

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.

MainActivity.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.

The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `MainActivity.java` file, which contains Java code for a contact management application. The code includes imports for `AppCompatActivity`, `Intent`, `Bundle`, `View`, `Button`, and `ArrayList`. It defines a `MainActivity` class that extends `AppCompatActivity`. The `onCreate` method initializes a `ListView` named `listViewContacts`, creates a `CustomAdapter` named `adapter`, and sets it as the adapter for the list view. It also sets an `OnItemClickListener` for the list view to handle item selection. Finally, it finds a button with ID `R.id.button_add_contact` and sets its `onClickListener` to a lambda expression that starts an activity for contact details.

```
package com.example.hw8c2844629;

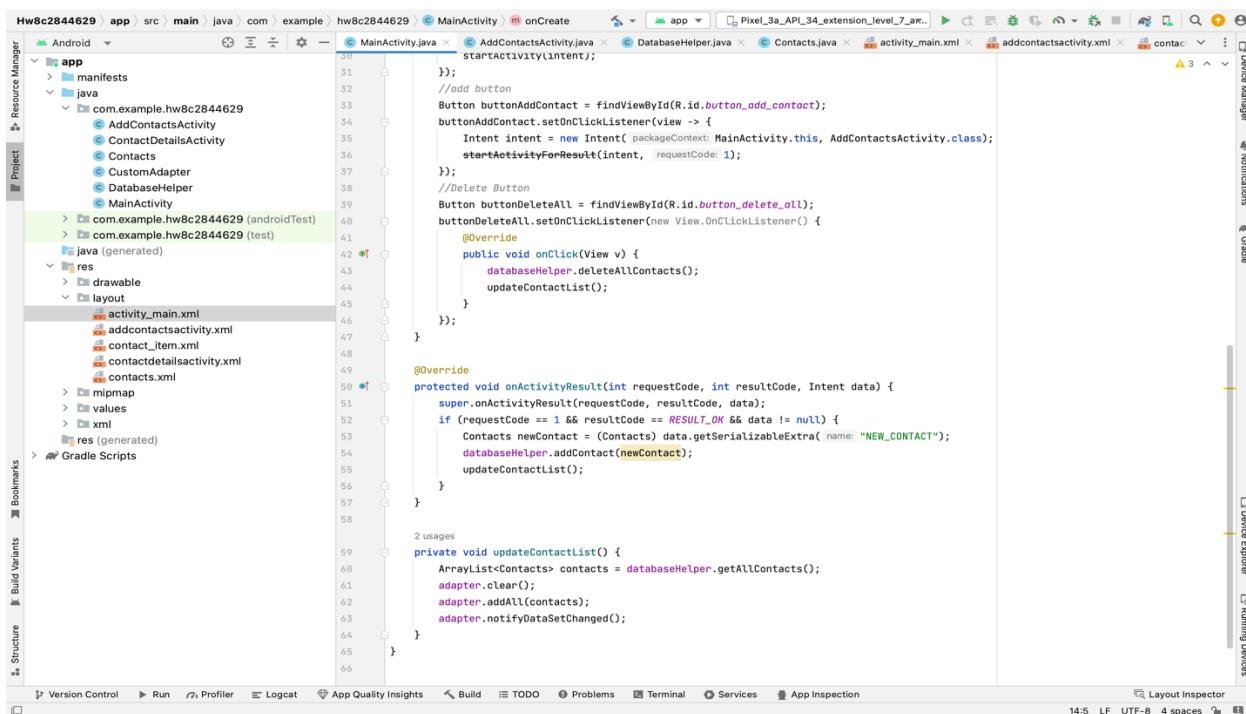
import android.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {
    ListView listViewContacts;
    CustomAdapter adapter;
    DatabaseHelper databaseHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        databaseHelper = new DatabaseHelper(this);

        listViewContacts = findViewById(R.id.listView_contacts);
        ArrayList<Contact> contacts = databaseHelper.getAllContacts();
        adapter = new CustomAdapter(context, this, contacts);
        listViewContacts.setAdapter(adapter);

        listViewContacts.setOnItemClickListener((parent, view, position, id) -> {
            Contact selectedContact = adapter.getItem(position);
            Intent intent = new Intent(packageContext, MainActivity.this, ContactDetailsActivity.class);
            intent.putExtra("CONTACT", selectedContact);
            startActivity(intent);
        });
    }
    Button buttonAddContact = findViewById(R.id.button_add_contact);
    buttonAddContact.setOnClickListener(view -> {
        // Add contact logic here
    });
}
```

