StudentDatabaseApp: Android Application

By

Sourabh Sinha (23SCSE2150018)

1. Introduction

The Student Database App is an Android application designed to streamline the management of student records using a local SQLite database. The app allows users to add, update, delete, and retrieve student information through an intuitive and user-friendly interface. The primary goal of this application is to simplify CRUD (Create, Read, Update, Delete) operations, enabling users to perform these tasks efficiently with minimal technical knowledge. Android's SQLite database system is employed to store the data locally on the device, making the app ideal for environments where internet connectivity is unreliable or unavailable.

The inspiration behind this app comes from the need to maintain student records in small educational institutions or as a personal record-keeping tool. The app can store details like Student ID, Name, Age, and Class. The user interacts with the database through four main activities: adding a new student, updating existing student records, deleting a student's information, and retrieving the full list of students.

The app design adheres to Android development best practices, ensuring that it is responsive and user-friendly. Each activity is clearly separated by functionality, ensuring ease of use even for non-technical users. SQLite was chosen as the database due to its simplicity and its ability to function without any setup or external dependencies, unlike more complex database management systems. It also integrates seamlessly with Android, allowing developers to manage data without requiring additional third-party libraries.

In summary, the Student Database App simplifies the management of student records, combining functionality, ease of use, and performance to deliver an application that meets the needs of educational institutions, teachers, and students. Whether adding new students, updating information, or retrieving the complete database, the app offers a one-stop solution for efficient data management.

2. Activities Details

The Student Database App consists of five major activities: **MainActivity**, **AddStudentActivity**, **UpdateStudentActivity**, **DeleteStudentActivity**, and **RetrieveStudentActivity**. Each of these activities is designed to handle one core function of the app, enabling seamless interaction with the database.

- **MainActivity:** This is the landing page of the application, where users can choose from four buttons—Add, Update, Delete, or Retrieve. Each button is assigned a specific task and directs the user to the respective activity. The Intent class in Android is used to switch between activities. The user experience is kept simple, with a straightforward layout and easily recognizable button labels.

- **AddStudentActivity:** This activity provides a form with fields for entering the student's details: Name, Age, and Class. Upon entering the required information and hitting the "Submit" button, the data is stored in the SQLite database using the insertData() method in the DatabaseHelper class. Input validation is implemented to ensure no fields are left empty, and appropriate error messages are displayed in case of invalid entries.

- **UpdateStudentActivity:** In this activity, users can update a student's existing details. The user first enters the Student ID, which triggers a database query to fetch the existing details. Once fetched, the data is displayed in editable text fields, allowing the user to modify the details. After the updates are made, the user submits the form, and the data is updated in the database using the updateData() method. Error handling is employed to manage cases where a non-existent Student ID is entered.

- **DeleteStudentActivity:** This activity allows users to delete a student record by entering the Student ID. After input, the app searches for the record in the database and deletes it using the deleteData() method. A confirmation message is displayed upon successful deletion. Error handling ensures that an appropriate message is shown if the Student ID does not exist.

- **RetrieveStudentActivity:** In this activity, all student records stored in the SQLite database are retrieved and displayed in a list. The app uses a Cursor object to fetch the data from the database, which is then displayed in a ListView or ScrollView for easy viewing. Each student's details—Student ID, Name, Age, and Class—are displayed in a structured format, making it easy for users to navigate through the records.

Together, these activities ensure that all CRUD operations are implemented and executed effectively, providing a complete solution for student database management.

3. Layout Details

The design of the Student Database App revolves around simplicity, ease of use, and clarity. The UI is developed using XML in Android Studio, and the layout for each activity is designed to provide a clean and intuitive interface for users.

- **MainActivity Layout**: The main screen contains four buttons—Add, Update, Delete, and Retrieve—arranged vertically using a LinearLayout. The layout is designed to make navigation straightforward for the user, ensuring that each button is clearly labeled and accessible. The background is kept simple, with padding and margins adjusted to enhance the visual hierarchy.

- **AddStudentActivity Layout:** This layout consists of three EditText fields for entering the student's Name, Age, and Class, along with a Button for submitting the data. The EditText fields use input constraints to ensure valid data is entered (e.g., numeric input for age). The button triggers the insertData() method to store the data in the database.

- **UpdateStudentActivity Layout:** The layout here is similar to AddStudentActivity, but with an additional field for entering the Student ID. The ID field is used to fetch existing details, which are then displayed in the editable EditText fields. After the user makes changes, the updated information is submitted using a button.

- **DeleteStudentActivity Layout:** The layout for this activity contains a single EditText field for entering the Student ID and a button to trigger the deletion of the corresponding record. The layout is minimalist to ensure the user can focus on the task at hand.

- **RetrieveStudentActivity Layout:** The retrieved student data is displayed in a ListView, where each record is listed in a structured format. The layout is scrollable, ensuring that even large datasets can be viewed easily.

Each activity's layout is designed with user experience in mind, ensuring that the app remains easy to navigate while providing a visually appealing interface.

4. Steps

Developing the Student Database App involves several crucial steps, each contributing to the overall functionality and user experience. Below is a step-by-step guide to the development process:

1. Setting Up Android Studio: Start by setting up a new project in Android Studio, configuring it for Java as the primary programming language. Ensure that all necessary dependencies for SQLite are available. This step includes setting up the project structure, configuring Gradle, and preparing the AndroidManifest file.

2. Designing the UI: Begin by designing the layouts for all the activities using XML. Start with the MainActivity, where users can choose between Add, Update, Delete, or Retrieve actions. Each activity should have its own dedicated layout that matches the required functionality (e.g., input fields for adding students, list views for retrieving records). Focus on creating an intuitive interface with minimal complexity.

