

Mobile_Lesson5: Working with SQLite and Firebase

ICP GROUP: 38

ICP 12

Name: Anil Kumar Reddy

Email: anggp@umsystem.edu

ICP12 Repository: <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-AnilkumarreddyNandikonda>

ICP12 source code link:

Firebase: <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-AnilkumarreddyNandikonda/tree/main/Source/Firebase>

SQLite: <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-AnilkumarreddyNandikonda/tree/main/Source/SQLite>

ICP12 Video: <https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4614827d-77f0-443b-b23b-ade005c632b>

My Partner

Partner name: Abhinay Yadav

Partner Email: ayr6y@umsystem.edu

Partner Repository: <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-YAbhinay>

Source Code Link

Firebase : <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-YAbhinay/tree/main/Source/Firebase/app/src>

SQLite : <https://github.com/UMKC-APL-WebMobileProgramming/ICP12-YAbhinay/tree/main/Source/SQLite/app/src>

ICP12 video: <https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4614827d-77f0-443b-b23b-ade005c632b>

Lesson Overview: In this lesson, we are going to discuss SQLite and Firebase databases.

In Class Programming (ICP):

SQLite:

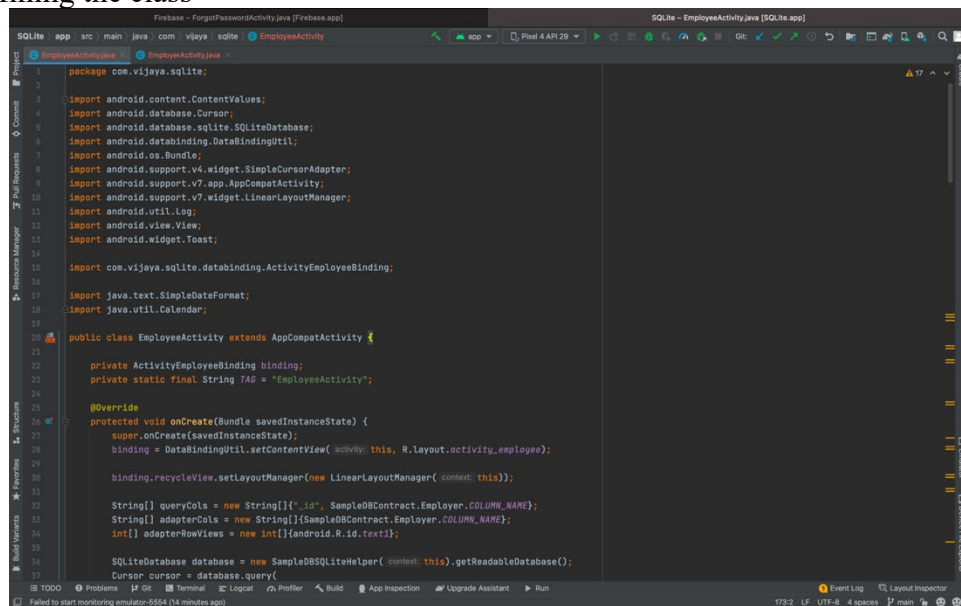
1. Open the use case SQLite provided in the source code and understand how the control flows from source code to UI
2. Add the following functionality to the app:
 - deleting employee or employer details
 - Updating employee details

Firebase:

1. Open the use case 'Firebase' provided in source code and understand how the control flows from source code to UI
2. Add the following functionalities to the app:
 - Log out
 - Delete feature

EmployeeActivity:

- Defining the class



```
1 package com.vijaya.sqlite;
2
3 import android.content.ContentValues;
4 import android.database.Cursor;
5 import android.database.sqlite.SQLiteDatabase;
6 import android.databinding.DataBindingUtil;
7 import android.os.Bundle;
8 import android.support.v4.widget.SimpleCursorAdapter;
9 import android.support.v7.app.AppCompatActivity;
10 import android.support.v7.widget.LinearLayoutManager;
11 import android.support.v7.widget.RecyclerView;
12 import android.util.Log;
13 import android.view.View;
14 import android.widget.Toast;
15
16 import com.vijaya.sqlite.databinding.ActivityEmployeeBinding;
17
18 import java.text.SimpleDateFormat;
19 import java.util.Calendar;
20
21 public class EmployeeActivity extends AppCompatActivity {
22     private ActivityEmployeeBinding binding;
23     private static final String TAG = "EmployeeActivity";
24
25     @Override
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         binding = DataBindingUtil.setContentView(this, R.layout.activity_employee);
29         binding.recyclerView.setLayoutManager(new LinearLayoutManager(this));
30
31         String[] queryCols = new String[]{"_id", SampleDBContract.Employer.COLUMN_NAME};
32         String[] adapterCols = new String[]{SampleDBContract.Employer.COLUMN_NAME};
33         int[] adapterRowViews = new int[]{android.R.id.text1};
34
35         SQLiteDatabase database = new SampleDBSQLiteHelper(this).getReadableDatabase();
36         Cursor cursor = database.query(
```

- Creating the buttons search, update, save

```

36 SQLiteDatabase database = new SampleDBSQLiteHelper(context: this).getReadableDatabase();
37 Cursor cursor = database.query(
38     SampleDBContract.Employee.TABLE_NAME, // The table to query
39     queryCols, // The columns to return
40     selection: null, // The columns for the WHERE clause
41     selectionArgs: null, // The values for the WHERE clause
42     groupBy: null, // don't group the rows
43     having: null, // don't filter by row groups
44     orderBy: null // don't sort
45 );
46
47 SimpleCursorAdapter cursorAdapter = new SimpleCursorAdapter(
48     context: this, android.R.layout.simple_spinner_item, cursor, adapterCols, adapterRowViews, flags: 0);
49 cursorAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
50 binding.employerSpinner.setAdapter(cursorAdapter);
51
52 binding.saveButton.setOnClickListener(new View.OnClickListener() {
53     @Override
54     public void onClick(View view) {
55         saveToDB();
56     }
57 });
58
59 binding.searchButton.setOnClickListener(new View.OnClickListener() {
60     @Override
61     public void onClick(View view) {
62         readFromDB();
63     }
64 });
65
66 binding.updateButton.setOnClickListener(new View.OnClickListener() {
67     @Override
68     public void onClick(View view) {
69         update();
70     }
71 });

```

- For each TODO function method i.e save, read, update, delete. Following screenshots as the code of TODO list for Employee

