NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273

HOMEWORK 6

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.

= In this homework I have created a contact app where the first screen will have the name, number and the email where a person can fill out his data and submit once submitted the second screen has all the names of the contact data saved and when you click on that name it will take you to the third screen where you find all details of that person.

In layout: ActivityOne.java this code represents a simple Android application with three activities. ActivityOne displays a list of contacts, allows the user to add new contacts using ActivityTwo, and view details of a contact by clicking on an item in the list using ActivityThree.

The FloatingActionButton (fab) is initialized using its ID from the layout.
A click listener is set on the FloatingActionButton to open ActivityTwo when clicked.

```
fab = findViewById(R.id.floatingActionButton);
fab.setOnClickListener(e -> {
   Intent intent = new Intent(ActivityOne.this, ActivityTwo.class);
   startActivityForResult(intent, 1000);
});
```

An ArrayAdapter is created to bind the contacts ArrayList to the ListView. The ListView is initialized using its ID from the layout. The adapter is set on the ListView to display the list of contacts.

```
ArrayAdapter<Contact> adapter = new ArrayAdapter<>(
        this,
        android.R.layout.simple_list_item_1,
        contacts
);
listView = findViewById(R.id.listView);
listView.setAdapter(adapter);
```

An item click listener is set on the ListView. When an item is clicked, it retrieves the selected Contact from the list and creates an Intent to open ActivityThree. Contact information is added as extras to the Intent.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Contact selectedContact = contacts.get(position);
        Intent intent = new Intent(ActivityOne.this, ActivityThree.class);
        intent.putExtra("contact_name", selectedContact.name);
        intent.putExtra("contact_phone", selectedContact.phone);
        intent.putExtra("contact_email", selectedContact.email);
        startActivity(intent);
    }
});
```

The onActivityResult method is called when ActivityTwo finishes. It checks if the requestCode is 1000 and if the resultCode is RESULT_OK. If conditions are met, it retrieves contact information from the Intent returned by ActivityTwo. It creates a new Contact object and adds it to the contacts ArrayList. The ListView is then updated with the new data using a new instance of the ArrayAdapter.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1000 && resultCode == RESULT_OK) {
        String name = data.getStringExtra("name");
        String phone = data.getStringExtra("phone");
        String email = data.getStringExtra("email");
        contacts.add(new Contact(name, phone, email));
        ArrayAdapter<Contact> adapter = new ArrayAdapter<>(
                this,
                android.R.layout.simple_list_item_1,
                contacts
        );
        listView.setAdapter(adapter);
    }
}
```

In layout: ActivityTwo.java This code defines an activity (ActivityTwo) with a simple user interface consisting of three EditText fields for entering contact information (name, phone, email) and a submit button. When the user clicks the submit button, the entered information is packaged into an Intent and sent back to the calling activity, which is presumably ActivityOne. The calling activity can then handle this result in its onActivityResult method.

These are member variables for the ActivityTwo class. editTextName, editTextPhone, editTextEmail: EditText widgets for user input. buttonSubmit: Button widget to submit the entered information.

```
private EditText editTextName;
private EditText editTextPhone;
private EditText editTextEmail;
private Button buttonSubmit;
```

The onCreate method is called when the activity is created. It sets the content view to the layout defined in activity_two.xml.

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

The EditText and Button widgets are initialized using their respective IDs from the layout.

```
editTextName = findViewById(R.id.editTextTextPersonName);
editTextPhone = findViewById(R.id.editTextPhone);
editTextEmail = findViewById(R.id.editTextTextEmailAddress);
buttonSubmit = findViewById(R.id.buttonSubmit);
```

In layout: ActivityThree.java this code defines an activity (ActivityThree) that displays contact details received through an Intent. It retrieves the contact information from the Intent's extras and updates the corresponding TextView widgets with the contact's name, phone, and email. This activity is likely opened from ActivityOne when a user clicks on a contact in the list.

