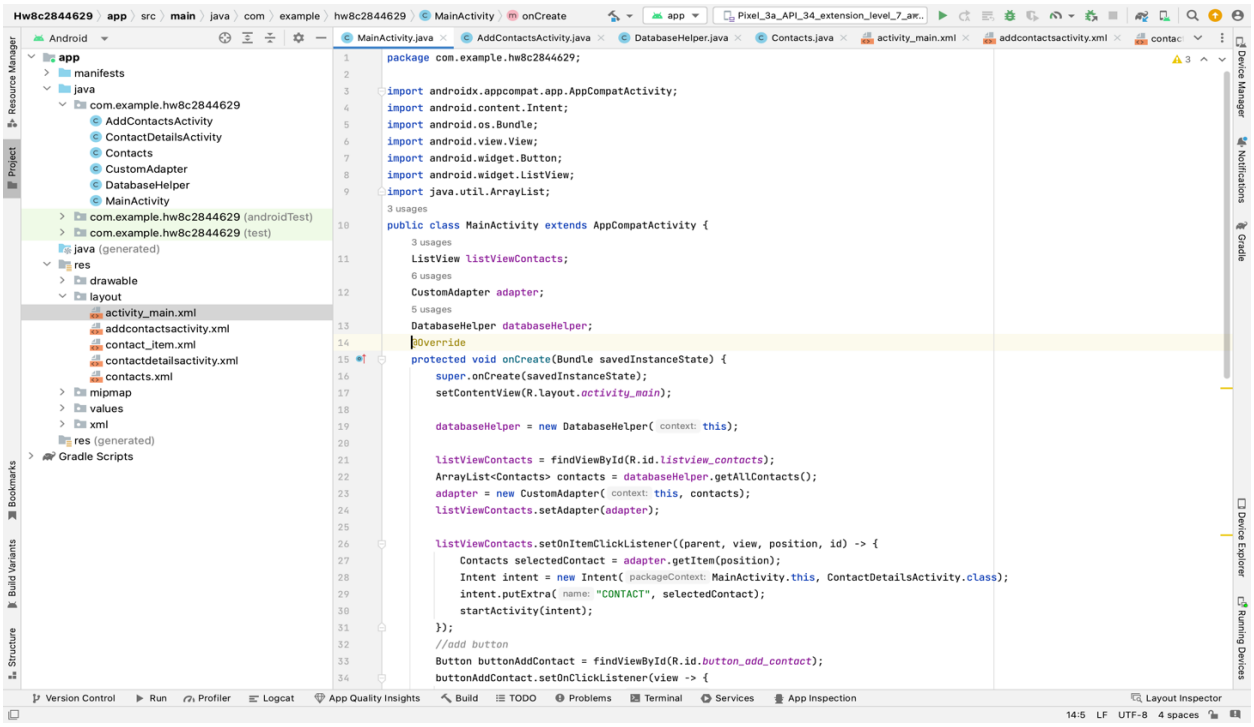
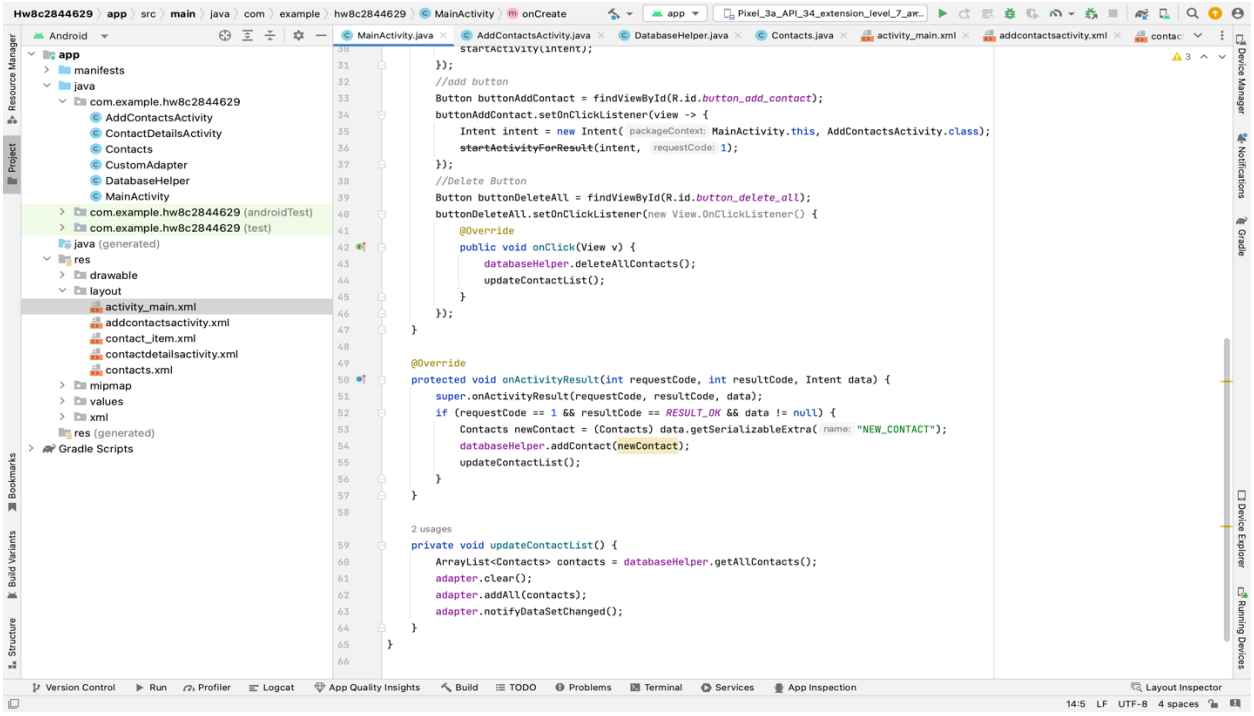
Contact management app version 3:

- Based on contact management app version 2, please use SQLite database to store the collected contact information.
- By using SQLite to store the contacts, your app now can store the contacts persistently, i.e., even if you close your app and restart it, all the contacts will not be lost, and they should still show in the ListView.

