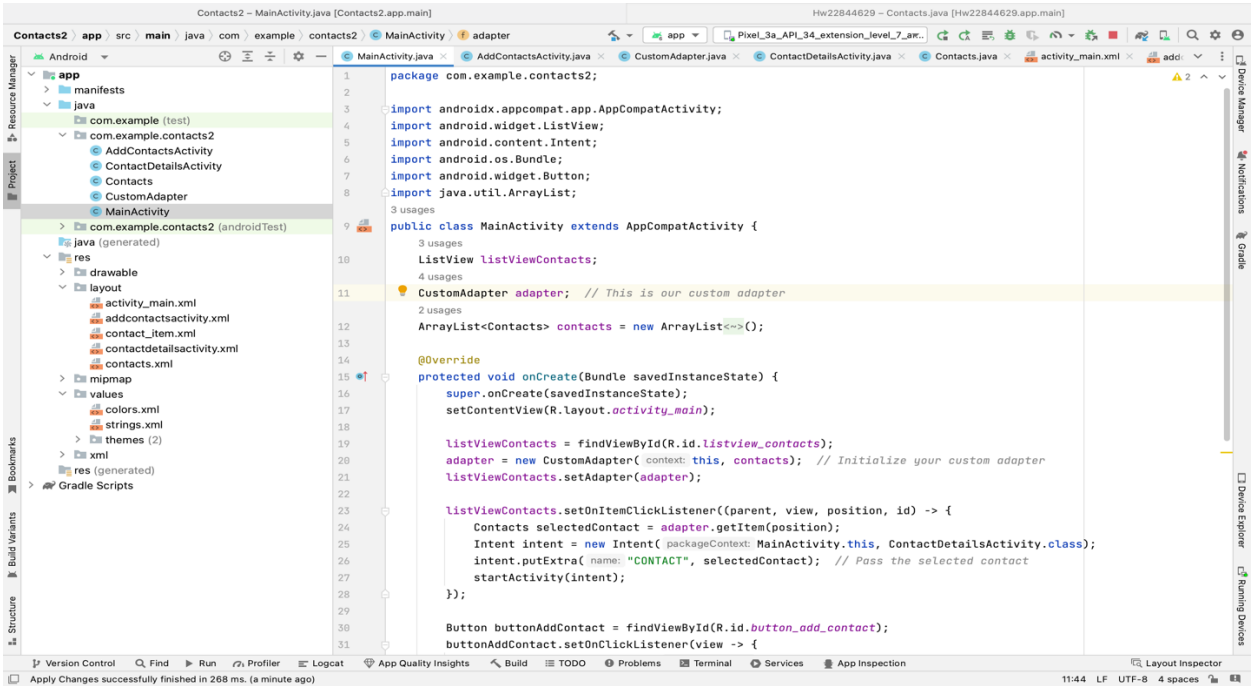
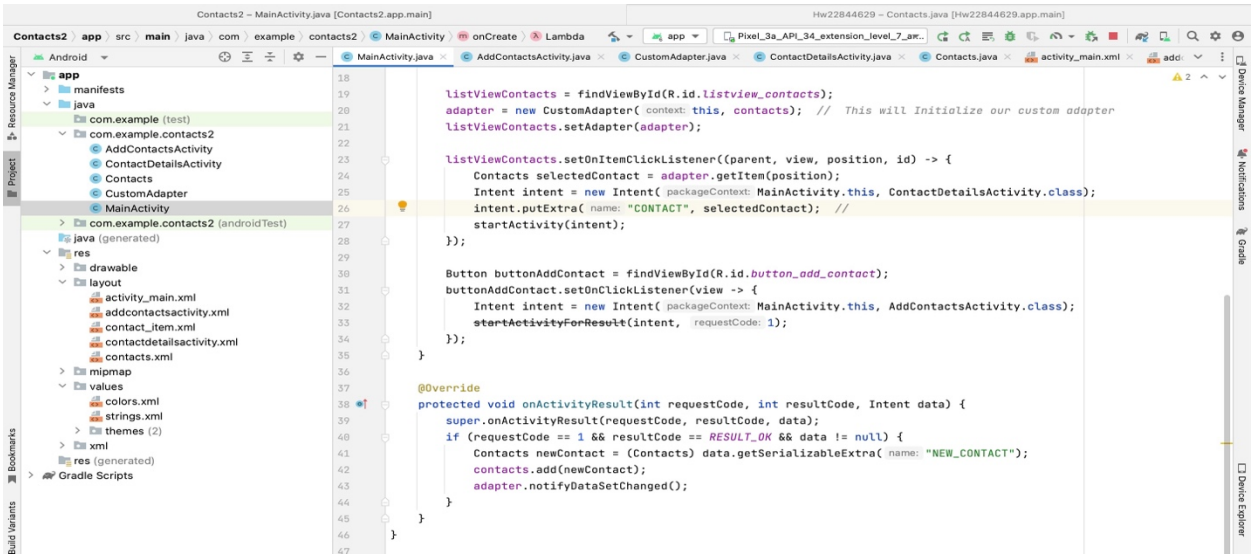
Contact management app version 2

- Like version 1, the layout for the main activity should have a button (at the bottom of the layout) for users to add a new contact, and the new contact information should be passed back to the main activity on exit of that activity.
