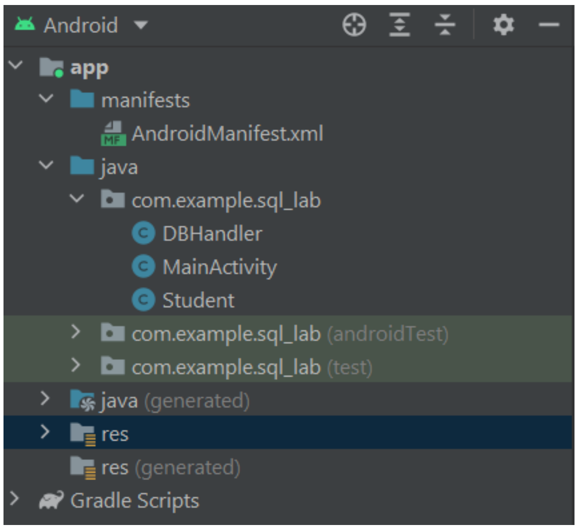
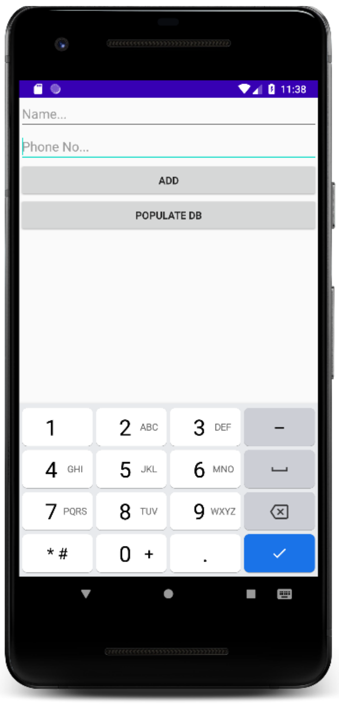
CSE2MAD Lab 6 – Part B SQLite

Aim: To give you some experience with persistent data storage on mobile devices. The aim of this app is to store students in the database.

Keep in mind, your project structure will be in the form,



1) Create the main activity with the UI below.



2) Let’s Model a Student by creating an Object

```java
package com.example.sql_lab;

/**
 * Created by smann
 */
public class Student {
    int _id;
    String _name;
    String _phoneNumber;

    public Student() {}

    public Student(int id, String name, String phoneNum) {
        this._id = id; this._name = name;
        this._phoneNumber = phoneNum;

    }
    public Student(String name, String phoneNumber) {
        this._name = name; this._phoneNumber =
                phoneNumber;
    }
    public int getID() {
        return this._id;
    }
    public void setID(int id) {
        this._id = id;
    }
    public String getName() {
        return this._name;
    }
    public void setName(String name) {
        this._name = name;
    }
    public String getPhone() {
        return this._phoneNumber;
    }
    public void setPhone(String phone) {
        this._phoneNumber = phone;
    }



}
```

