Introduction to Android Development

winc.cs.ucr.edu/android

Karen Kong x ACM

Overview

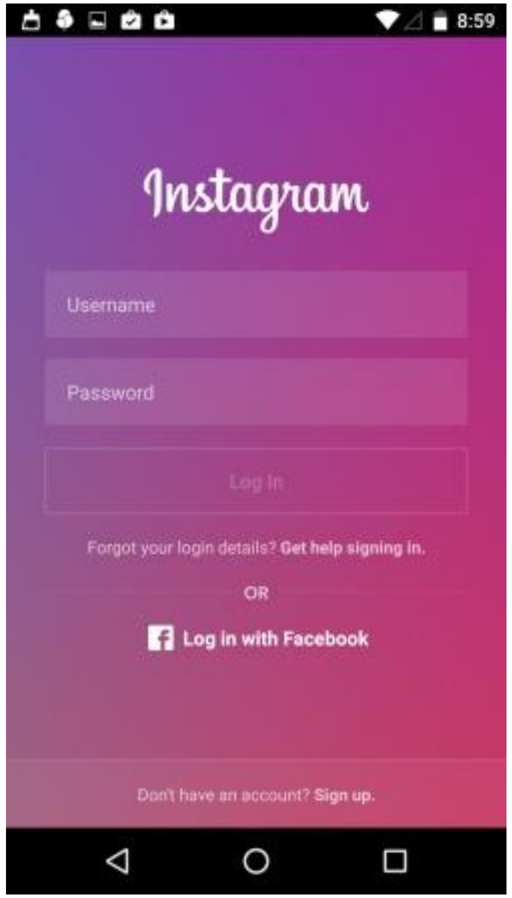
- Java/XML
- Activity
- Context
- Layout
- View Elements
- Manifest
- Gradle

Java/XML

- From C++ to Java crash course
  - Similar to C++
- XML tutorial
  - Similar to HTML

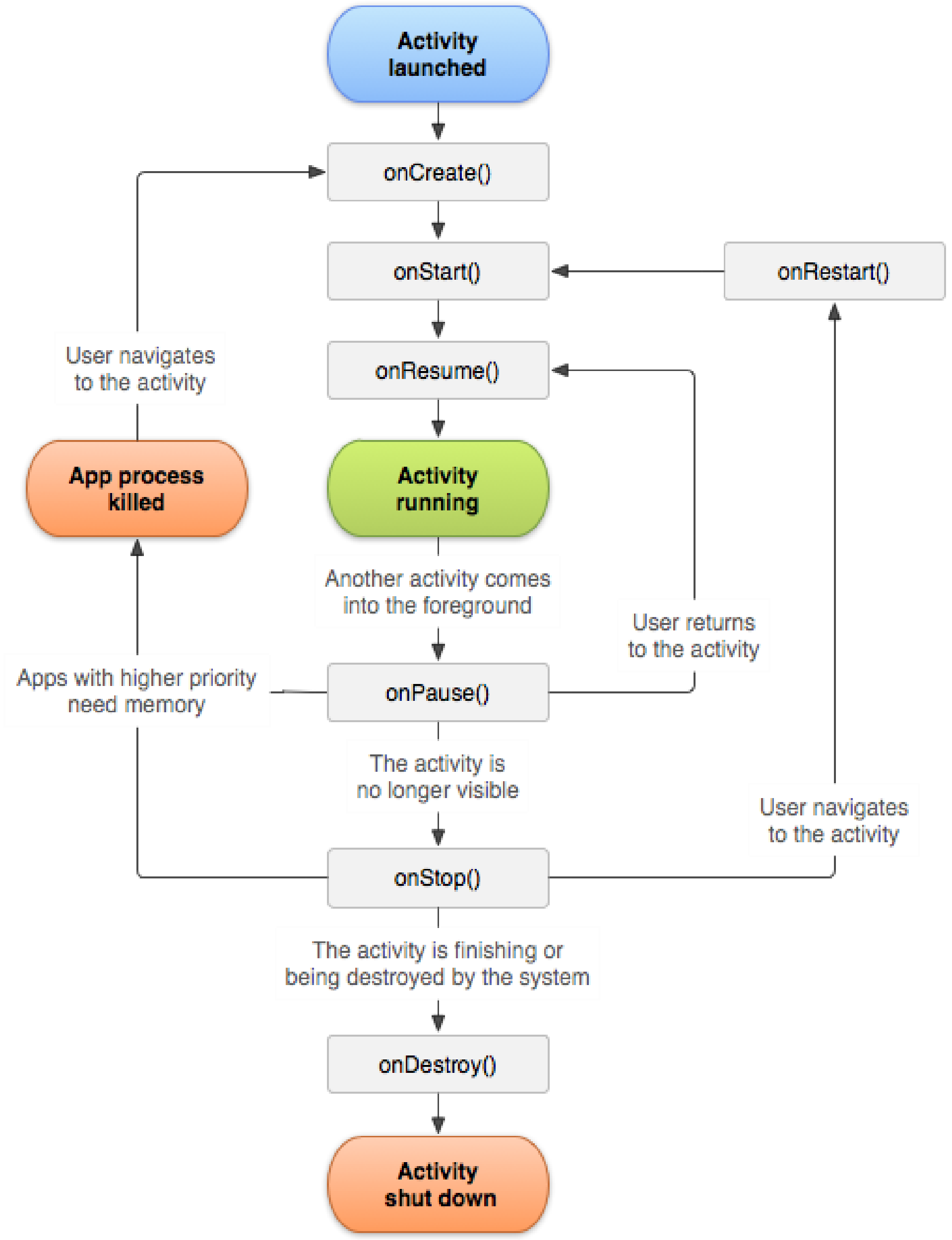
Activity

- A page of the app
- Creates a window to put UI elements
- Arranged in a stack, with the top activity running and other ones paused
- Fragments inside activities can better modularize code
- app -> java -> package -> __Activity


8:59
Instagram
Username
Password
Log In
Forgot your login details? Get help signing in.
OR
Log in with Facebook
Don't have an account? Sign up.

Activity Lifecycle

- Active
  - Visible, in focus
- Paused
  - Visible, not in focus
- Stopped
  - Not visible, not in focus
- Killed
  - Removed from memory during paused or stopped state by system or user



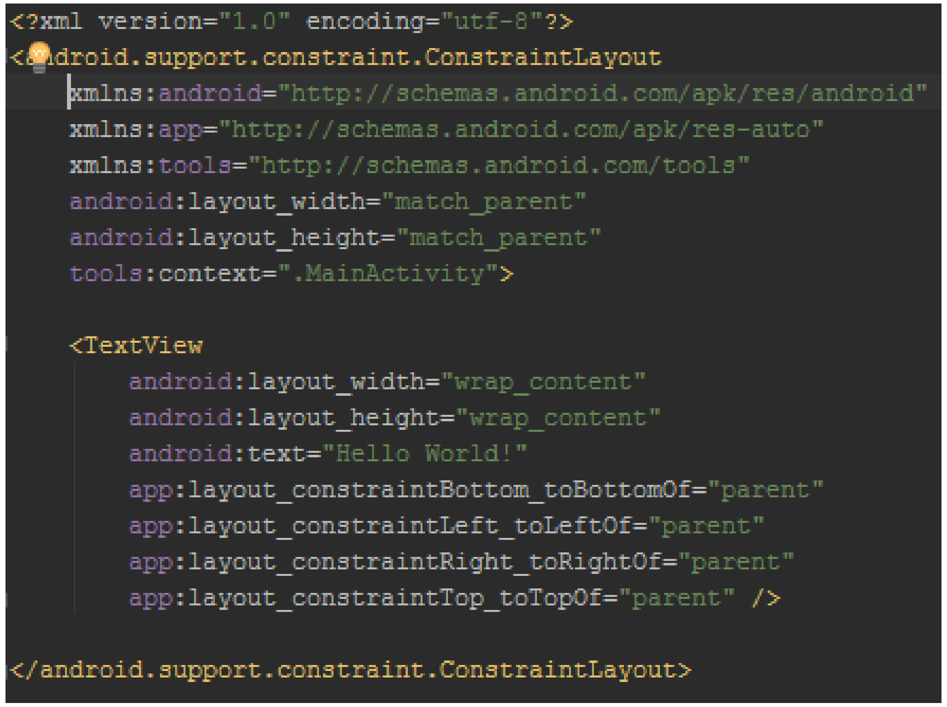
Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Context

- Global resources
- Access to resources and classes
- In an Activity, access context through the keyword “this”
- In an inner class, getApplicationContext()
- Can pass context from an Activity to non-activity classes