1)

```

72
73 binding.deleteButton.setOnClickListener(new View.OnClickListener() {
74     @Override
75     public void onClick(View view) {
76         delete();
77     }
78 });
79
80
81 private void saveToDB() {
82     SQLiteDatabase database = new SampleDBSQLiteHelper(context: this).getWritableDatabase();
83     ContentValues values = new ContentValues();
84     values.put(SampleDBContract.Employee.COLUMN_FIRSTNAME, binding.firstnameEditText.getText().toString());
85     values.put(SampleDBContract.Employee.COLUMN_LASTNAME, binding.lastnameEditText.getText().toString());
86     values.put(SampleDBContract.Employee.COLUMN_JOB_DESCRIPTION, binding.jobDescEditText.getText().toString());
87     values.put(SampleDBContract.Employee.COLUMN_EMPLOYER_ID,
88         ((Cursor) binding.employerSpinner.getSelectedItem()).getInt(0));
89
90     Log.d("getINT", msg: ((Cursor) binding.employerSpinner.getSelectedItem()).getInt(0) + "");
91     Log.d("getColumn", ((Cursor) binding.employerSpinner.getSelectedItem()).getColumnName(0));
92
93     try {
94         Calendar calendar = Calendar.getInstance();
95         calendar.setTime((new SimpleDateFormat(pattern: "dd/MM/yyyy")).parse(
96             binding.dobEditText.getText().toString()));
97         long date = calendar.getTimeInMillis();
98         values.put(SampleDBContract.Employee.COLUMN_DATE_OF_BIRTH, date);
99
100         calendar.setTime((new SimpleDateFormat(pattern: "dd/MM/yyyy")).parse(
101             binding.employedEditText.getText().toString()));
102         date = calendar.getTimeInMillis();
103         values.put(SampleDBContract.Employee.COLUMN_EMPLOYED_DATE, date);
104     } catch (Exception e) {
105         Log.e(TAG, msg: "Error", e);
106         Toast.makeText(context: this, text: "Date is in the wrong format", Toast.LENGTH_LONG).show();
107         return;
108     }
109 }

```

2)

```

108    }
109    long newRowId = database.insert(SampleDBContract.Employee.TABLE_NAME, nullColumnHack: null, values);
110
111    Toast.makeText( context: this, text: "The new Row Id is " + newRowId, Toast.LENGTH_LONG).show();
112  }
113
114  private void readFromDB() {
115    String firstname = binding.firstnameEditText.getText().toString();
116    String lastname = binding.lastnameEditText.getText().toString();
117
118    SQLiteDatabase database = new SampleDBSQLiteHelper( context: this).getReadableDatabase();
119
120    String[] selectionArgs = {"%" + firstname + "%", "%" + lastname + "%"};
121
122    Cursor cursor = database.rawQuery(SampleDBContract.SELECT_EMPLOYEE_WITH_EMPLOYER, selectionArgs);
123    binding.recycleView.setAdapter(new SampleJoinRecyclerViewCursorAdapter( context: this, cursor));
124  }
125
126  // Update
127  private void update() {
128    String lastname = binding.lastnameEditText.getText().toString();
129    SQLiteDatabase database = new SampleDBSQLiteHelper( context: this).getWritableDatabase();
130    ContentValues values = new ContentValues();
131    values.put(SampleDBContract.Employee.COLUMN_FIRSTNAME, binding.firstnameEditText.getText().toString());
132    values.put(SampleDBContract.Employee.COLUMN_JOB_DESCRIPTION, binding.jobDescEditText.getText().toString());
133    values.put(SampleDBContract.Employee.COLUMN_EMPLOYER_ID,
134      ((Cursor)binding.employerSpinner.getSelectedItemAt( 0)).getInt( 0));
135
136    Log.d( tag: "getINT", msg: ((Cursor)binding.employerSpinner.getSelectedItemAt( 0)).getInt( 0) + "");
137    Log.d( tag: "getColumnname", ((Cursor)binding.employerSpinner.getSelectedItemAt( 0)).getColumnname( 0));
138
139    try {
140      Calendar calendar = Calendar.getInstance();
141      calendar.setTime(new SimpleDateFormat( pattern: "dd/MM/yyyy").parse(
142        binding.dobEditText.getText().toString()));
143      long date = calendar.getTimeInMillis();
144      values.put(SampleDBContract.Employee.COLUMN_DATE_OF_BIRTH, date);
145    }
146  }

```

3)

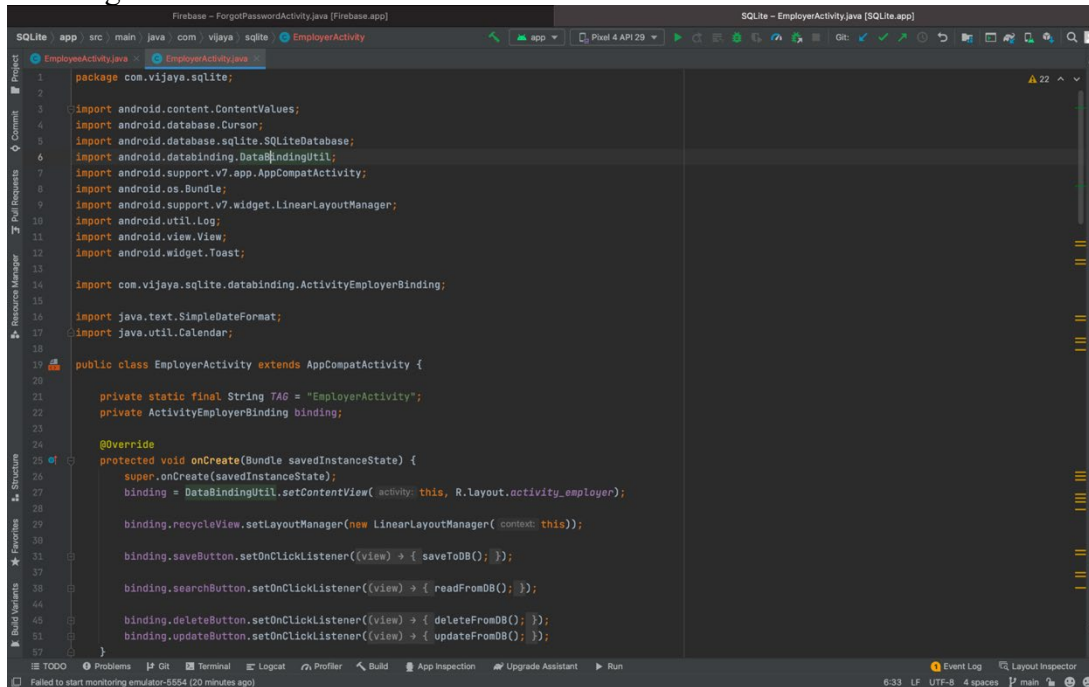
```

142    binding.dobEditText.getText().toString());
143    long date = calendar.getTimeInMillis();
144    values.put(SampleDBContract.Employee.COLUMN_DATE_OF_BIRTH, date);
145
146    calendar.setTime(new SimpleDateFormat( pattern: "dd/MM/yyyy").parse(
147      binding.employedEditText.getText().toString()));
148    date = calendar.getTimeInMillis();
149    values.put(SampleDBContract.Employee.COLUMN_EMPLOYED_DATE, date);
150  }
151  catch (Exception e) {
152    Log.e(TAG, msg: "Error", e);
153    Toast.makeText( context: this, text: "Date is in the wrong format", Toast.LENGTH_LONG).show();
154    return;
155  }
156
157  String filter = SampleDBContract.Employee.COLUMN_LASTNAME+"="+lastname+"";
158  database.update(SampleDBContract.Employee.TABLE_NAME, values, filter, whereArgs: null);
159
160  Toast.makeText( context: this, text: "Updated !!", Toast.LENGTH_LONG).show();
161  }
162
163  // Delete Employee
164  private void delete() {
165    String lastname = binding.lastnameEditText.getText().toString();
166    SQLiteDatabase db = new SampleDBSQLiteHelper( context: this).getWritableDatabase();
167    //fill this method to delete the row
168    db.delete(SampleDBContract.Employee.TABLE_NAME, whereClause: SampleDBContract.Employee.COLUMN_LASTNAME
169      +"="+lastname+"", whereArgs: null);
170
171    Toast.makeText( context: this, text: "Deleted !!", Toast.LENGTH_LONG).show();
172  }
173  }