- The created contacts should be displayed as a ListView in the main activity.
- In the ListView, each contact's full name should be displayed.
- In the main activity, a user could click any row of the contact, a third activity would open to display the contact details.

MainActivity.java : This is the MainActivity of the app we are using many methods in this applications like "onCreate" to initialize the main activity, "onActivityResult" to handle the result of adding new contact and update the contact list.

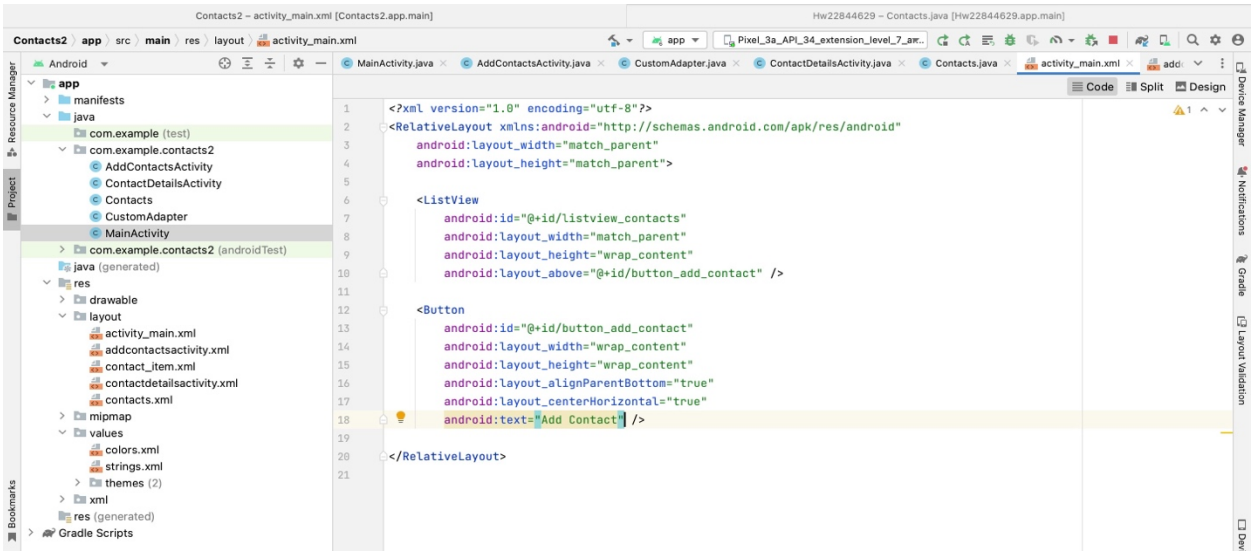


```
package com.example.contacts2;

import androidx.appcompat.app.AppCompatActivity;
import android.widget.ListView;
import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import java.util.ArrayList;
3 usages
public class MainActivity extends AppCompatActivity {
    3 usages
    ListView listViewContacts;
    4 usages
    CustomAdapter adapter;  // This is our custom adapter
    2 usages
    ArrayList<Contacts> contacts = new ArrayList<>();

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

        listViewContacts = findViewById(R.id.listview_contacts);
        adapter = new CustomAdapter( context: this, contacts);  // Initialize your custom adapter
        listViewContacts.setAdapter(adapter);

        listViewContacts.setOnItemClickListener((parent, view, position, id) -> {
            Contacts selectedContact = adapter.getItem(position);
            Intent intent = new Intent( packageContext: MainActivity.this, ContactDetailsActivity.class);
            intent.putExtra( name: "CONTACT", selectedContact);  // Pass the selected contact
            startActivity(intent);
