Project: Simply TO DO

By Hardik Hajela/105070182

Group 4

Link to GitHub:

https://github.com/HardikHajela/SImply-TODO

Table of Contents

1. Acknowledgement:

I hereby acknowledge that the work submitted in this document and in the project is solely done by me with integrity.

H.H.

2. Introduction:

Simply To-Do is a comprehensive and easy app for to store Daily ‘To-Do’s’ and other chores. It consists of three activities mainly: The Home Page(activity_main.xml), AddTask(activity_add_tasks.xml) and ChangeTask(activity_change_task.xml) and consists of five java files essentially: MainActivity.java, CustomAdapter.java, AddTask.java, ChangeTask.java and SQL.java.

3. Objectives and Project Requirements:

The objective is to make the most out of Project Requirements in given time:

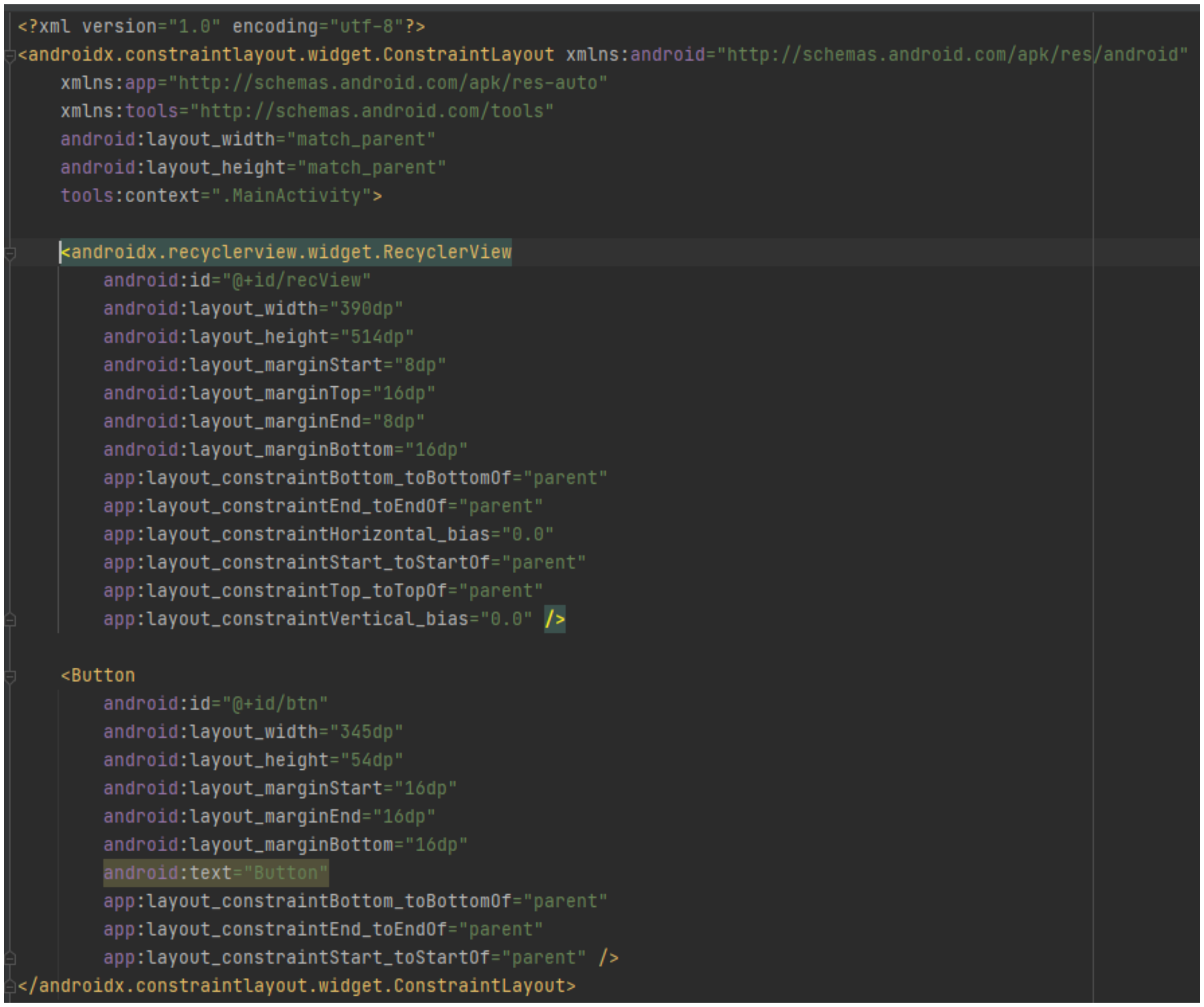
Project must have multiple activities. Project must have databases connectivity and related functionality. The project codes must be uploaded to Github, where each member has to push his/her contribution. The project should not be the same or similar to projects demonstrated during class/assignment.

4. Tools and Components

1. Android studio: official integrated development environment(ide) for Development of Applications for Android operating system
2. Java: High Level Object-Oriented Language used for Android Development
3. XML: Markup Language without pre-defined tags, used for representing structured information
4. SQLite: Database engine written in C, used for development of embedded softwares for cell phones and televisions.
5. Recycler View: It is a ViewGroup that contains the views corresponding to our respective data
6. Card View: An API used to show information inside card-like structures that have constraint look!
7. TextView: it is a user interface element that displays text to the user
8. PlainText: A text field used mostly for taking input from the users
9. Button: A user interface element that when tapped, can perform a desired reaction.
10. Linear Layout: A layout that arranges other views either in vertical signle row or horizontal single column.
11. Constraint Layout: A view group that enables the positioning and size of Widgets in a flexible way.
12. Manifest: A manifest file for any application describes essential information about the App build tools.
13. Date picker: It lets a user select a date of dates in a range!
14. Spinner: A view used to select one option off a list of several options.
15. Gradle: Provides a flexible way to compile, build and package Andriod apps and libraries!