```

EmployerActivity:

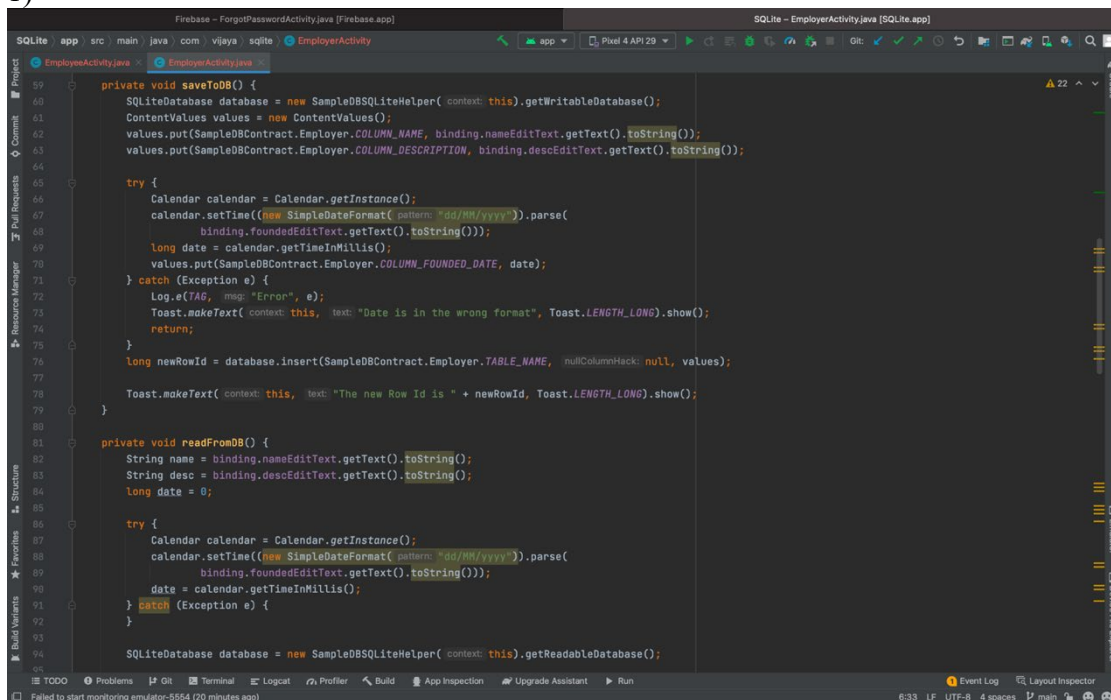
Defining the class



```
1 package com.vijaya.sqlite;
2
3 import android.content.ContentValues;
4 import android.database.Cursor;
5 import android.database.sqlite.SQLiteDatabase;
6 import android.databinding.DataBindingUtil;
7 import android.support.v7.app.AppCompatActivity;
8 import android.os.Bundle;
9 import android.support.v7.widget.LinearLayoutManager;
10 import android.util.Log;
11 import android.view.View;
12 import android.widget.Toast;
13
14 import com.vijaya.sqlite.databinding.ActivityEmployerBinding;
15
16 import java.text.SimpleDateFormat;
17 import java.util.Calendar;
18
19 public class EmployerActivity extends AppCompatActivity {
20
21     private static final String TAG = "EmployerActivity";
22     private ActivityEmployerBinding binding;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         binding = DataBindingUtil.setContentView(this, R.layout.activity_employer);
28
29         binding.recyclerView.setLayoutManager(new LinearLayoutManager(this));
30
31         binding.saveButton.setOnClickListener((view) -> { saveToDB(); });
32
33         binding.searchButton.setOnClickListener((view) -> { readFromDB(); });
34
35         binding.deleteButton.setOnClickListener((view) -> { deleteFromDB(); });
36         binding.updateButton.setOnClickListener((view) -> { updateFromDB(); });
37     }
38 }
```

- For each TODO function method i.e save, read, update, delete. Following screenshots as the code of TODO list for EMPLOYER

1)



```
59 private void saveToDB() {
60     SQLiteDatabase database = new SampleDBSQLiteHelper(this).getWritableDatabase();
61     ContentValues values = new ContentValues();
62     values.put(SampleDBContract.Employer.COLUMN_NAME, binding.nameEditText.getText().toString());
63     values.put(SampleDBContract.Employer.COLUMN_DESCRIPTION, binding.descEditText.getText().toString());
64
65     try {
66         Calendar calendar = Calendar.getInstance();
67         calendar.setTime(new SimpleDateFormat(pattern: "dd/MM/yyyy").parse(
68             binding.foundedEditText.getText().toString()));
69         long date = calendar.getTimeInMillis();
70         values.put(SampleDBContract.Employer.COLUMN_FOUNDED_DATE, date);
71     } catch (Exception e) {
72         Log.e(TAG, msg: "Error", e);
73         Toast.makeText(this, text: "Date is in the wrong format", Toast.LENGTH_LONG).show();
74         return;
75     }
76     long newRowId = database.insert(SampleDBContract.Employer.TABLE_NAME, nullColumnHack: null, values);
77
78     Toast.makeText(this, text: "The new Row Id is " + newRowId, Toast.LENGTH_LONG).show();
79 }
80
81 private void readFromDB() {
82     String name = binding.nameEditText.getText().toString();
83     String desc = binding.descEditText.getText().toString();
84     long date = 0;
85
86     try {
87         Calendar calendar = Calendar.getInstance();
88         calendar.setTime(new SimpleDateFormat(pattern: "dd/MM/yyyy").parse(
89             binding.foundedEditText.getText().toString()));
90         date = calendar.getTimeInMillis();
91     } catch (Exception e) {
92         Log.e(TAG, msg: "Error", e);
93         Toast.makeText(this, text: "Date is in the wrong format", Toast.LENGTH_LONG).show();
94         return;
95     }
96     SQLiteDatabase database = new SampleDBSQLiteHelper(this).getReadableDatabase();
97 }
```

2)


```
110 SQLiteDatabase database = new SampleDBSQLiteHelper( context, this).getReadableDatabase();
111
112 String[] projection = {
113     SampleDBContract.EMPLOYER._ID,
114     SampleDBContract.EMPLOYER.COLUMN_NAME,
115     SampleDBContract.EMPLOYER.COLUMN_DESCRIPTION,
116     SampleDBContract.EMPLOYER.COLUMN_FOUNDED_DATE
117 };
118
119 String selection =
120     SampleDBContract.EMPLOYER.COLUMN_NAME + " like ? and " +
121     SampleDBContract.EMPLOYER.COLUMN_FOUNDED_DATE + " > ? and " +
122     SampleDBContract.EMPLOYER.COLUMN_DESCRIPTION + " like ?";
123
124 String[] selectionArgs = {"%" + name + "%", date + "", "%" + desc + "%"};
125
126 Cursor cursor = database.query(
127     SampleDBContract.EMPLOYER.TABLE_NAME, // The table to query
128     projection, // The columns to return
129     selection, // The columns for the WHERE clause
130     selectionArgs, // The values for the WHERE clause
131     groupBy: null, // don't group the rows
132     having: null, // don't filter by row groups
133     orderBy: null // don't sort
134 );
135
136 binding.recycleView.setAdapter(new SampleRecyclerViewCursorAdapter( context, this, cursor));
137
138 private void updateFromDB(){
139     String name = binding.nameEditText.getText().toString();
140     String desc = binding.descEditText.getText().toString();
141     long date = 0;
142
143     try {
144         Calendar calendar = Calendar.getInstance();
145         calendar.setTimeInMillis(date);
146     } catch (Exception e) {
147         // Handle exception
148     }
149
150     SQLiteDatabase database = new SampleDBSQLiteHelper( context, this).getWritableDatabase();
151     ContentValues values = new ContentValues();
152     values.put(SampleDBContract.EMPLOYER.COLUMN_DESCRIPTION, binding.descEditText.getText().toString());
153     String selection =
154         SampleDBContract.EMPLOYER.COLUMN_NAME + " like ?";
155     String[] selectionArgs = {"%" + name + "%"};
156     int newRowId = database.update(SampleDBContract.EMPLOYER.TABLE_NAME, values, selection, selectionArgs );
157     Toast.makeText( context, this, "The updated Row Id is " + newRowId, Toast.LENGTH_LONG).show();
158 }
```