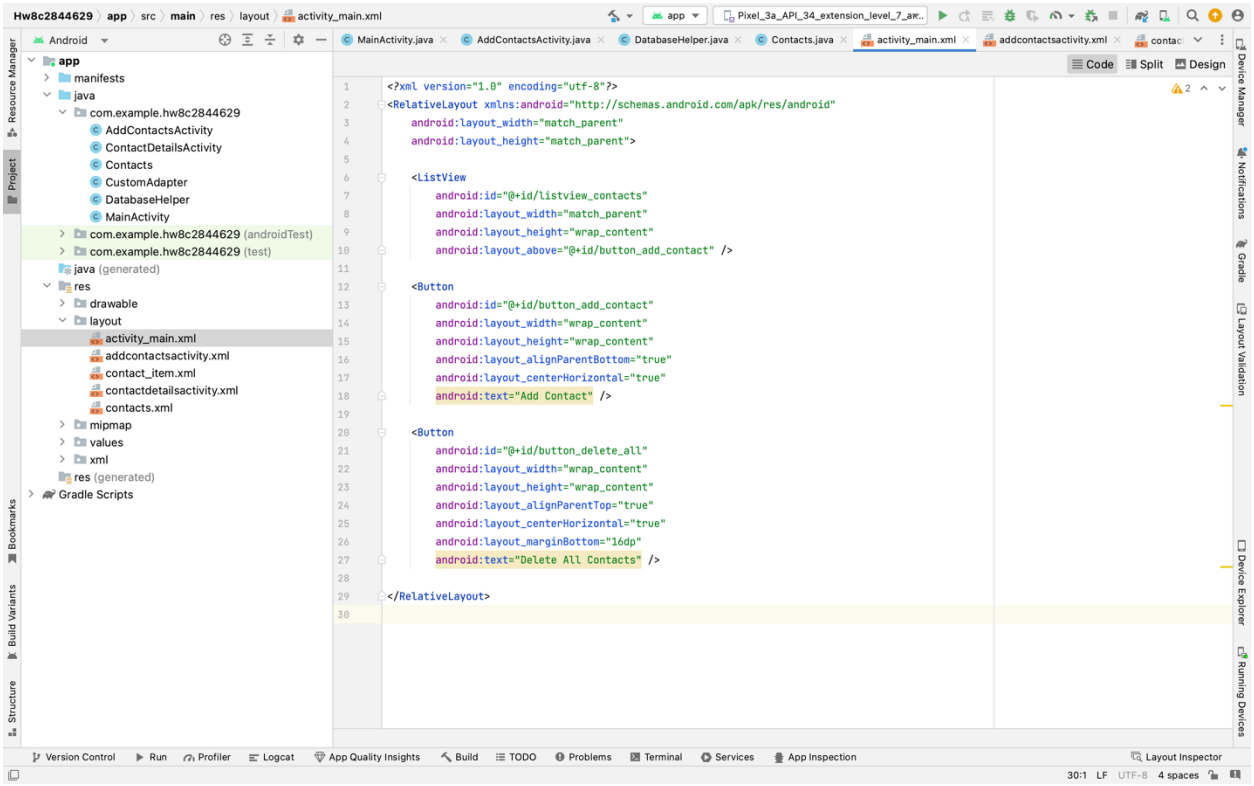
NOTE : I added a delete button too in the mainactivity so we can delete the whole list of contacts at once so it will be easy to test the app.

MainAcctivity.java : This is the MainActivity of the app we are using many methods in this application like “onCreate” to initialize the main activity, “onActivityResult” to handle the result of adding new contact and update the contact list. The main activity also makes use of a ListView and a "custom adapter" to display contacts retrieved from an SQLite database, which is handled by the “DatabaseHelper” class.

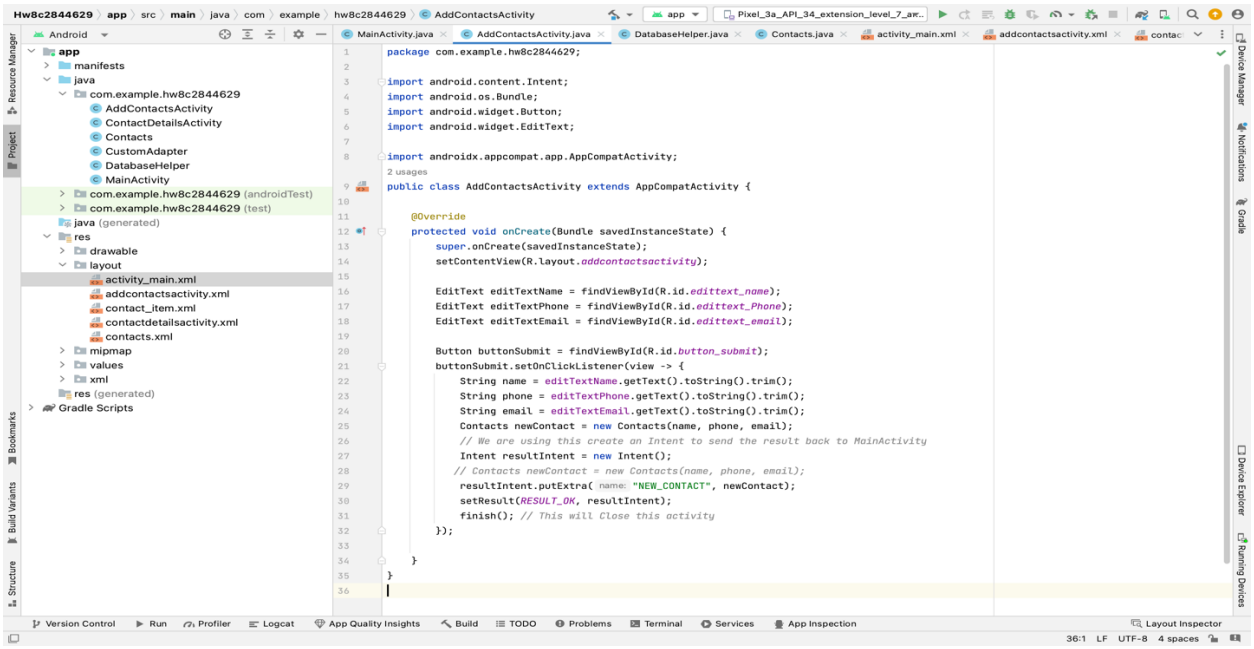




Activity_main.xml : This is a layout file arranges elements on the screen using “RelativeLayout” which contains “ListView” to display contacts entries , a Button using positioned at the bottom for adding new contacts and the top to delete all contacts.