        });

        Button buttonAddContact = findViewById(R.id.button_add_contact);
        buttonAddContact.setOnClickListener(view -> {
```

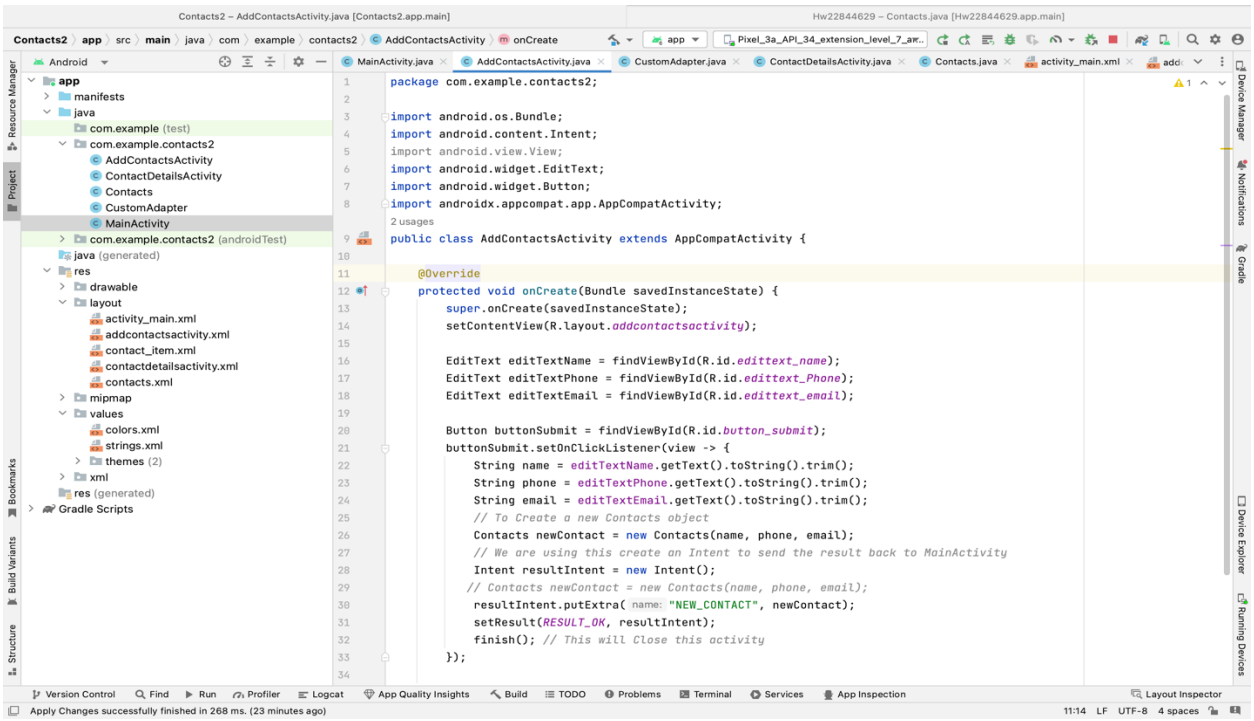


```
        listViewContacts = findViewById(R.id.listview_contacts);
        adapter = new CustomAdapter( context: this, contacts);  //  This will Initialize our custom adapter
        listViewContacts.setAdapter(adapter);

        listViewContacts.setOnItemClickListener((parent, view, position, id) -> {
            Contacts selectedContact = adapter.getItem(position);
            Intent intent = new Intent( packageContext: MainActivity.this, ContactDetailsActivity.class);
            intent.putExtra( name: "CONTACT", selectedContact);  //
            startActivity(intent);
        });

        Button buttonAddContact = findViewById(R.id.button_add_contact);
        buttonAddContact.setOnClickListener(view -> {
            Intent intent = new Intent( packageContext: MainActivity.this, AddContactsActivity.class);
            startActivityForResult(intent,  requestCode: 1);
        });
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1 && resultCode == RESULT_OK && data != null) {
            Contacts newContact = (Contacts) data.getSerializableExtra( name: "NEW_CONTACT");
            contacts.add(newContact);
            adapter.notifyDataSetChanged();
        }
    }
}
```

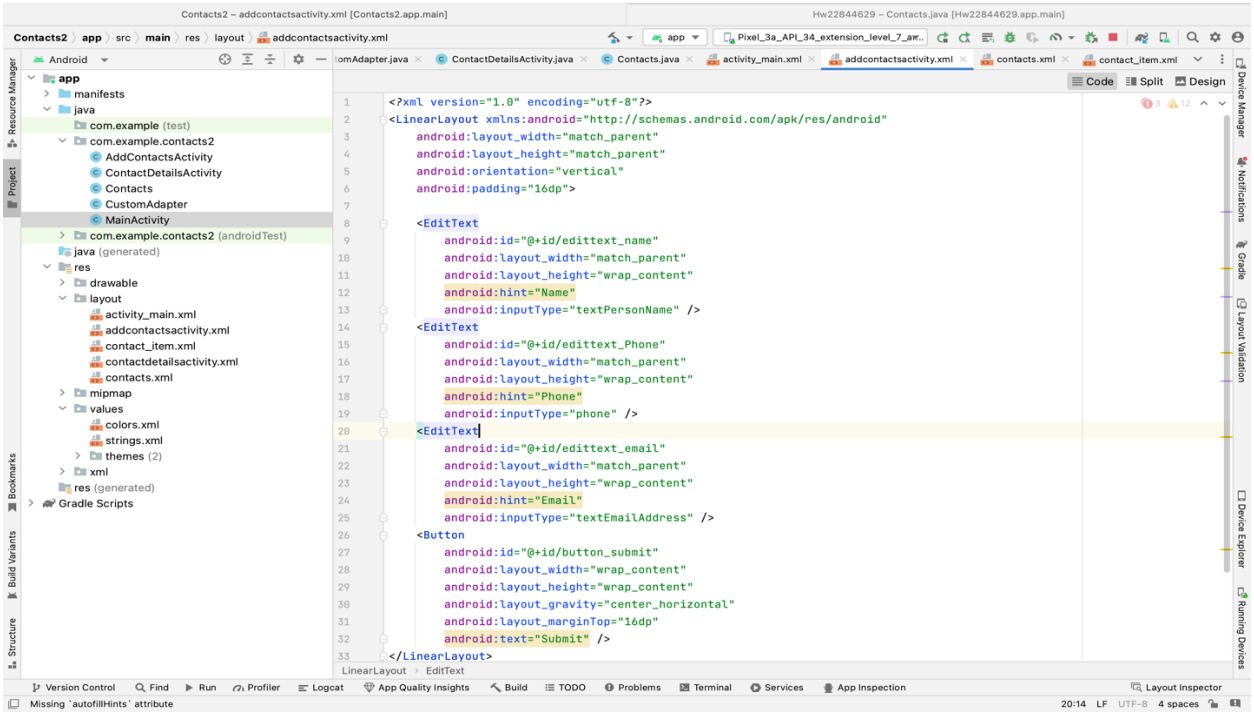
Activity_main.xml : this is a layout file arranges elements on the screen using “RelativeLayout” which contains “ListView” to display contacts entries and a Button using positioned at the bottom for adding new contacts.