3)

```
159 try {
160     Calendar calendar = Calendar.getInstance();
161     calendar.setTime(new SimpleDateFormat( pattern: "dd/MM/yyyy").parse(
162         binding.foundedEditText.getText().toString()));
163     date = calendar.getTimeInMillis();
164 } catch (Exception e) {
165     // Handle exception
166 }
167
168 SQLiteDatabase database = new SampleDBSQLiteHelper( context, this).getWritableDatabase();
169 ContentValues values = new ContentValues();
170 values.put(SampleDBContract.EMPLOYER.COLUMN_DESCRIPTION, binding.descEditText.getText().toString());
171 String selection =
172     SampleDBContract.EMPLOYER.COLUMN_NAME + " like ?";
173 String[] selectionArgs = {"%" + name + "%"};
174 int newRowId = database.update(SampleDBContract.EMPLOYER.TABLE_NAME, values, selection, selectionArgs );
175 Toast.makeText( context, this, "The updated Row Id is " + newRowId, Toast.LENGTH_LONG).show();
176 }
177
178 private void deleteFromDB(){
179     String name = binding.nameEditText.getText().toString();
180     String desc = binding.descEditText.getText().toString();
181     long date = 0;
182
183     try {
184         Calendar calendar = Calendar.getInstance();
185         calendar.setTime(new SimpleDateFormat( pattern: "dd/MM/yyyy").parse(
186             binding.foundedEditText.getText().toString()));
187         date = calendar.getTimeInMillis();
188     } catch (Exception e) {
189         // Handle exception
190     }
191
192     SQLiteDatabase database = new SampleDBSQLiteHelper( context, this).getWritableDatabase();
193     ContentValues values = new ContentValues();
194     values.put(SampleDBContract.EMPLOYER.COLUMN_NAME, binding.nameEditText.getText().toString());
195     String selection =
196         SampleDBContract.EMPLOYER.COLUMN_NAME + " like ? and " +
197         SampleDBContract.EMPLOYER.COLUMN_FOUNDED_DATE + " > ? and " +
198         SampleDBContract.EMPLOYER.COLUMN_DESCRIPTION + " like ?";
199     String[] selectionArgs = {"%" + name + "%", date + "", "%" + desc + "%"};
200
201     long newRowId = database.delete(SampleDBContract.EMPLOYER.TABLE_NAME, selection, selectionArgs );
202     Toast.makeText( context, this, "The deleted Row Id is " + newRowId, Toast.LENGTH_LONG).show();
203 }
```

4)

```

137 values.put(SampleDBContract.Employer.COLUMN_DESCRIPTION, binding.descEditText.getText().toString());
138 String selection =
139     SampleDBContract.Employer.COLUMN_NAME + " like ?";
140 String[] selectionArgs = {"%" + name};
141 int newRowId = database.update(SampleDBContract.Employer.TABLE_NAME, values, selection, selectionArgs);
142 Toast.makeText(context, this, "The updated Row Id is " + newRowId, Toast.LENGTH_LONG).show();
143 }
144 private void deleteFromDB(){
145     String name = binding.nameEditText.getText().toString();
146     String desc = binding.descEditText.getText().toString();
147     long date = 0;
148
149     try {
150         Calendar calendar = Calendar.getInstance();
151         calendar.setTime(new SimpleDateFormat(pattern: "dd/MM/yyyy").parse(
152             binding.foundedEditText.getText().toString()));
153         date = calendar.getTimeInMillis();
154     }
155     catch (Exception e) {}
156     SQLiteDatabase database = new SampleDBSQLiteHelper(context).getWritableDatabase();
157     ContentValues values = new ContentValues();
158     values.put(SampleDBContract.Employer.COLUMN_NAME, binding.nameEditText.getText().toString());
159     String selection =
160         SampleDBContract.Employer.COLUMN_NAME + " like ? and " +
161         SampleDBContract.Employer.COLUMN_FOUNDED_DATE + " > ? and " +
162         SampleDBContract.Employer.COLUMN_DESCRIPTION + " like ?";
163     String[] selectionArgs = {"%" + name + "%", date + "", "%" + desc + "%"};
164     long newRowId = database.delete(SampleDBContract.Employer.TABLE_NAME, selection, selectionArgs);
165     Toast.makeText(context, this, "The deleted Row Id is " + newRowId, Toast.LENGTH_LONG).show();
166 }
167 }
168 }

```

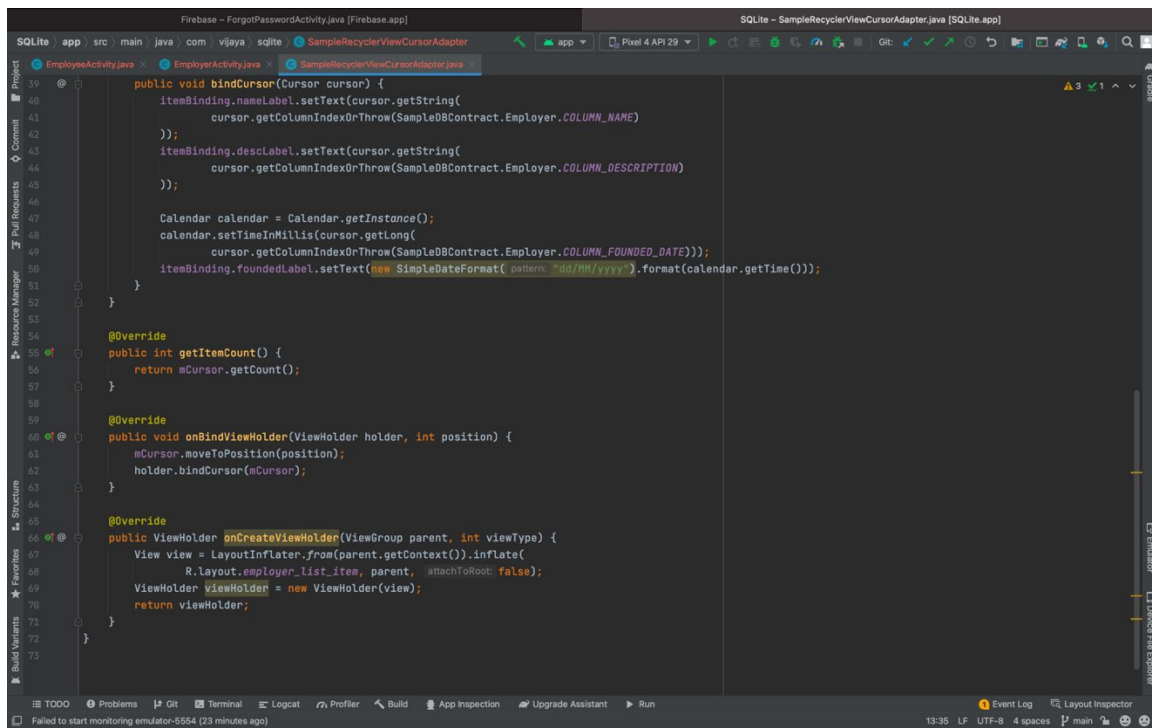
4

Sample Recycler View Cursor Adapter:

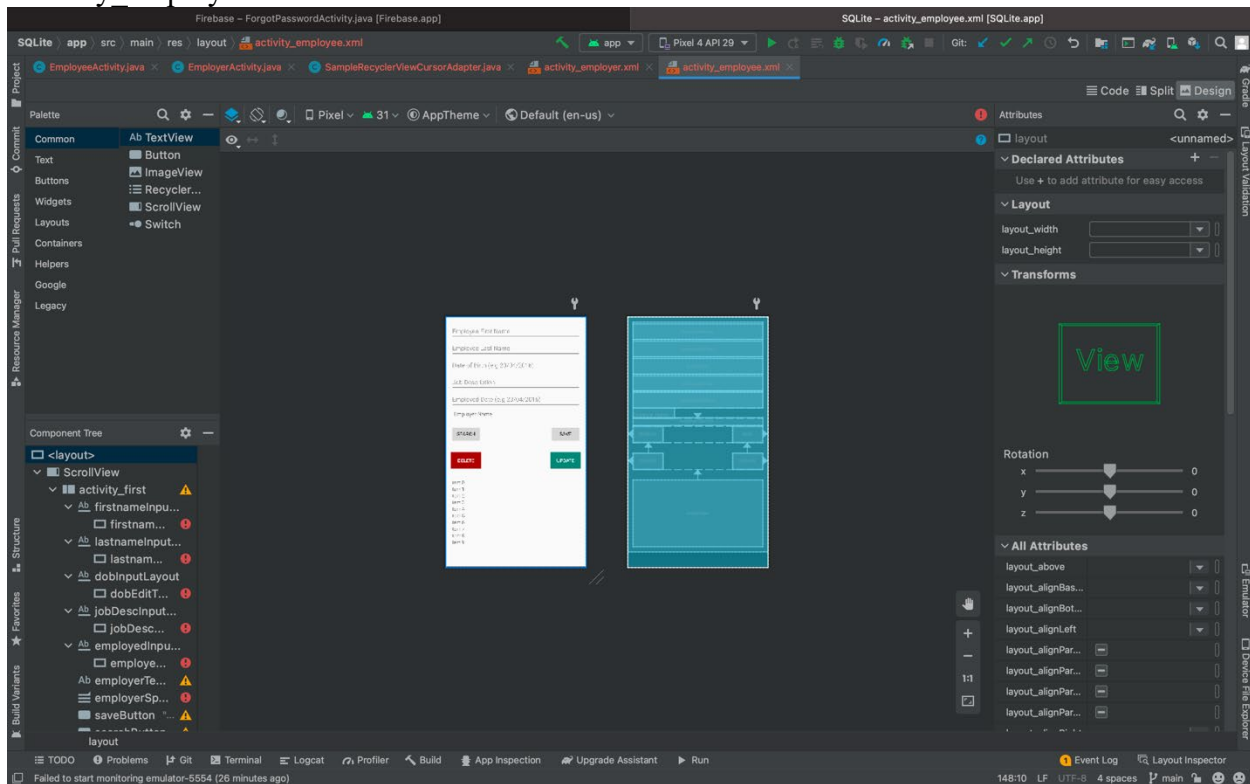
```

1 package com.vijaya.sqlite;
2
3 import android.content.Context;
4 import android.database.Cursor;
5 import android.databinding.DataBindingUtil;
6 import android.support.v7.widget.RecyclerView;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10
11 import com.vijaya.sqlite.databinding.EmployerListItemBinding;
12
13 import java.text.SimpleDateFormat;
14 import java.util.Calendar;
15
16 /**
17  * Created by gharg on 26/09/2016.
18  */
19
20 public class SampleRecyclerViewCursorAdapter extends RecyclerView.Adapter<SampleRecyclerViewCursorAdapter.ViewHolder> {
21
22     Context mContext;
23     Cursor mCursor;
24
25     public SampleRecyclerViewCursorAdapter(Context context, Cursor cursor) {
26
27         mContext = context;
28         mCursor = cursor;
29     }
30
31     public static class ViewHolder extends RecyclerView.ViewHolder {
32         EmployerListItemBinding itemBinding;
33
34         public ViewHolder(View itemView) {
35             super(itemView);
36             itemBinding = DataBindingUtil.bind(itemView);
37         }
38     }
39 }

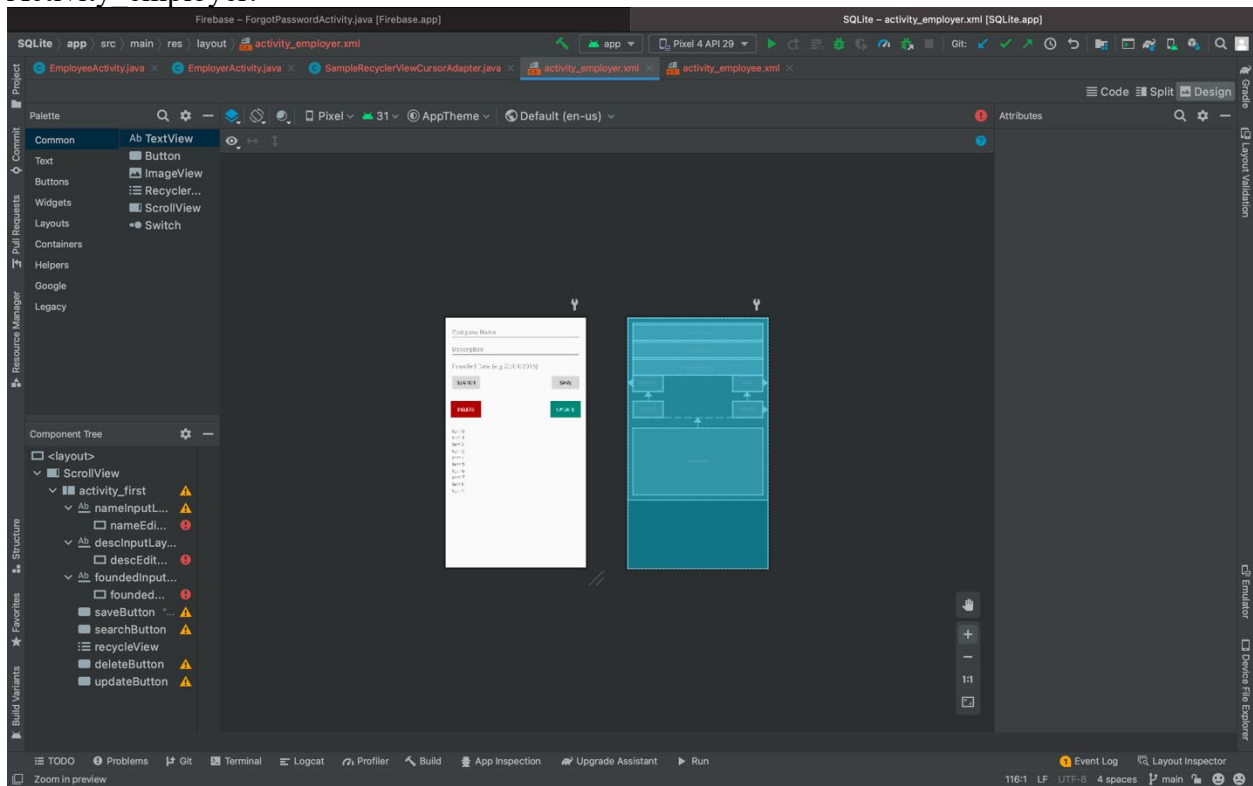
```



Layouts: Activity employee:



Activity employer:



Output:



Employer: Here we can search, save, delete, update the employer details in the same page.