Layout

- Structure for UI
  - Views: widgets, like Buttons, TextViews, etc.
  - ViewGroup: layouts, like Linear, Constraint, etc.
- Declare UI elements in XML or instantiate at runtime
  - Each element has attributes
- Identify view elements using the ID attribute
- app -> res -> layout -> __.xml



```xml
<?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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

View Elements

- TextView: displays text
- EditText: user enters text
- Button: captures presses
- ListView: a list of items
  - More memory efficient to use a RecyclerView
- To access view elements from activity code, bind the view elements to Java variables
  - `TextView tv = findViewById(R.id.tv);`

Manifest

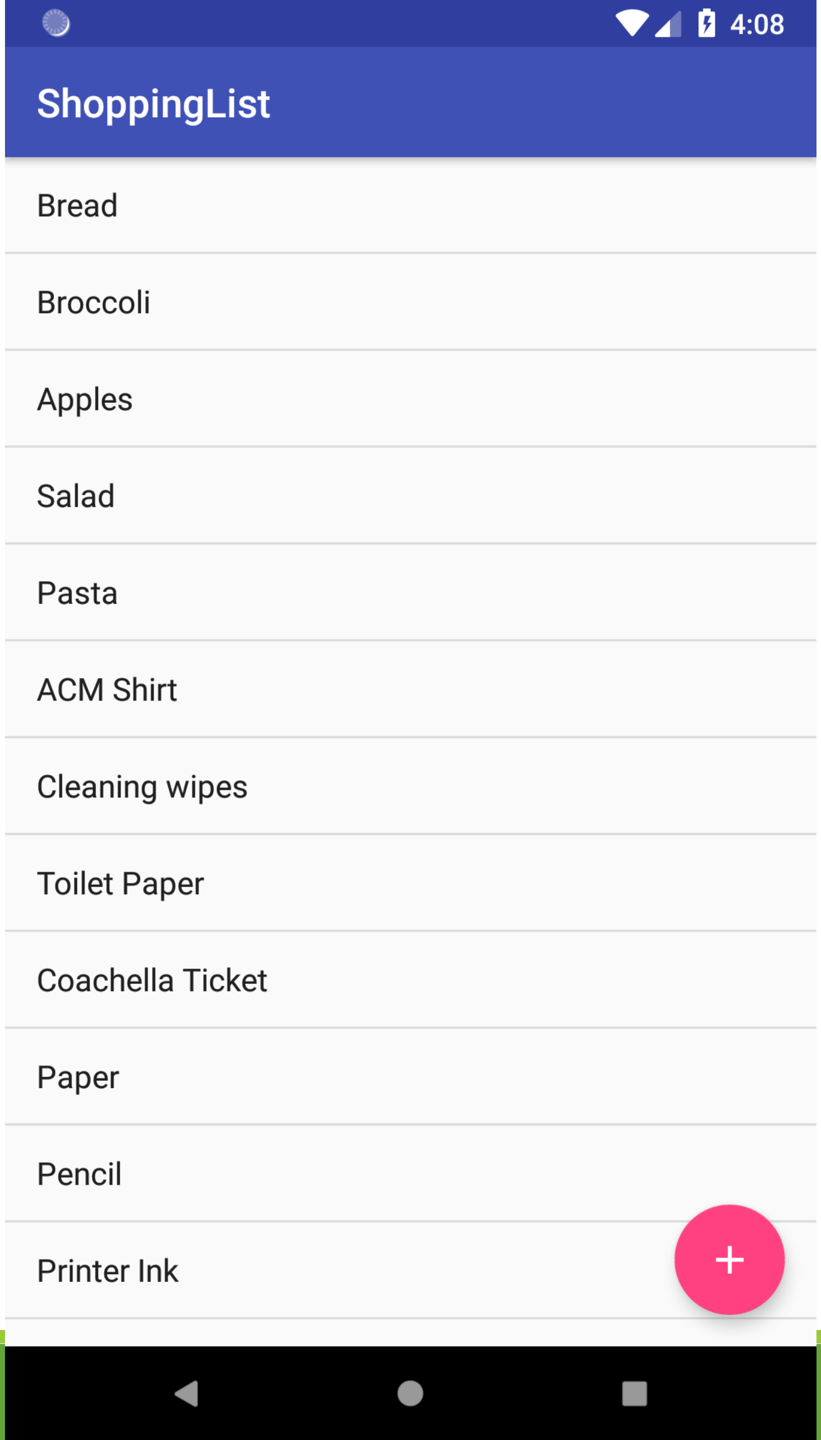
- app -> manifests -> AndroidManifest.xml
- App information for build tools, Android, and Google Play to use
- Package name
- Activities
- Services
- Permissions (internet, wakelock, camera, bluetooth…)

Gradle

- Build tool
- Gradle Scripts -> build.gradle (Module: app)
- SDK (Android) versions supported by the app
- Dependencies
  - `implementation 'com.android.support:design:27.1.1'`

Shopping List App

- Show shopping list
- Add new items to shopping list
- Firebase integration to save items



Bread

Broccoli

Apples

Salad

Pasta

ACM Shirt

Cleaning wipes

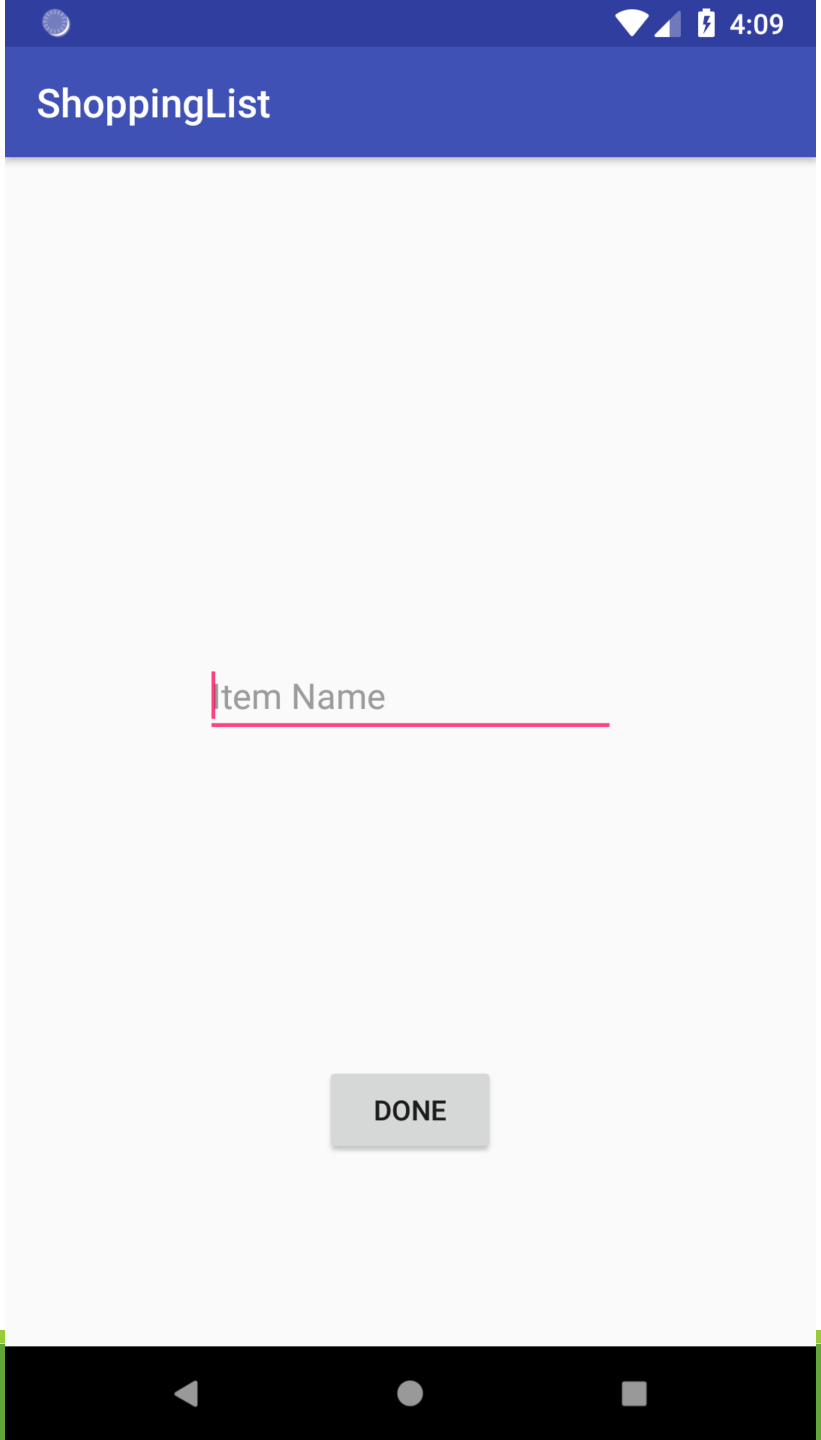
Toilet Paper

Coachella Ticket

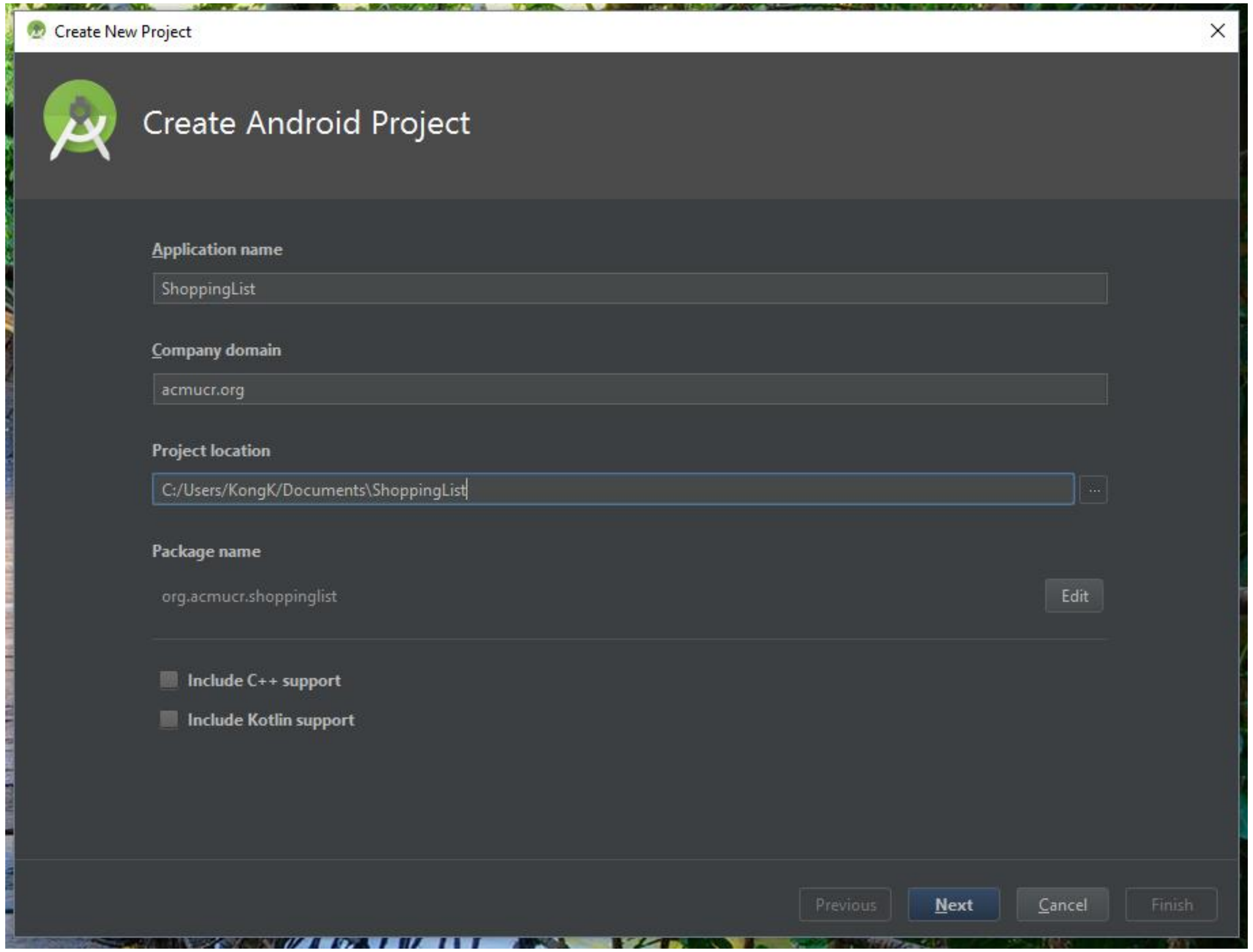