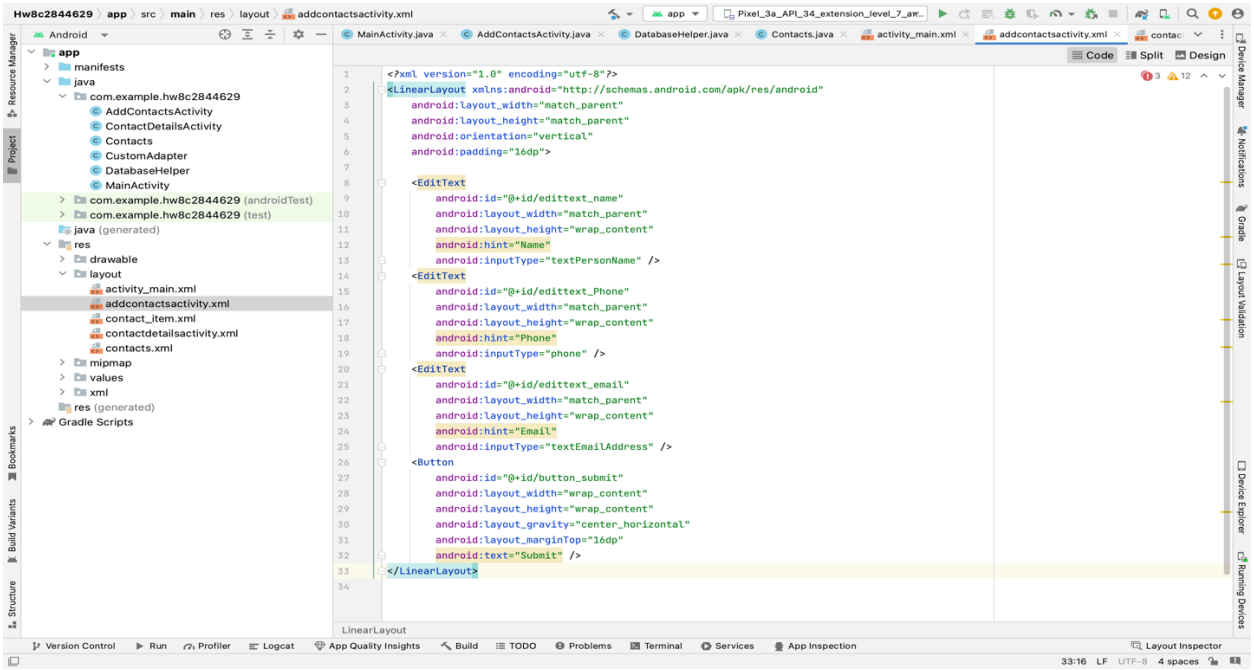


AddContactActivity.java : In this class users can add new contacts ,it captures the name, phone and email input. “onCreate” is used to set up a layout for adding new contact and handles the submission of contact information. Then it closes the contact creation screen.



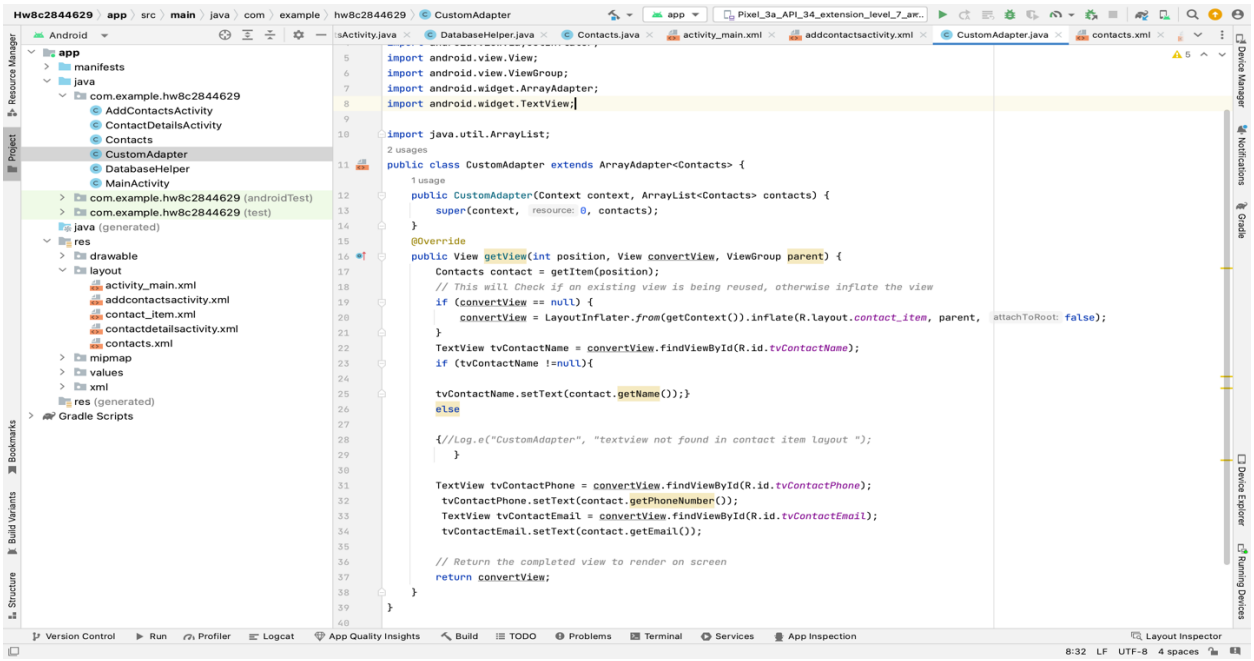
```java
package com.example.hw8c2844629;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;

import androidx.appcompat.app.AppCompatActivity;
2 usages
public class AddContactsActivity extends AppCompatActivity {

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

        EditText editTextName = findViewById(R.id.edittext_name);
        EditText editTextPhone = findViewById(R.id.edittext_Phone);
        EditText editTextEmail = findViewById(R.id.edittext_email);

        Button buttonSubmit = findViewById(R.id.button_submit);
        buttonSubmit.setOnClickListener(view -> {
            String name = editTextName.getText().toString().trim();
            String phone = editTextPhone.getText().toString().trim();
            String email = editTextEmail.getText().toString().trim();
            Contacts newContact = new Contacts(name, phone, email);
            // We are using this create an Intent to send the result back to MainActivity
            Intent resultIntent = new Intent();
           // Contacts newContact = new Contacts(name, phone, email);
            resultIntent.putExtra( name: "NEW_CONTACT", newContact);
            setResult(RESULT_OK, resultIntent);
            finish(); // This will Close this activity
        });

    }
}
```