9:57

SQLite

Company Name
Amazon

Description
Service Based

Founded Date (e.g 23/04/2016)
10/01/2000

SEARCH SAVE

DELETE UPDATE

Employee: Here we can search, save, delete, update the employee details in the same page.

9:59

SQLite

Abhinay

Employee Last Name
Yadav

Date of Birth (e.g 23/04/2016)
21/12/1995

Job Description
Developer

Employed Date (e.g 23/04/2016)
10/12/2010

Employer Name
cap

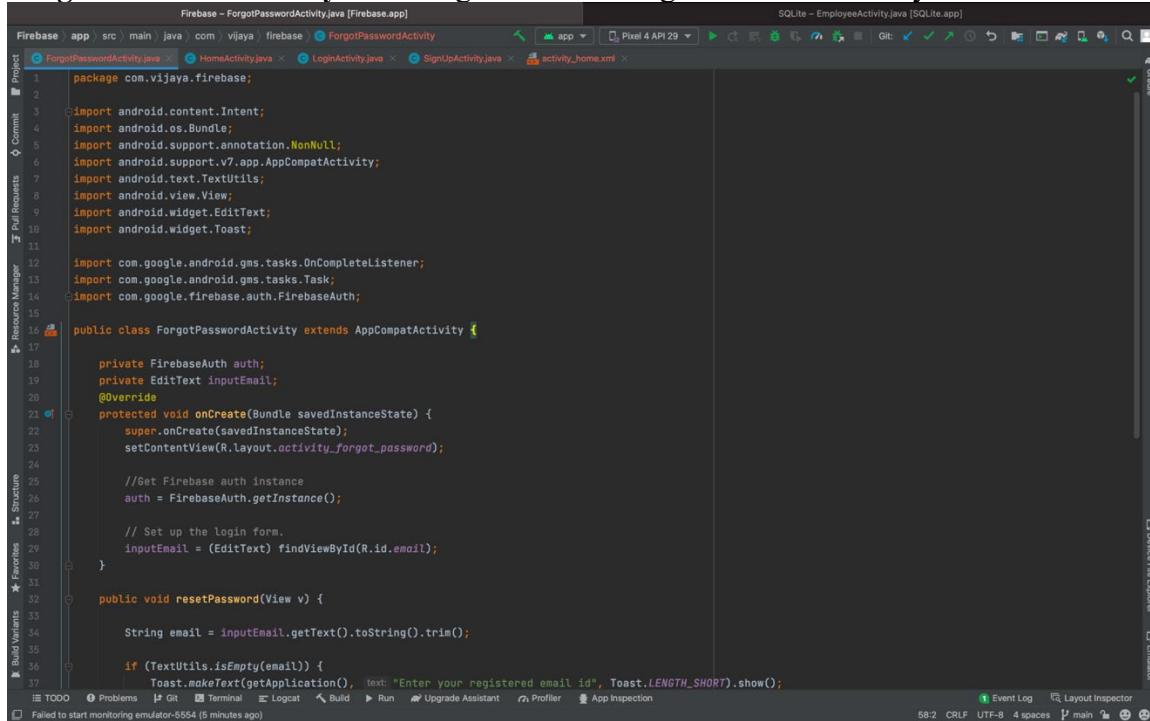
SEARCH SAVE

DELETE UPDATE

Employee Details
Abhinay
Yadav
21/12/1995
Developer
Employer
cap
sql
23/04/2020

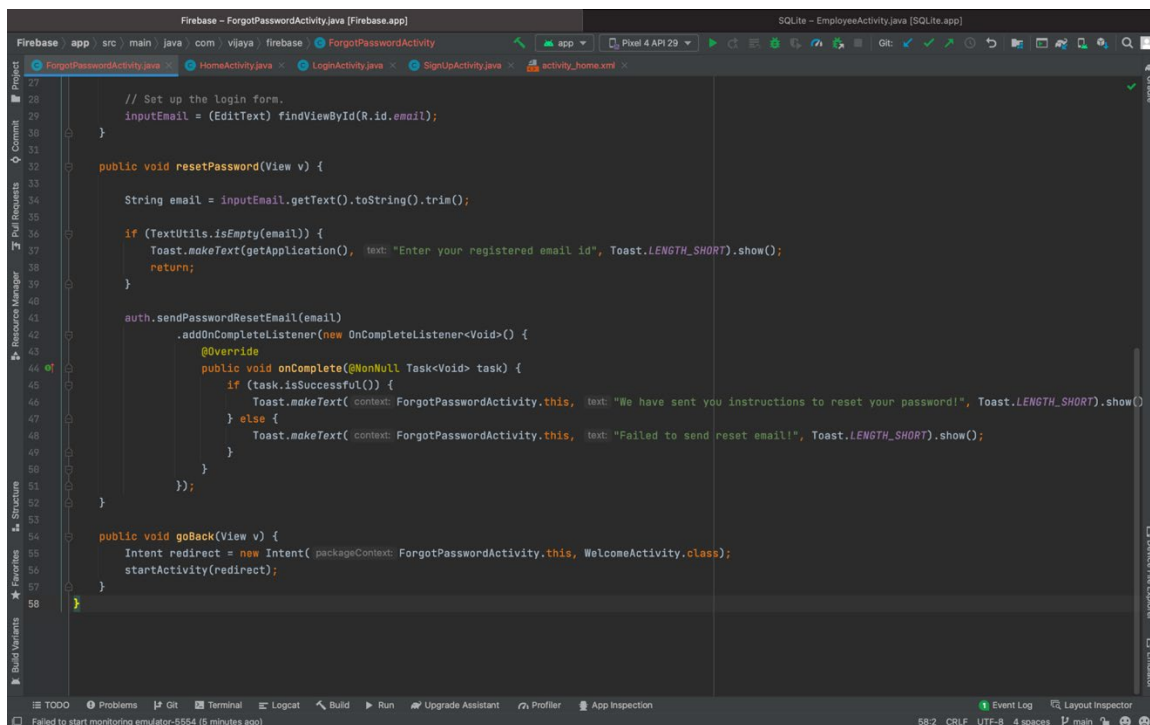
Firestore:

ForgotPasswordActivity: Defining the class ‘ForgotPasswordActivity’



```
1 package com.vijaya.firestore;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.support.annotation.NonNull;
6 import android.support.v7.app.AppCompatActivity;
7 import android.text.TextUtils;
8 import android.view.View;
9 import android.widget.EditText;
10 import android.widget.Toast;
11
12 import com.google.android.gms.tasks.OnCompleteListener;
13 import com.google.android.gms.tasks.Task;
14 import com.google.firebase.auth.FirebaseAuth;
15
16 public class ForgotPasswordActivity extends AppCompatActivity {
17
18     private FirebaseAuth auth;
19     private EditText inputEmail;
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_forgot_password);
24
25         //Get Firebase auth instance
26         auth = FirebaseAuth.getInstance();
27
28         // Set up the login form.
29         inputEmail = (EditText) findViewById(R.id.email);
30     }
31
32     public void resetPassword(View v) {
33
34         String email = inputEmail.getText().toString().trim();
35
36         if (TextUtils.isEmpty(email)) {
37             Toast.makeText(getApplicationContext(), "Enter your registered email id", Toast.LENGTH_SHORT).show();
38         }
39     }
40
41     public void goBack(View v) {
42         Intent redirect = new Intent(packageContext, WelcomeActivity.class);
43         startActivity(redirect);
44     }
45 }
```

The reset password method as follows:



```
27 // Set up the login form.
28 inputEmail = (EditText) findViewById(R.id.email);
29 }
30
31 public void resetPassword(View v) {
32
33     String email = inputEmail.getText().toString().trim();
34
35     if (TextUtils.isEmpty(email)) {
36         Toast.makeText(getApplicationContext(), "Enter your registered email id", Toast.LENGTH_SHORT).show();
37         return;
38     }
39
40     auth.sendPasswordResetEmail(email)
41         .addOnCompleteListener(new OnCompleteListener<Void>() {
42             @Override
43             public void onComplete(@NonNull Task<Void> task) {
44                 if (task.isSuccessful()) {
45                     Toast.makeText(ForgotPasswordActivity.this, "We have sent you instructions to reset your password!", Toast.LENGTH_SHORT).show();
46                 } else {
47                     Toast.makeText(ForgotPasswordActivity.this, "Failed to send reset email", Toast.LENGTH_SHORT).show();
48                 }
49             }
50         });
51 }
52
53 public void goBack(View v) {
54     Intent redirect = new Intent(packageContext, WelcomeActivity.class);
55     startActivity(redirect);
56 }
57 }
```

HomeActivity: Defining the class “HomeActivity”

```
1 package com.vijaya.firebase;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.support.annotation.NonNull;
6 import android.support.v7.app.AppCompatActivity;
7 import android.text.TextUtils;
8 import android.util.Log;
9 import android.view.View;
10 import android.widget.Button;
11 import android.widget.EditText;
12 import android.widget.TextView;
13
14 import com.google.android.gms.tasks.OnCompleteListener;
15 import com.google.android.gms.tasks.Task;
16 import com.google.firebase.auth.FirebaseAuth;
17 import com.google.firebase.auth.FirebaseUser;
18 import com.google.firebase.database.DataSnapshot;
19 import com.google.firebase.database.DatabaseError;
20 import com.google.firebase.database.DatabaseReference;
21 import com.google.firebase.database.FirebaseDatabase;
22 import com.google.firebase.database.ValueEventListener;
23
24 public class HomeActivity extends AppCompatActivity {
25
26     private static final String TAG = HomeActivity.class.getSimpleName();
27     private TextView txtDetails;
28     private EditText inputName, inputPhone;
29     private Button btnSave;
30     private DatabaseReference mFirebaseDatabase;
31
32     private String userId;
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_home);
38     }
39 }
```