5. Project Development and User manual

Activity_main.XML:



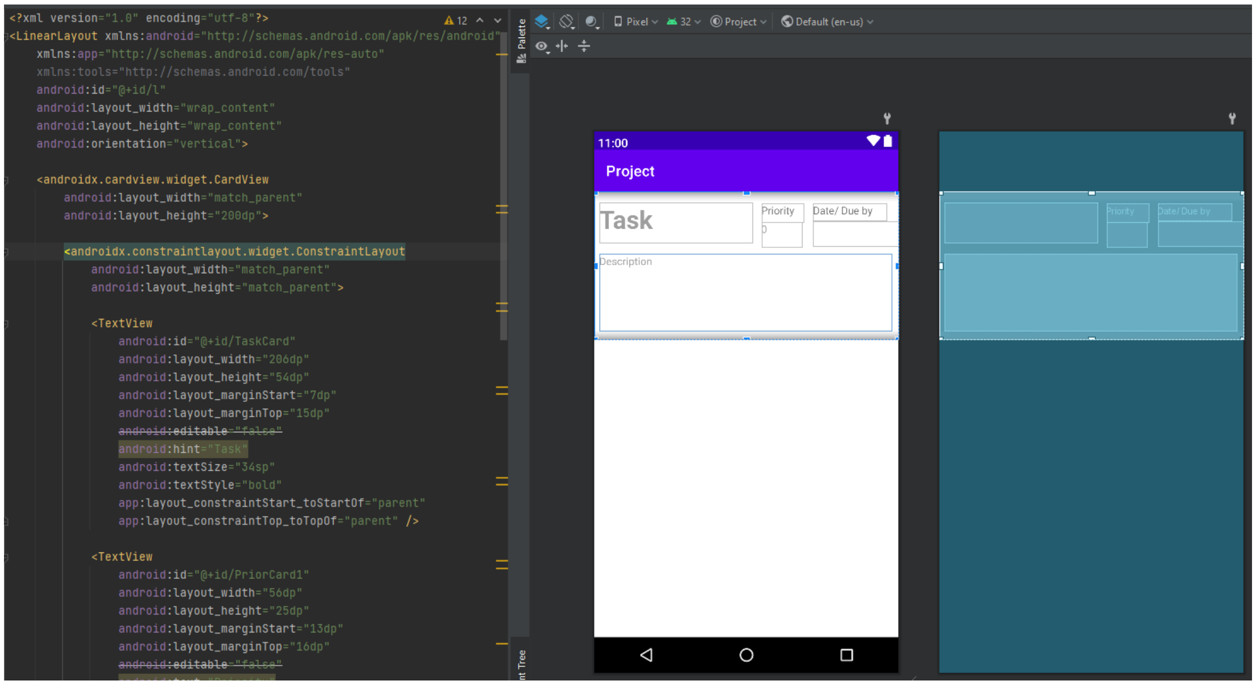
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recView"
        android:layout_width="390dp"
        android:layout_height="514dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

    <Button
        android:id="@+id/btn"
        android:layout_width="345dp"
        android:layout_height="54dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="16dp"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

This is the Home page of the Application where a User can see their Added tasks.

When they Click on a Particular Task, the App allows them to Update and Delete it. The Button towards the bottom help users add a new item to this list!

Made in Recycler View, the individual records are displayed using cardview from Card.xml.