Paper

Pencil

Printer Ink



Create the Project



Application name
ShoppingList

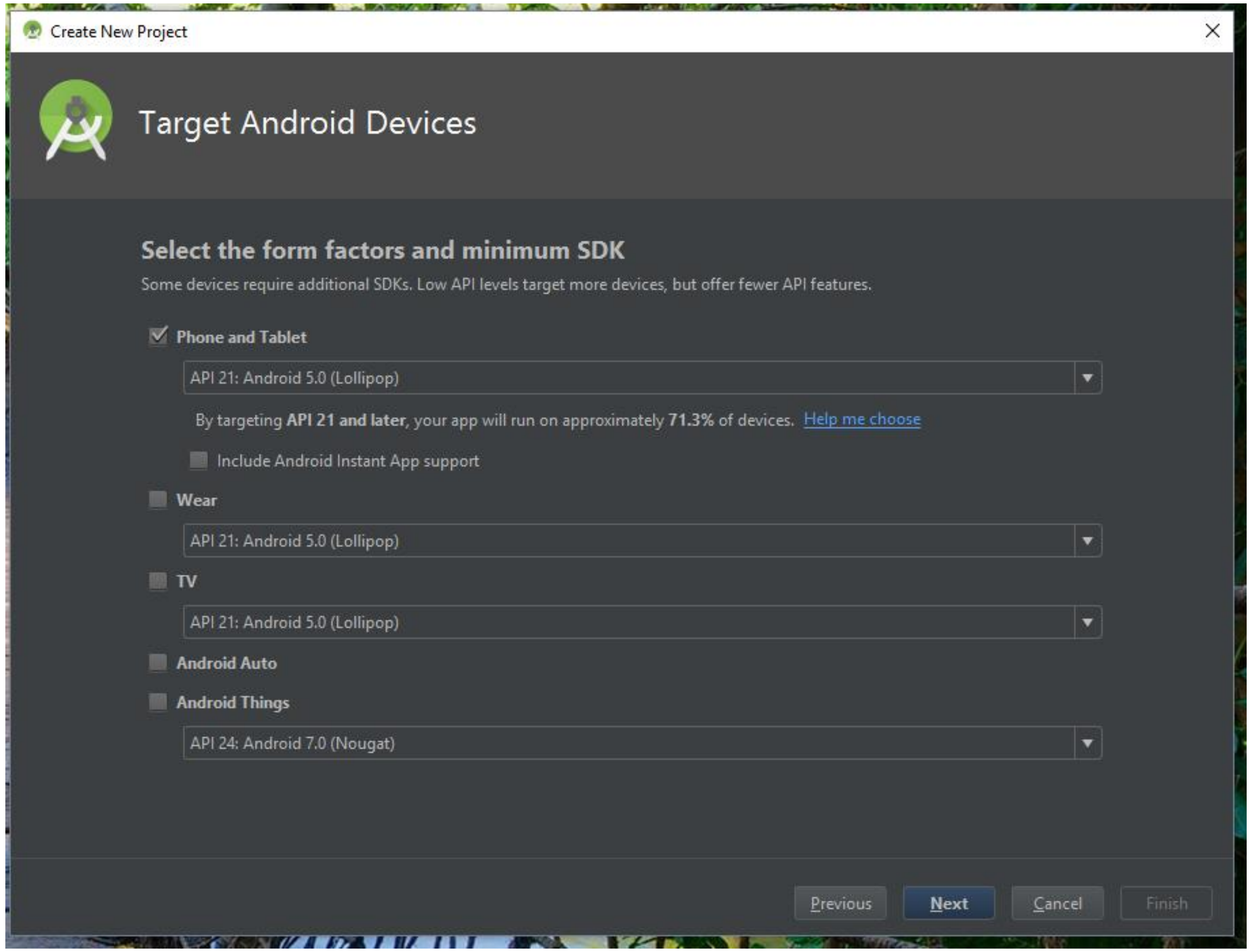
Company domain
acmucr.org

Project location
C:/Users/KongK/Documents\ShoppingList

Package name
org.acmucr.shoppinglist

Include C++ support
Include Kotlin support

Choose the Target Android Devices


Create New Project

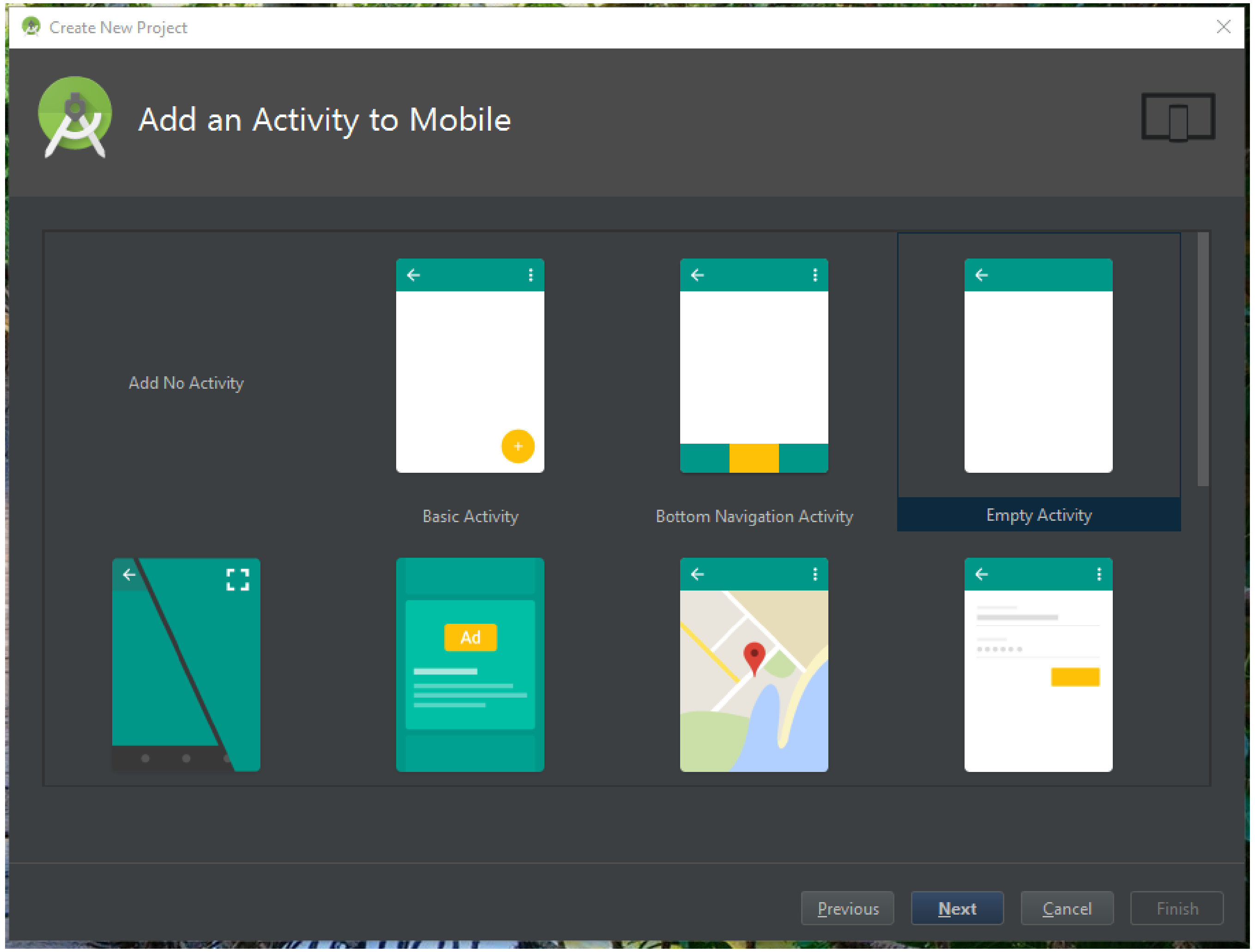
Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

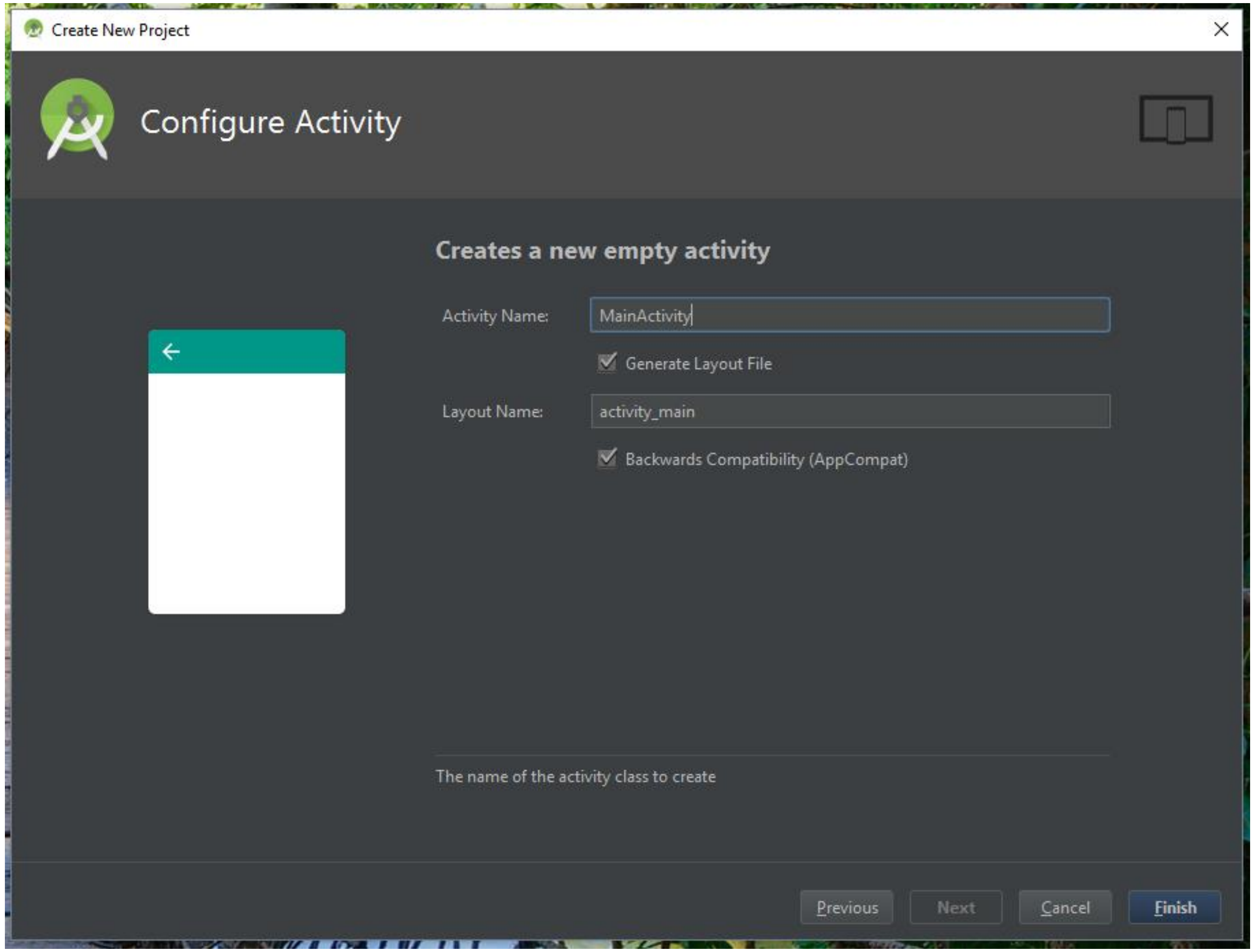
- [✓] **Phone and Tablet**
  - API 21: Android 5.0 (Lollipop)
  - By targeting **API 21 and later**, your app will run on approximately **71.3%** of devices. Help me choose
  - [ ] Include Android Instant App support
- [ ] **Wear**
  - API 21: Android 5.0 (Lollipop)
- [ ] **TV**
  - API 21: Android 5.0 (Lollipop)
- [ ] **Android Auto**
- [ ] **Android Things**
  - API 24: Android 7.0 (Nougat)

Choose Activity Template


Create New Project

Add an Activity to Mobile

Name Activity



Creates a new empty activity

Activity Name: MainActivity

[✓] Generate Layout File

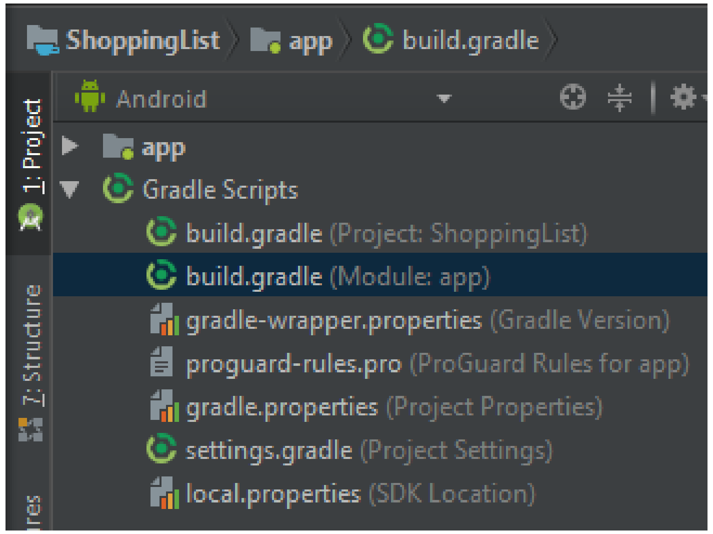
Layout Name: activity_main

[✓] Backwards Compatibility (AppCompat)

The name of the activity class to create

Add Gradle Dependencies

- Gradle Scripts -> build.gradle (Module: app)



build.gradle (Project: ShoppingList)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)

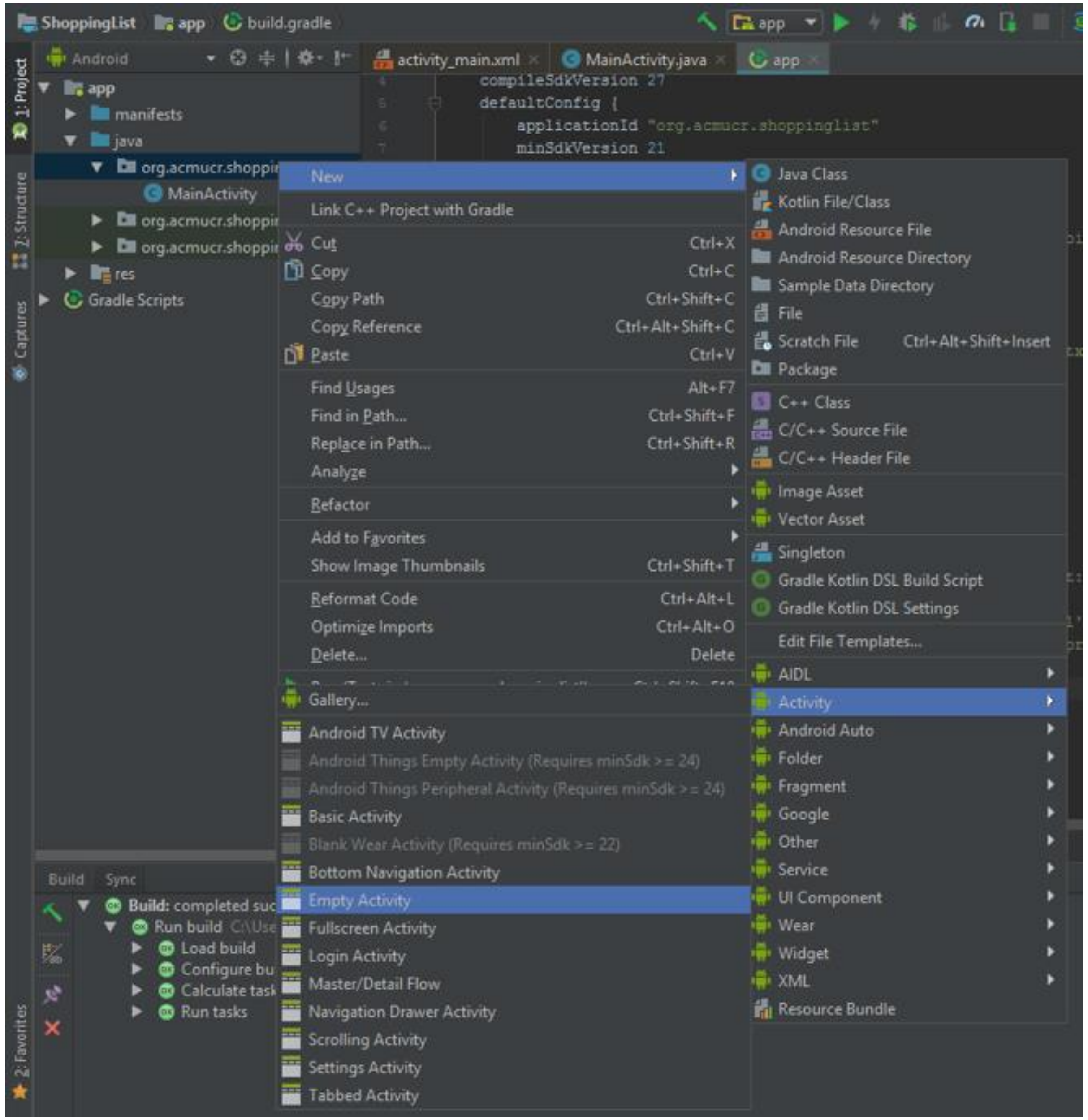
- Add to dependencies {…}

