify the kind of Intent your activity will accept.

Intent filters may contain the following three elements

<action>: The Intent action that the activity accepts

<data>: The type of data accepted by activity. The data can be represented by MIME type or other attributes of the data URI.

<category>: the intent category

- The main activity of your app needs an Intent filter that defines the "main" action and the "launcher" category so the system can launch your app.

```
<intent-filters>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
```

This specifies that this is the app's "main" entry point

This activity should be listed in the systems app launcher

Intent Action Types

- Our Application can request different action depending on our interest.
- This are commonly used Action Types

Action Type	Description
ACTION_CALL	Brings up a phone dialer and immediately initiates a call using the number supplied in the Intent's data URI
ACTION_SEND	Also known as the *share* intent, when you have some data that the user can share through another app, such as an email app or social sharing app.
ACTION_VIEW	when you have some information that an activity can show to the user. E.g webpages, photos

Intent Filter Example

E.g. Here's the entry for an activity that can handle an action of ACTION_SEND. The activity is able to accept data with MIME types of text/plain or image:

<activity android:name="ShareActivity">

<intent-filter>

<action android:name="android.intent.action.SEND"/>

<category android:name="android.intent.category.DEFAULT"/>

<data android:mimeType="text/plain"/>

<data android:mimeType="image/*"/>

</intent-filter>

</activity>

N.B If an Activity does not include Intent filters, it can only be launched with an explicit intent

Any Activity that you want to accept an implicit intent must include the android.intent.category.Default intent filter

Implicit Intent Example

- The intent requests applications that can handle send action having textual data.

```
Intent intent = new
Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT,
"Hello");
startActivity(intent)
```

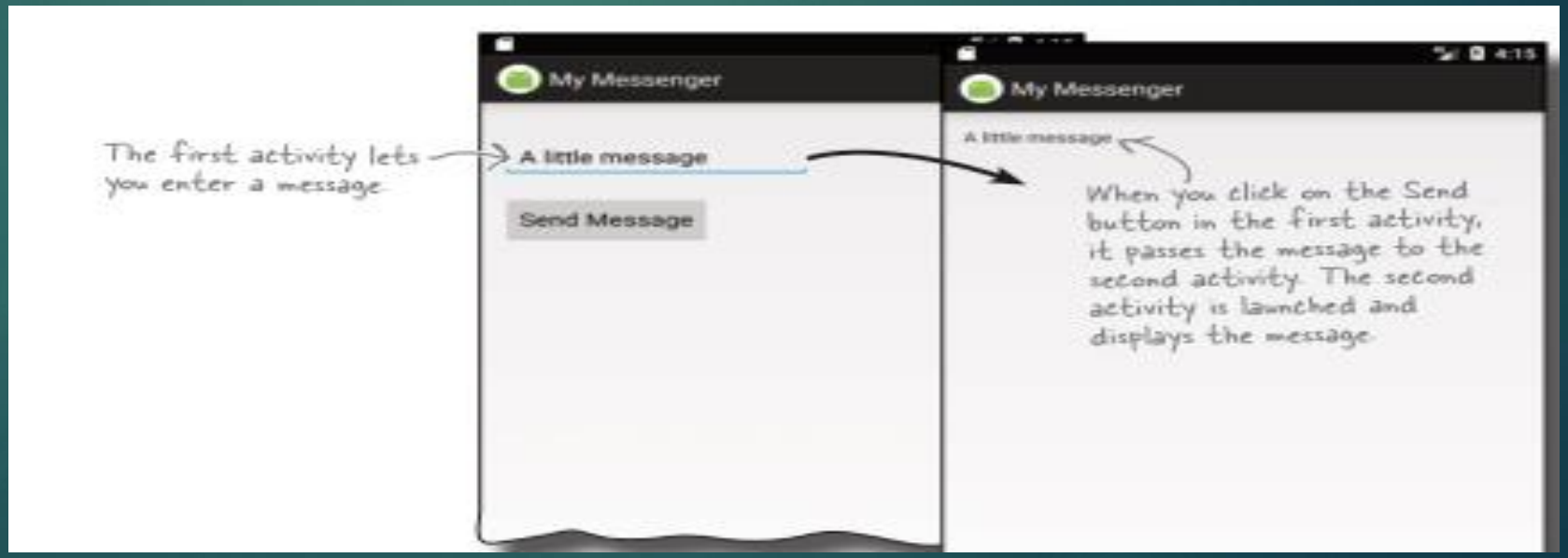
If you always want your user to choose an Activity

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, "Hello");
String chooserTitle = getString(R.string.chooser);
Intent chosen= Intent.createChooser(intent,chooserTitle);
startActivity(chosen);
```

Explicit Intent

- Explicit intent: Explicit Intent specifies the component. In such case, intent provides the external class to be invoked in our app. We are explicitly tell Android which class is expected to run.
- Explicit Intent work internally within an application to perform navigation and data transfer.

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);
startActivity(i);
```



Create the second activity and layout

- In the app/src/main/java folder, choose
- File → New → Activity, and choose the option for Empty Activity. You will be presented with a new screen where you can choose options for your new activity.
- Check the activity in AndroidManifest.xml

Let say your First activity is sendMessageActivity and your second activity is receiveMessageActivity

Send message Activity

```
public void onSendMessage(View view) {
EditText messageView = (EditText)
findViewById(R.id.message);
String messageText =
messageView.getText().toString();
Intent intent = new Intent(this,
ReceiveMessageActivity.class);
intent.putExtra("message", messageText);
startActivity(intent);
}
}
```

Receive message Activity

```
Intent intent = getIntent();
String messageText =
intent.getStringExtra("message");
TextView messageView =
(TextView)findViewById(R.id.message);
messageView.setText(messageText);
```

Getting data from Activity

- The steps to get result from Activity
  - We use Start Activity for result with the Intent and a Request code
  - Create a new Intent in the launched activity and add the return data to that intent
  - Implement onActivityResult in the original activity to process the returned data
- In most cases this method takes two parameters; intent object and request code. To pass the information to another activity and to uniquely identify our request to that activity respectively.
- The request code is an integer that identifies the request and can be used to differentiate between results when you process the return data

For e.g. if you launch one activity to take a photo and another to pick a photo from a gallery, we need different request codes to identify which request the returned data belongs to.

- Our Activity needs to override onActivityResult method that is invoked automatically when the second activity returns result.

onActivityResult() method

- The three arguments to onActivityResult() contain all the information you need to handle the return data.
  - Request code: The request code you set when you launched the Activity with startActivityForResult()
  - Result code: the result code set in the launched Activity, usually one of RESULT_OK or RESULT_CANCELED
  - Intent data: The intent that contains the data returned from the launched Activity.
- In the next, we will see how we get results from an activity.

Get Activity class

final int REQUEST_Code = 2;

```
public void onClick(View view) {
Intent intent = new Intent(this, GiveActivity.class);
intent.putExtra("message", messageText);
startActivityForResult(intent,2);
}

Protected void onActivityResult(int requestCode, int resultCode,Intent
intent){
If (requestCode == REQUEST_Code && resultCode == RESULT_OK){
Textview.setText(intent.getStringExtra("message"));
}
}
```

Give Activity class

```
public void onClick(View view) {
Intent intent = new Intent();
intent.putExtra("message", "Hi");
setResult(RESULT_OK,intent);
finish();
}
```