he getIntent method retrieves the Intent that started this activity. If the Intent is not null, it extracts contact information (name, phone, email) from the Intent's extras. The received data is then used to set the text of the TextView widgets to display the contact details.

```
Intent intent = getIntent();
if (intent != null) {
    String name = intent.getStringExtra("contact_name");
    String phone = intent.getStringExtra("contact_phone");
    String email = intent.getStringExtra("contact_email");

    // Use the received data as needed
    TextView textViewName = findViewById(R.id.textViewName);
    TextView textViewPhone = findViewById(R.id.textViewPhone);
    TextView textViewEmail = findViewById(R.id.textViewEmail);

    textViewName.setText("Name: " + name);
    textViewPhone.setText("Phone: " + phone);
textViewEmail.setText("Email: " + email) ;
}
```

In layout: Contact.java This Contact class serves as a data model for representing contact information. It has fields for the name, phone number, and email address of a contact. The class provides a constructor to initialize these fields, and it overrides the toString method to return the name of the contact when a string representation is needed. The class can be used to create instances of contacts with specific information, and the toString method is useful when displaying a list of contacts, such as in a ListView or other UI elements.

The class has a constructor that takes three parameters (name, phone, email) and initializes the corresponding fields.

```
public Contact(String name, String phone, String email) {
    this.name = name;
    this.phone = phone;
    this.email = email;
}
```

The toString method is overridden to provide a custom string representation of a Contact object.
In this case, it returns the name of the contact when the toString method is called.

```
@NonNull
@Override
public String toString() {
    return this.name;
}
```

In layout: AndroidManifest.xml This AndroidManifest.xml file defines the configuration for the entire Android application, including the application's metadata, activities, and their intent filters. It specifies the main (launcher) activity, which is ActivityOne, and it declares two other activities (ActivityTwo and ActivityThree). The manifest also includes some application-level configurations, such as icons, labels, themes, and backup settings.

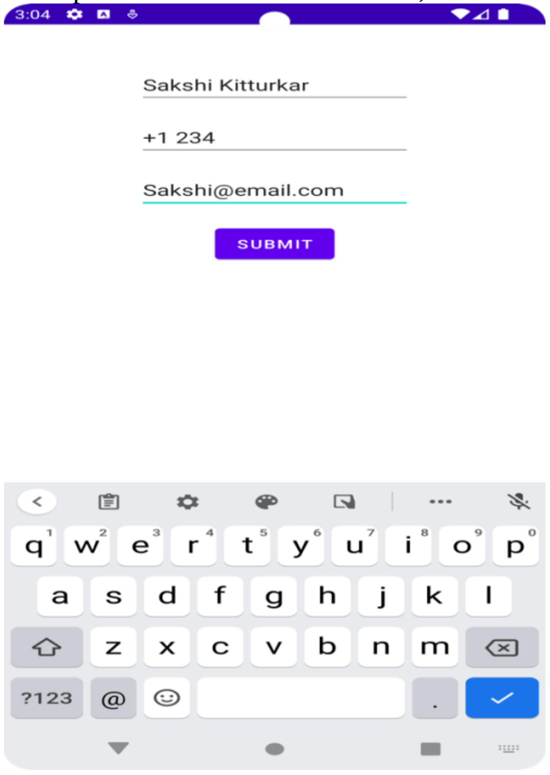
Configuration for the entire application.
allowBackup: Specifies whether to allow the application to participate in the backup and restore framework.
dataExtractionRules: Reference to XML rules for data extraction.
fullBackupContent: Reference to XML rules for full backup content.
icon: Specifies the application icon.
label: Specifies the application label.
supportsRtl: Specifies whether the application supports right-to-left (RTL) layout.
theme: Specifies the default theme for the application.
tools:targetApi: Specifies the target API level for the application.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/Theme.Hw6"
    tools:targetApi="31">
```

each <activity> element represents a screen or UI component in the application.
android:name: Specifies the Java class associated with the activity.
android:exported: Specifies whether the activity can be used by other applications. In this case, both ActivityThree and ActivityTwo are not exported (exported="false"), meaning they are only for internal use. ActivityOne is exported, and it is also specified as the main (launcher) activity.
<intent-filter>: Defines the types of intents that an activity can respond to.
<action>: Specifies the main action to be performed (in this case, the main entry point).
<category>: Specifies the category of the intent (in this case, the launcher category).