```
implementation 'com.android.support:design:27.1.1'
```

- Press “Sync Now”

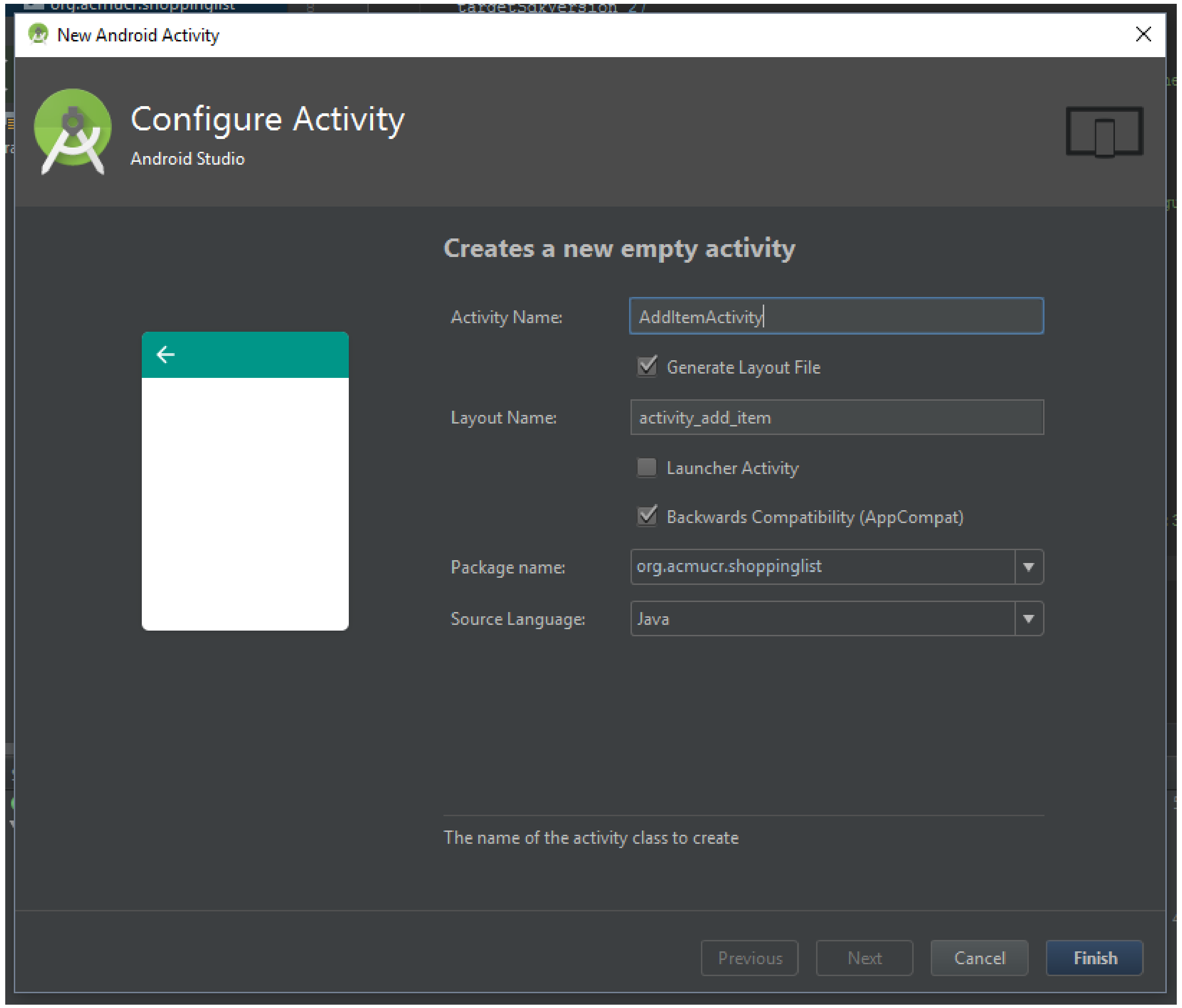
Add AddItemActivity

- app -> java -> right click on first folder
- New -> Activity -> Empty Activty