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.



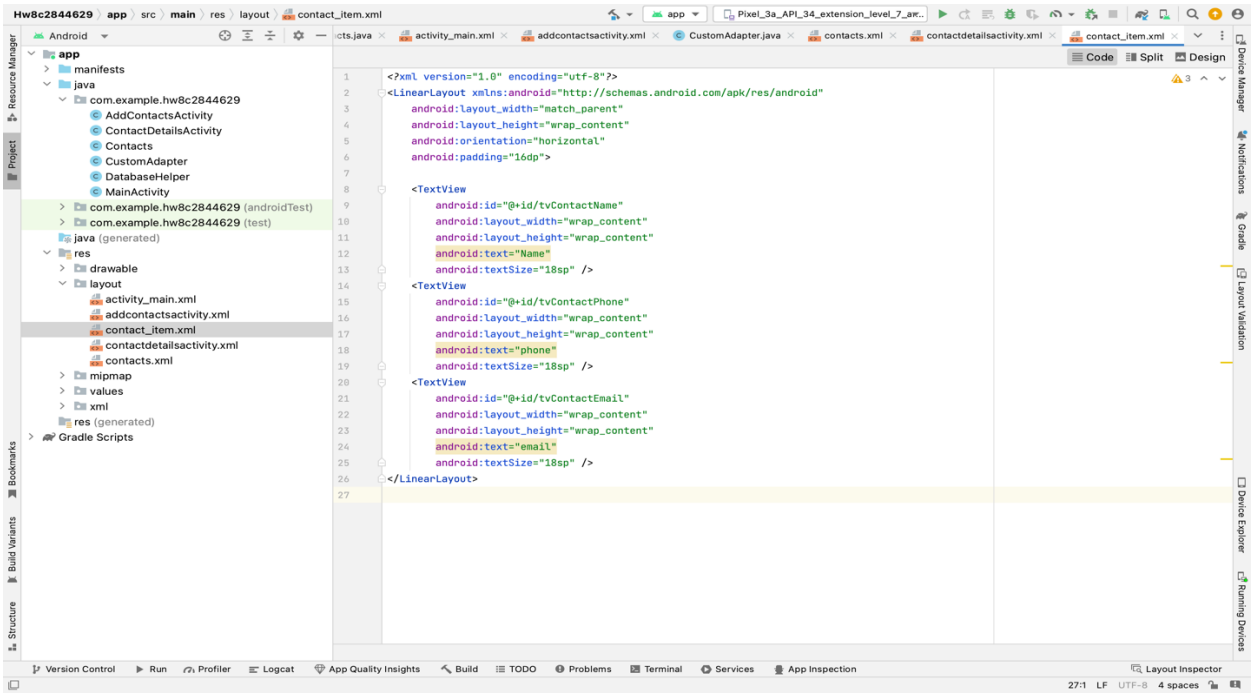
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <EditText
        android:id="@+id/edittext_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:inputType="textPersonName" />
    <EditText
        android:id="@+id/edittext_Phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Phone"
        android:inputType="phone" />
    <EditText
        android:id="@+id/edittext_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />
    <Button
        android:id="@+id/button_submit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="16dp"
        android:text="Submit" />
</LinearLayout>
```

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.



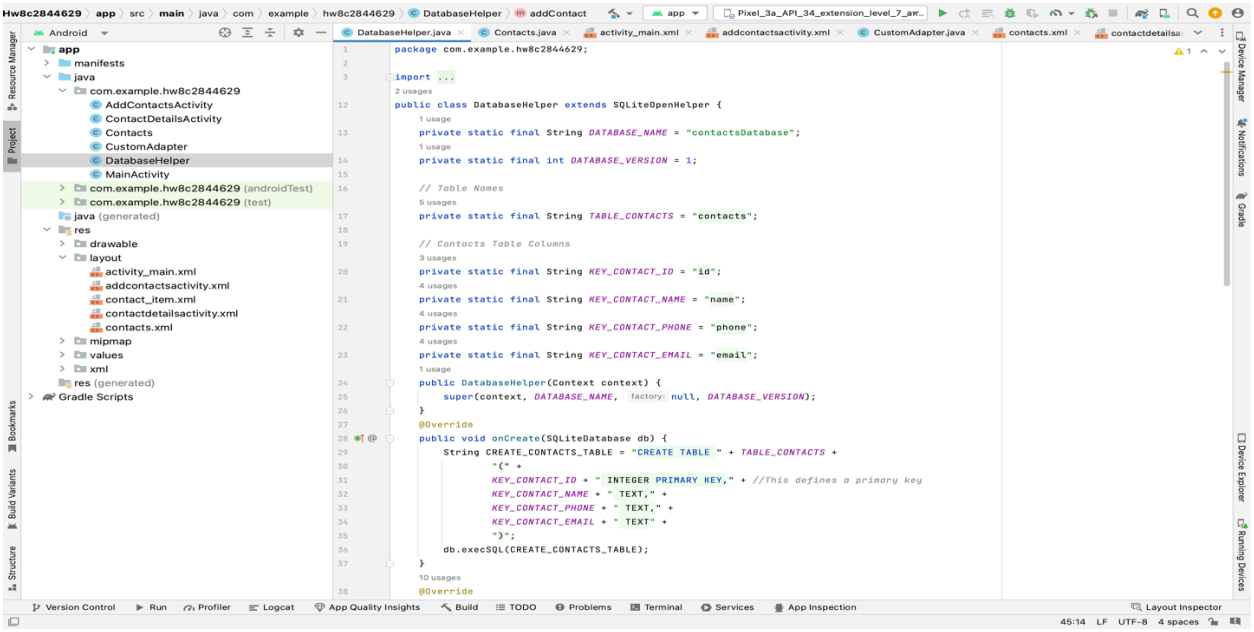
```java
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import java.util.ArrayList;
2 usages
public class CustomAdapter extends ArrayAdapter<Contacts> {
    1 usage
    public CustomAdapter(Context context, ArrayList<Contacts> contacts) {
        super(context, resource: 0, contacts);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Contacts contact = getItem(position);
        // This will Check if an existing view is being reused, otherwise inflate the view
        if (convertView == null) {
            convertView = LayoutInflater.from(getContext()).inflate(R.layout.contact_item, parent, attachToRoot: false);
        }
        TextView tvContactName = convertView.findViewById(R.id.tvContactName);
        if (tvContactName !=null){