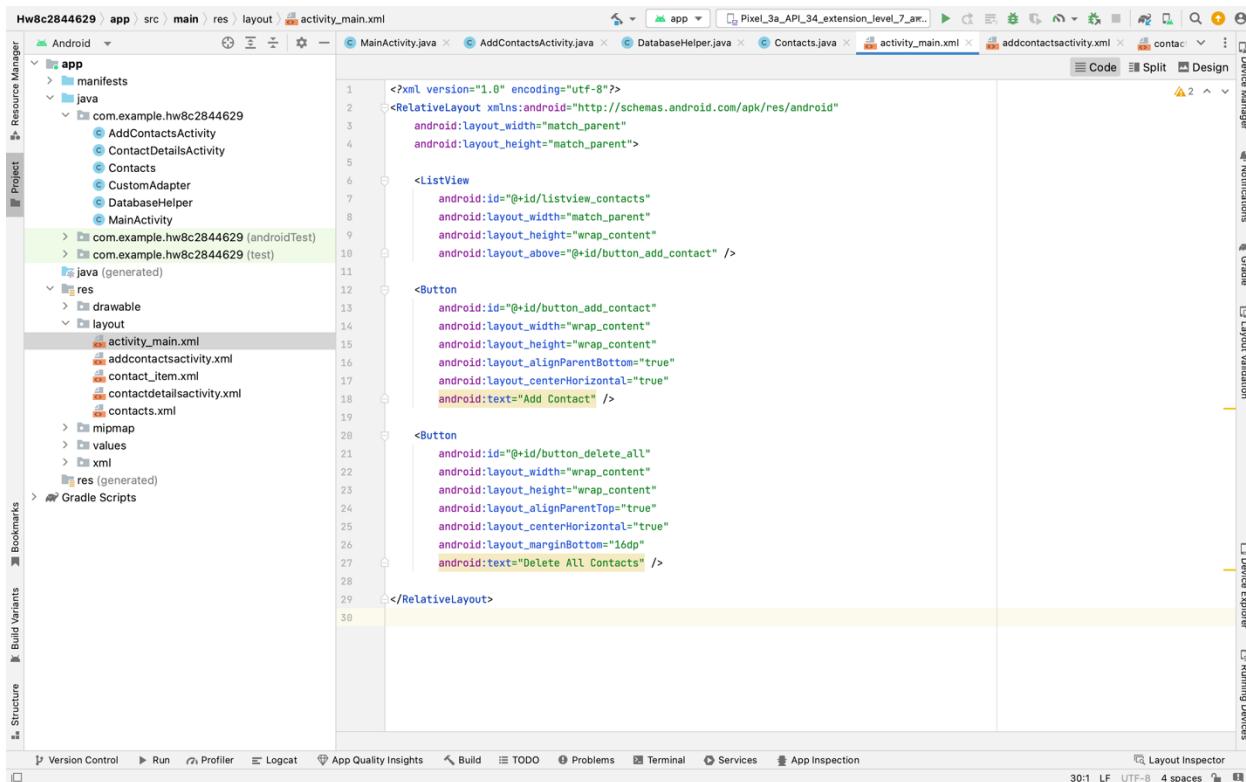


```

    MainActivity.java
    package com.example.hw8c2844629;
    ...
    public class MainActivity extends AppCompatActivity {
        ...
        Button buttonAddContact = findViewById(R.id.button_add_contact);
        buttonAddContact.setOnClickListener(v -> {
            Intent intent = new Intent(getApplicationContext(), AddContactsActivity.class);
            startActivityForResult(intent, requestCode 1);
        });
        //Delete Button
        Button buttonDeleteAll = findViewById(R.id.button_delete_all);
        buttonDeleteAll.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                databaseHelper.deleteAllContacts();
                updateContactList();
            }
        });
        ...
        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("NEW_CONTACT");
                databaseHelper.addContact(newContact);
                updateContactList();
            }
        }
        ...
        private void updateContactList() {
            ArrayList<Contacts> contacts = databaseHelper.getAllContacts();
            adapter.clear();
            adapter.addAll(contacts);
            adapter.notifyDataSetChanged();
        }
    }

```

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.



```

    activity_main.xml
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ListView
            android:id="@+id/listview_contacts"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_above="@+id/button_add_contact" />

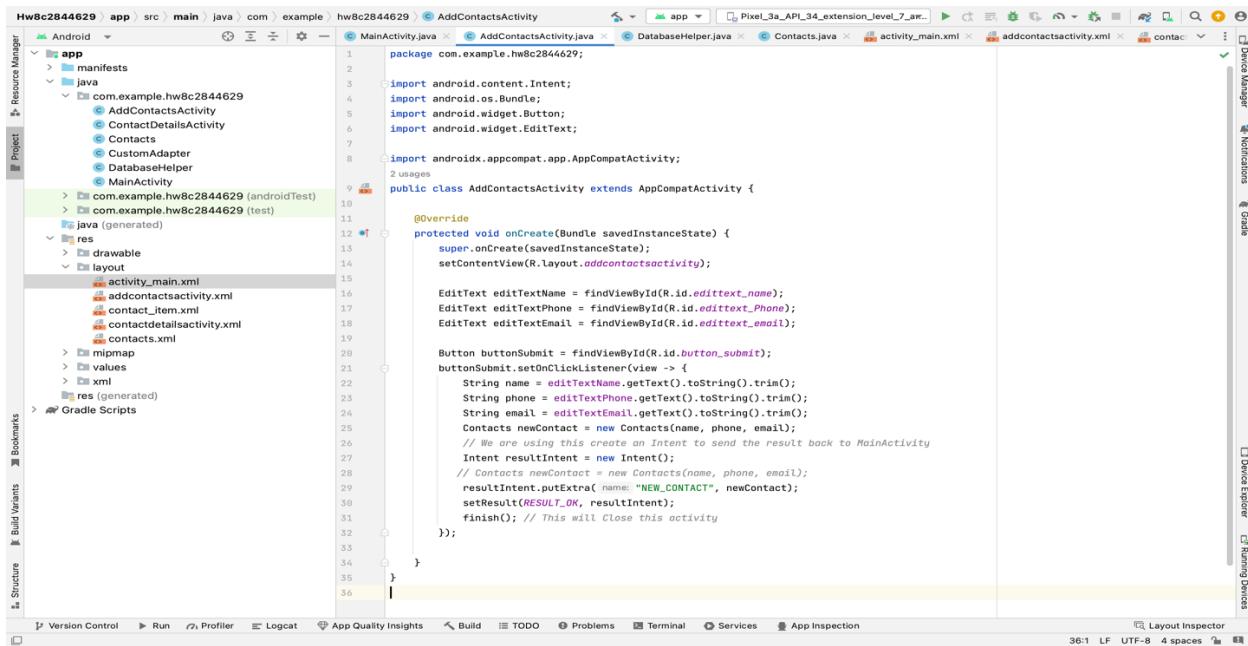
        <Button
            android:id="@+id/button_add_contact"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_centerHorizontal="true"
            android:text="Add Contact" />

        <Button
            android:id="@+id/button_delete_all"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="16dp"
            android:text="Delete All Contacts" />

    </RelativeLayout>