Card.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/l"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <androidx.cardview.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="200dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/TaskCard"
                android:layout_width="206dp"
                android:layout_height="54dp"
                android:layout_marginStart="7dp"
                android:layout_marginTop="15dp"
                ~~android:editable="false"~~
                android:hint="Task"
                android:textSize="34sp"
                android:textStyle="bold"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/PriorCard1"
                android:layout_width="56dp"
                android:layout_height="25dp"
                android:layout_marginStart="13dp"
                android:layout_marginTop="16dp"
                ~~android:editable="false"~~
```

Card for Home Page. It consists of a total of 6 elements: Task name, Task Description, Task Priority and the day, task is due by!

This cardview is then implement in Main Activity using CustomAdapter.java !

MainActivity.java :

https://github.com/HardikHajela/SImply-TODO/blob/main/MainActivity.java

This is the Brains behind Activity_main.xml. We have five parameters in table “TODO”: id, task, description, dueby and priority! Arraylist of each and every Parameter is declared here.

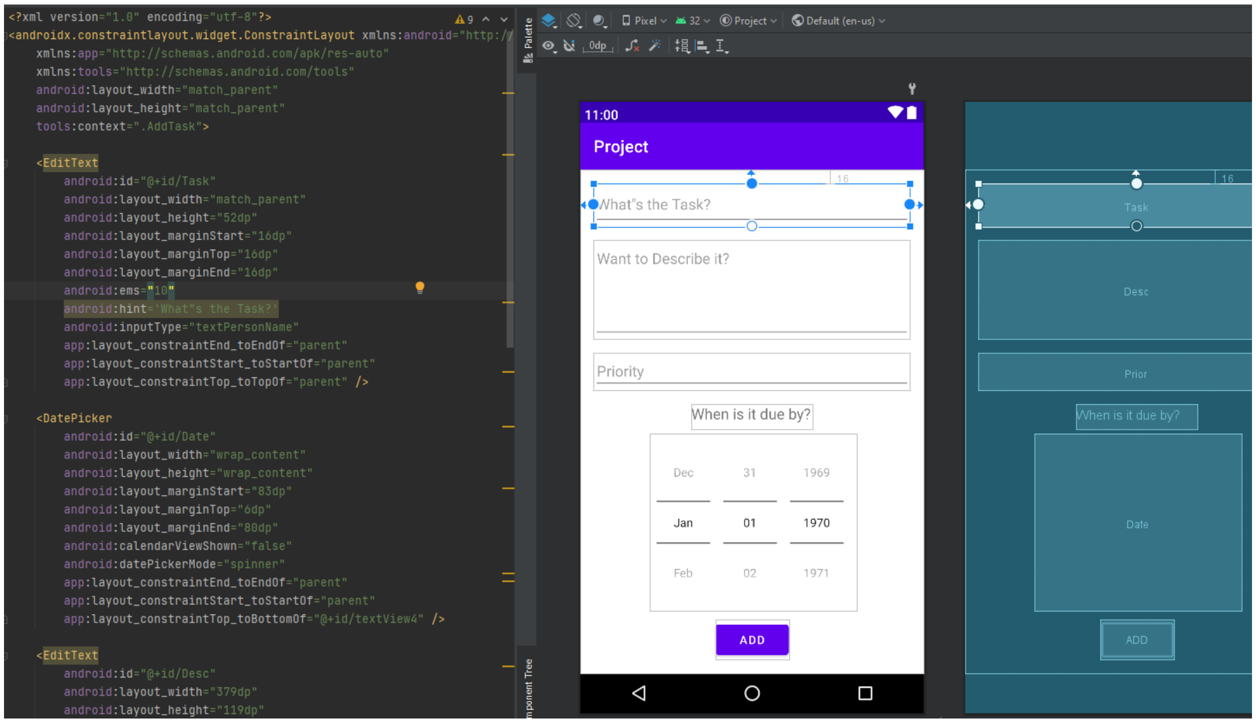
After declaring the ArrayList, we then store the Data in Arrays retrieved from the SQLite Database through SQL.java.

Custom Adapter:

https://github.com/HardikHajela/SImply-TODO/blob/main/CustomAdapter.java

The is the adapter class for the app. It’s a given to have an adapter if Recycler View is implemented in the main Activity. public void onBindViewHolder and class MyViewHolder were used to implement cardview from card.xml to RecyclerView in Main activity!

Activity_add_task.xml



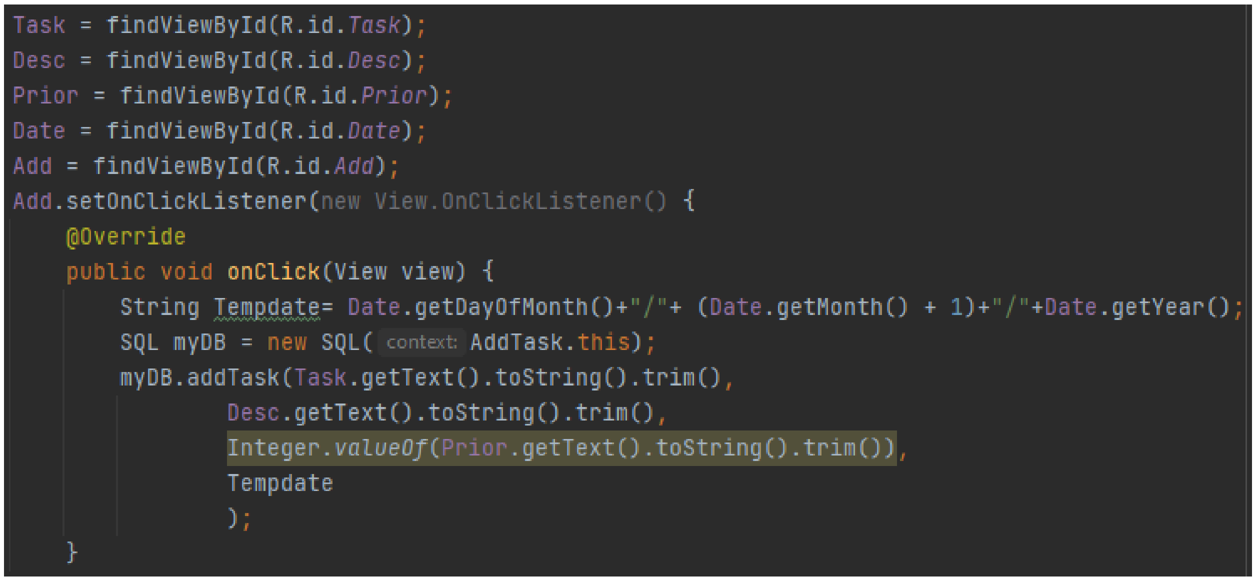
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AddTask">

    <EditText
        android:id="@+id/Task"
        android:layout_width="match_parent"
        android:layout_height="52dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:ems="10"
        android:hint='What"s the Task?'
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <DatePicker
        android:id="@+id/Date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="83dp"
        android:layout_marginTop="6dp"
        android:layout_marginEnd="80dp"
        android:calendarViewShown="false"
        android:datePickerMode="spinner"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />

    <EditText
        android:id="@+id/Desc"
        android:layout_width="379dp"
        android:layout_height="119dp"
```

When a user presses the button at the bottom of main Activity, they come to this page. It's in order to update new task in the List. After Filling all the fields above, when they press add, on the Back-end, a new record is created, which instantly reflects on the Home Screen, with other records!

AddTask.java

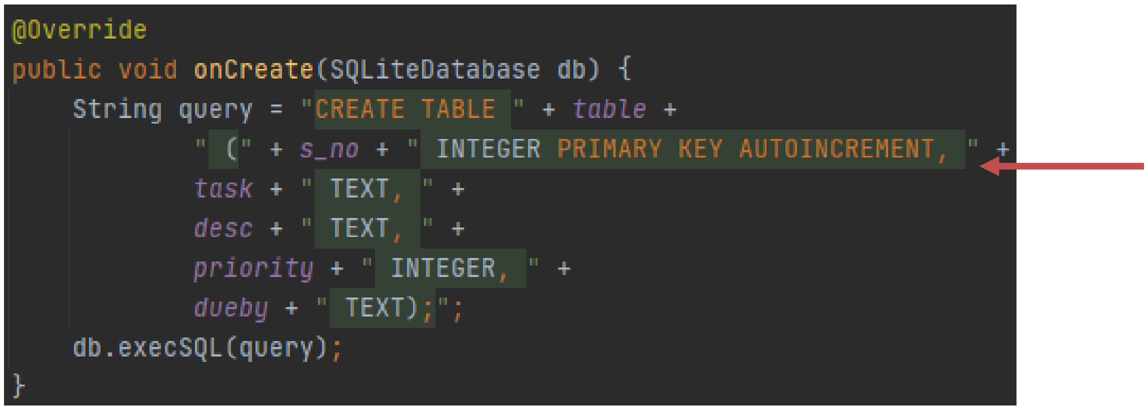
https://github.com/HardikHajela/SImply-TODO/blob/main/AddTask.java

This the Brains of Activity_add_task.xml. When user Lands on this page and fills out the information, They are stored in 4 different variables: Task(Name of the Task), Discription, priority as int and due by date using Date Picker with Spinner! Making a new new Instance of SQL.java in this file, all four variable are than added to table TODO using addTask Function. After that, one record is updated.