3. Implementing the SQLite Database: Develop the DatabaseHelper class, which extends SQLiteOpenHelper. This class is responsible for creating the database and managing the CRUD operations (Create, Read, Update, Delete). Define the schema for the student_table with fields for ID, Name, Age, and Class. Implement methods such as insertData(), updateData(), deleteData(), and getAllData() to handle the respective database operations.

4. Coding CRUD Functionalities:

- In the AddStudentActivity, collect user input and pass it to the insertData() method, which inserts the record into the database.
- In the UpdateStudentActivity, use the provided Student ID to fetch the current record, display it in editable fields, and update the record in the database using updateData().
- In the DeleteStudentActivity, delete the student record based on the entered Student ID and display a confirmation message.
- In the RetrieveStudentActivity, retrieve all student records from the database and display them in a list view using a Cursor object.

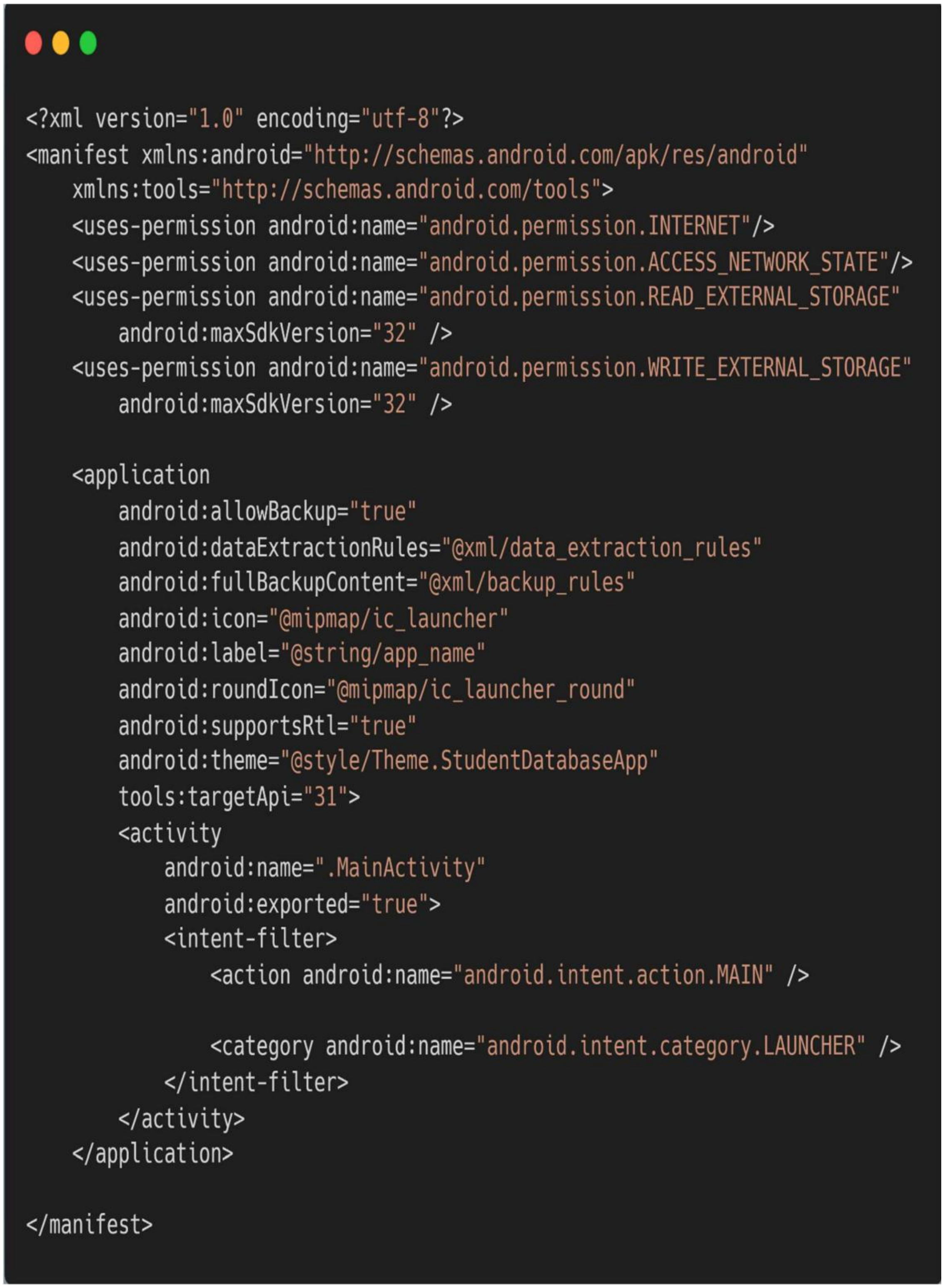
5. Testing and Debugging: Once the functionalities are implemented, thoroughly test each activity on both the Android emulator and a physical device. Test edge cases, such as entering invalid data or attempting to delete non-existent records. Debug any issues that arise and ensure smooth functionality across all activities.

6. Finalizing the App: After testing, refine the UI to make it more visually appealing. Consider adding additional error handling, user feedback (e.g., toast messages), and visual enhancements like animations or color changes to improve the user experience.

By following these steps, the app is developed in a systematic manner, ensuring that each feature is fully functional and integrated into the larger system.