```

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.

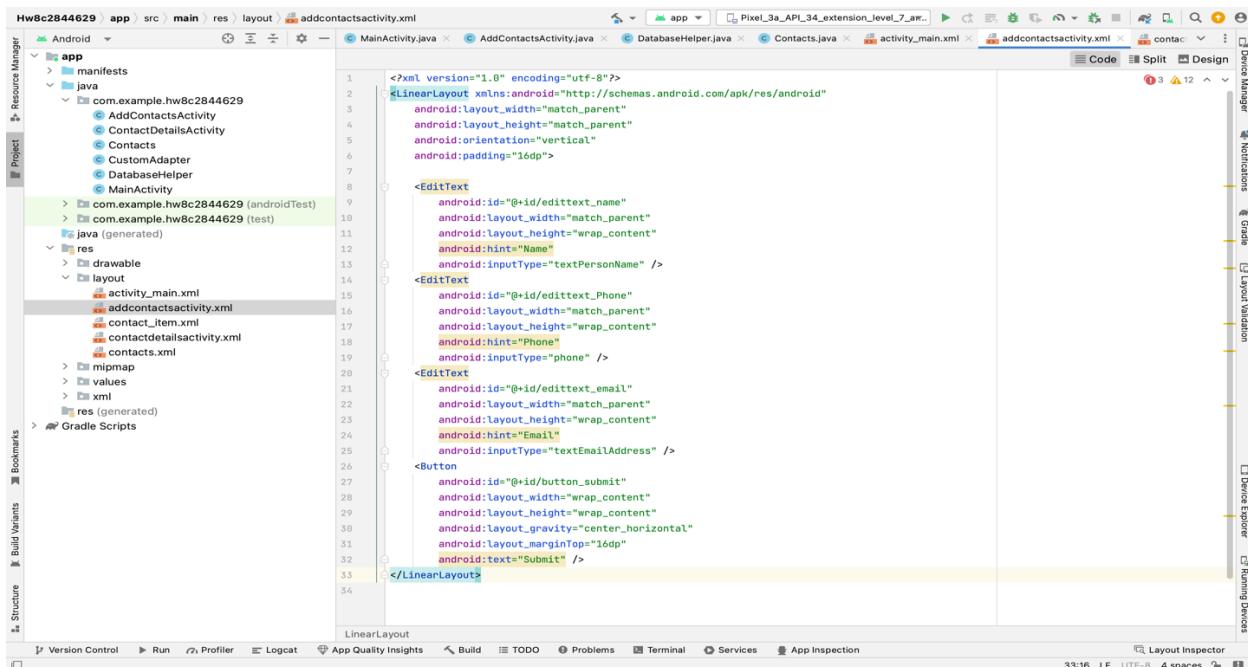


```

Hw8c2844629 > app > src > main > java > com > example > hw8c2844629 > AddContactsActivity.java
1 package com.example.hw8c2844629;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.widget.Button;
6 import android.widget.EditText;
7
8 import androidx.appcompat.app.AppCompatActivity;
9
10 public class AddContactsActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         EditText editTextName = findViewById(R.id.edittext_name);
18         EditText editTextPhone = findViewById(R.id.edittext_phone);
19         EditText editTextEmail = findViewById(R.id.edittext_email);
20
21         Button buttonSubmit = findViewById(R.id.button_submit);
22         buttonSubmit.setOnClickListener(view -> {
23             String name = editTextName.getText().toString().trim();
24             String phone = editTextPhone.getText().toString().trim();
25             String email = editTextEmail.getText().toString().trim();
26             Contacts newContact = new Contacts(name, phone, email);
27
28             // We are using this create an Intent to send the result back to MainActivity
29             Intent resultIntent = new Intent();
30             // Contacts newContact = new Contacts(name, phone, email);
31             resultIntent.putExtra("name", "NEW_CONTACT", newContact);
32             setResult(RESULT_OK, resultIntent);
33             finish(); // This will Close this activity
34         });
35     }
36

```

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.