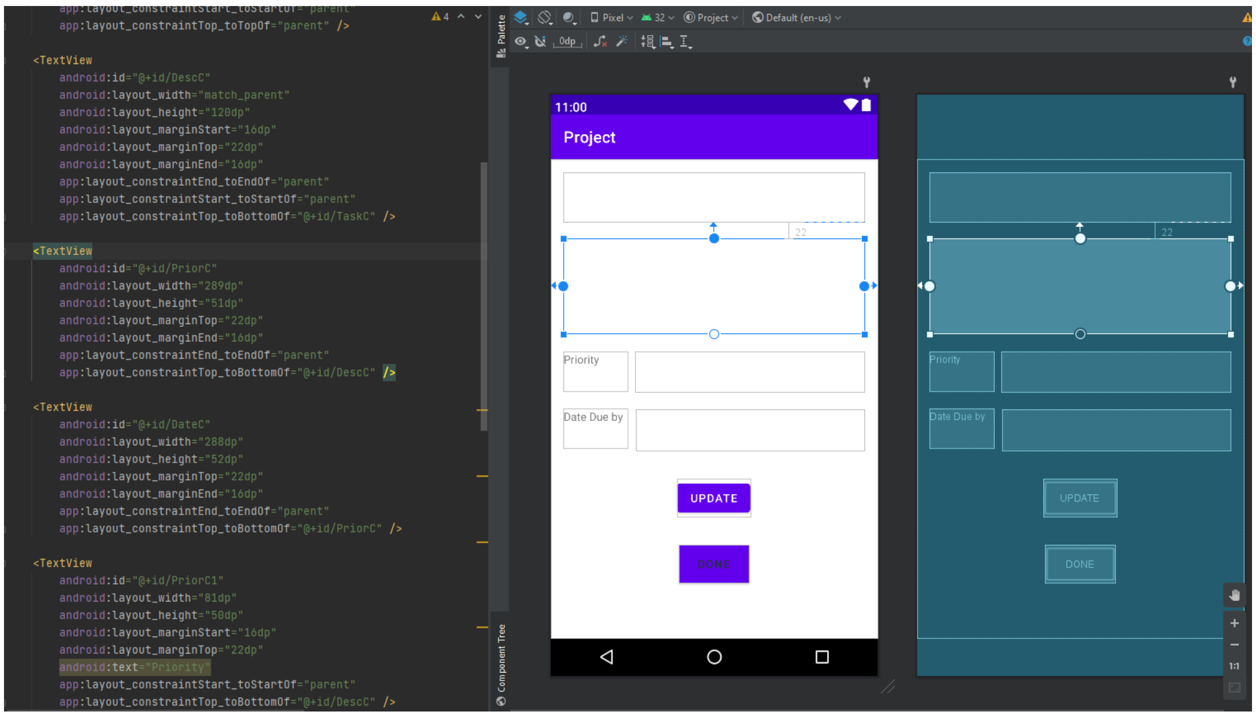
```
Task = findViewById(R.id.Task);
Desc = findViewById(R.id.Desc);
Prior = findViewById(R.id.Prior);
Date = findViewById(R.id.Date);
Add = findViewById(R.id.Add);
Add.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String Tempdate= Date.getDayOfMonth()+"/"+ (Date.getMonth() + 1)+"/"+Date.getYear();
        SQL myDB = new SQL( context: AddTask.this);
        myDB.addTask(Task.getText().toString().trim(),
                Desc.getText().toString().trim(),
                Integer.valueOf(Prior.getText().toString().trim()),
                Tempdate
                );
    }
```

Note: There’s 5 columns in the table TODO, but only 4 are stored and changed as the first column(S. No) is auto-incrementing per record!



```
@Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + table +
            " (" + s_no + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            task + " TEXT, " +
            desc + " TEXT, " +
            priority + " INTEGER, " +
            dueby + " TEXT);";
    db.execSQL(query);
}
```