5. Manifest File Code

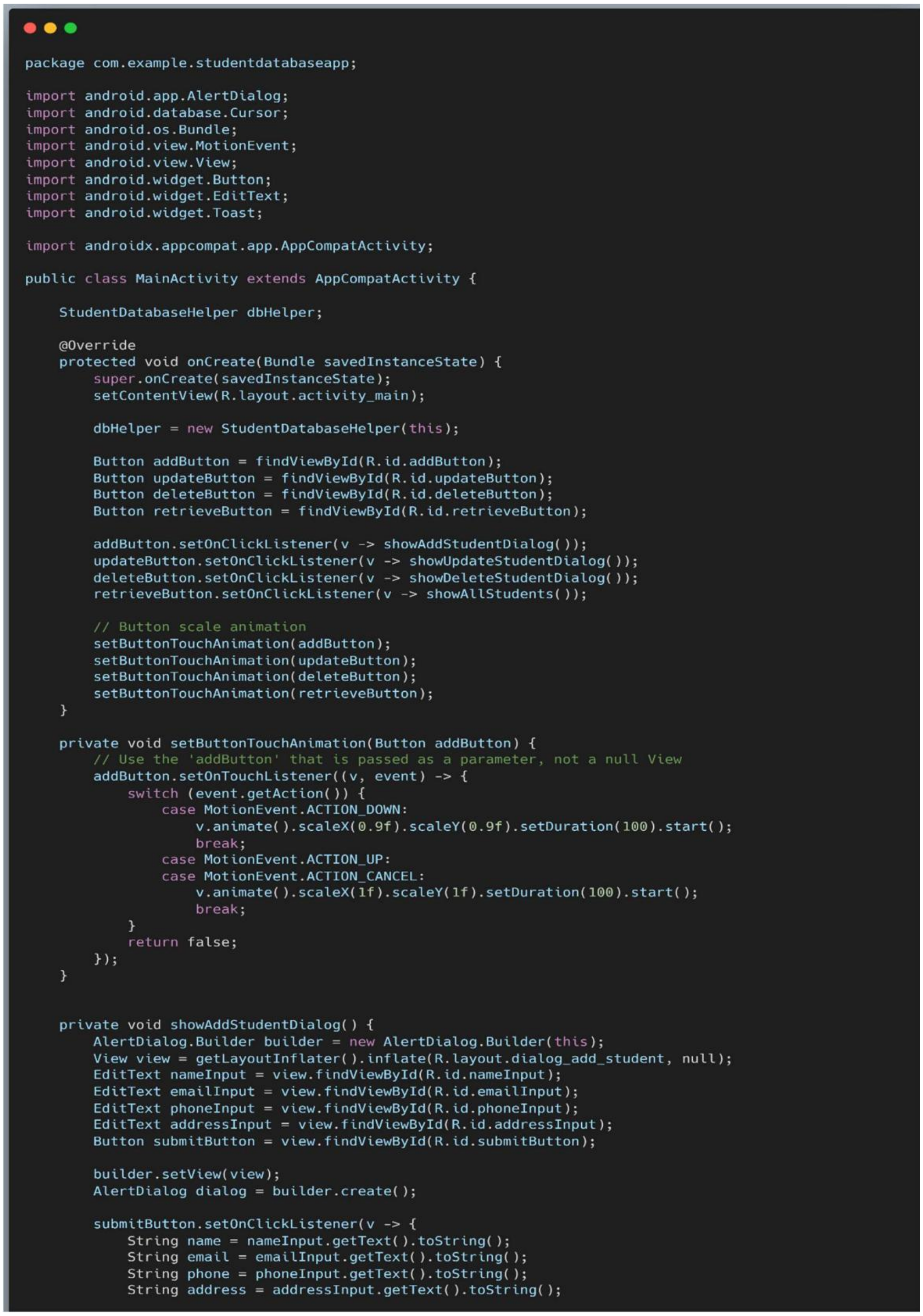
The AndroidManifest.xml file is essential for any Android application as it provides the system with critical information about the app, its components, and its permissions. For the Student Database App, the manifest file ensures that the necessary activities are registered and defines any permissions that the app requires for its functionality.



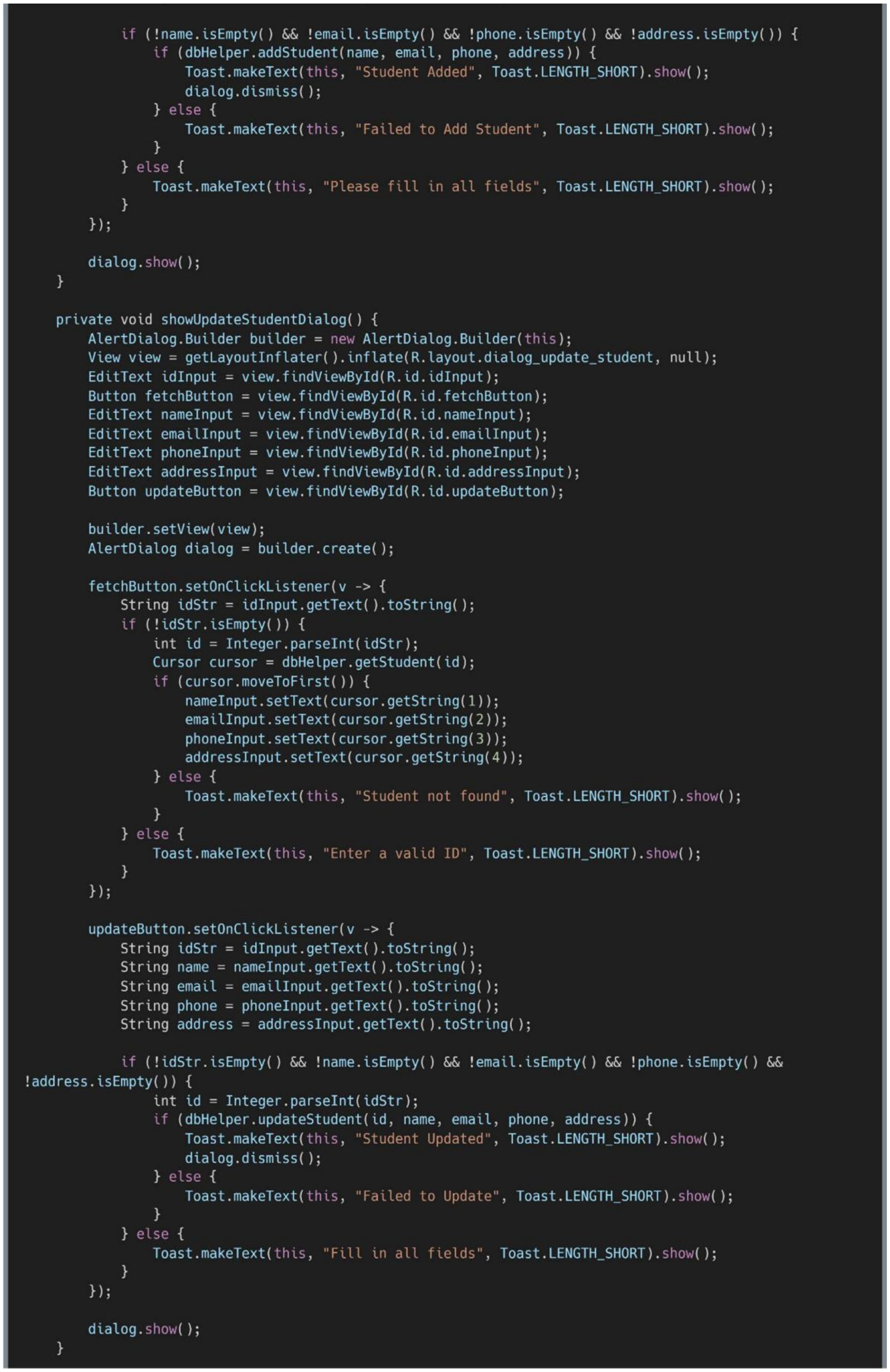
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />

    <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:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.StudentDatabaseApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

6. Supporting Files/Code

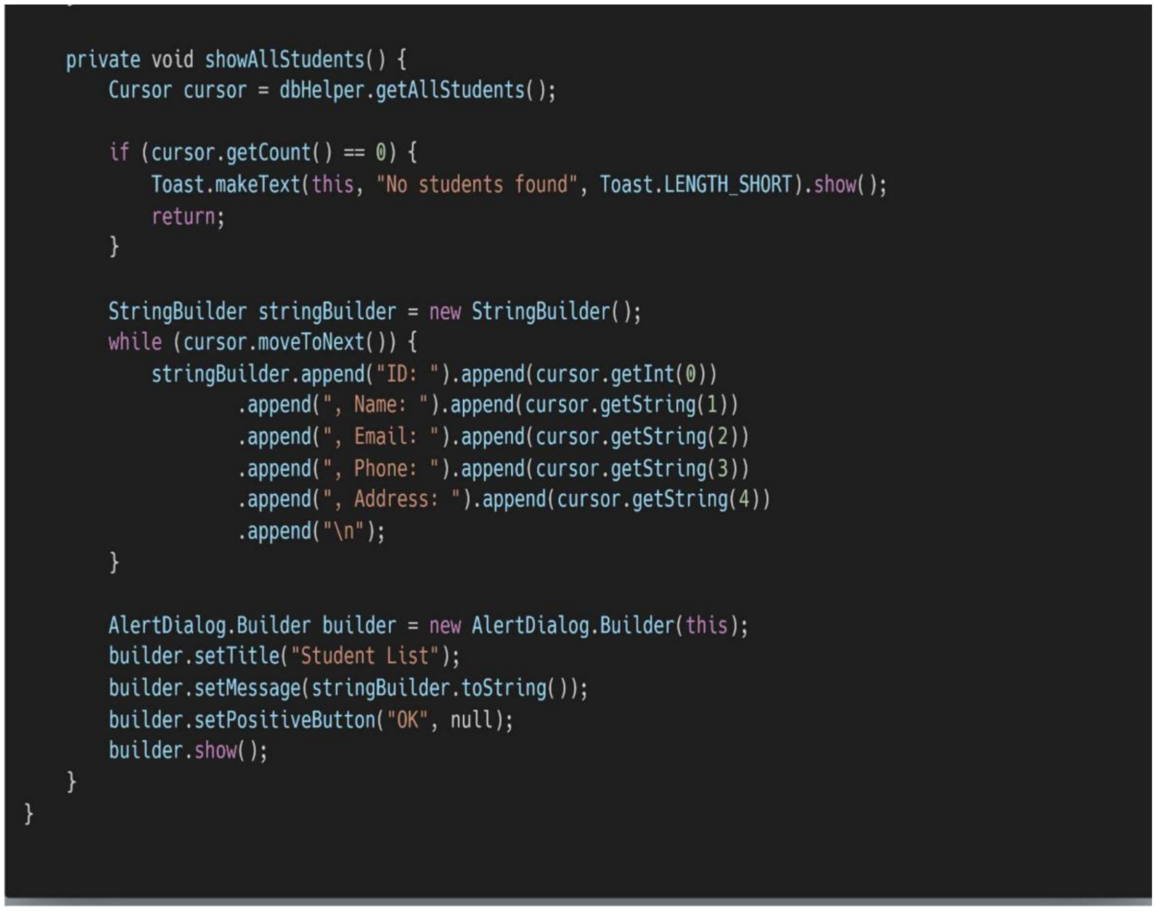
MainActivity.java



```java
package com.example.studentdatabaseapp;

import android.app.AlertDialog;
import android.database.Cursor;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    StudentDatabaseHelper dbHelper;

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

        dbHelper = new StudentDatabaseHelper(this);

        Button addButton = findViewById(R.id.addButton);
        Button updateButton = findViewById(R.id.updateButton);
        Button deleteButton = findViewById(R.id.deleteButton);
        Button retrieveButton = findViewById(R.id.retrieveButton);

        addButton.setOnClickListener(v -> showAddStudentDialog());
        updateButton.setOnClickListener(v -> showUpdateStudentDialog());
        deleteButton.setOnClickListener(v -> showDeleteStudentDialog());
        retrieveButton.setOnClickListener(v -> showAllStudents());

        // Button scale animation
        setButtonTouchAnimation(addButton);
        setButtonTouchAnimation(updateButton);
        setButtonTouchAnimation(deleteButton);
        setButtonTouchAnimation(retrieveButton);
    }

    private void setButtonTouchAnimation(Button addButton) {
        // Use the 'addButton' that is passed as a parameter, not a null View
        addButton.setOnTouchListener((v, event) -> {
            switch (event.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    v.animate().scaleX(0.9f).scaleY(0.9f).setDuration(100).start();
                    break;
                case MotionEvent.ACTION_UP:
                case MotionEvent.ACTION_CANCEL:
                    v.animate().scaleX(1f).scaleY(1f).setDuration(100).start();
                    break;
            }
            return false;
        });
    }


    private void showAddStudentDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        View view = getLayoutInflater().inflate(R.layout.dialog_add_student, null);
        EditText nameInput = view.findViewById(R.id.nameInput);
        EditText emailInput = view.findViewById(R.id.emailInput);
        EditText phoneInput = view.findViewById(R.id.phoneInput);
        EditText addressInput = view.findViewById(R.id.addressInput);
        Button submitButton = view.findViewById(R.id.submitButton);

        builder.setView(view);
        AlertDialog dialog = builder.create();

        submitButton.setOnClickListener(v -> {
            String name = nameInput.getText().toString();
            String email = emailInput.getText().toString();
            String phone = phoneInput.getText().toString();
            String address = addressInput.getText().toString();
```