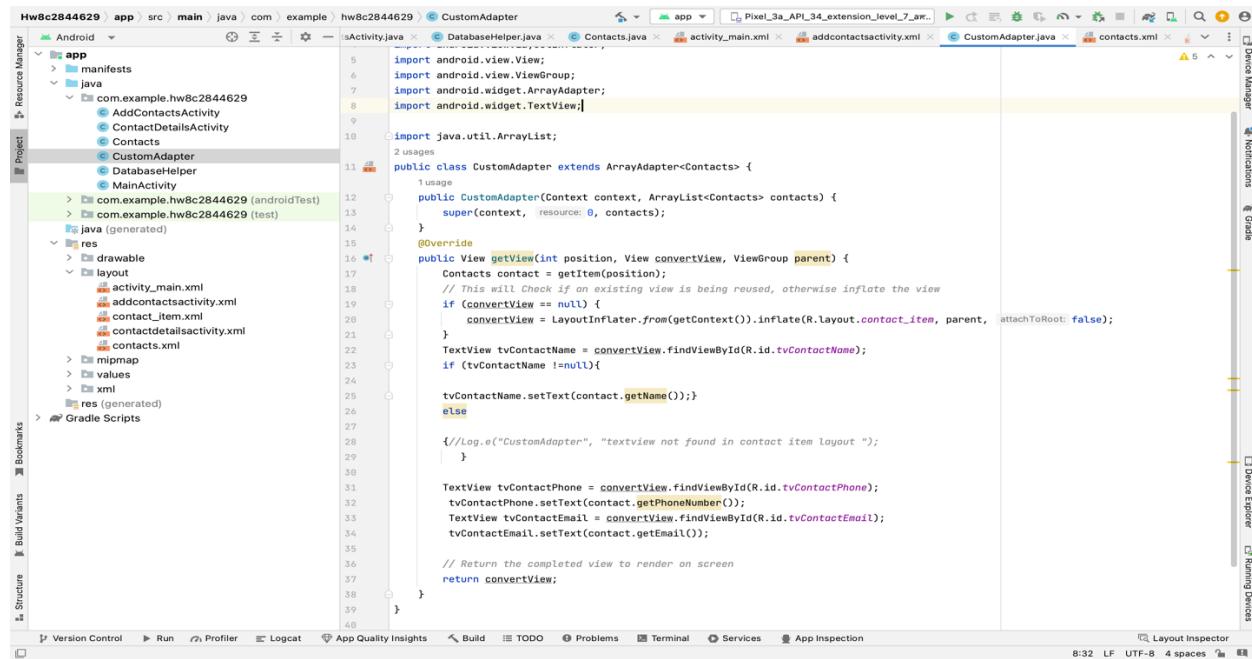


```

Hw8c2844629 > app > src > main > res > layout > addcontactsactivity.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical"
6     android:padding="16dp">
7
8     <EditText
9         android:id="@+id/edittext_name"
10        android:layout_width="match_parent"
11        android:layout_height="wrap_content"
12        android:hint="Name"
13        android:inputType="textPersonName" />
14
15     <EditText
16         android:id="@+id/edittext_phone"
17        android:layout_width="match_parent"
18        android:layout_height="wrap_content"
19        android:hint="Phone"
20        android:inputType="phone" />
21
22     <EditText
23         android:id="@+id/edittext_email"
24        android:layout_width="match_parent"
25        android:layout_height="wrap_content"
26        android:hint="Email"
27        android:inputType="textEmailAddress" />
28
29     <Button
30         android:id="@+id/button_submit"
31        android:layout_width="wrap_content"
32        android:layout_height="wrap_content"
33        android:layout_gravity="center_horizontal"
34        android:layout_marginTop="16dp"
35        android:text="Submit" />
36 </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.



```

Hw8c2844629 app src main java com example hw8c2844629 CustomAdapter
  android.view.View;
  android.widget.ArrayAdapter;
  android.widget.TextView;
import java.util.ArrayList;

public class CustomAdapter extends ArrayAdapter<Contacts> {
    public CustomAdapter(Context context, ArrayList<Contacts> contacts) {
        super(context, resource, 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