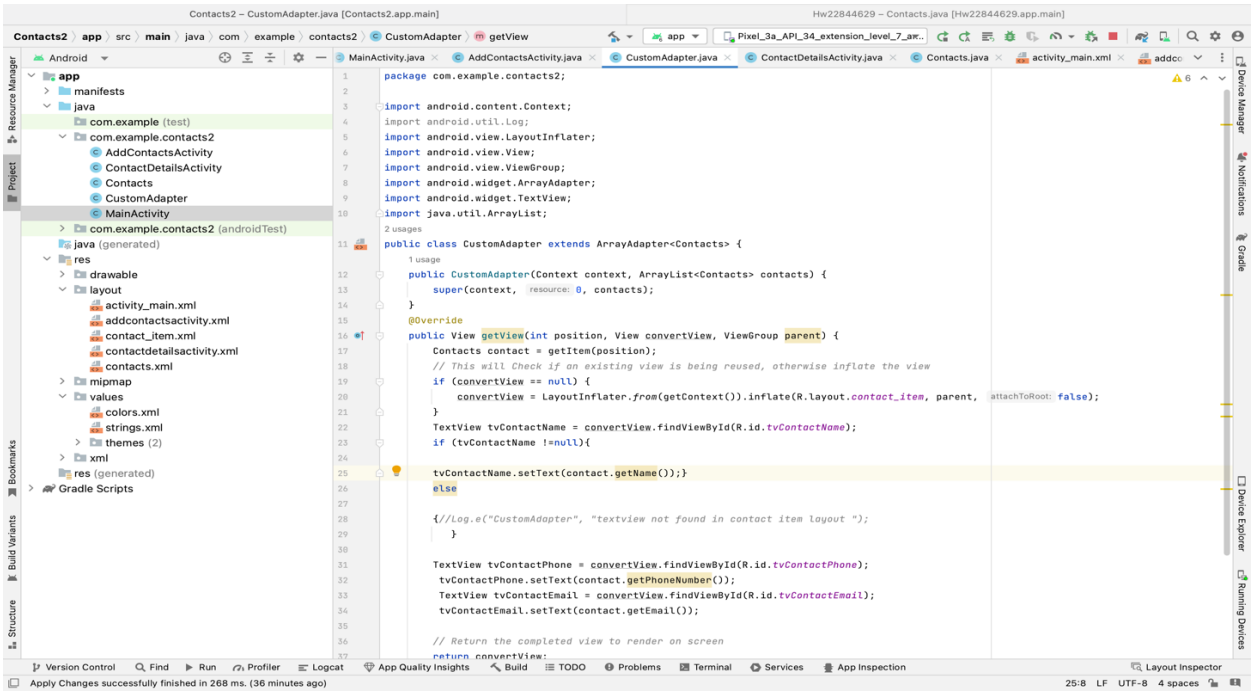
AddContactActivity.java : In this class users can add new contacts ,it captures the name, phone and email input. “onCreate” is used to set up a layout for adding new contact and handles the submission of contact information. Then it closes the contact creation screen.