Activity_chane_task.xml



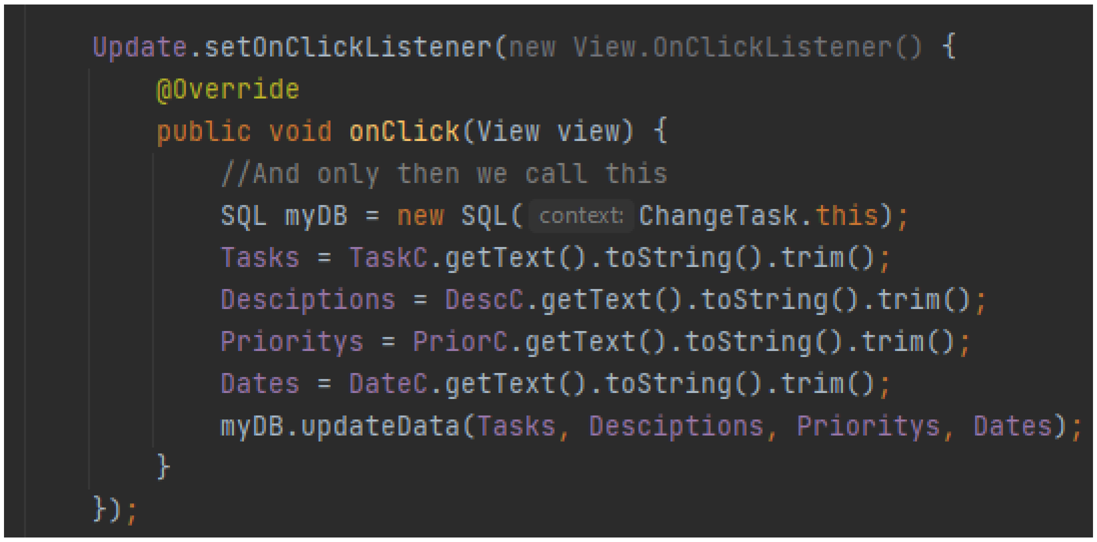
```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/DescC"
        android:layout_width="match_parent"
        android:layout_height="120dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="22dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/TaskC" />

    <TextView
        android:id="@+id/PriorC"
        android:layout_width="289dp"
        android:layout_height="51dp"
        android:layout_marginTop="22dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/DescC" />

    <TextView
        android:id="@+id/DateC"
        android:layout_width="288dp"
        android:layout_height="52dp"
        android:layout_marginTop="22dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/PriorC" />

    <TextView
        android:id="@+id/PriorC1"
        android:layout_width="81dp"
        android:layout_height="50dp"
        android:layout_marginStart="16dp"
        android:layout_marginTop="22dp"
        android:text="Priority"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/DescC" />
```

This is the most import part and motive for the To-Do list: A ‘Done Task!’ If a user taps on any one of the records, they land here, where they can see a full description of their Task, if need be, Information can easily be updated with a click of a button, and the task is Done, they can press the ‘Done’ button in the bottom to remove the record!

ChangeTask.java

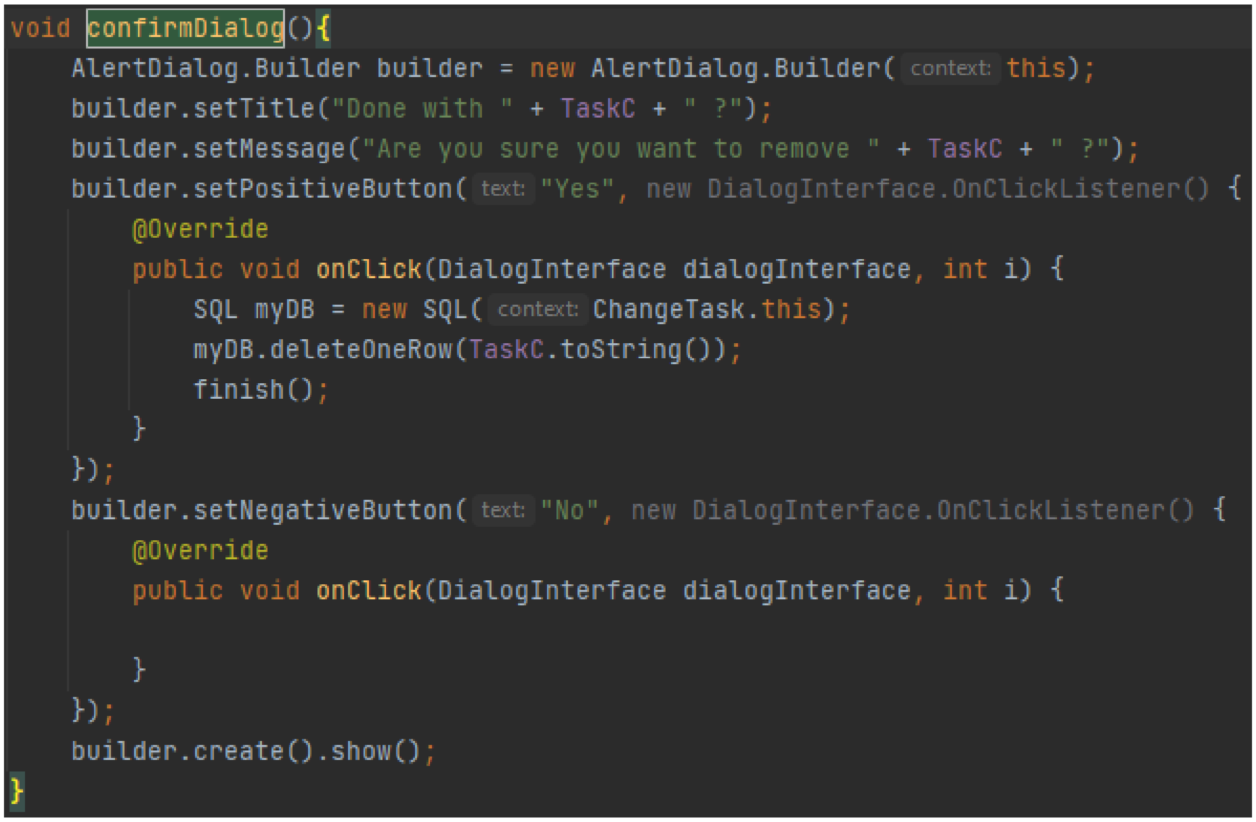
https://github.com/HardikHajela/SImply-TODO/blob/main/ChangeTask.java

This is the brains behind Activity_change_task.xml. Here, we again create an instance of SQL.java, we Update and Delete the record the user can see in Activity_change_task.xml, after simply tapping on it! Once they are on Activity_change_task.xml, a user can update a record by typing the very updates in designated fields, followed by pressing the Update Key!