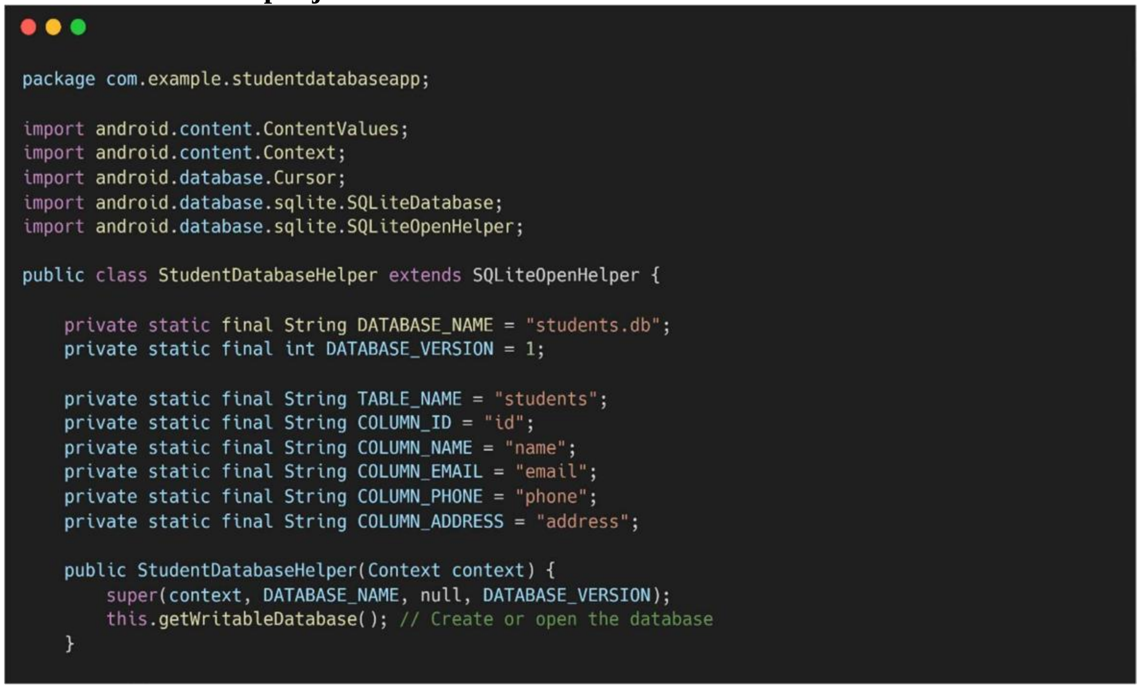


```
            if (!name.isEmpty() && !email.isEmpty() && !phone.isEmpty() && !address.isEmpty()) {
                if (dbHelper.addStudent(name, email, phone, address)) {
                    Toast.makeText(this, "Student Added", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                } else {
                    Toast.makeText(this, "Failed to Add Student", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Please fill in all fields", Toast.LENGTH_SHORT).show();
            }
        });

        dialog.show();
    }

    private void showUpdateStudentDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        View view = getLayoutInflater().inflate(R.layout.dialog_update_student, null);
        EditText idInput = view.findViewById(R.id.idInput);
        Button fetchButton = view.findViewById(R.id.fetchButton);
        EditText nameInput = view.findViewById(R.id.nameInput);
        EditText emailInput = view.findViewById(R.id.emailInput);
        EditText phoneInput = view.findViewById(R.id.phoneInput);
        EditText addressInput = view.findViewById(R.id.addressInput);
        Button updateButton = view.findViewById(R.id.updateButton);

        builder.setView(view);
        AlertDialog dialog = builder.create();

        fetchButton.setOnClickListener(v -> {
            String idStr = idInput.getText().toString();
            if (!idStr.isEmpty()) {
                int id = Integer.parseInt(idStr);
                Cursor cursor = dbHelper.getStudent(id);
                if (cursor.moveToFirst()) {
                    nameInput.setText(cursor.getString(1));
                    emailInput.setText(cursor.getString(2));
                    phoneInput.setText(cursor.getString(3));
                    addressInput.setText(cursor.getString(4));
                } else {
                    Toast.makeText(this, "Student not found", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Enter a valid ID", Toast.LENGTH_SHORT).show();
            }
        });

        updateButton.setOnClickListener(v -> {
            String idStr = idInput.getText().toString();
            String name = nameInput.getText().toString();
            String email = emailInput.getText().toString();
            String phone = phoneInput.getText().toString();
            String address = addressInput.getText().toString();

            if (!idStr.isEmpty() && !name.isEmpty() && !email.isEmpty() && !phone.isEmpty() &&
!address.isEmpty()) {
                int id = Integer.parseInt(idStr);
                if (dbHelper.updateStudent(id, name, email, phone, address)) {
                    Toast.makeText(this, "Student Updated", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                } else {
                    Toast.makeText(this, "Failed to Update", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "Fill in all fields", Toast.LENGTH_SHORT).show();
            }
        });

        dialog.show();
    }
```