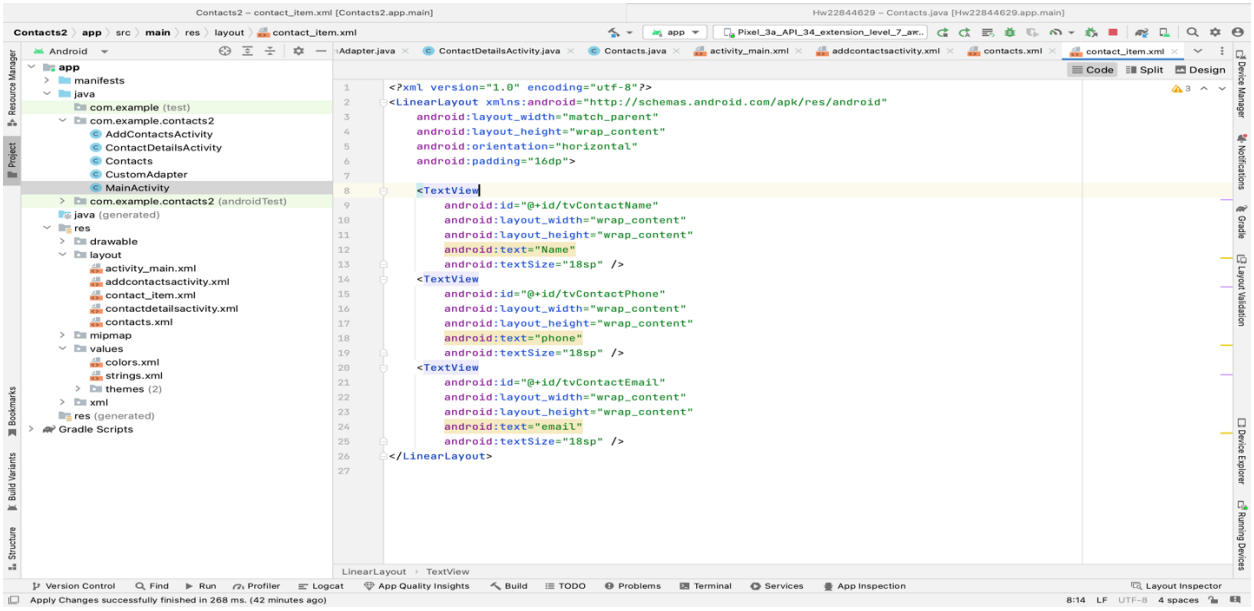
Addcontactsactivity.xml : this defines a form screen for adding contact details like name, phone and email and after the information is entered then it can be submitted using submit.



CustomAdapter.java : This custom adaptor allows you to change the list view appearance of the contact data. It takes a contact list, formats it so that it can be shown in the app, and makes sure the arrangement is ideal.

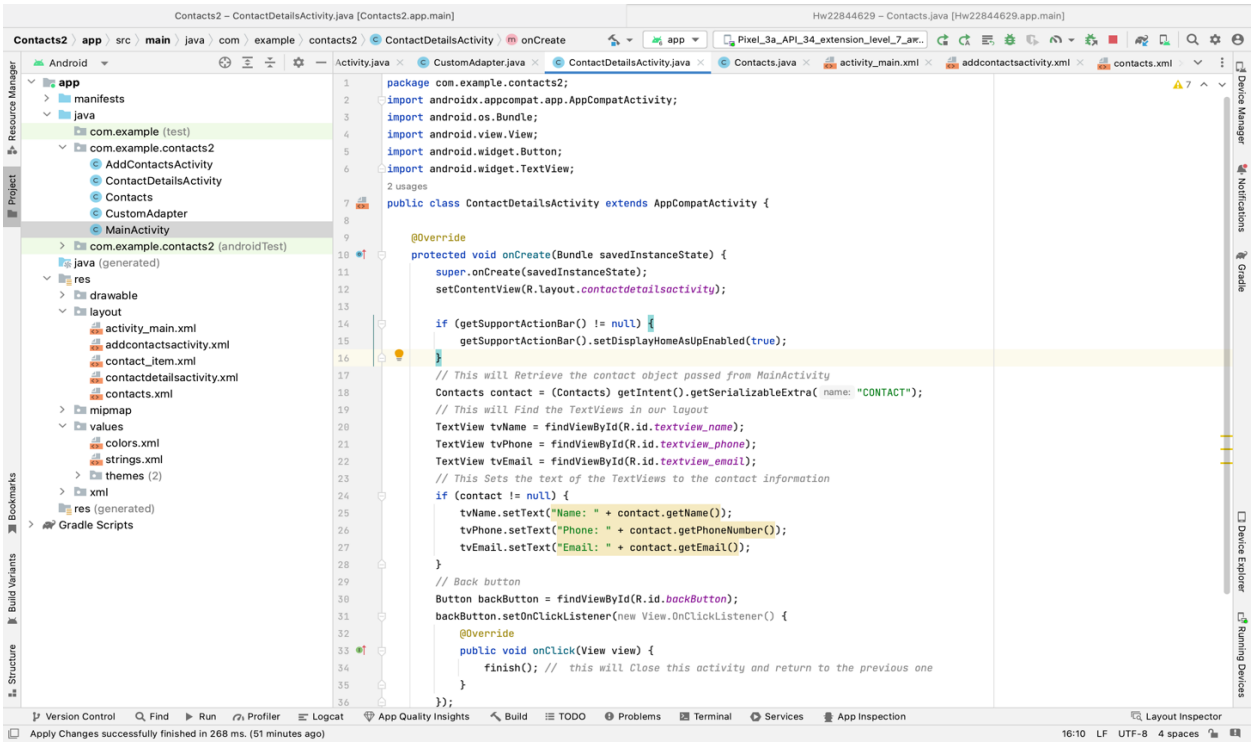


Contacts_item.xml : This layout defines a horizontal layout for displaying contact information like name, phone and email and each of them is represented by a “TextView”.