```
compileSdkVersion 27
defaultConfig {
    applicationId "org.acmucr.shoppinglist"
    minSdkVersion 21
```

Add AddItemActivity


New Android Activity

Configure Activity

Android Studio

Creates a new empty activity

Activity Name: AddItemActivity

[✓] Generate Layout File

Layout Name: activity_add_item

[ ] Launcher Activity

[✓] Backwards Compatibility (AppCompat)

Package name: org.acmucr.shoppinglist

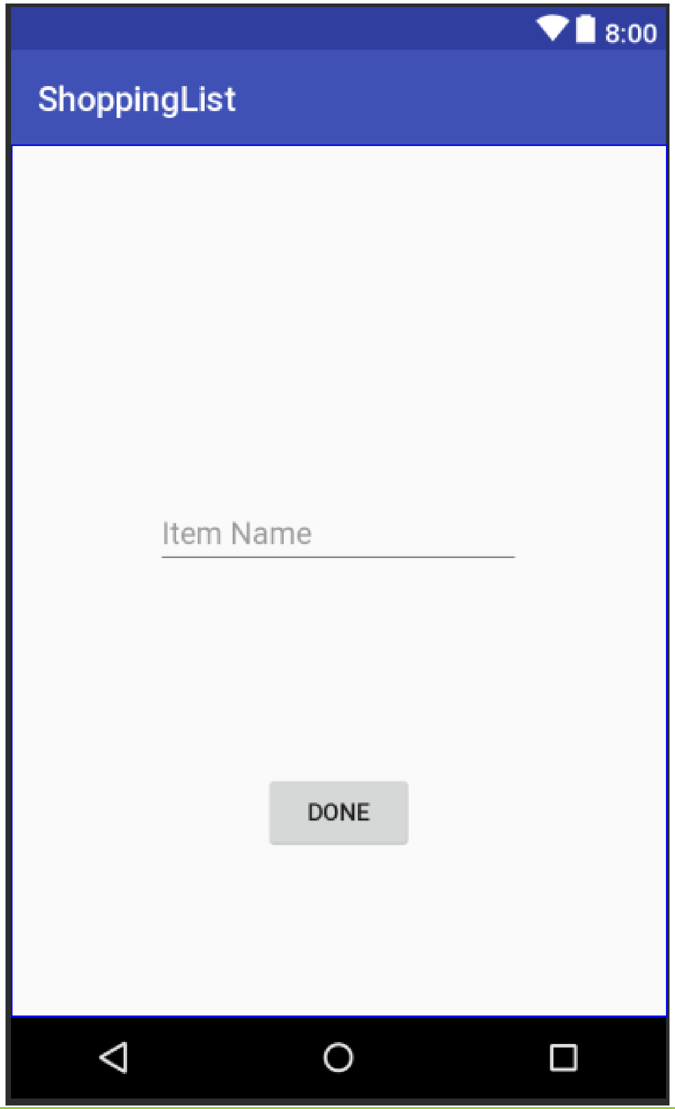
Source Language: Java

The name of the activity class to create

Design Add Item Activity Layout

- res -> layout -> activity_add_item.xml
- Add an EditText
- Add a Button

activity_add_item.xml

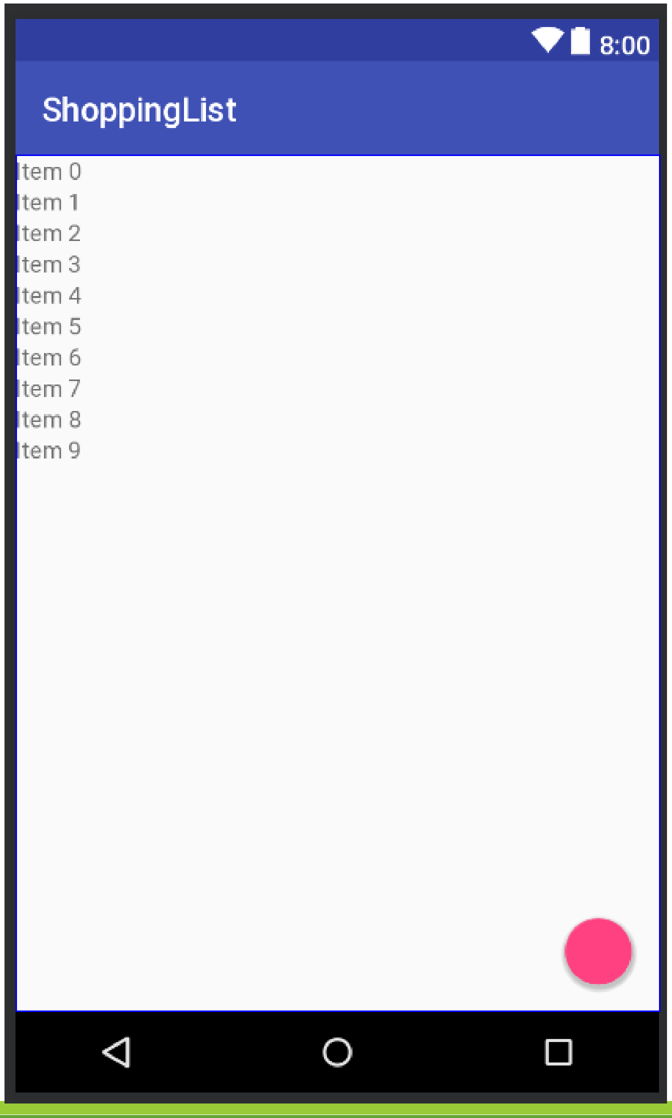


```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddItemActivity">

    <EditText
        android:id="@+id/etItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginTop="96dp"
        android:ems="10"
        android:hint="Item Name"
        android:inputType="text"
        app:layout_constraintBottom_toTopOf="@+id/btDone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btDone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="96dp"
        android:text="Done"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

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

Design the Main Activity Layout

- res -> layout -> activity_main.xml
- Add a ListView
- Add a FloatingActionButton

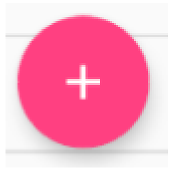
activity_main.xml



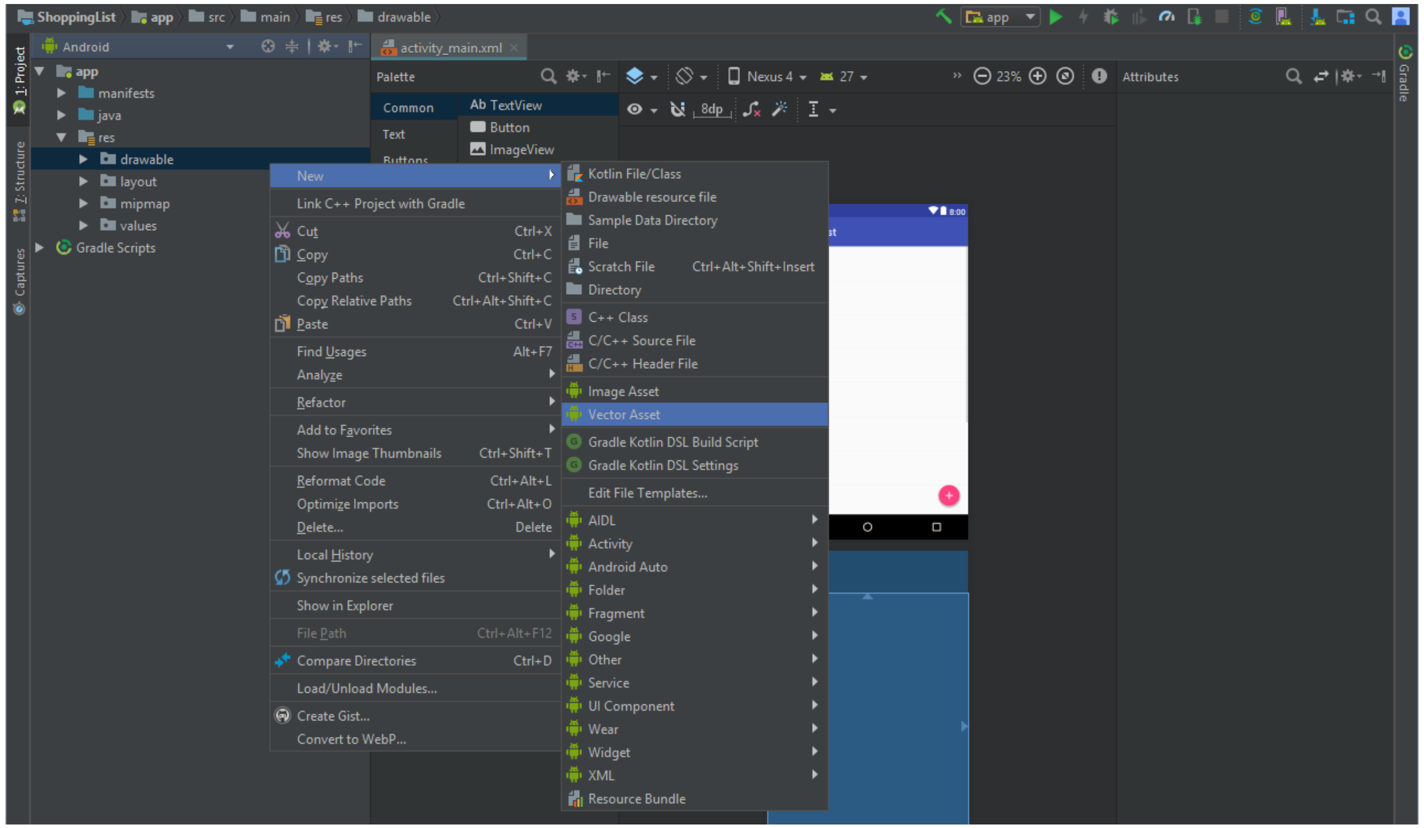
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/rvShoppingList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/btAddItem"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