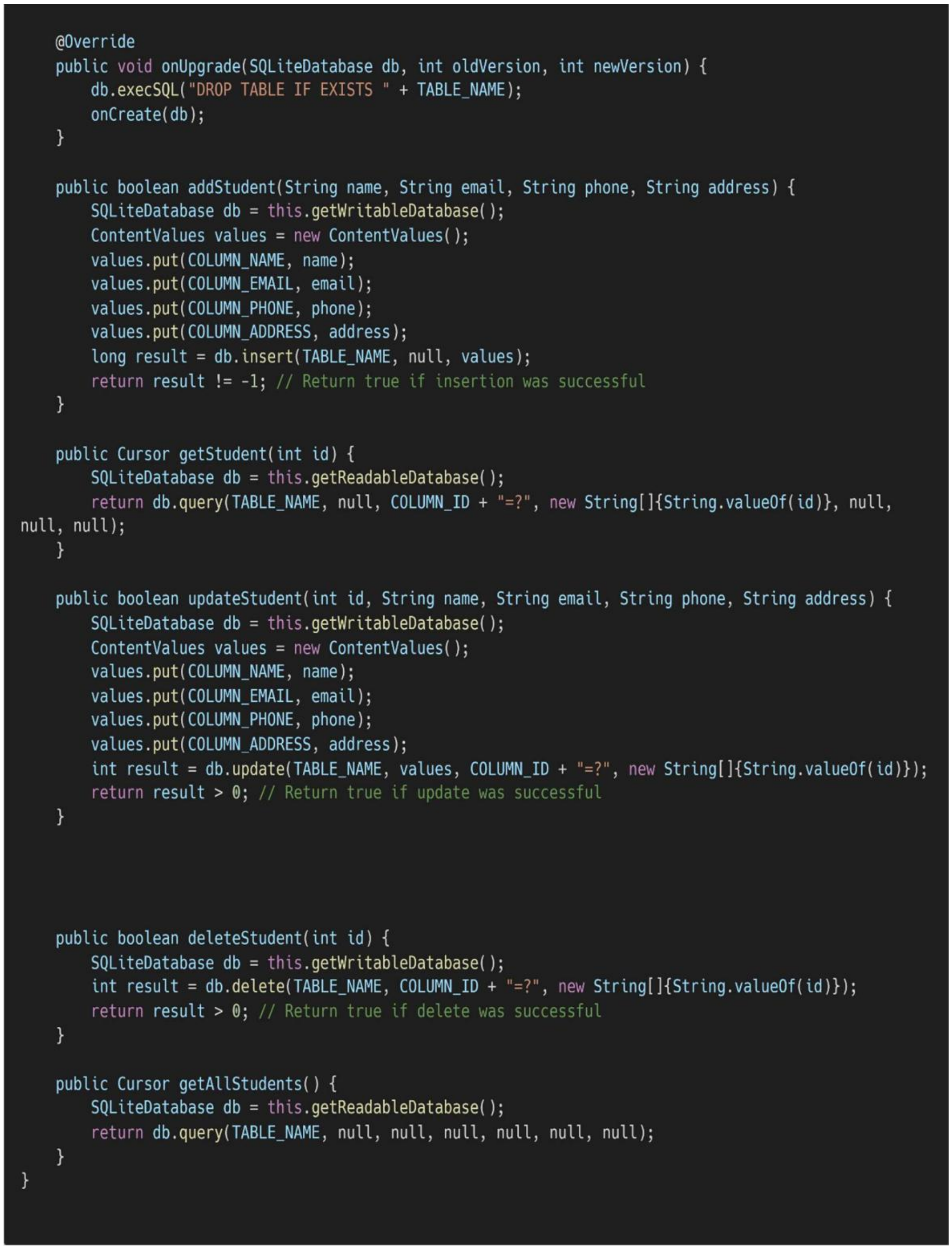


```
    private void showAllStudents() {
        Cursor cursor = dbHelper.getAllStudents();

        if (cursor.getCount() == 0) {
            Toast.makeText(this, "No students found", Toast.LENGTH_SHORT).show();
            return;
        }

        StringBuilder stringBuilder = new StringBuilder();
        while (cursor.moveToNext()) {
            stringBuilder.append("ID: ").append(cursor.getInt(0))
                    .append(", Name: ").append(cursor.getString(1))
                    .append(", Email: ").append(cursor.getString(2))
                    .append(", Phone: ").append(cursor.getString(3))
                    .append(", Address: ").append(cursor.getString(4))
                    .append("\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Student List");
        builder.setMessage(stringBuilder.toString());
        builder.setPositiveButton("OK", null);
        builder.show();
    }
}
```