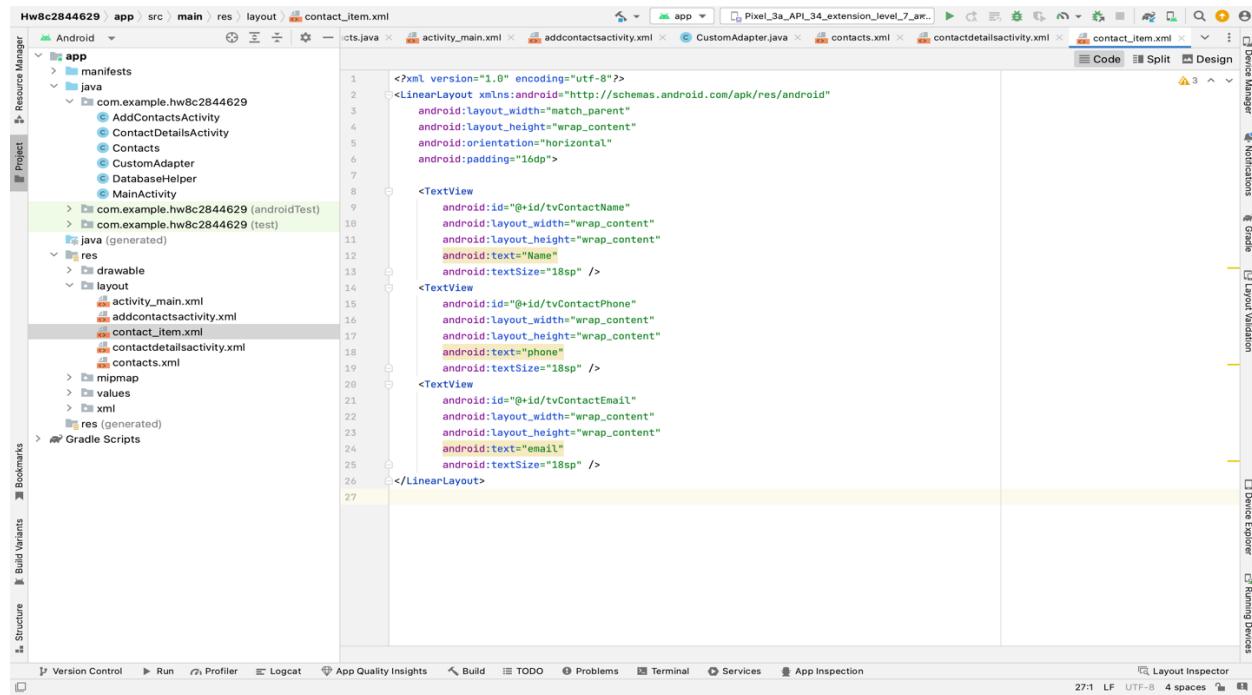
        //Log.e("CustomAdapter", "textview not found in contact item layout ");
    }

    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”.



```

Hw8c2844629 app src main res layout contact_item.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">

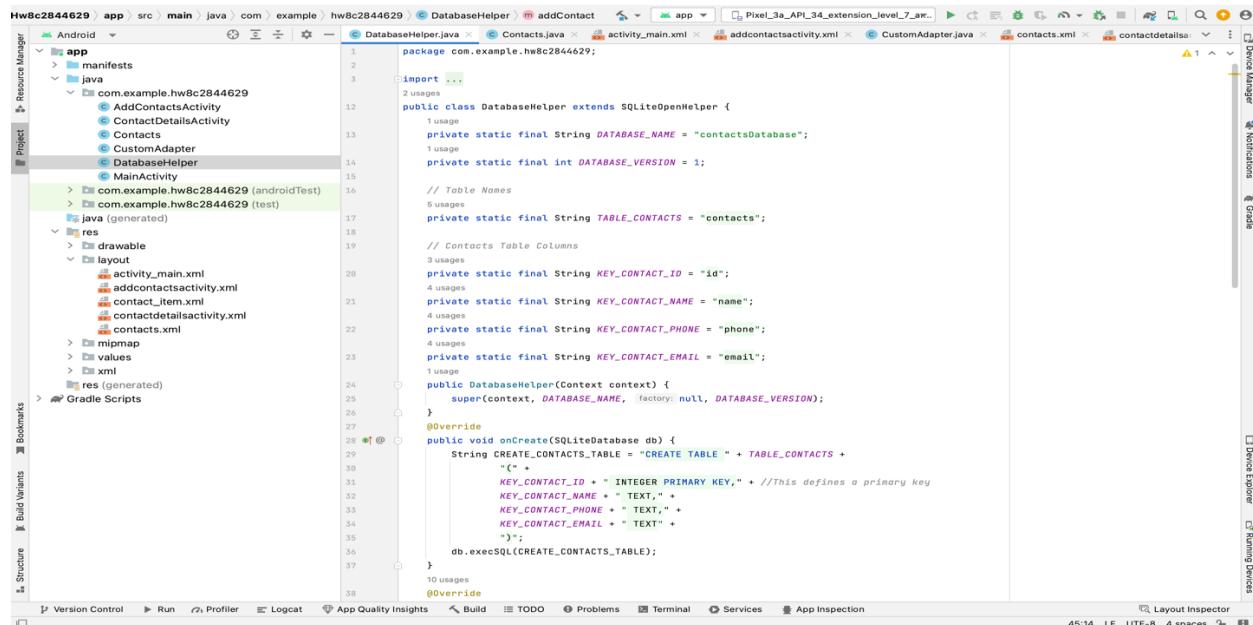
    <TextView
        android:id="@+id/tvContactName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Name"
        android:textSize="18sp" />

    <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" />

```

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 ...

public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String DATABASE_NAME = "contactsDatabase";
    private static final int DATABASE_VERSION = 1;

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

    // Contacts Table Columns
    private static final String KEY_CONTACT_ID = "id";
    private static final String KEY_CONTACT_NAME = "name";
    private static final String KEY_CONTACT_PHONE = "phone";
    private static final String KEY_CONTACT_EMAIL = "email";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, 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);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        if (oldVersion != newVersion) {
            db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS);
            onCreate(db);
        }
    }