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

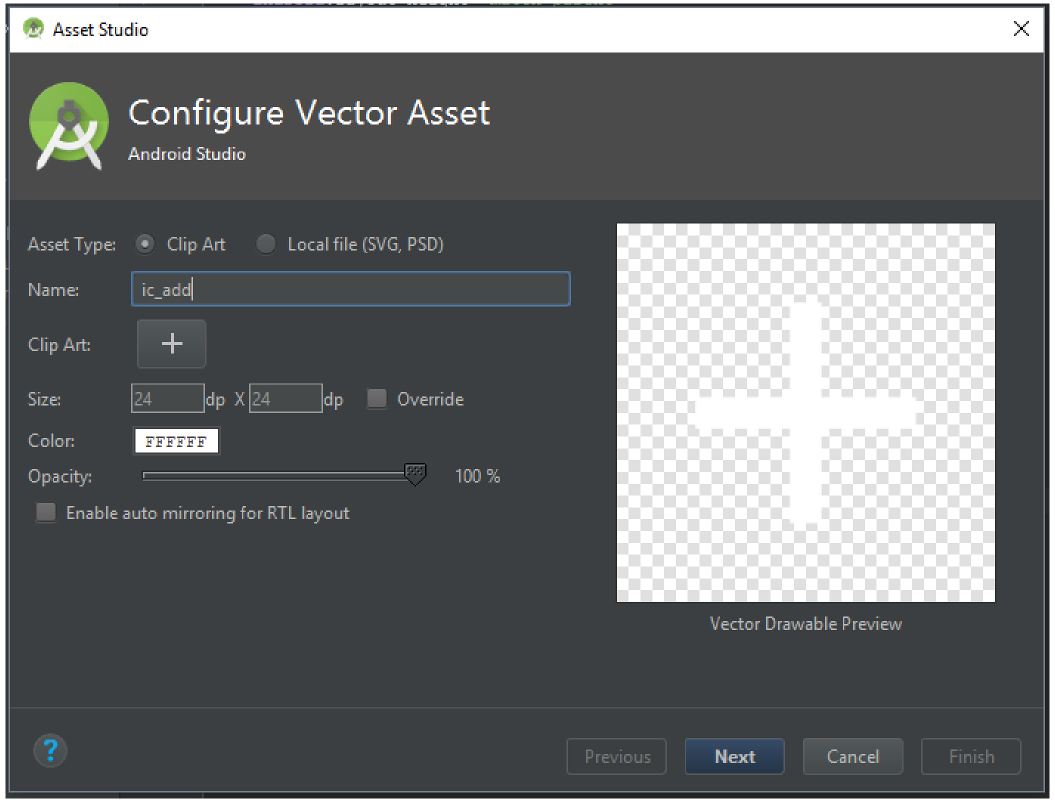
Design the Button

- Add a vector asset to put a “+” on the button



Design the Button




Asset Studio

Configure Vector Asset

Android Studio

Asset Type: [✓ Clip Art] Local file (SVG, PSD)

Name: ic_add

Clip Art:

Size: 24 dp X 24 dp Override

Color: FFFFFF

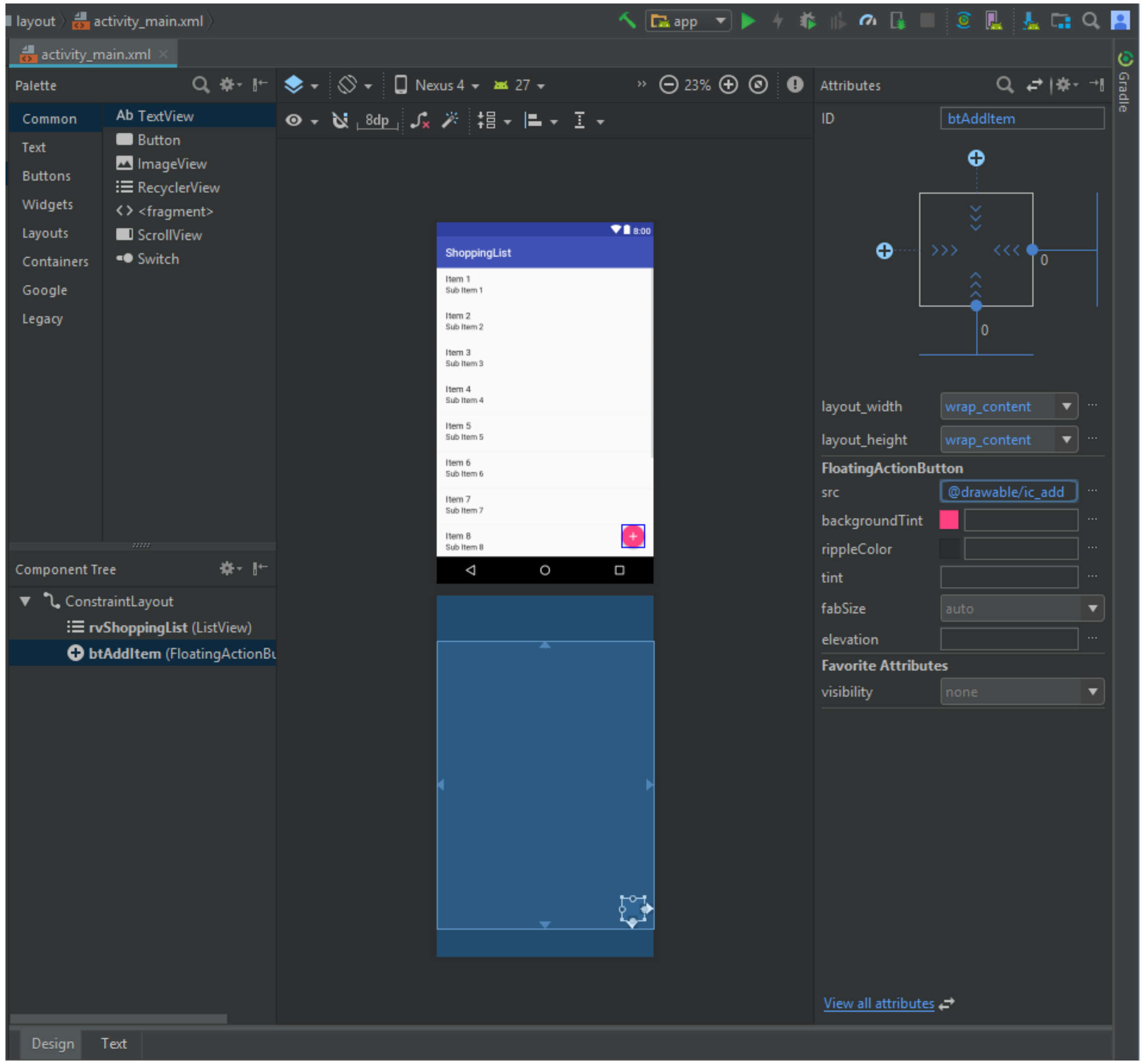
Opacity: 100 %

Enable auto mirroring for RTL layout

Design the Button

- Specify the image resource for the button
- Set the src attribute to @drawable/ic_add

Design the Button



Component Tree

- ConstraintLayout
  - rvShoppingList (ListView)
  - btAddItem (FloatingActionBu

Attributes

ID	btAddItem
layout_width	wrap_content
layout_height	wrap_content

FloatingActionButton

src	@drawable/ic_add
backgroundTint	🟥
rippleColor	
tint	
fabSize	auto
elevation	

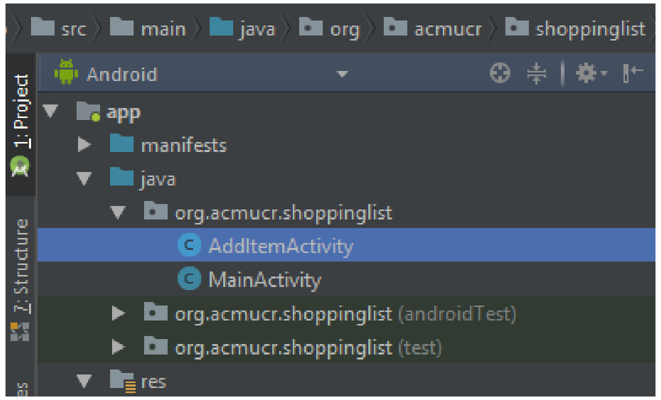
Favorite Attributes

visibility	none

View all attributes

Write the Controller for AddItemActivity

- app -> java -> first folder -> AddItemActivity.java



- Bind the view elements to variables

```java
public class AddItemActivity extends AppCompatActivity {

    private EditText etItem;
    private Button btDone;

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

        // Bind the variables to the view elements
        etItem = findViewById(R.id.etItem);
        btDone = findViewById(R.id.btDone);
    }
}
```

Write the Controller for AddItemActivity

- Add an OnClick to the Done button in onCreate()
- Pass the text the user entered back to the MainActivity

```java
// Add the OnClick listener to the button
btDone.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Get the text the user entered
        String itemName = etItem.getText().toString();

        // Pass the information and transition back to the MainActivity
        Intent mainActivityIntent = new Intent(getApplicationContext(), MainActivity.class);
        mainActivityIntent.putExtra("ITEM_NAME", itemName);
        setResult(RESULT_OK, mainActivityIntent);
        finish();
    }
});
```

Write the Controller for Main Activity

- app -> java -> first folder -> MainActivity.java
- Bind the view elements to variables

```java
public class MainActivity extends AppCompatActivity {

    private ListView lvShoppingList;
    private FloatingActionButton btAddItem;

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

        // Bind the variables to the view elements
        lvShoppingList = findViewById(R.id.rvShoppingList);
        btAddItem = findViewById(R.id.btAddItem);

    }
}
```

Write the Controller for Main Activity

- Add a REQUEST_CODE to MainActivity

```
private static final int REQUEST_CODE = 20;
```

- Add the OnClick listener to the button to transition to the AddItemActivity

```
// OnClick listener for the button to transition to AddItemActivity
btAddItem.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent addItemActivityIntent = new Intent(getApplicationContext(), AddItemActivity.class);
        startActivityForResult(addItemActivityIntent, REQUEST_CODE);
    }
});
```

Write the Controller for Main Activity

- Add a list to hold items and adapter to MainActivity

```
private List<String> itemNames;
private ArrayAdapter<String> itemsAdapter;
```

- Set up an adapter for the ListView in onCreate()

```
// Initialize the list of names
itemNames = new ArrayList<>();

// Create the adapter to send the list of names to the list view
itemsAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, itemNames);

// Set the list view's adapter
lvShoppingList.setAdapter(itemsAdapter);
```

Write the Controller for Main Activity

- Override onActivityResult to get the text passed from AddItemActivity
- Add that text to the list of items and alert the adapter

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
        // If returning from AddItemActivity, get the text the user entered
        String itemName = data.getStringExtra("ITEM_NAME");

        // Add the item to the list and notify the adapter
        itemNames.add(itemName);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

