

NAME: SAKSHI VINAYAK KITTURKAR
CSU ID: 2860273
HOMEWORK 8

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.

= I have created app where the contact information will be saved in SQLite database even when we close the app and open it again, we will be able to see the contacts list.

In layout: MainActivity The code represents an Android app for managing contacts. It uses a Room database to store contact information and features a main activity (MainActivity) with a FloatingActionButton for adding contacts and a ListView for displaying them. The onCreate method initializes UI components, sets up event listeners, and asynchronously retrieves contacts from the database. The app launches a second activity (ActivityTwo) to add new contacts and a third activity (ActivityThree) to display contact details. The code uses coroutines for asynchronous tasks and demonstrates activity result handling for updating the contact list.

Here, the MainActivity class is defined, extending AppCompatActivity. This is the main activity of the Android application.

```
class MainActivity : AppCompatActivity() {  
    // ... (rest of the class code)  
}
```

This section declares member variables, including a list to store contacts, UI components (FloatingActionButton, ListView, ContactAdapter), and a ContactDao instance for database operations.

```
// Member variables  
private var contactsList: List<Contact> = emptyList()  
private lateinit var fabAdd: FloatingActionButton  
lateinit var listView: ListView  
lateinit var adapter: ContactAdapter  
private lateinit var contactDao: ContactDao
```

The `startForResult` is an `ActivityResultLauncher` created using `registerForActivityResult`. It handles results returned from other activities, specifically launched to add new contacts.

```
// Activity result launcher
val startForResult =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result:
ActivityResult ->
    // ... (rest of the callback code)
}
```

The `onCreate` method is called when the activity is created. It sets the content view, initializes UI components, sets up event listeners, and initializes the Room database.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    // ... (rest of the method code)
}
```

This is the callback for the activity result. It checks if the result is successful, extracts contact information, inserts a new contact into the Room database, and updates the UI.

```
val startForResult =
registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result:
ActivityResult ->
    if (result.resultCode == Activity.RESULT_OK) {
        val results = result.data?.extras
        // ... (rest of the callback code)
    }
}
```

In layout: `ActivityTwo` The code represents an Android activity (`ActivityTwo`) for entering contact information. It includes three `EditText` fields for name, phone, and email, along with a submit Button. When the user clicks the submit button, the entered data is retrieved, packaged into an Intent, and sent back to the calling activity with a result code indicating success (`RESULT_OK`). The activity then finishes, returning to the calling activity.

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);
// ... (rest of the method)
}

```

These lines initialize the member variables by finding the corresponding views in the layout using their resource IDs.

```

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

```

Sets up a click listener for the buttonSubmit. When the button is clicked, the code inside the onClick method will be executed.

```

buttonSubmit.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // ... (rest of the click listener)
    }
});

```

In the onClick method, the code retrieves the text entered into the EditText fields for name, phone, and email. It then creates an Intent to pass this data back to the calling activity.

```

String userName = editTextName.getText().toString();
String userPhone = editTextPhone.getText().toString();
String userEmail = editTextEmail.getText().toString();

```

```

Intent resultIntent = new Intent();
resultIntent.putExtra("name", userName);
resultIntent.putExtra("phone", userPhone);
resultIntent.putExtra("email", userEmail);

```

```

setResult(Activity.RESULT_OK, resultIntent);
finish();

```

In layout: ActivityThree The code represents an activity designed to display contact details. It retrieves contact information from the incoming intent, assuming it's not null, and updates the UI accordingly by setting the text of the relevant TextView elements with the received data.

Declares member variables for three TextView elements that will display the contact's name, phone, and email.

```
private TextView textViewName;  
private TextView textViewPhone;  
private TextView textViewEmail;
```

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

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_three);  
    // ... (rest of the method)  
}
```

These lines initialize the member variables by finding the corresponding TextView views in the layout using their resource IDs.

```
textViewName = findViewById(R.id.textViewName);  
textViewPhone = findViewById(R.id.textViewPhone);  
textViewEmail = findViewById(R.id.textViewEmail);
```

Retrieves the intent that started this activity. Checks if the intent is not null before proceeding.

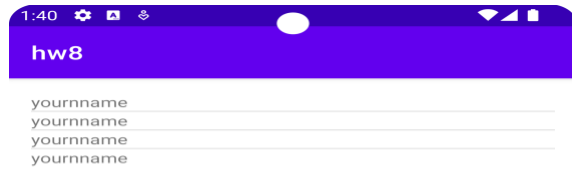
```
Intent intent = getIntent();  
if (intent != null) {  
    // ... (rest of the code)  
}
```

Uses the extracted data to set the text for the TextView elements, displaying the contact's name, phone, and email.

```
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);
```

First Screen:



1:40

hw8

yourname

yourname

yourname

yourname

This is a mobile app UI mockup for the first screen. It features a white status bar at the top with the time '1:40' and system icons. Below is a blue header bar with the text 'hw8'. The main content area is white and contains four horizontal input fields, each with the placeholder text 'yourname'.



Second Screen adding up contacts:

1:40

hw8

sakshi

+1

email@email.com

SUBMIT



Third Screen adding up contacts:

1:41

hw8

gowtham

+1

email@email.com

SUBMIT



After closing the screen when you open you will get all the contacts