StudentDatabaseHelper.java

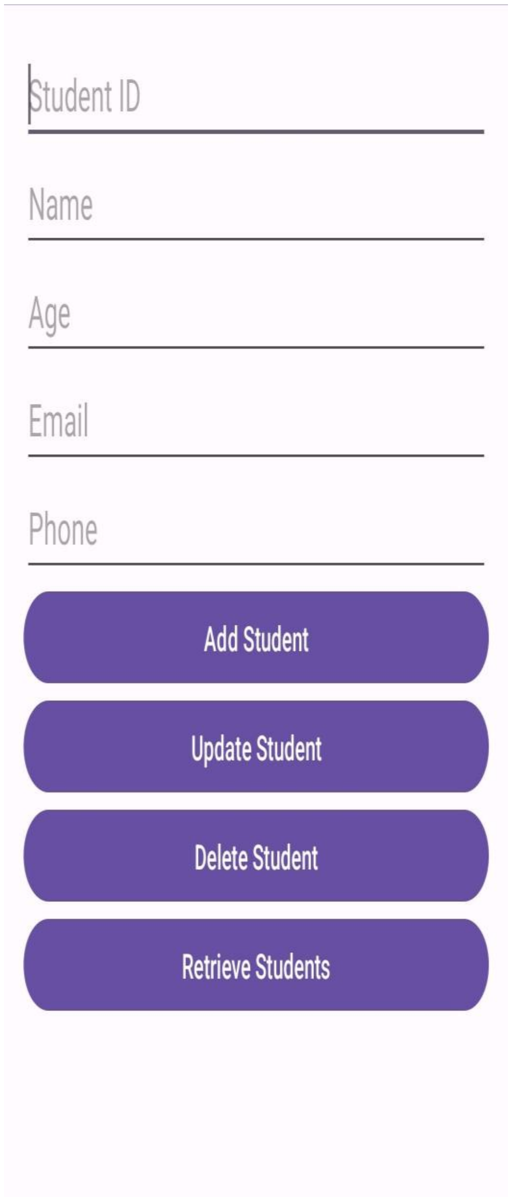


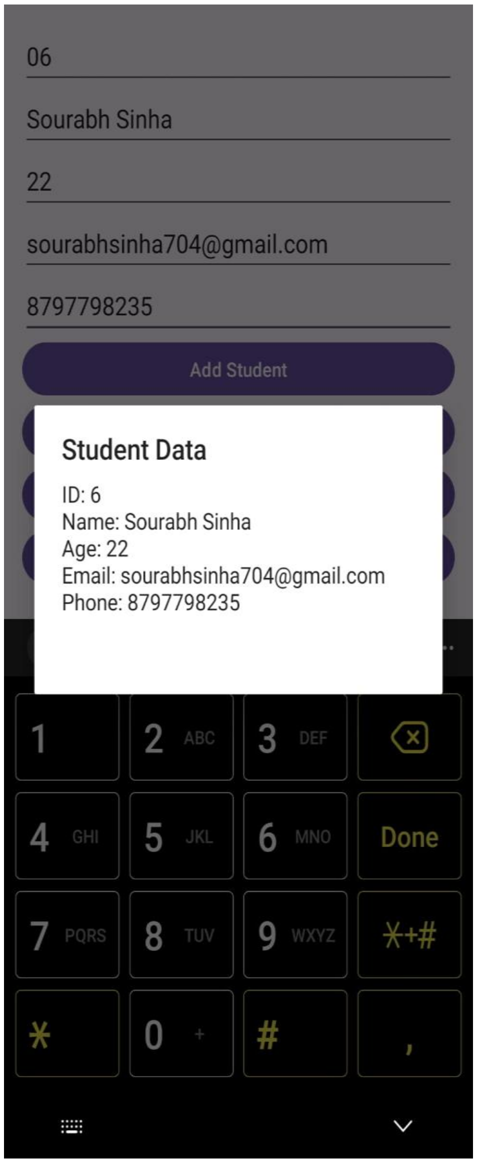
```
package com.example.studentdatabaseapp;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class StudentDatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "students.db";
    private static final int DATABASE_VERSION = 1;

    private static final String TABLE_NAME = "students";
    private static final String COLUMN_ID = "id";
    private static final String COLUMN_NAME = "name";
    private static final String COLUMN_EMAIL = "email";
    private static final String COLUMN_PHONE = "phone";
    private static final String COLUMN_ADDRESS = "address";

    public StudentDatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        this.getWritableDatabase(); // Create or open the database
    }
```



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

    public boolean addStudent(String name, String email, String phone, String address) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_NAME, name);
        values.put(COLUMN_EMAIL, email);
        values.put(COLUMN_PHONE, phone);
        values.put(COLUMN_ADDRESS, address);
        long result = db.insert(TABLE_NAME, null, values);
        return result != -1; // Return true if insertion was successful
    }

    public Cursor getStudent(int id) {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.query(TABLE_NAME, null, COLUMN_ID + "=?", new String[]{String.valueOf(id)}, null,
null, null);
    }

    public boolean updateStudent(int id, String name, String email, String phone, String address) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
        values.put(COLUMN_NAME, name);
        values.put(COLUMN_EMAIL, email);
        values.put(COLUMN_PHONE, phone);
        values.put(COLUMN_ADDRESS, address);
        int result = db.update(TABLE_NAME, values, COLUMN_ID + "=?", new String[]{String.valueOf(id)});
        return result > 0; // Return true if update was successful
    }



    public boolean deleteStudent(int id) {
        SQLiteDatabase db = this.getWritableDatabase();
        int result = db.delete(TABLE_NAME, COLUMN_ID + "=?", new String[]{String.valueOf(id)});
        return result > 0; // Return true if delete was successful
    }