        tvContactName.setText(contact.getName());}
        else

        {//Log.e("CustomAdapter", "textview not found in contact item layout ");
            }

        TextView tvContactPhone = convertView.findViewById(R.id.tvContactPhone);
         tvContactPhone.setText(contact.getPhoneNumber());
         TextView tvContactEmail = convertView.findViewById(R.id.tvContactEmail);
         tvContactEmail.setText(contact.getEmail());

        // Return the completed view to render on screen
        return convertView;
    }
}
```

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”.

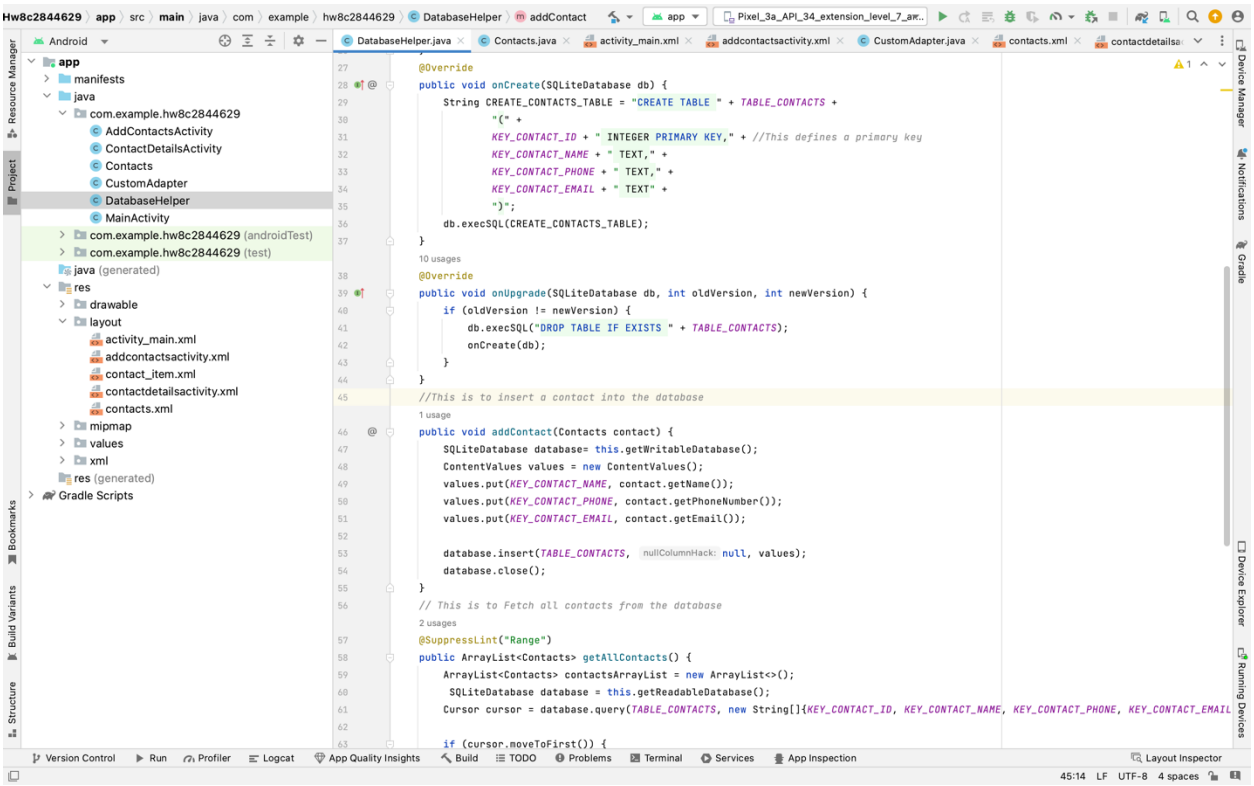


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="16dp">

    <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>
```