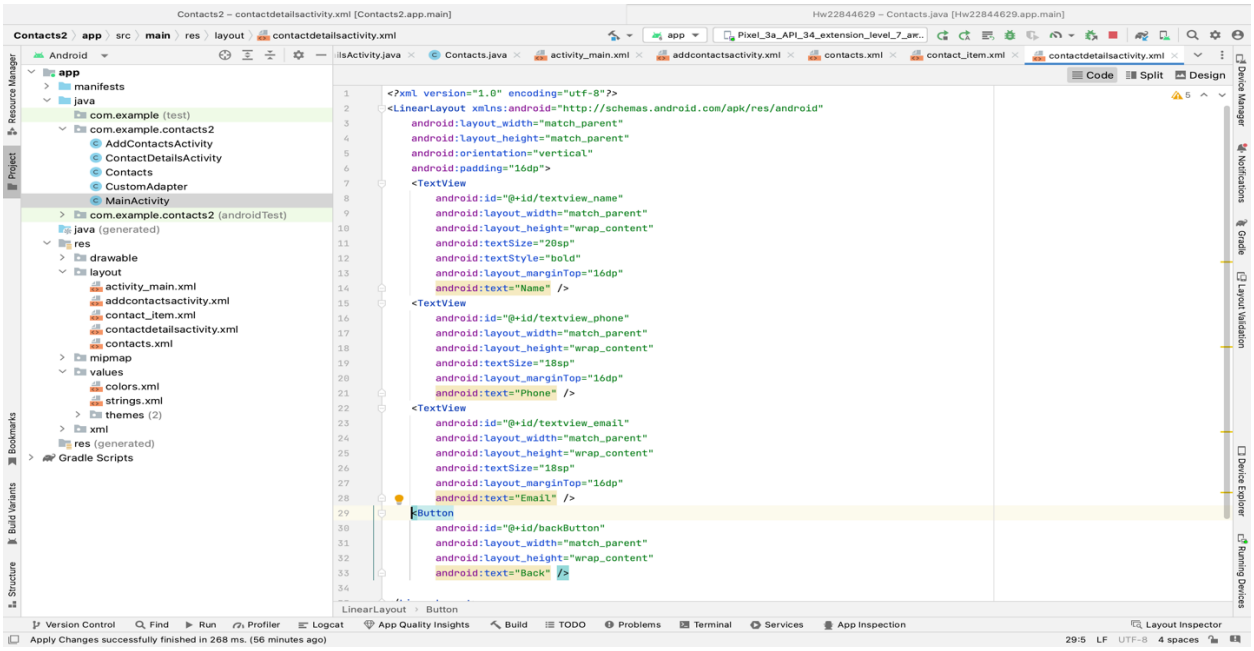
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvContactName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name"
        android:textSize="18sp" />
    <TextView
        android:id="@+id/tvContactPhone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="phone"
        android:textSize="18sp" />
    <TextView
        android:id="@+id/tvContactEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="email"
        android:textSize="18sp" />
</LinearLayout>
```

ContactDetailActivity.java : This shows the specific details of a submitted contact. It obtains the contact information from the "MainActivity," configures "TextViews" to show the contact's name, phone number, and email, and includes a back button to return to the list view of other saved contacts.