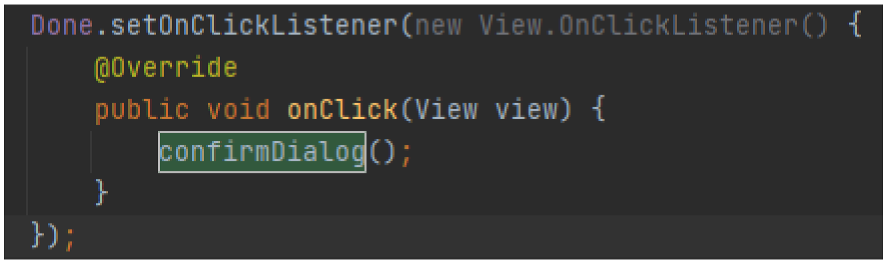


```
Update.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //And only then we call this
        SQL myDB = new SQL( context: ChangeTask.this);
        Tasks = TaskC.getText().toString().trim();
        Desciptions = DescC.getText().toString().trim();
        Prioritys = PriorC.getText().toString().trim();
        Dates = DateC.getText().toString().trim();
        myDB.updateData(Tasks, Desciptions, Prioritys, Dates);
    }
});
```

When they have done/completed/accomplished a task, they can open the task record from Home Page and land on Activity_change_task.xml, after which they just have to press the Done key and the record will be deleted!



```
void confirmDialog(){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Done with " + TaskC + " ?");
    builder.setMessage("Are you sure you want to remove " + TaskC + " ?");
    builder.setPositiveButton( text: "Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            SQL myDB = new SQL( context: ChangeTask.this);
            myDB.deleteOneRow(TaskC.toString());
            finish();
        }
    });
    builder.setNegativeButton( text: "No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {

        }
    });
    builder.create().show();
}
```



```
Done.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        confirmDialog();
    }
});
```

One's the user taps the Done Button, the App sends an alert dialog confirming if they want to remove the record. If the press Yes, the record is permanently deleted!

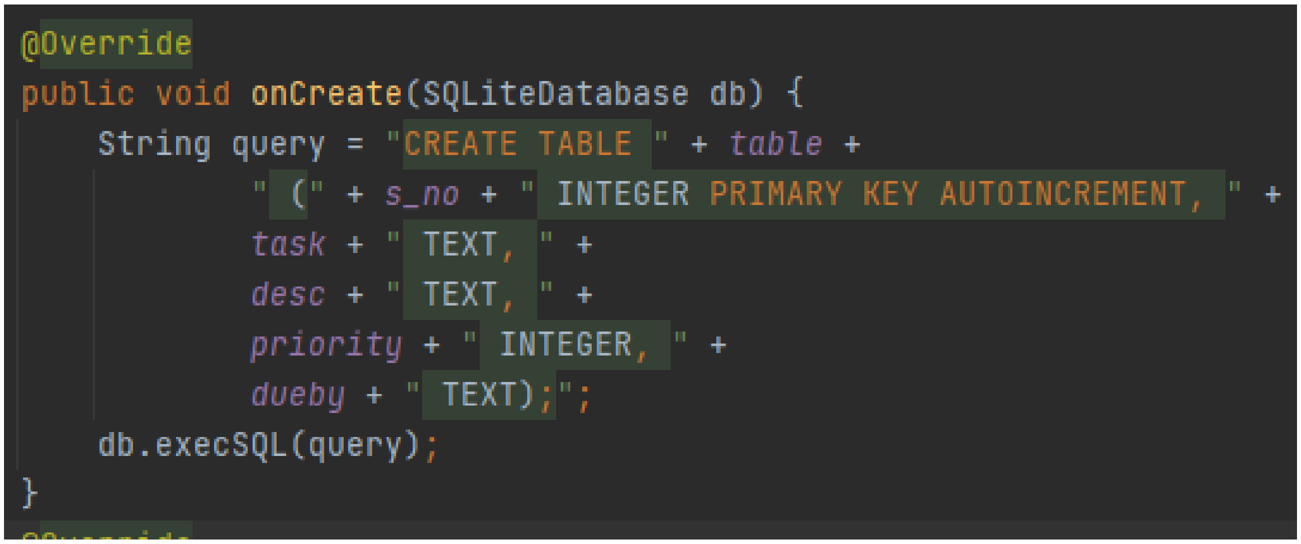
SQL.java :

https://github.com/HardikHajela/SImply-TODO/blob/main/SQL.java

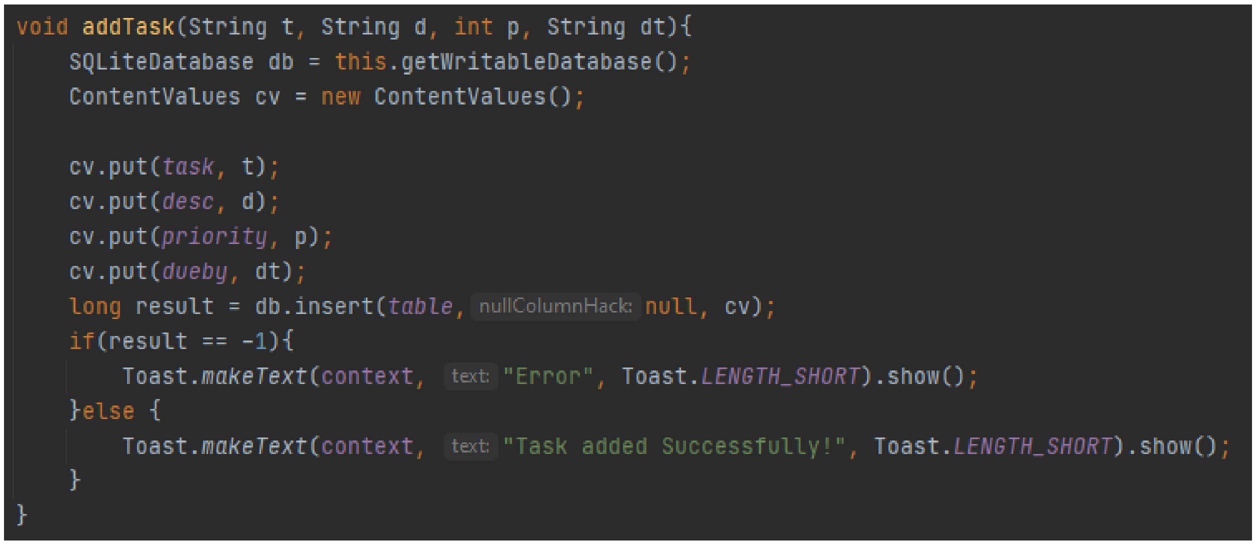
The CRUD functions: creating, reading, updating and deleting of the TODO table is written is this class and then used by other classes for performing all the CRUD Functions.