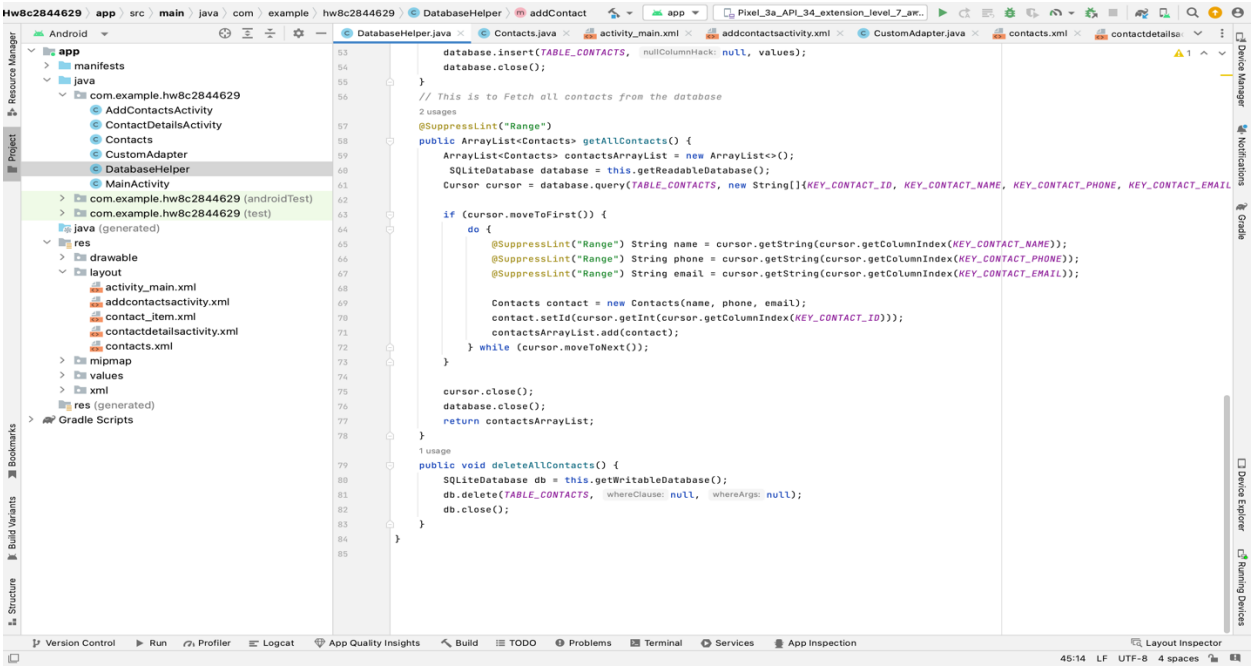
DatabaseHelper.java : This class acts like a bridge between “SQLLite database” and the app.it uses standard SQLLite functionality in android to handle database interactions. It supports database creation and versioning, as well as providing critical Create, Read, Update, Delete (CRUD) actions on the contacts database. It also includes the building of a contacts database with columns for ID, name, phone, and email, methods for adding and retrieving contact information, and a function for deleting all contacts.



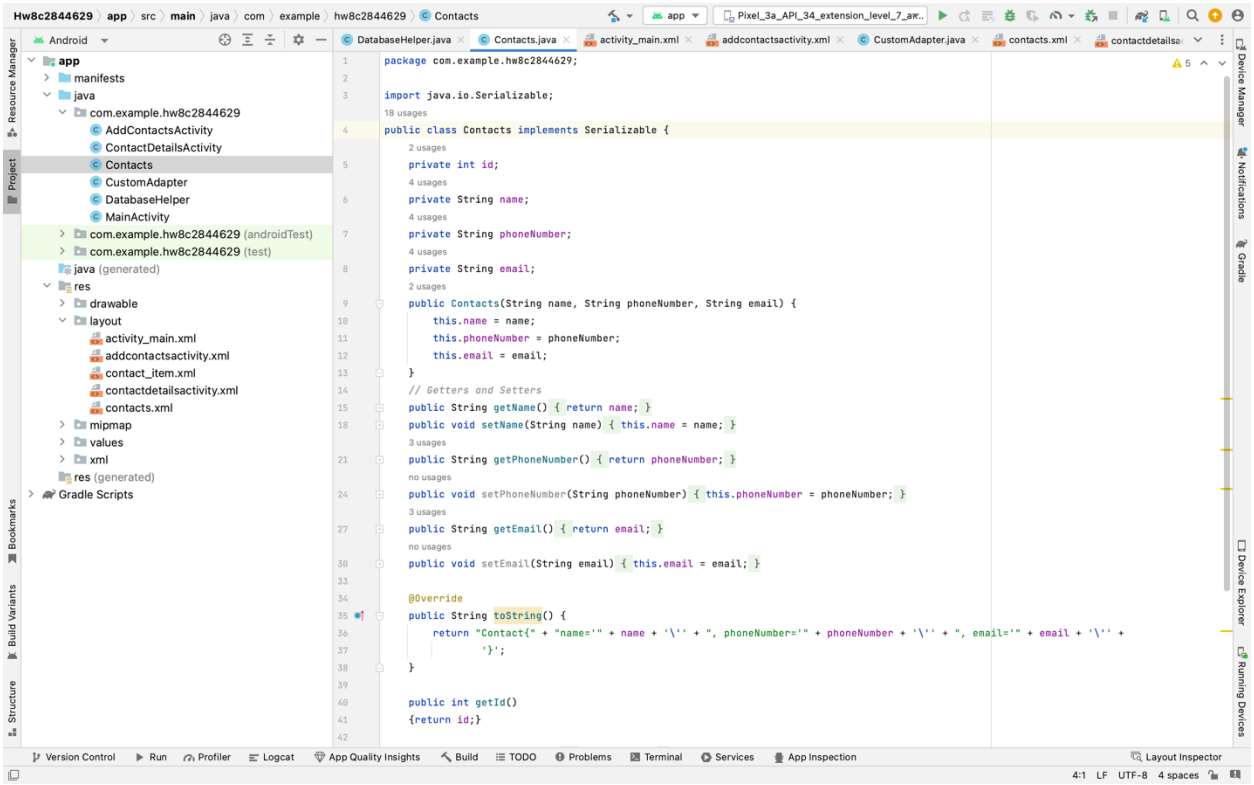
```
package com.example.hw8c2844629;

import ...
2 usages
public class DatabaseHelper extends SQLiteOpenHelper {
    1 usage
    private static final String DATABASE_NAME = "contactsDatabase";
    1 usage
    private static final int DATABASE_VERSION = 1;

    // Table Names
    5 usages
    private static final String TABLE_CONTACTS = "contacts";

    // Contacts Table Columns
    3 usages
    private static final String KEY_CONTACT_ID = "id";
    4 usages
    private static final String KEY_CONTACT_NAME = "name";
    4 usages
    private static final String KEY_CONTACT_PHONE = "phone";
    4 usages
    private static final String KEY_CONTACT_EMAIL = "email";
    1 usage
    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS +
                "(" +
                KEY_CONTACT_ID + " INTEGER PRIMARY KEY," + //This defines a primary key
                KEY_CONTACT_NAME + " TEXT," +
                KEY_CONTACT_PHONE + " TEXT," +
                KEY_CONTACT_EMAIL + " TEXT" +
                ")";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }
    10 usages
    @Override
```