```
<activity
    android:name=".ActivityThree"
    android:exported="false" />
<activity
    android:name=".ActivityTwo"
    android:exported="false" />
<activity
    android:name=".ActivityOne"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Example 1: First Screen with name, contact and email

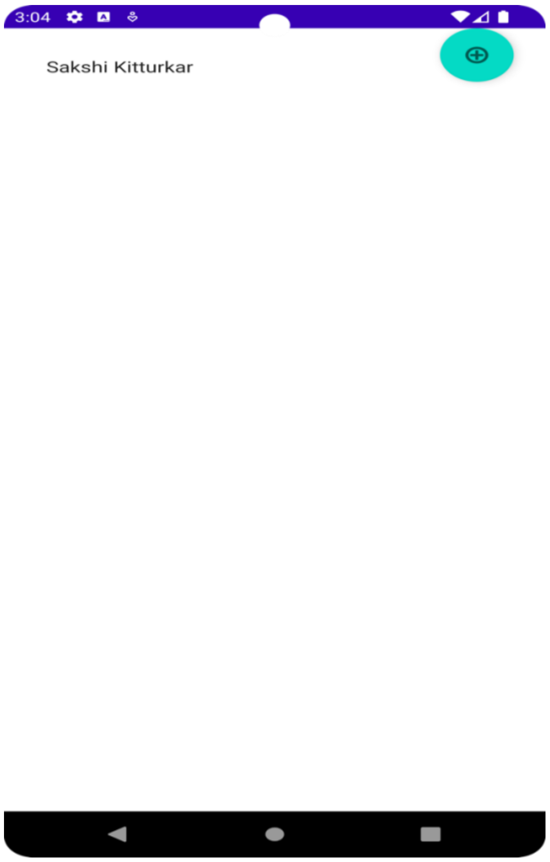


Sakshi Kitturkar

+1 234

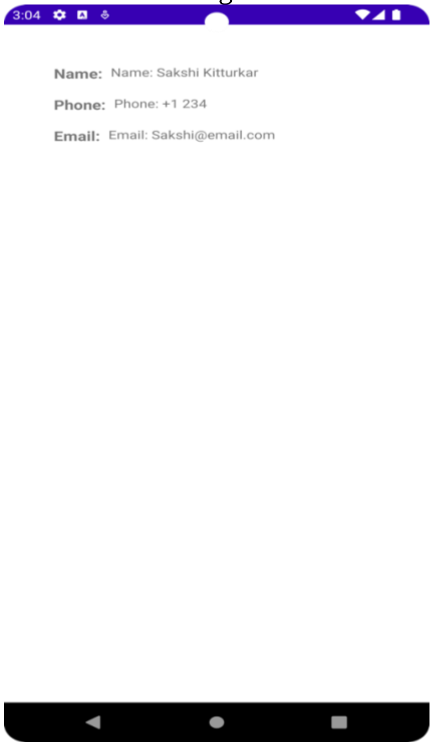
Sakshi@email.com

Second screen with Full name shown in ListView



Sakshi Kitturkar

Third after clicking the name the whole information about that person is shown.