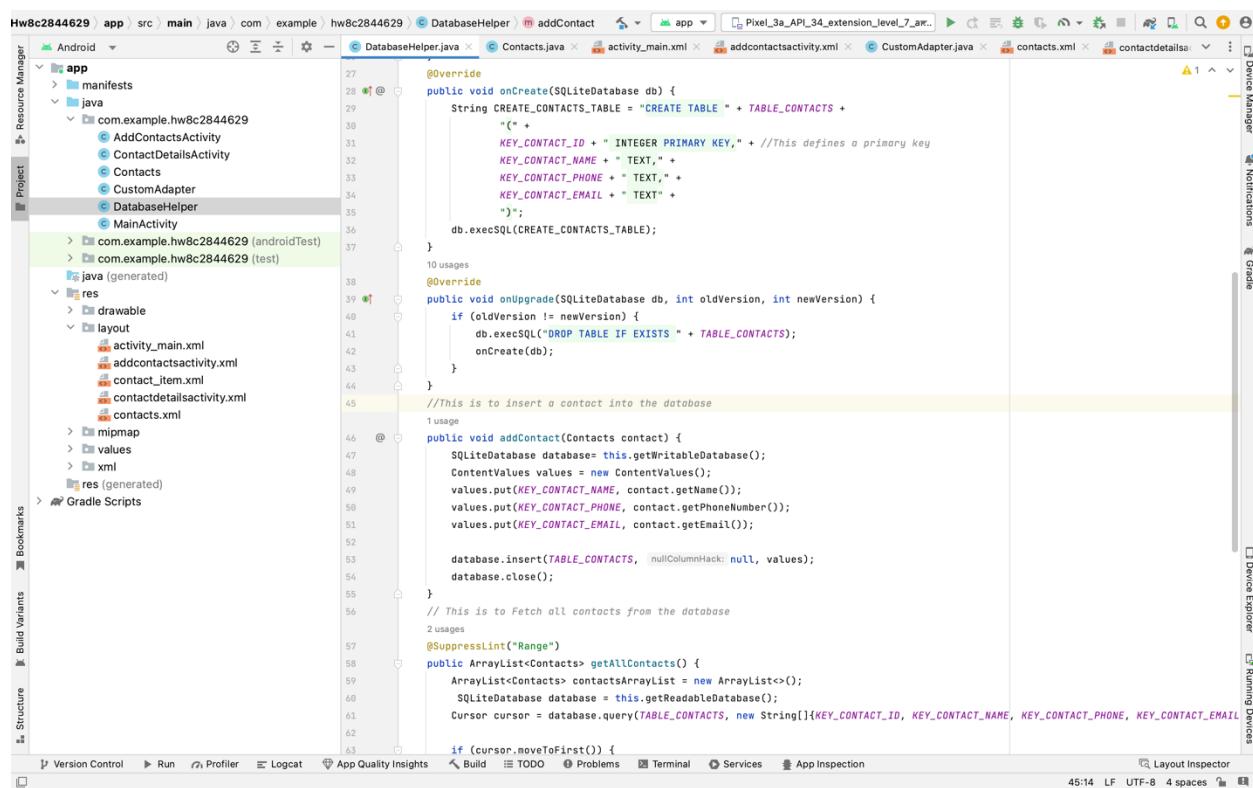
    // This is to insert a contact into the database
    public void addContact(Contact contact) {
        SQLiteDatabase database = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(KEY_CONTACT_NAME, contact.getName());
        values.put(KEY_CONTACT_PHONE, 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
    public ArrayList<Contact> getAllContacts() {
        ArrayList<Contact> 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},
            null, null, null, null, null);

        if (cursor.moveToFirst()) {
            do {
                Contact contact = new Contact();
                contact.setId(cursor.getInt(0));
                contact.setName(cursor.getString(1));
                contact.setPhoneNumber(cursor.getString(2));
                contact.setEmail(cursor.getString(3));
                contactsArrayList.add(contact);
            } while (cursor.moveToNext());
        }
        cursor.close();
        return contactsArrayList;
    }
}

```



```

@Override
public void addContact(Contact 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
public ArrayList<Contact> getAllContacts() {
    ArrayList<Contact> 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},
        null, null, null, null, null);

    if (cursor.moveToFirst()) {
        do {
            Contact contact = new Contact();
            contact.setId(cursor.getInt(0));
            contact.setName(cursor.getString(1));
            contact.setPhoneNumber(cursor.getString(2));
            contact.setEmail(cursor.getString(3));
            contactsArrayList.add(contact);
        } while (cursor.moveToNext());
    }
    cursor.close();
    return contactsArrayList;
}

```