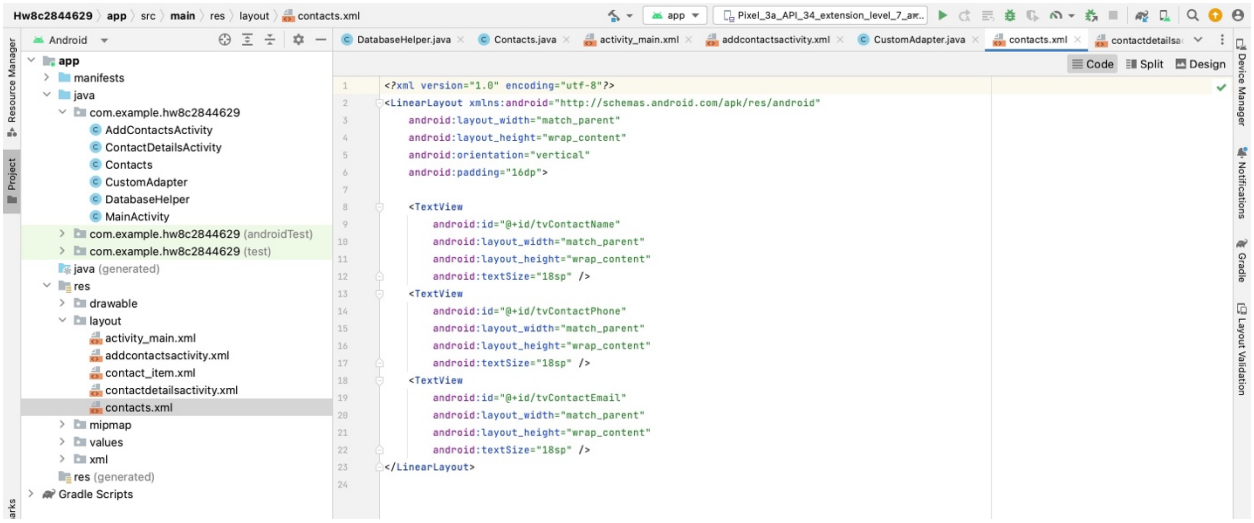
```
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_CONTACTS_TABLE = "CREATE TABLE " + TABLE_CONTACTS +
                "(" +
                KEY_CONTACT_ID + " INTEGER PRIMARY KEY," + //This defines a primary key
                KEY_CONTACT_NAME + " TEXT," +
                KEY_CONTACT_PHONE + " TEXT," +
                KEY_CONTACT_EMAIL + " TEXT" +
                ")";
        db.execSQL(CREATE_CONTACTS_TABLE);
    }
    10 usages
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        if (oldVersion != newVersion) {
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
            onCreate(db);
        }
    }
    //This is to insert a contact into the database
    1 usage
    public void addContact(Contacts contact) {
        SQLiteDatabase database= this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(KEY_CONTACT_NAME, contact.getName());
        values.put(KEY_CONTACT_PHONE, contact.getPhoneNumber());
        values.put(KEY_CONTACT_EMAIL, contact.getEmail());

        database.insert(TABLE_CONTACTS,  nullColumnHack: null, values);
        database.close();
    }
    // This is to Fetch all contacts from the database
    2 usages
    @SuppressLint("Range")
    public ArrayList<Contacts> getAllContacts() {
        ArrayList<Contacts> contactsArrayList = new ArrayList<>();
         SQLiteDatabase database = this.getReadableDatabase();
        Cursor cursor = database.query(TABLE_CONTACTS, new String[]{KEY_CONTACT_ID, KEY_CONTACT_NAME, KEY_CONTACT_PHONE, KEY_CONTACT_EMAIL

        if (cursor.moveToFirst()) {
```



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 and we are using “int id” which serves as a unique identifier for each contact entity.