3) Create the DatabaseHandler. We extend SQLiteOpenHelper and provide implementations of the callback methods.

```java
package com.example.sql_lab;

import java.util.ArrayList;
import java.util.List;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by smann
 */
public class DBHandler extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_NAME = "studentsManager";
    private static final String TABLE_STUDENTS = "students";
    private static final String KEY_ID = "id"; private
    static final String KEY_NAME = "name"; private static
    final String KEY_PH_NO = "phone_number";

    public DBHandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    // Creating Table
    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_STUDENTS_TABLE = "CREATE TABLE " + TABLE_STUDENTS + "("
                + KEY_ID + " INTEGER PRIMARY KEY," + KEY_NAME + " TEXT,"
                + KEY_PH_NO + " TEXT" + ")";
        db.execSQL(CREATE_STUDENTS_TABLE);
    }


    // Upgrading database
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if existed
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_STUDENTS);

        // Create table again
        onCreate(db);
    }

    // Adding new student
    void addStudent(Student student) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_NAME, student.getName()); // Contact Name
        values.put(KEY_PH_NO, student.getPhone()); // Contact Phone
        // Inserting Row
        db.insert(TABLE_STUDENTS, null, values);
//2nd argument is String containing nullColumnHack
        db.close(); // Closing database connection
    }

    // Getting single student
    Student getStudent(int id) {
        SQLiteDatabase db = this.getReadableDatabase();

        Cursor cursor = db.query(TABLE_STUDENTS, new String[] { KEY_ID,
                        KEY_NAME, KEY_PH_NO }, KEY_ID + "=?",
                new String[] { String.valueOf(id) }, null, null, null, null);
        if (cursor != null) cursor.moveToFirst();

        Student student = new Student(Integer.parseInt(cursor.getString(0)),
                cursor.getString(1), cursor.getString(2));
        // return student
        return student;
    }

    // Getting All Contacts
    public List<Student> getAllStudents() {
        List<Student> studentList = new ArrayList<Student>();
        // Select All Query
        String selectQuery = "SELECT * FROM " + TABLE_STUDENTS;

        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        // looping through all rows and adding to list
        if (cursor.moveToFirst()) { do {
            Student student = new Student();
            student.setID(Integer.parseInt(cursor.getString(0)));
            student.setName(cursor.getString(1));
            student.setPhone(cursor.getString(2));
            // Adding student to list
            studentList.add(student); }
        while (cursor.moveToNext());
        }

        // return student list
        return studentList;
    }

    // Updating single student
    public int updateStudent(Student student) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(KEY_NAME, student.getName()); values.put(KEY_PH_NO,
                student.getPhone());

        // updating row
        return db.update(TABLE_STUDENTS, values, KEY_ID + " = ?",
                new String[] { String.valueOf(student.getID()) });
    }

    // Deleting single student
    public void deleteStudent(Student student) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_STUDENTS, KEY_ID + " = ?",
                new String[] { String.valueOf(student.getID()) });
        db.close();
    }

    // Getting student Count
    public int getStudentsCount() {
        String countQuery = "SELECT * FROM " + TABLE_STUDENTS;
        SQLiteDatabase db = this.getReadableDatabase();
```

```java
        Cursor cursor = db.rawQuery(countQuery, null);
        cursor.close();

        // return count
        return cursor.getCount();
    } }
```

These should map to typical CRUD (create, read, update and delete) functionality.

4) In the java code for the main activity, we can now insert some students. Also reading them out to log for debug.

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

    DBHandler db = new DBHandler(this);

    // Inserting students
    Log.d("Insert: ", "Inserting ..");
    db.addStudent(new Student("Mat", "43540"));
    db.addStudent(new Student("Alex", "54334"));
    db.addStudent(new Student("Sameer", "34422"));
    db.addStudent(new Student("Shaz", "48465"));

    // Reading all students
    Log.d("Reading: ", "Reading all students..");
    List<Student> students = db.getAllStudents();

    for (Student cn : students) {
        String log = "Id: "+cn.getID()+" ,Name: " + cn.getName() + " ,Phone: "
                + cn.getPhone();
        // Writing Contacts to log
        Log.d("Name: ", log);
    }
}
```

Run the app, open the logcat in Android Studio and you will see the lines below if you have created the app correctly.

```
2020-09-06 13:06:21.235 13698-13698/com.example.sql_lab D/Read Name:: Id: 1 ,Name: Mat ,Phone: 43540

2020-09-06 13:06:21.235 13698-13698/com.example.sql_lab D/Read Name:: Id: 2 ,Name: Alex ,Phone: 54334

2020-09-06 13:06:21.235 13698-13698/com.example.sql_lab D/Read Name:: Id: 3 ,Name: Sameer ,Phone: 34422

2020-09-06 13:06:21.235 13698-13698/com.example.sql_lab D/Read Name:: Id: 4 ,Name: Shaz ,Phone: 48465
```

5) **Your goal** is to now implement the adding of students as above using the populate button and to also add individual students using the ADD button and the EditText fields.