```

53     database.insert(TABLE_CONTACTS, nullColumnHack: null, values);
54
55     database.close();
56 }
57 // This is to Fetch all contacts from the database
58 @SuppressWarnings("Range")
59 public ArrayList<Contacts> getAllContacts() {
60     ArrayList<Contacts> contactsArrayList = new ArrayList<>();
61     SQLiteDatabase database = this.getReadableDatabase();
62     Cursor cursor = database.query(TABLE_CONTACTS, new String[]{KEY_CONTACT_ID, KEY_CONTACT_NAME, KEY_CONTACT_PHONE, KEY_CONTACT_EMAIL});
63
64     if (cursor.moveToFirst()) {
65         do {
66             @SuppressLint("Range") String name = cursor.getString(cursor.getColumnIndex(KEY_CONTACT_NAME));
67             @SuppressLint("Range") String phone = cursor.getString(cursor.getColumnIndex(KEY_CONTACT_PHONE));
68             @SuppressLint("Range") String email = cursor.getString(cursor.getColumnIndex(KEY_CONTACT_EMAIL));
69
70             Contacts contact = new Contacts(name, phone, email);
71             contact.setId(cursor.getInt(cursor.getColumnIndex(KEY_CONTACT_ID)));
72             contactsArrayList.add(contact);
73         } while (cursor.moveToNext());
74
75         cursor.close();
76         database.close();
77     }
78     return contactsArrayList;
79 }
80
81 public void deleteAllContacts() {
82     SQLiteDatabase db = this.getWritableDatabase();
83     db.delete(TABLE_CONTACTS, whereClause: null, whereArgs: null);
84     db.close();
85 }

```

Contacts.java : This is essential for creating and storing the contact information.

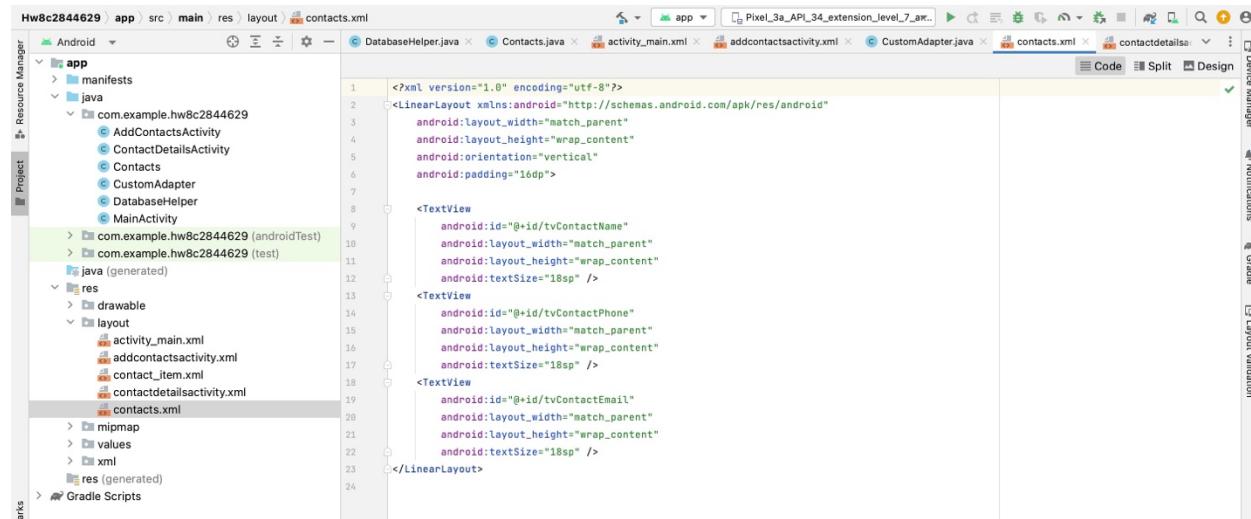
This class's "Serializable" interface enables objects to be serialized, enabling the transfer of contact information between other app components and we are using "int id" which serves as a unique identifier for each contact entity.