Main Activity using Read, Add Task using create and Change Task using both Update and Delete!

Create:

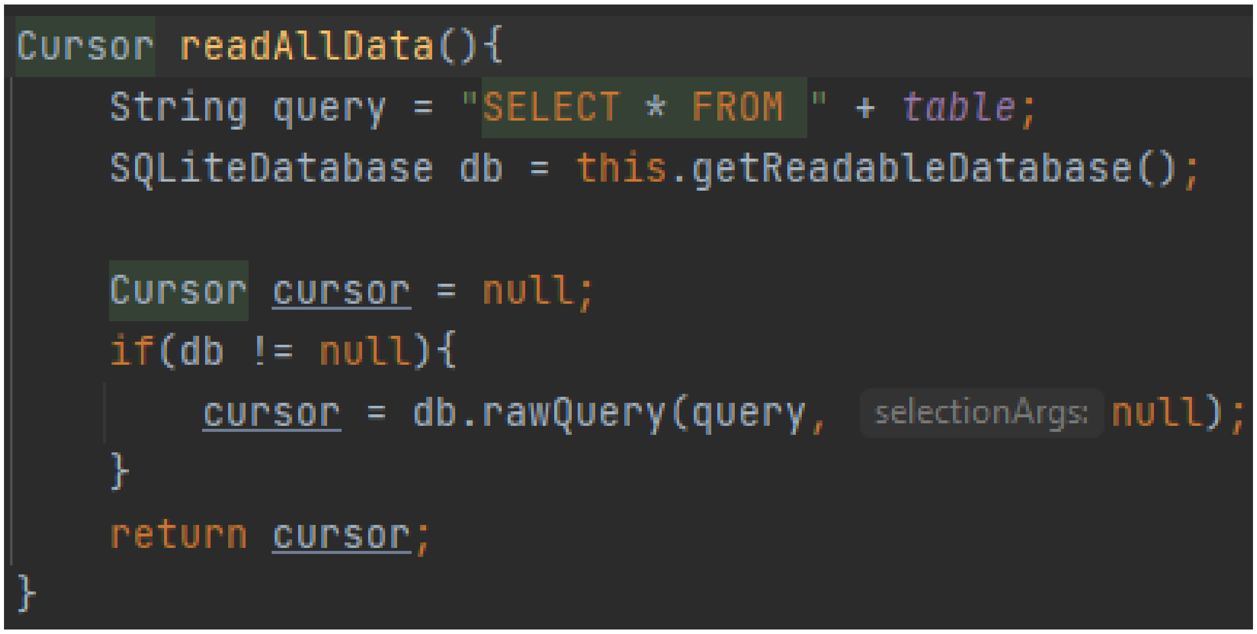


```
@Override
public void onCreate(SQLiteDatabase db) {
    String query = "CREATE TABLE " + table +
            " (" + s_no + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
            task + " TEXT, " +
            desc + " TEXT, " +
            priority + " INTEGER, " +
            dueby + " TEXT);";
    db.execSQL(query);
}
```



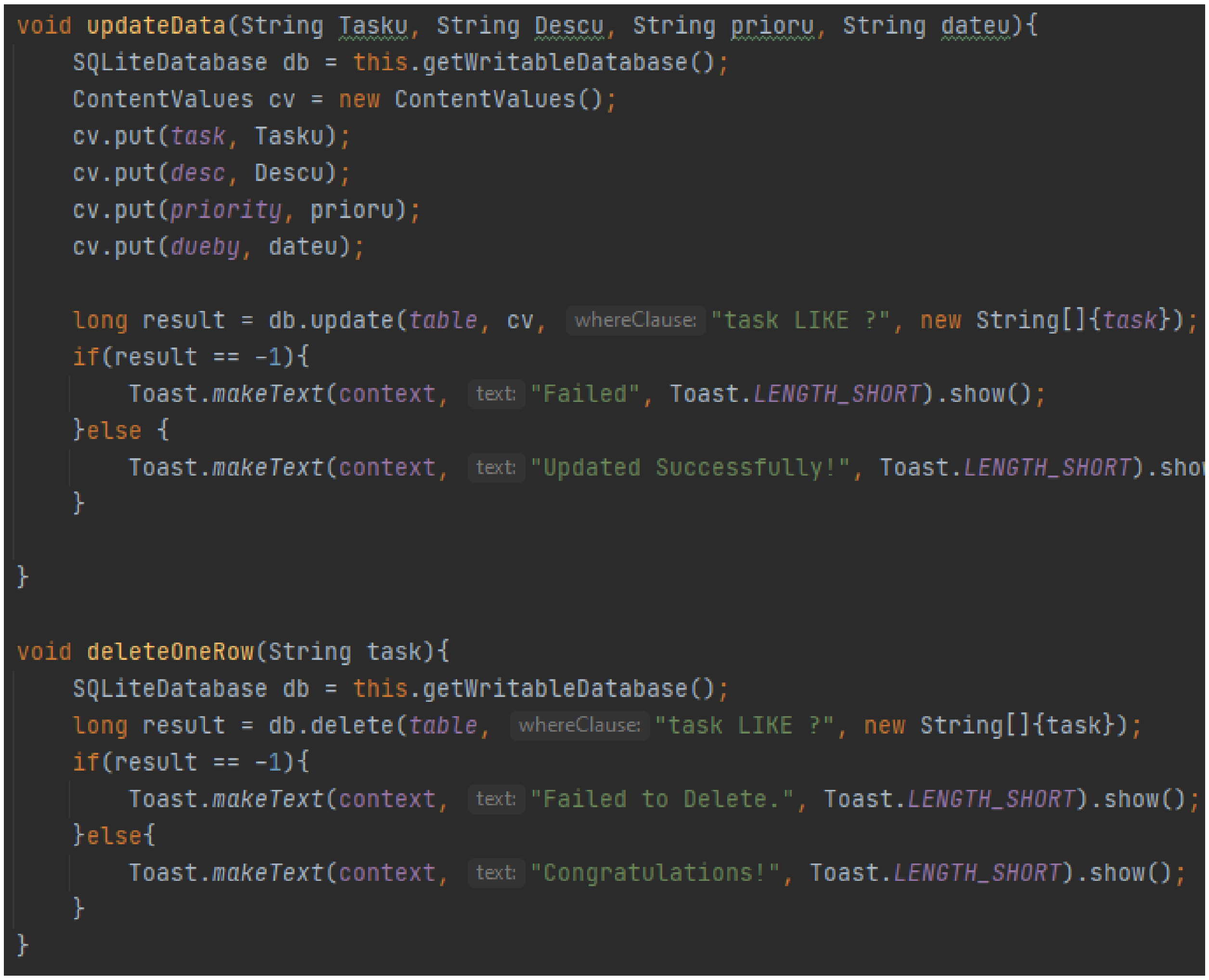
```java
void addTask(String t, String d, int p, String dt){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();

    cv.put(task, t);
    cv.put(desc, d);
    cv.put(priority, p);
    cv.put(dueby, dt);
    long result = db.insert(table, nullColumnHack: null, cv);
    if(result == -1){
        Toast.makeText(context, text: "Error", Toast.LENGTH_SHORT).show();
    }else {
        Toast.makeText(context, text: "Task added Successfully!", Toast.LENGTH_SHORT).show();
    }
}
```