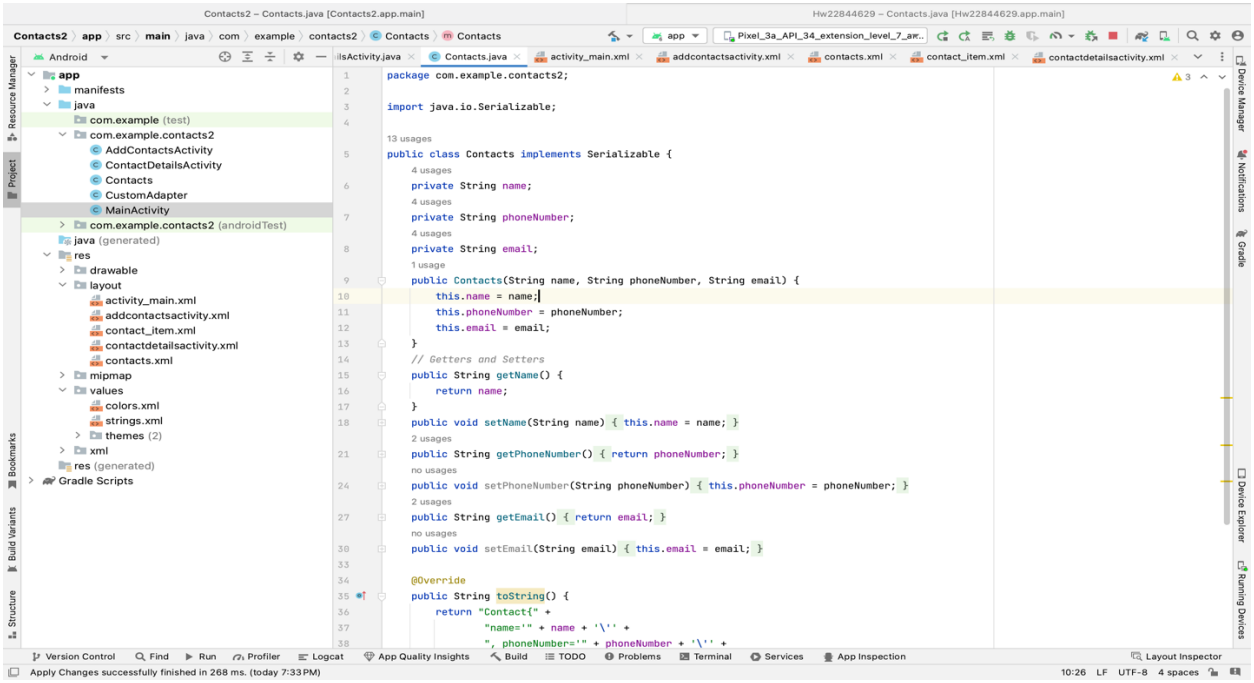


```
package com.example.contacts2;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
2 usages
public class ContactDetailsActivity extends AppCompatActivity {

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

        if (getSupportActionBar() != null) {
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        }
        // This will Retrieve the contact object passed from MainActivity
        Contacts contact = (Contacts) getIntent().getSerializableExtra( name: "CONTACT");
        // This will Find the TextViews in our layout
        TextView tvName = findViewById(R.id.textview_name);
        TextView tvPhone = findViewById(R.id.textview_phone);
        TextView tvEmail = findViewById(R.id.textview_email);
        // This Sets the text of the TextViews to the contact information
        if (contact != null) {
            tvName.setText("Name: " + contact.getName());
            tvPhone.setText("Phone: " + contact.getPhoneNumber());
            tvEmail.setText("Email: " + contact.getEmail());
        }
        // Back button
        Button backButton = findViewById(R.id.backButton);
        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish(); //  this will Close this activity and return to the previous one
            }
        });
```

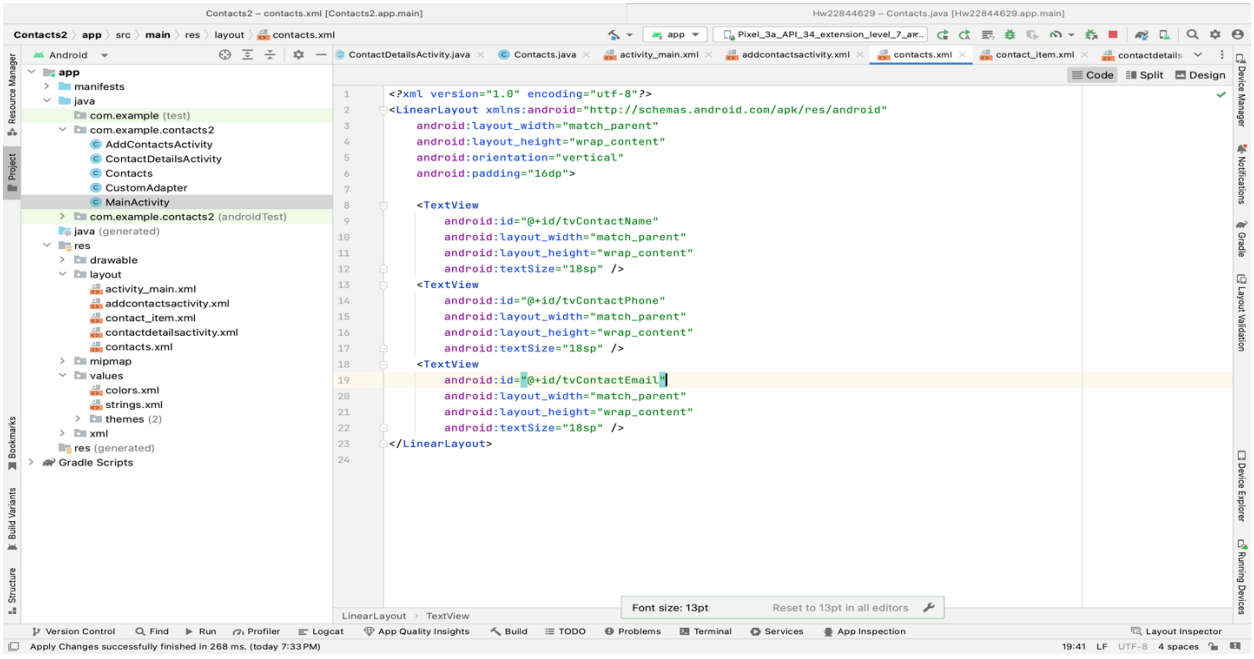
Contactdetailsactivity.xml : This is the "ContactDetailsActivity.java" layout, which uses "TextViews" to show contact details like name, phone number, and email in addition to a back button to go back to the previous activity.