```

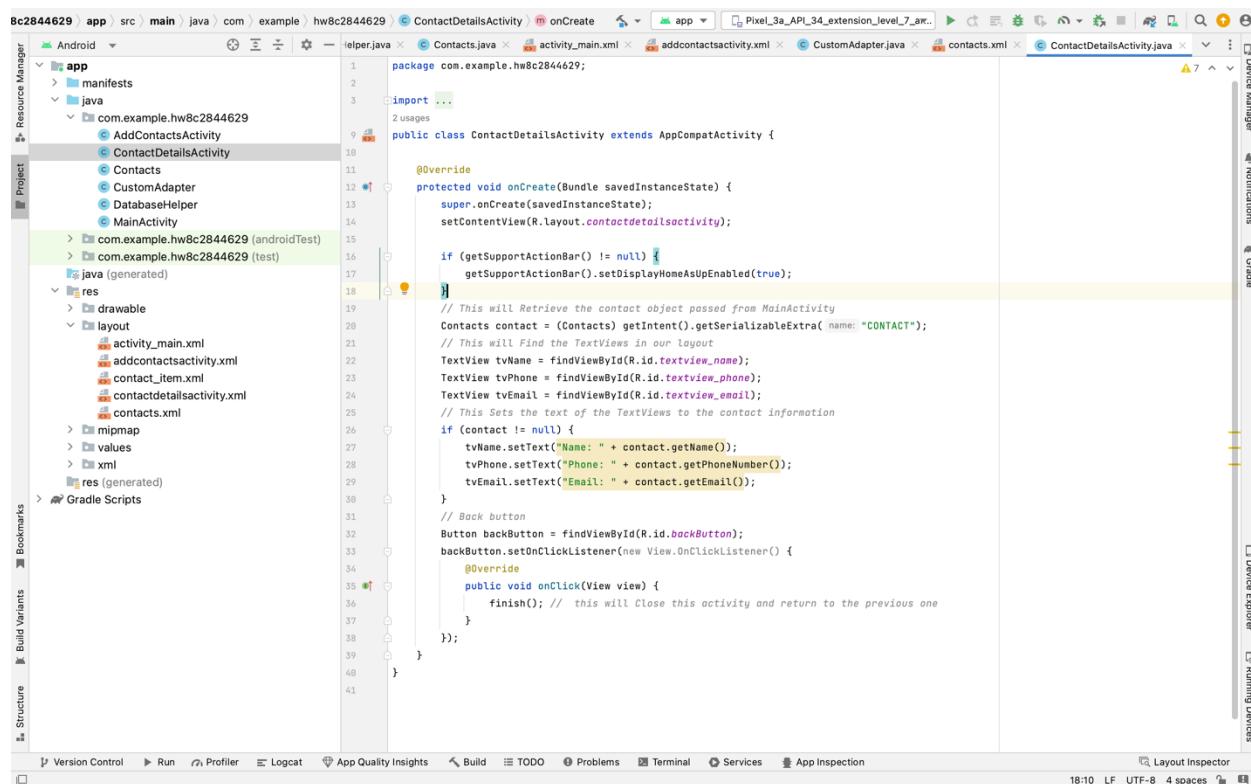
1 package com.example.hw8c2844629;
2
3 import java.io.Serializable;
4
5 public class Contacts implements Serializable {
6     private int id;
7     private String name;
8     private String phoneNumber;
9     private String email;
10
11     public Contacts(String name, String phoneNumber, String email) {
12         this.name = name;
13         this.phoneNumber = phoneNumber;
14         this.email = email;
15     }
16
17     // Getters and Setters
18     public String getName() { return name; }
19     public void setName(String name) { this.name = name; }
20
21     public String getPhoneNumber() { return phoneNumber; }
22
23     public void setPhoneNumber(String phoneNumber) { this.phoneNumber = phoneNumber; }
24
25     public String getEmail() { return email; }
26
27     public void setEmail(String email) { this.email = email; }
28
29     @Override
30     public String toString() {
31         return "Contact{" + "name='" + name + '\'' + ", phoneNumber='" + phoneNumber + '\'' + ", email='" + email + '\'';
32     }
33
34     public int getId()
35     {return id;}
36
37
38
39
40
41
42

```

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.



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.



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.

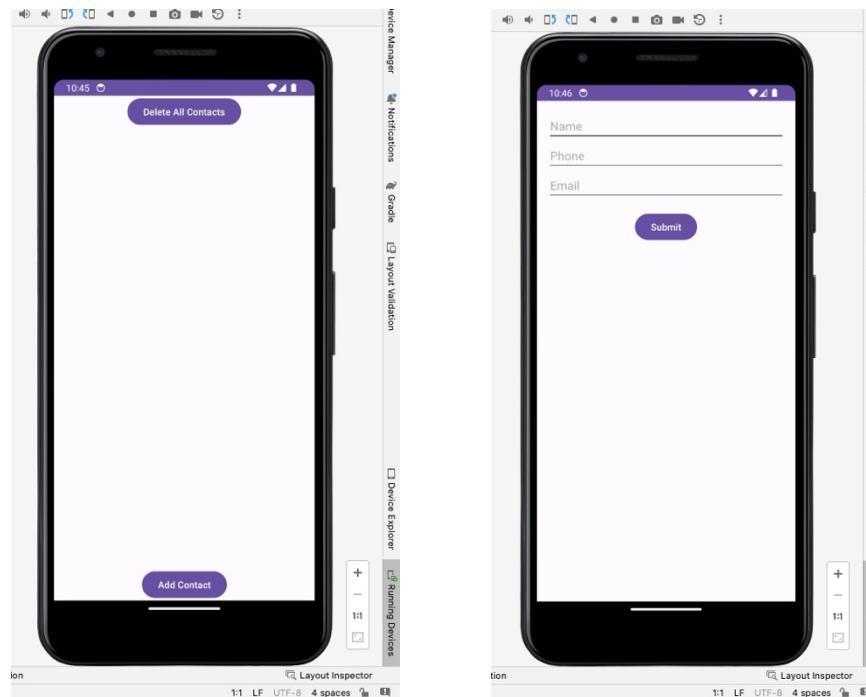
The screenshot shows the Android Studio interface with the project structure on the left and the XML code for `contactdetailsactivity.xml` on the right. The code defines a linear layout containing three text views for Name, Phone, and Email, and a back button.

```

<?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/textview_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_marginTop="16dp"
        android:text="Name" />
    <TextView
        android:id="@+id/textview_phone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_marginTop="16dp"
        android:text="Phone" />
    <TextView
        android:id="@+id/textview_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_marginTop="16dp"
        android:text="Email" />
    <Button
        android:id="@+id/backButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Back" />
</LinearLayout>

```

Below are the Screenshots of the app :



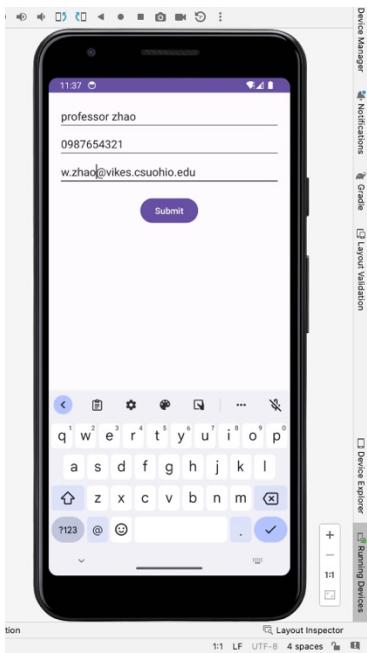
SS1 (MainActivity)

SS2 (Addcontactactvity)

2844629

KARRAR NAWAZ KHAN

ASP HW8



SS3 (giving input).

SS4 (list of added contacts)



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