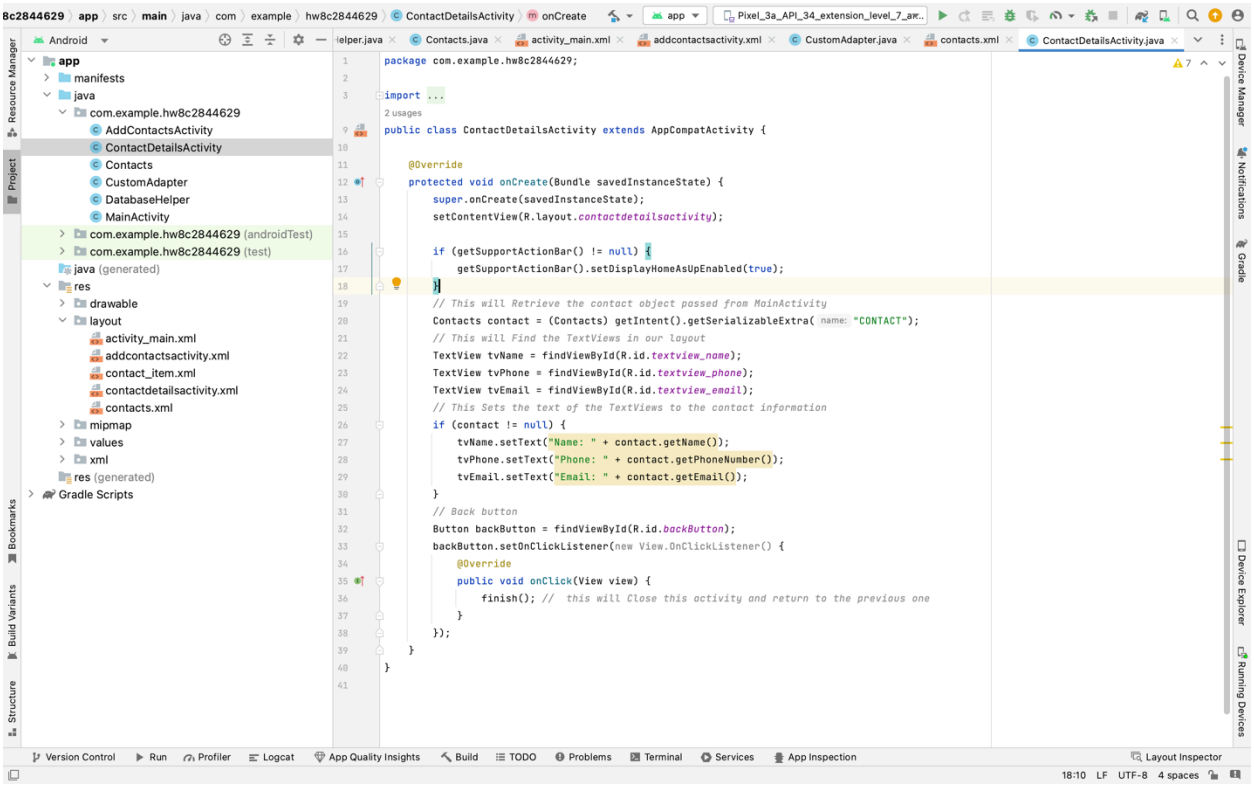


Contacts.xml : this defines the vertical list of “TextViews” used for displaying contact information i.e., name, phone and email in a simple and organized way.



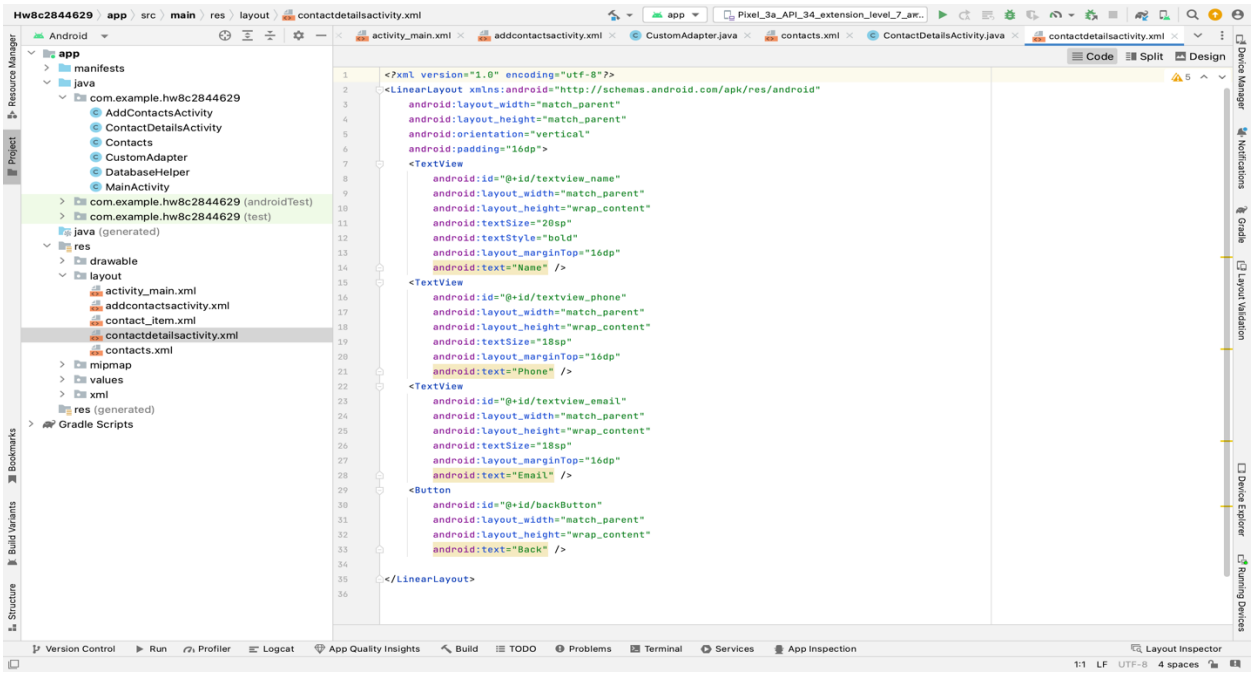
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvContactName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />
    <TextView
        android:id="@+id/tvContactPhone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp" />
    <TextView
        android:id="@+id/tvContactEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        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.



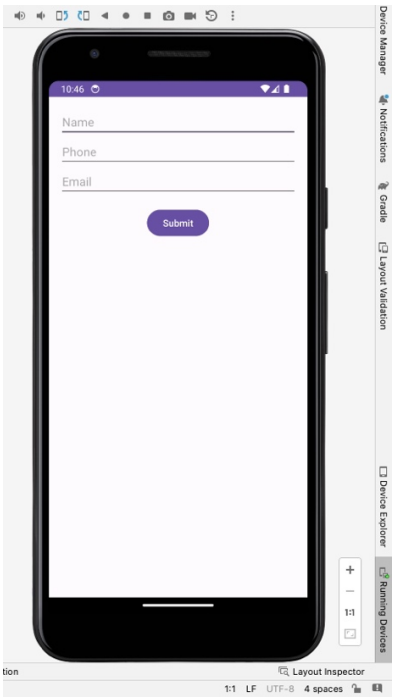
```java
package com.example.hw8c2844629;

import ...
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.

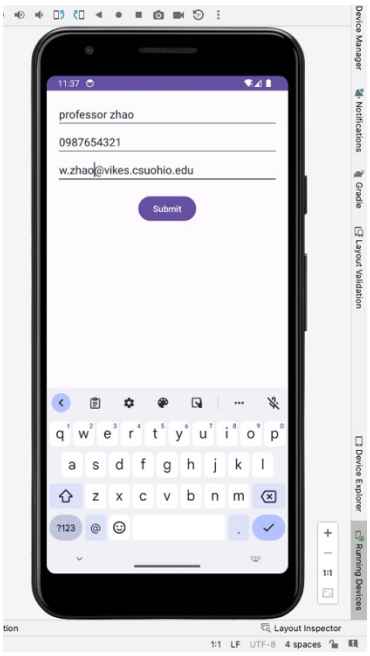


Below are the Screenshots of the app :

SS1 (MainActivity)



SS2 (Addcontactactvity)



SS3 (giving input).

SS4 (list of added contacts)

SS5 (Rerunning the app after clearing it from recent).