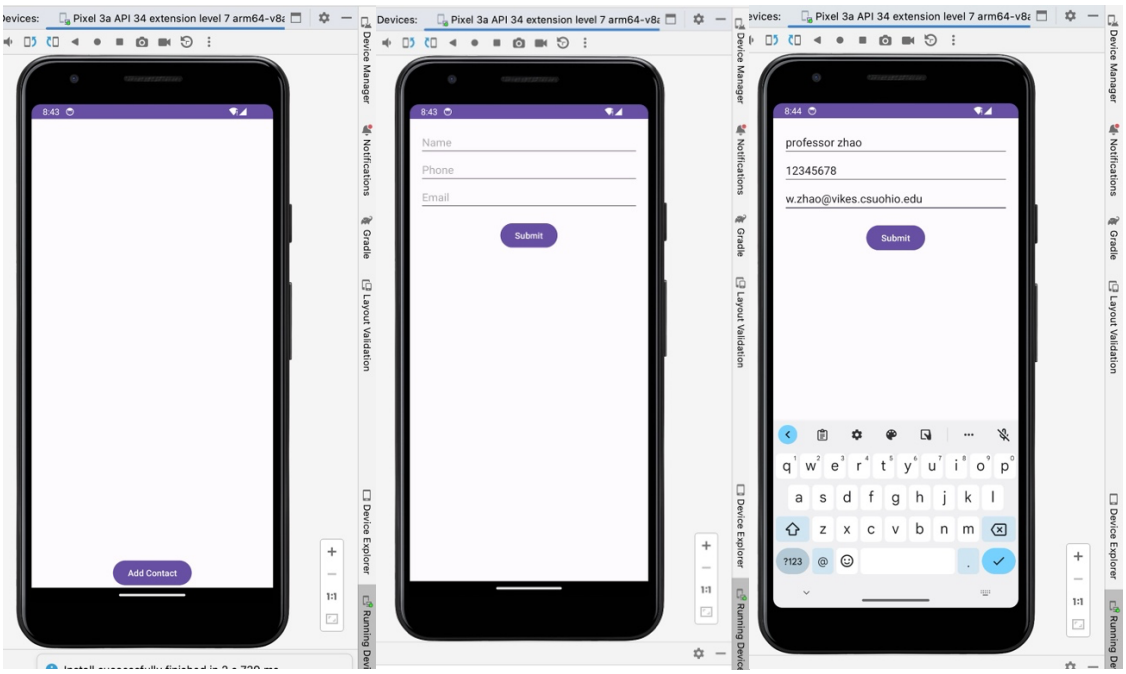
Contacts.java : This is essential for creating and storing the contact information
This class's "Serializable" interface enables objects to be serialized, enabling the transfer of contact information between other app components.



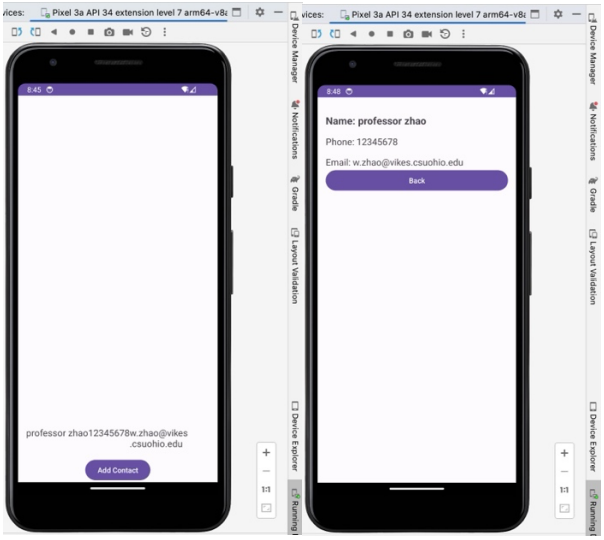
Contacts.xml : this defines the vertical list of “TextViews” used for displaying contact information i.e., name, phone and email in a simple and organized way.



Below are the screenshots of the app :



MainActivity. Addcontactactivty. Entering and submitting contact details



Contacts saved contactDetailsActivity