How to Create Contacts App in Android Studio?

Last Updated : 28 Mar, 2025

Contacts app in android device is a system app that comes installed on your android device. Different devices have different UI for the contacts app. In this article, we will take a look at how we can build our own contacts app in Android Studio.

What we are going to build in this article?

We will be building a simple application in which we will be displaying the list of contacts that are stored in the user's device. Along with that we will be also adding a feature to save a new contact, filter contacts from the list using the search bar, and many more. Below is the video in which we will get to see what we are going to build in this article.

Step by Step Implementation

Step 1: Create a New Project

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio.

> ***Note:*** *that select Java/Kotlin as the programming language.*

Step 2: Add dependency and JitPack Repository

As we will have to request the user's permissions to display contacts from the device in the app, that we have to ask for the user's permissions. So for getting user's permissions we will be using **Dexter** for getting user's permissions in runtime. For using **Dexter**, we have added the below dependency in

build.gradle.kts file. Navigate to the **Gradle Scripts > build.gradle.kts (Module:app)** and add the below dependency in the dependencies section.

```
dependencies {
    ...
    implementation ("com.karumi:dexter:6.2.2")
}
```

Add the JitPack repository to **settings.gradle.kts**

```
dependencyResolutionManagement {
    ...
    repositories {
        ...
        maven { url = uri("https://jitpack.io") }
    }
}
```

Now **sync** your project.

Step 3: Adding permissions in the AndroidManifest.xml

Navigate to the **app > manifests > AndroidManifest.xml** file and add the below permissions to it.

AndroidManifest.xml:

```
<uses-feature
    android:name="android.hardware.telephony"
    android:required="false" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
```

Step 4: Working with the Main Layout

Navigate to the **app > res > layout > activity_main.xml** and add the below code to that file. Now, create a new layout file for each item of the recyclerview.

Set the name as **contact_rv_item.xml** and add the below code in the file.

activity_main.xml | contact_rv_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">

    <!--Recycler view for displaying list of contacts-->
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/contacts_rv_item"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <!--fab for adding a new contact-->
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/addButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="20dp"
        android:backgroundTint="@color/colorPrimary"
        android:importantForAccessibility="no"
        android:src="@android:drawable/ic_input_add"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:tint="@color/white" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 5: Creating a modal class for each contacts

Navigate to the **app > java > {package-name}**, Right-click on it, **New > Java/Kotlin class** and name it as **Contacts** and add below code to it. Comments are added in the code to get to know in more detail.

Contacts.java | Contacts.kt

```
package org.geeksforgeeks.demo;

public class Contacts {

    // user name and contact number.
    private String userName;
    private String contactNumber;

    public Contacts(String userName, String contactNumber) {
        this.userName = userName;
        this.contactNumber = contactNumber;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }
}
```

Step 6: Creating an adapter class

Navigate to the **app > java > {package-name}**, Right-click on it, **New > Java/Kotlin class** and name it as **Adapter** and add the below code to it. Comments are added in the code to get to know in more detail.

Adapter.java Adapter.kt

```
package org.geeksforgeeks.demo;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

// Adapter class for handling the display
```

```java
// of contacts in a RecyclerView
public class Adapter extends RecyclerView.Adapter<Adapter.ViewHolder> {

    private final Context context;
    private ArrayList<Contacts> contactsArrayList;

    public Adapter(Context context, ArrayList<Contacts> contactsArrayList) {
        this.context = context;
        this.contactsArrayList = contactsArrayList;
    }

    // Creates and returns a ViewHolder object
    // for each item in the RecyclerView.
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        View view =
LayoutInflater.from(context).inflate(R.layout.contacts_rv_item, parent, false);
        return new ViewHolder(view);
    }

    // Updates the contact list with a filtered
    // list and notifies the adapter.
    public void filterList(ArrayList<Contacts> filterList) {
        this.contactsArrayList = filterList;

        // Notify adapter about dataset change
        notifyItemRangeChanged(0, getItemCount());
    }

    // Binds data to the ViewHolder for a specific position.
    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
        Contacts contact = contactsArrayList.get(position);
        holder.contactTV.setText(contact.getUserName());

        // Set click listener to open ContactDetailActivity
        // with selected contact details
        holder.itemView.setOnClickListener(v -> {
            Intent intent = new Intent(context, ContactDetailActivity.class);
            intent.putExtra("name", contact.getUserName());
            intent.putExtra("contact", contact.getContactNumber());
            context.startActivity(intent);
        });
    }

    /**
     * Returns the total number of items in the list.
     */
    @Override
    public int getItemCount() {
        return contactsArrayList.size();
    }

    /**
     * ViewHolder class to hold and manage views
     * for each RecyclerView item.
     */
```

```
    public static class ViewHolder extends RecyclerView.ViewHolder {
        private final TextView contactTV;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            contactTV = itemView.findViewById(R.id.contactName);
        }
    }
}
```

Step 8: Working with the MainActivity file

Go to the **MainActivity** file and refer to the following code. Below is the code for the **MainActivity** file. Comments are added inside the code to understand the code in more detail.