Read:



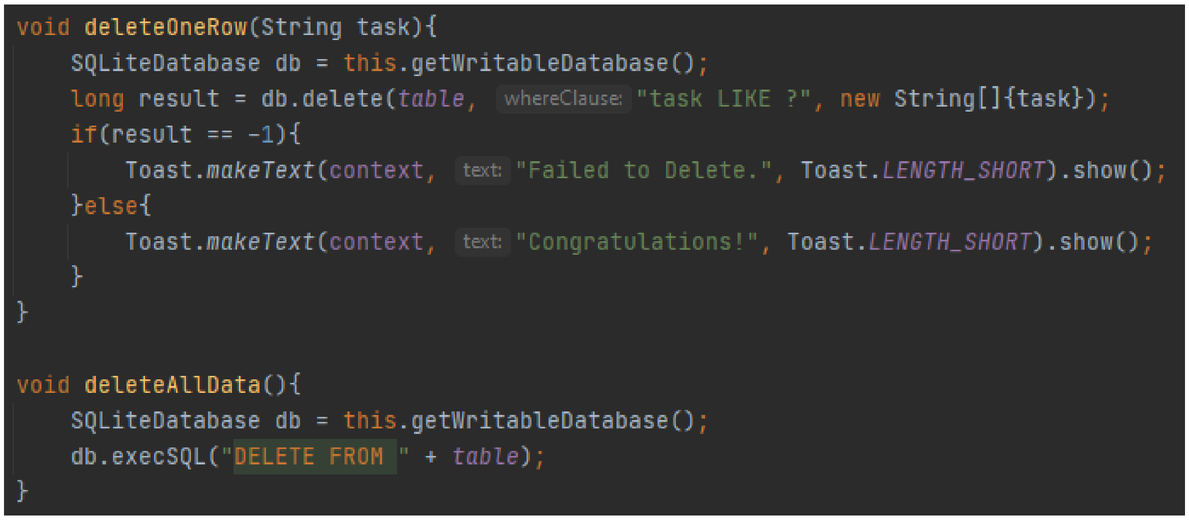
```java
Cursor readAllData(){
    String query = "SELECT * FROM " + table;
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = null;
    if(db != null){
        cursor = db.rawQuery(query, selectionArgs: null);
    }
    return cursor;
}
```

Update:



```
void updateData(String Tasku, String Descu, String prioru, String dateu){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put(task, Tasku);
    cv.put(desc, Descu);
    cv.put(priority, prioru);
    cv.put(dueby, dateu);

    long result = db.update(table, cv, whereClause: "task LIKE ?", new String[]{task});
    if(result == -1){
        Toast.makeText(context, text: "Failed", Toast.LENGTH_SHORT).show();
    }else {
        Toast.makeText(context, text: "Updated Successfully!", Toast.LENGTH_SHORT).sho
    }

}

void deleteOneRow(String task){
    SQLiteDatabase db = this.getWritableDatabase();
    long result = db.delete(table, whereClause: "task LIKE ?", new String[]{task});
    if(result == -1){
        Toast.makeText(context, text: "Failed to Delete.", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText(context, text: "Congratulations!", Toast.LENGTH_SHORT).show();
    }
}
```

Delete:



```
void deleteOneRow(String task){
    SQLiteDatabase db = this.getWritableDatabase();
    long result = db.delete(table, whereClause: "task LIKE ?", new String[]{task});
    if(result == -1){
        Toast.makeText(context, text: "Failed to Delete.", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText(context, text: "Congratulations!", Toast.LENGTH_SHORT).show();
    }
}

void deleteAllData(){
    SQLiteDatabase db = this.getWritableDatabase();
    db.execSQL("DELETE FROM " + table);
}
```

6. Conclusion, Limitations and Future scopes.

   The Simply TO-DO App is a good start towards my future endeavors as an Android Developer. The biggest limitation was time for this project, and we had just the basics to works with, hence, there are infinite possibilities for its further interpretations of the application, considering we have much more to work upon it in the future, to make it a better App. with more features. If possible it'll be soon uploaded to play story!

Link to GitHub: https://github.com/HardikHajela/SImply-TODO