Write the Controller for Main Activity

- Set an onLongClickListener on the ListView in onCreate() so a long press deletes an item

```
// Remove an item from the list if the user long presses on it
lvShoppingList.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
    @Override
    public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
        // Remove the clicked item from the ListView and notify the adapter
        itemsAdapter.remove(itemNames.get(position));
        itemsAdapter.notifyDataSetChanged();
        return false;
    }
});
```

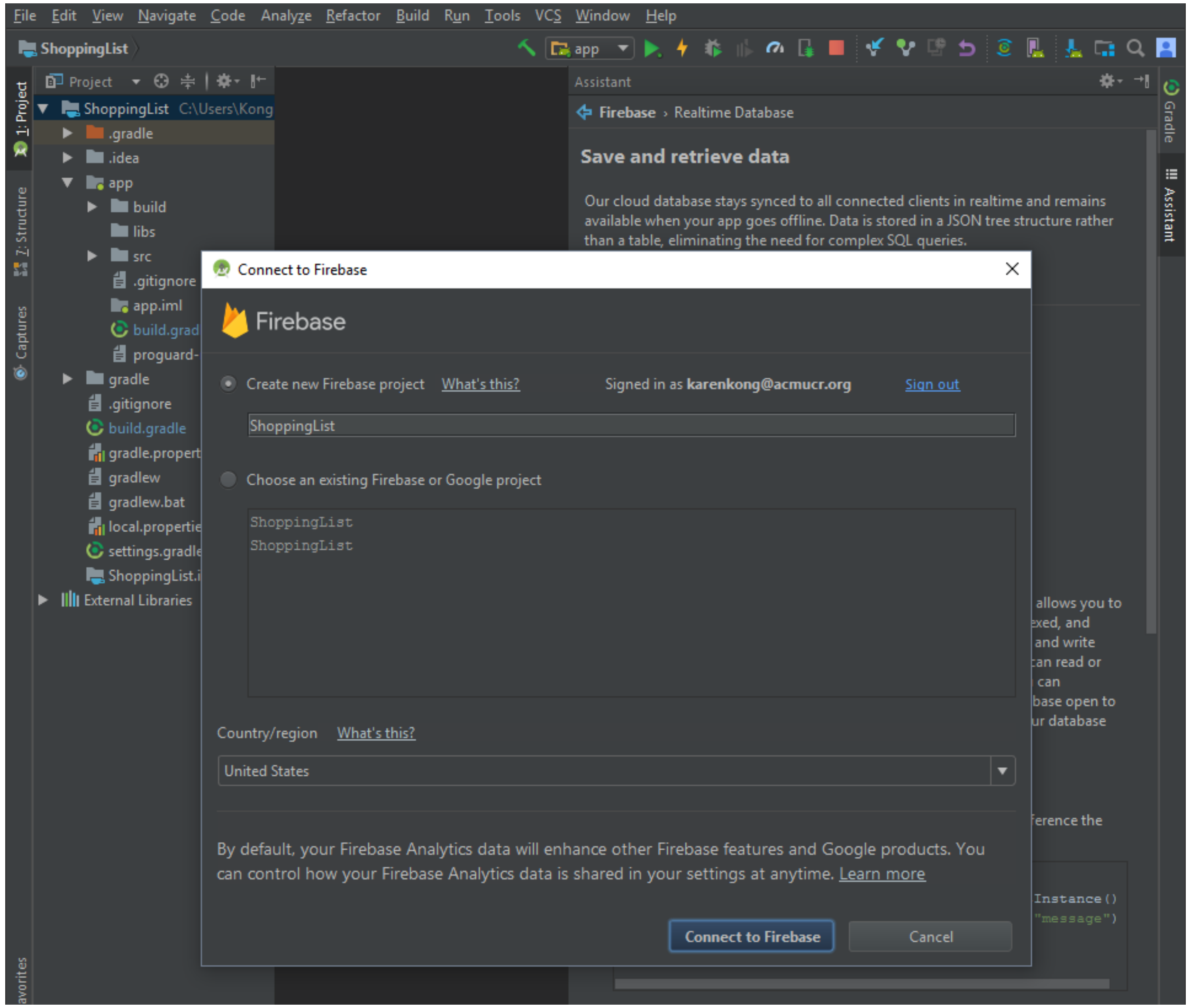
Firebase

- Backend as a service
- Ideal for fast development
- Easier to set up than a database + API
- Versus traditional backend options
  - Generally more expensive
  - Can be harder to scale
- Realtime Database
  - Stores data in JSON file

Firebase Integration

- Tools -> Firebase
- Connect to Firebase
- Sign in to your Google account
- Create new Firebase project

Firebase Integration



Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

Connect to Firebase

Firebase

Create new Firebase project What's this? Signed in as karenkong@acmucr.org Sign out

ShoppingList

Choose an existing Firebase or Google project

```
ShoppingList
ShoppingList
```

Country/region What's this?

United States

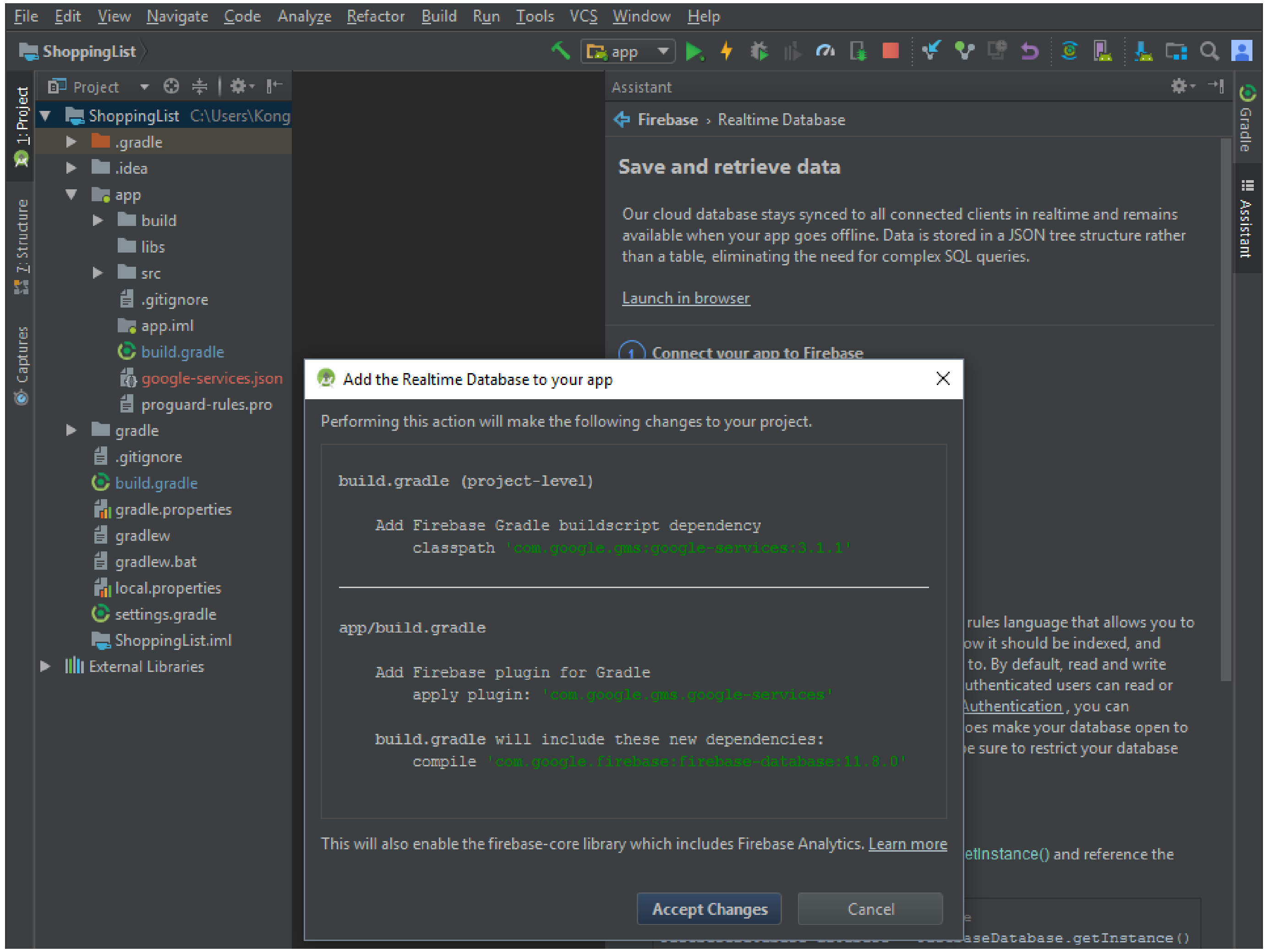
By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. Learn more

Connect to Firebase Cancel

Firebase Integration

- Add the Realtime Database to your App
- Accept Changes
- (Sync Now)

Firebase Integration



Assistant

Firebase › Realtime Database

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

Launch in browser

1 Connect your app to Firebase

Add the Realtime Database to your app

Performing this action will make the following changes to your project.

```
build.gradle (project-level)

    Add Firebase Gradle buildscript dependency
        classpath 'com.google.gms:google-services:3.1.1'

app/build.gradle

    Add Firebase plugin for Gradle
        apply plugin: 'com.google.gms.google-services'

    build.gradle will include these new dependencies:
        compile 'com.google.firebase:firebase-database:11.8.0'
```

This will also enable the firebase-core library which includes Firebase Analytics. Learn more