    public Cursor getAllStudents() {
        SQLiteDatabase db = this.getReadableDatabase();
        return db.query(TABLE_NAME, null, null, null, null, null, null);
    }
}
```

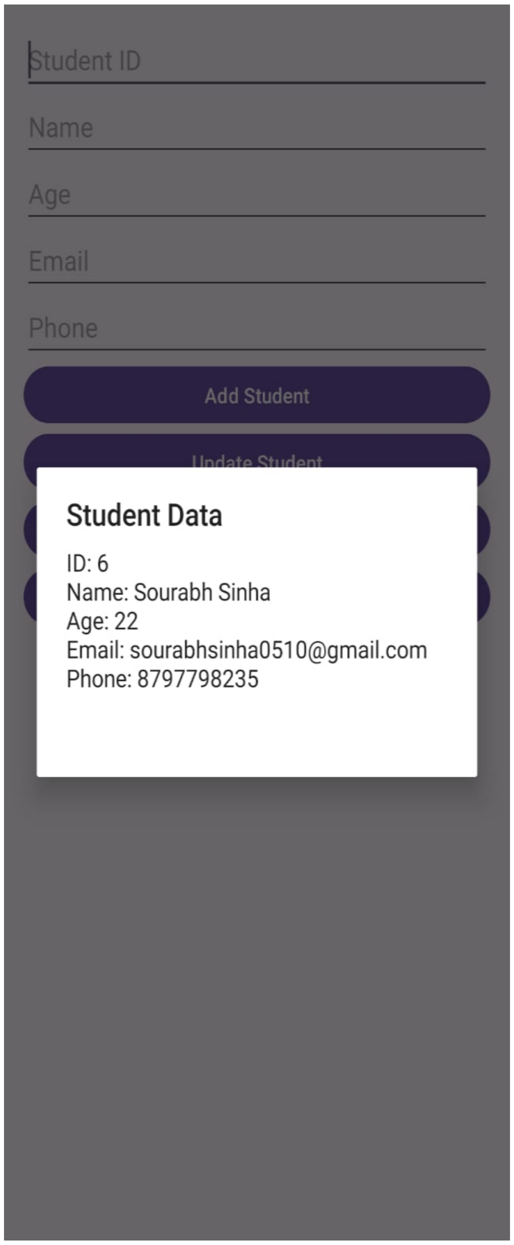
7. Screenshot





Student Data

ID: 6
Name: Sourabh Sinha
Age: 22
Email: sourabhsinha704@gmail.com
Phone: 8797798235


Student ID

Name

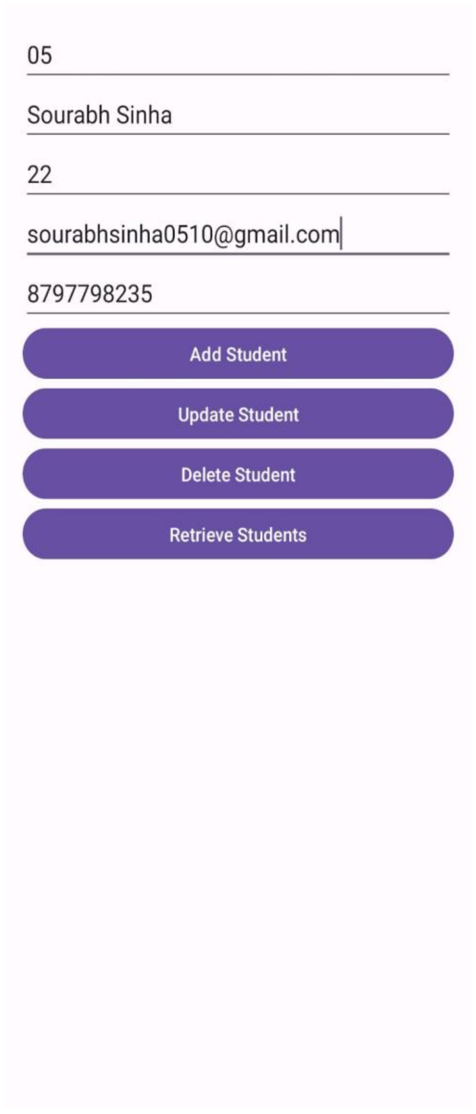
Age

Email

Phone

Student Data

ID: 6
Name: Sourabh Sinha
Age: 22
Email: sourabhsinha0510@gmail.com
Phone: 8797798235


05

Sourabh Sinha

22

sourabhsinha0510@gmail.com

8797798235

8. Setup Instructions

Prerequisites

- Android Studio installed on your development machine.
- Basic understanding of Java and Android development.

Installation

Clone this repository:
https://github.com/sourabhsinha0510/android_studentdatabase

Open the project in Android Studio:

- Go to Android Studio and select "Open an Existing Project."
- Navigate to the folder where you cloned the project.

Build and run:

Click on the "Run" button or press Shift + F10 to build and run the app on an emulator or connected device.

Usage

- Launch the app.
- Navigate to the Add Student page to enter a new student's details.
- Use the Update and Delete options to modify or remove student information.
- Go to the Retrieve Student section to view stored records.

Future Enhancements

- Search Functionality: Implement a search feature to quickly find student records.
- Export Data: Add functionality to export student data as CSV or PDF.
- Cloud Integration: Store student records in the cloud for real-time access across multiple devices.

Contact Information

For any inquiries, feel free to reach out:

 Email: sourabhsinha0510@gmail.com
 LinkedIn:/sourabh-sinha
 GitHub: sourabhsinha0510

Feel free to modify this template and adapt it to your specific app details! Let me know if you'd like help with any particular part.