MainActivity.java MainActivity.kt

```
package org.geeksforgeeks.demo;

import android.Manifest;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.provider.Settings;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.karumi.dexter.Dexter;
import com.karumi.dexter.MultiplePermissionsReport;
import com.karumi.dexter.PermissionToken;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.multi.MultiplePermissionsListener;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    // List to store contacts
    private final ArrayList<Contacts> contactsArrayList = new ArrayList<>();

    // RecyclerView for displaying contacts
    private RecyclerView contactRV;

    // Adapter for RecyclerView
    private Adapter adapter;
```

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

        // Initialize RecyclerView
        contactRV = findViewById(R.id.rv);

        // Floating Action Button
        // to add new contact
        FloatingActionButton addNewContactFAB = findViewById(R.id.addButton);

        // Setup RecyclerView and request
        // necessary permissions
        prepareContactRV();
        requestPermissions();

        // Handle click event on FloatingActionButton
        // to open CreateNewContactActivity
        addNewContactFAB.setOnClickListener(v ->
            startActivity(new Intent(MainActivity.this,
CreateNewContactActivity.class))
        );
    }

    // Initialize RecyclerView with adapter
    // and layout manager
    private void prepareContactRV() {
        adapter = new Adapter(this, contactsArrayList);
        contactRV.setLayoutManager(new LinearLayoutManager(this));
        contactRV.setAdapter(adapter);
    }

    // Request necessary permissions using Dexter
    private void requestPermissions() {
        Dexter.withActivity(this)
            .withPermissions(
                Manifest.permission.READ_CONTACTS,
                Manifest.permission.CALL_PHONE,
                Manifest.permission.SEND_SMS,
                Manifest.permission.WRITE_CONTACTS
            )
            .withListener(new MultiplePermissionsListener() {
                @Override
                public void onPermissionsChecked(MultiplePermissionsReport report)
{
                    if (report.areAllPermissionsGranted()) {
                        Toast.makeText(MainActivity.this,
                                       "All permissions granted",
                                       Toast.LENGTH_SHORT).show();

                        getContacts();
                    }
                    if (report.isAnyPermissionPermanentlyDenied()) {
                        showSettingsDialog();
                    }
                }
```

```
                @Override
                public void onPermissionRationaleShouldBeShown(
                        List<PermissionRequest> permissions, PermissionToken
token) {
                    token.continuePermissionRequest();
                }
            })
            .withErrorListener(error ->
                Toast.makeText(getApplicationContext(), "Error occurred!",
Toast.LENGTH_SHORT).show()
            )
            .onSameThread()
            .check();
    }

    // Show a dialog directing the user to app
    // settings if permissions are denied permanently
    private void showSettingsDialog() {
        new AlertDialog.Builder(MainActivity.this)
            .setTitle("Need Permissions")
            .setMessage("This app needs permission to use this feature. You can
grant them in app settings.")
            .setPositiveButton("GOTO SETTINGS", (dialog, which) -> {
                dialog.cancel();
                Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                Uri uri = Uri.fromParts("package", getPackageName(), null);
                intent.setData(uri);
                startActivityForResult(intent, 101);
            })
            .setNegativeButton("Cancel", (dialog, which) -> dialog.cancel())
            .show();
    }

    // Fetch contacts from the device's contacts list
    private void getContacts() {
        Cursor cursor = getContentResolver().query(
            ContactsContract.Contacts.CONTENT_URI,
            null, null, null,
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + " ASC"
        );

        if (cursor != null) {
            while (cursor.moveToNext()) {
                String contactId = cursor.getString(

cursor.getColumnIndexOrThrow(ContactsContract.Contacts._ID)
                    );

                String displayName = cursor.getString(

cursor.getColumnIndexOrThrow(ContactsContract.Contacts.DISPLAY_NAME)
                    );

                // Check if the contact has a phone number
                int hasPhoneNumber = cursor.getInt(

cursor.getColumnIndexOrThrow(ContactsContract.Contacts.HAS_PHONE_NUMBER)
                    );
```

```
                if (hasPhoneNumber > 0) {
                    Cursor phoneCursor = getContentResolver().query(
                        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                        null,
                        ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " =
?",
                        new String[]{contactId},
                        null
                    );

                    if (phoneCursor != null) {
                        if (phoneCursor.moveToNext()) {
                            String phoneNumber = phoneCursor.getString(

phoneCursor.getColumnIndexOrThrow(ContactsContract.CommonDataKinds.Phone.NUMBER)
                                );
                            contactsArrayList.add(new Contacts(displayName,
phoneNumber));
                        }
                        phoneCursor.close();
                    }
                }
            }
            cursor.close();
        }

        // Notify the adapter about dataset changes
        adapter.notifyItemRangeChanged(0, contactsArrayList.size());
    }
}
```

Step 9: Working with the CreateNewContactActivity

Below is the code for both **activity_create_new_contact.xml** and **CreateNewContactActivity Java/Kotlin** file. Comments are added inside the code to understand the code in more detail.

CreateNewContactActivity.java CreateNewContactActivity.kt

activity_create_new_contact.xml