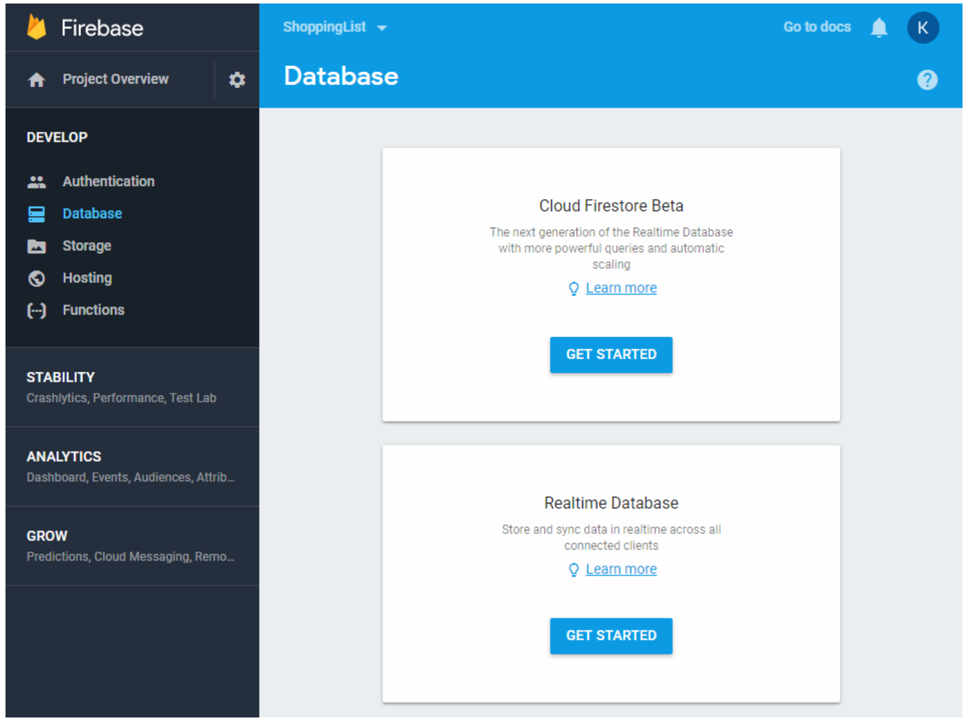
rules language that allows you to
ow it should be indexed, and
to. By default, read and write
uthenticated users can read or
Authentication, you can
oes make your database open to
e sure to restrict your database

etInstance() and reference the

aseDatabase.getInstance()

Firebase Integration

- Go to the Firebase console
- Open the ShoppingList project
- Develop -> Database -> Realtime Database -> Get Started



Cloud Firestore Beta

The next generation of the Realtime Database with more powerful queries and automatic scaling

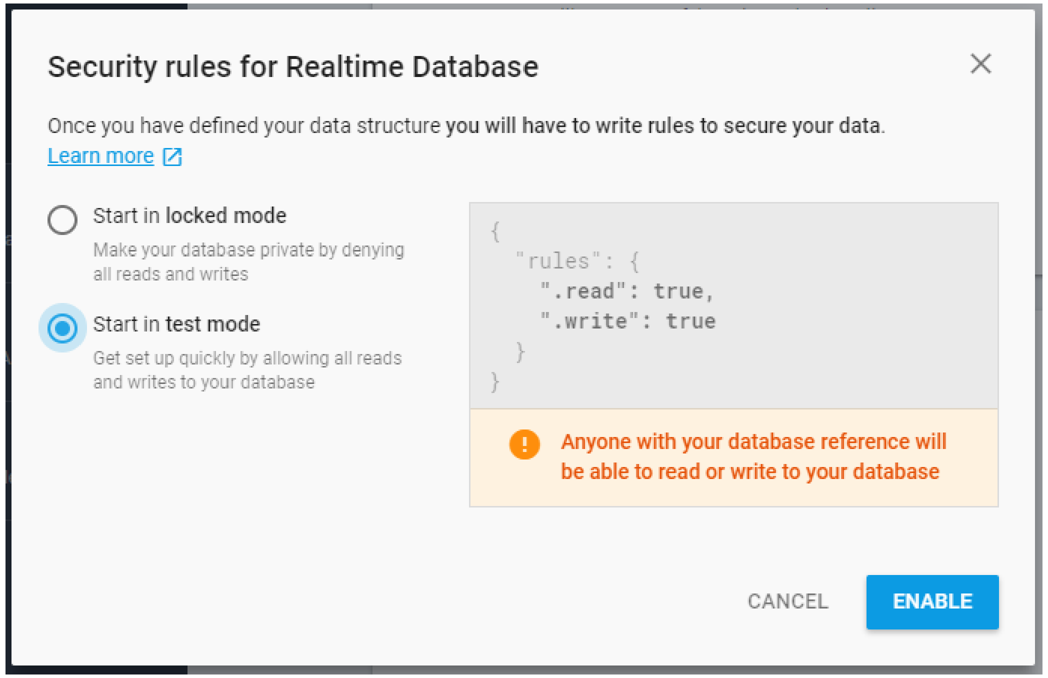
Learn more

Realtime Database

Store and sync data in realtime across all connected clients

Learn more

Firebase Integration



Security rules for Realtime Database

Once you have defined your data structure you will have to write rules to secure your data.
Learn more

( ) Start in locked mode
Make your database private by denying all reads and writes

[✓ Start in test mode]
Get set up quickly by allowing all reads and writes to your database

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Anyone with your database reference will be able to read or write to your database

Firebase Integration

- Add a DatabaseReference to MainActivity

```
private DatabaseReference dbReference;
```

- Get a reference to the root of your Firebase database in onCreate()

```
// Get a reference to the root of the Firebase database
dbReference = FirebaseDatabase.getInstance().getReference();
```

Firebase Integration

- Add a ValueEventListener to fetch the items from the Firebase database in onCreate()

```java
// Pull the items from the Firebase database and load them into the ListView
dbReference.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        for (DataSnapshot d : dataSnapshot.getChildren()) {
            String item = d.getValue(String.class);
            itemNames.add(item);
        }
        itemsAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Getting the items from the database failed
    }
});
```

Firebase Integration

- Add the item to the Firebase database when adding to the ListView
- In onActivityResult(), push the new item to the Firebase database

```
// Add the item to the Firebase database
dbReference.push().setValue(itemName);
```

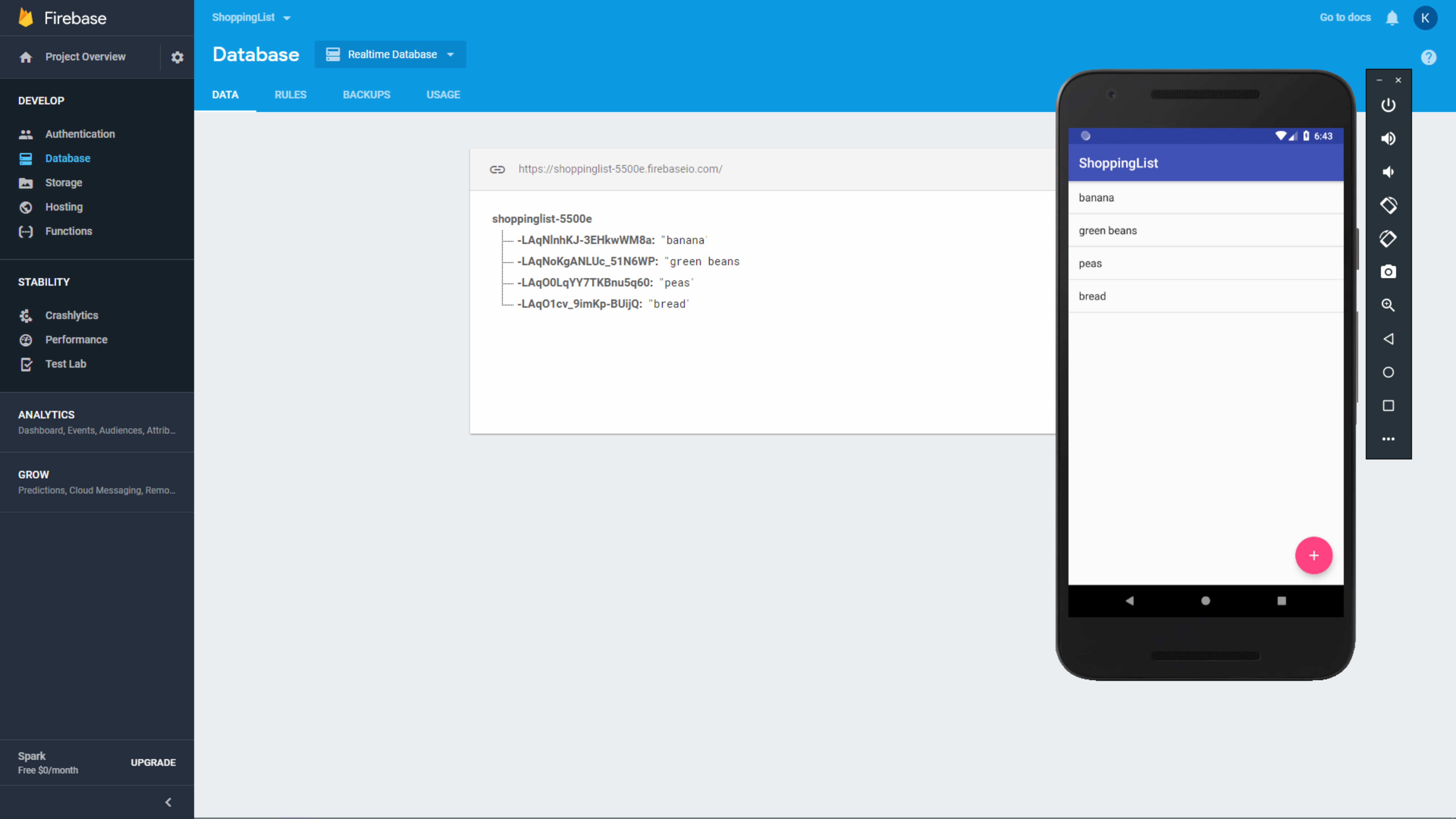
Firebase Integration

- Remove the item from the Firebase database when removing from the ListView
- In onCreate(), remove the item from the Firebase database in onItemLongClick()

```java
// Remove the item from the Firebase database
final String clickedItem = itemNames.get(position);
dbReference.orderByValue().equalTo(clickedItem).addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        for(DataSnapshot d : dataSnapshot.getChildren()) {
            if(d.getValue().toString().equals(clickedItem)) {
                dbReference.child(d.getKey()).removeValue();
                break;
            }
        }
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Getting the items from the database failed
    }
});
```

Final Result

- Press the green play button or Run -> Run app to see your app in action!
- Full code: https://github.com/acm-ucr/intro-to-android
- See the live updates in your Firebase project's Database: https://console.firebase.google.com/



https://shoppinglist-5500e.firebaseio.com/

```
shoppinglist-5500e
├── -LAqNlnhKJ-3EHkwWM8a: "banana"
├── -LAqNoKgANLUc_51N6WP: "green beans
├── -LAqO0LqYY7TKBnu5q60: "peas"
└── -LAqO1cv_9imKp-BUijQ: "bread"
```

banana

green beans

peas

bread

Debugging

- ~~System.out.println()~~
- Logcat
- Android Studio Debugger
- Android Profiler

Resources

- Android Documentation: https://developer.android.com/guide/index.html
- Firebase Documentation: https://firebase.google.com/docs/guides/
- Firebase Live Chat App Tutorial: https://codelabs.developers.google.com/codelabs/firebase-android
- UI/UX: https://developer.android.com/design/index.html
- Additional Guides
    - https://www.tutorialspoint.com/android/index.htm
    - https://guides.codepath.com/android
    - Searching for Android tutorials on Youtube