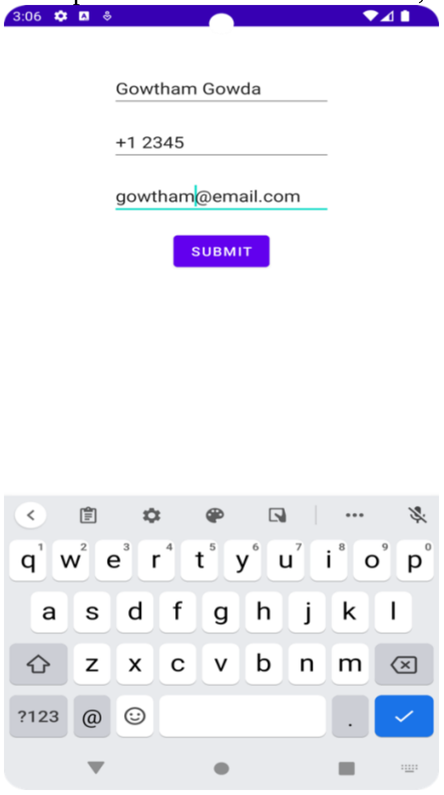


Name: Name: Sakshi Kitturkar

Phone: Phone: +1 234

Email: Email: Sakshi@email.com

Example 2: First Screen with name, contact and email

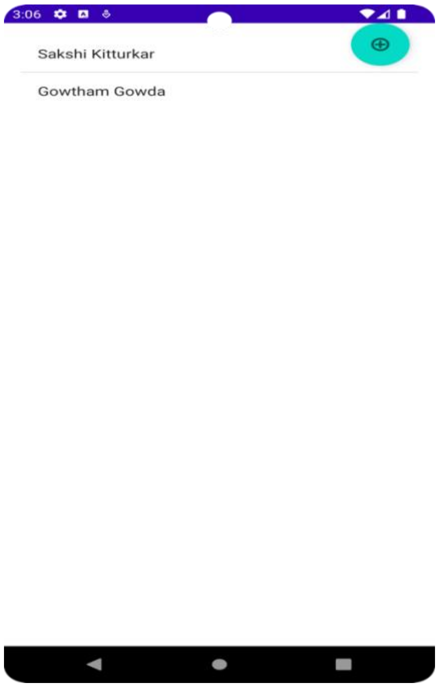


Gowtham Gowda

+1 2345

gowtham@email.com

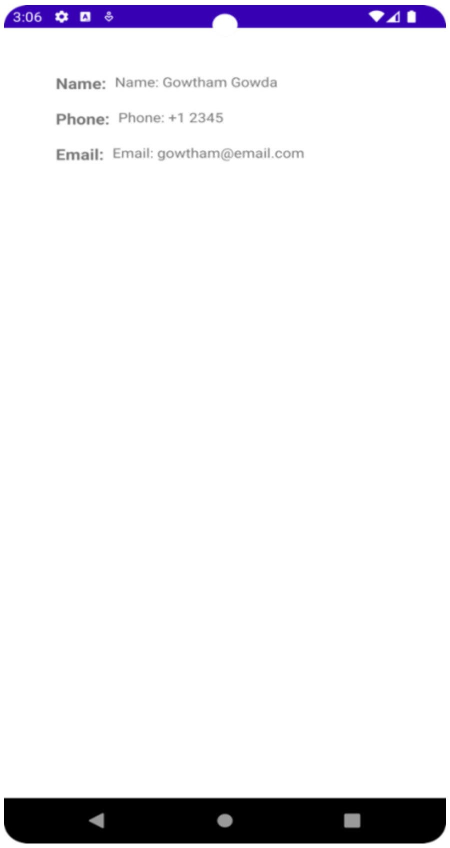
Second screen with Full name shown in ListView



Sakshi Kitturkar

Gowtham Gowda

Third after clicking the name the whole information about that person is shown.

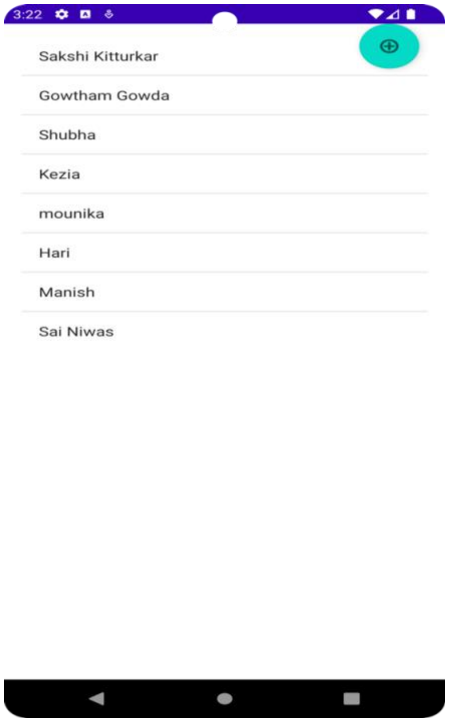


Name: Name: Gowtham Gowda

Phone: Phone: +1 2345

Email: Email: gowtham@email.com

List View



Sakshi Kitturkar

Gowtham Gowda

Shubha

Kezia

mounika

Hari

Manish

Sai Niwas