```
package org.geeksforgeeks.demo;

import android.content.Intent;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.text.TextUtils;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidx.annotation.Nullable;
```

```java
import androidx.appcompat.app.AppCompatActivity;

/**
 * Activity to create a new contact and
 * save it to the device's contact list.
 */
public class CreateNewContactActivity extends AppCompatActivity {

    // UI elements
    private EditText nameEdt, phoneEdt, emailEdt;
    private Button addContactBtn;

    private static final int REQUEST_CODE_ADD_CONTACT = 1;

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

        // Initialize UI elements
        nameEdt = findViewById(R.id.enterName);
        phoneEdt = findViewById(R.id.enterNumber);
        emailEdt = findViewById(R.id.enterEmail);
        addContactBtn = findViewById(R.id.saveButton);

        // Set click listener on the "Add Contact" button
        addContactBtn.setOnClickListener(v -> {
            String name = nameEdt.getText().toString().trim();
            String phone = phoneEdt.getText().toString().trim();
            String email = emailEdt.getText().toString().trim();

            // Check if any field is empty before proceeding
            if (TextUtils.isEmpty(name) || TextUtils.isEmpty(phone) ||
TextUtils.isEmpty(email)) {
                Toast.makeText(CreateNewContactActivity.this,
                               "Please enter data in all fields.",
                               Toast.LENGTH_SHORT).show();
            } else {
                addContact(name, phone, email);
            }
        });
    }

    /**
     * Opens the contacts app to add a new contact with the provided details.
     *
     * @param name  The contact's name.
     * @param phone The contact's phone number.
     * @param email The contact's email address.
     */
    private void addContact(String name, String phone, String email) {
        Intent contactIntent = new Intent(ContactsContract.Intents.Insert.ACTION);
        contactIntent.setType(ContactsContract.RawContacts.CONTENT_TYPE);
        contactIntent.putExtra(ContactsContract.Intents.Insert.NAME, name);
        contactIntent.putExtra(ContactsContract.Intents.Insert.PHONE, phone);
        contactIntent.putExtra(ContactsContract.Intents.Insert.EMAIL, email);
        startActivityForResult(contactIntent, REQUEST_CODE_ADD_CONTACT);
    }
```

```
    /**
     * Handles the result after attempting to add a contact.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_CODE_ADD_CONTACT) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(this, "Contact has been added.",
Toast.LENGTH_SHORT).show();
                navigateToMainActivity();
            } else if (resultCode == RESULT_CANCELED) {
                Toast.makeText(this, "Cancelled adding contact.",
Toast.LENGTH_SHORT).show();
            }
        }
    }

    /**
     * Navigates back to the main activity
     * after successfully adding a contact.
     */
    private void navigateToMainActivity() {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

Step 10: Working with the ContactDetailActivity

Below is the code for both **activity_contact_detail.xml** and **ContactDetailActivity Java/Kotlin** file. Comments are added inside the code to understand the code in more detail.

ContactDetailActivity.java ContactDetailActivity.kt activity_contact_detail.xml

```
package org.geeksforgeeks.demo;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
```

```java
import androidx.core.app.ActivityCompat;

/**
 * Activity to display contact details and provide call and messaging
functionalities.
 */
public class ContactDetailActivity extends AppCompatActivity {

    // Variables to store contact details
    private String contactName;
    private String contactNumber;

    // UI elements
    private TextView contactTV, nameTV;
    private ImageView contactIV, callIV, messageIV;

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

        // Retrieve contact details from intent extras
        contactName = getIntent().getStringExtra("name");
        contactNumber = getIntent().getStringExtra("contact");

        // Initialize UI elements
        nameTV = findViewById(R.id.name);
        contactTV = findViewById(R.id.number);
        contactIV = findViewById(R.id.profileImage);
        callIV = findViewById(R.id.callButton);
        messageIV = findViewById(R.id.messageButton);

        // Set contact details in UI
        nameTV.setText(contactName);
        contactTV.setText(contactNumber);

        // Handle call button click
        callIV.setOnClickListener(v -> makeCall(contactNumber));

        // Handle message button click
        messageIV.setOnClickListener(v -> sendMessage(contactNumber));
    }

    /**
     * Opens the default messaging app with the contact number pre-filled.
     *
     * @param contactNumber The phone number to send a message to.
     */
    private void sendMessage(String contactNumber) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("sms:" +
contactNumber));
        intent.putExtra("sms_body", "Enter your message");
        startActivity(intent);
    }

    /**
     * Initiates a phone call to the given contact number.
     * Checks for CALL_PHONE permission before making the call.
     *
```

```
     * @param contactNumber The phone number to call.
     */
    private void makeCall(String contactNumber) {
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:" + contactNumber));

        // Check if CALL_PHONE permission is granted before making the call
        if (ActivityCompat.checkSelfPermission(
                this, Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
            return;
        }
        startActivity(callIntent);
    }
}
```

Now run your app and see the output of the app.

> ***Note**: For all the drawable files you may refer to the GitHub link below or you may add it of your own.*

Output:

0:00

Check out the project on below Github link: *Contacts_App_Android*