Method for create, Data change, cancelled, delete, logout

```
34
35 @Override
36 protected void onCreate(Bundle savedInstanceState) {
37     super.onCreate(savedInstanceState);
38     setContentView(R.layout.activity_home);
39
40     // Displaying toolbar icon
41     getSupportActionBar().setDisplayHomeAsUpEnabled(true);
42     getSupportActionBar().setIcon(R.mipmap.ic_launcher);
43
44     txtDetails = (TextView) findViewById(R.id.txt_user);
45     inputName = (EditText) findViewById(R.id.name);
46     inputPhone = (EditText) findViewById(R.id.phone);
47     btnSave = (Button) findViewById(R.id.btn_save);
48     final Button btnDelete = (Button) findViewById(R.id.btn_delete_account);
49     final Button btnLogout = (Button) findViewById(R.id.btn_logout);
50
51     final FirebaseDatabase firebaseInstance = FirebaseDatabase.getInstance();
52
53     // get reference to 'users' node
54     mFirebaseDatabase = firebaseInstance.getReference(path: "users");
55
56     // store app title to 'app_title' node
57     firebaseInstance.getReference(path: "app_title").setValue("Realtime Database");
58
59     // app_title change listener
60     firebaseInstance.getReference(path: "app_title").addValueEventListener(new ValueEventListener() {
61         @Override
62         public void onDataChange(DataSnapshot dataSnapshot) {
63             Log.e(TAG, msg: "App title updated");
64
65             String appTitle = dataSnapshot.getValue(String.class);
66
67             // update toolbar title
68             getSupportActionBar().setTitle(appTitle);
69         }
70     });
71 }
```

```

    @Override
    public void onCancelled(DatabaseError error) {
        // Failed to read value
        Log.e(TAG, msg: "Failed to read app title value.", error.toException());
    }
}

// Save / update the user
btnSave.setOnClickListener((view) -> {
    String name = inputName.getText().toString();
    String phone = inputPhone.getText().toString();

    // Check for already existed userId
    if (TextUtils.isEmpty(userId)) {
        createUser(name, phone);
    } else {
        updateUser(name, phone);
    }
});

btnDelete.setOnClickListener((view) -> { deleteAccount(); });

btnLogout.setOnClickListener((view) -> { logout(); });

toggleButton();

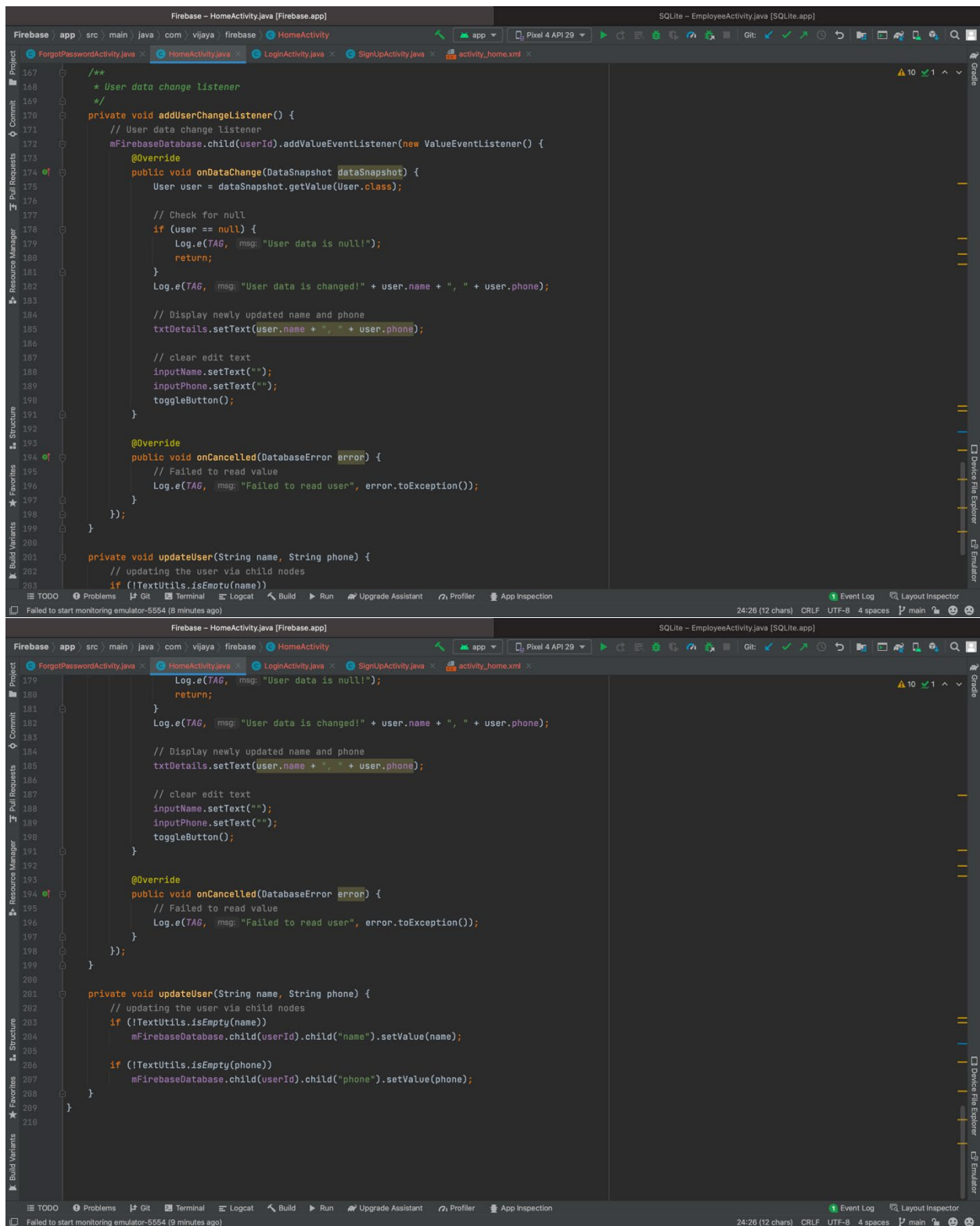
FirebaseAuth auth = FirebaseAuth.getInstance();
auth.addAuthStateListener((firebaseAuth) -> {
    FirebaseUser user = firebaseAuth.getCurrentUser();
    if (user == null) {
        // user authstate is changed -user is null
        // launch login activity
        startActivity(new Intent( packageContext: HomeActivity.this, WelcomeActivity.class));
        finish();
    }
});
});

private void deleteAccount() {
    final FirebaseAuth auth = FirebaseAuth.getInstance();
    FirebaseUser user = auth.getCurrentUser();
    if (user != null) {
        user.delete().addOnCompleteListener((task) -> { auth.signOut(); });
    }
}

private void logout() {
    final FirebaseAuth auth = FirebaseAuth.getInstance();
    auth.signOut();
}

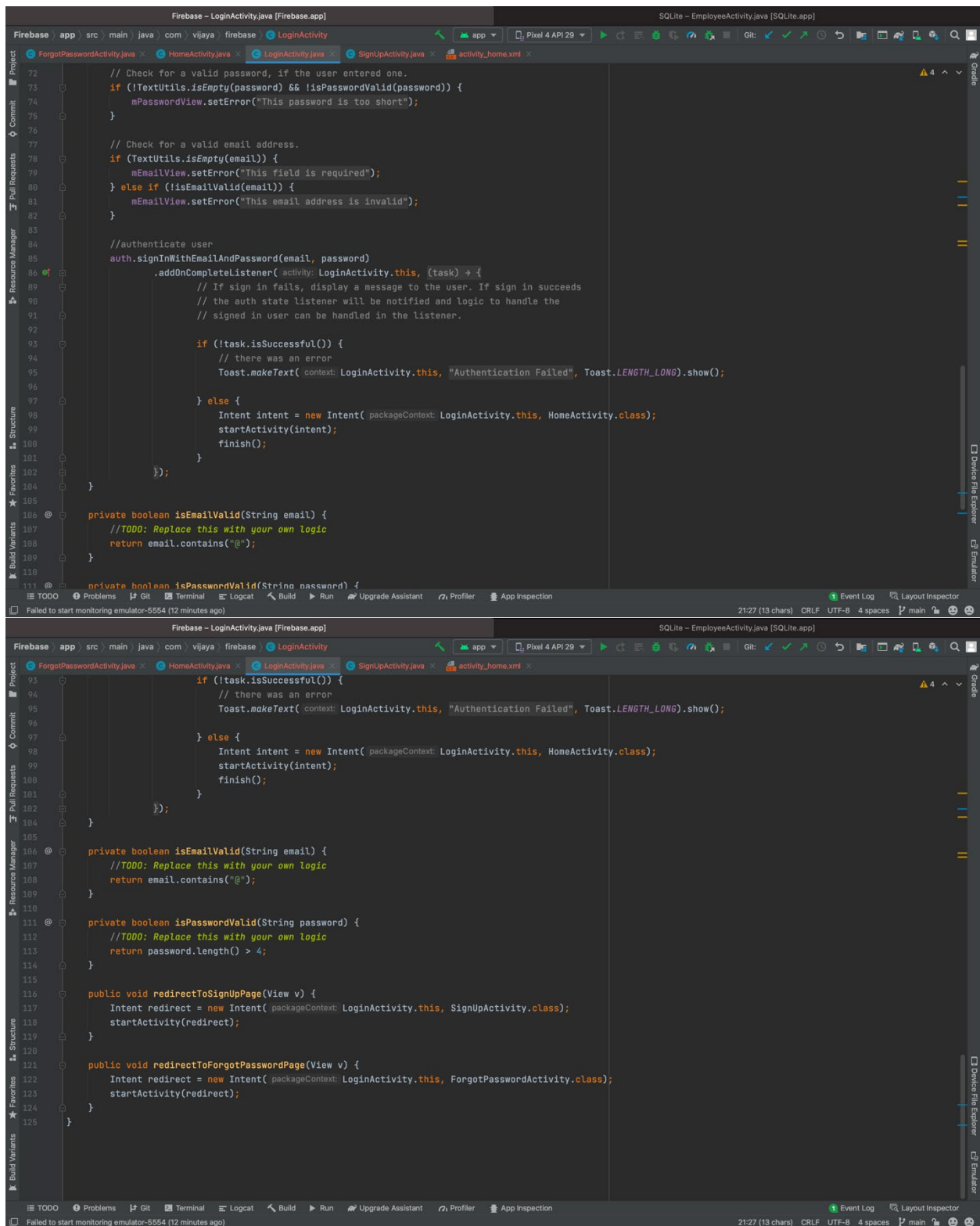
// Changing button text
private void toggleButton() {
    if (TextUtils.isEmpty(userId)) {
        btnSave.setText("Save");
    } else {
        btnSave.setText("Update");
    }
}

/**
 * Creating new user node under 'users'
 */
private void createUser(String name, String phone) {
    // TODO
    // In real apps this userId should be fetched
    // by implementing firebase auth
    if (TextUtils.isEmpty(userId)) {
        userId = mFirebaseDatabase.push().getKey();
    }
    User user = new User(name, phone);
    mFirebaseDatabase.child(userId).setValue(user);
    addUserChangeListener();
}
}
```

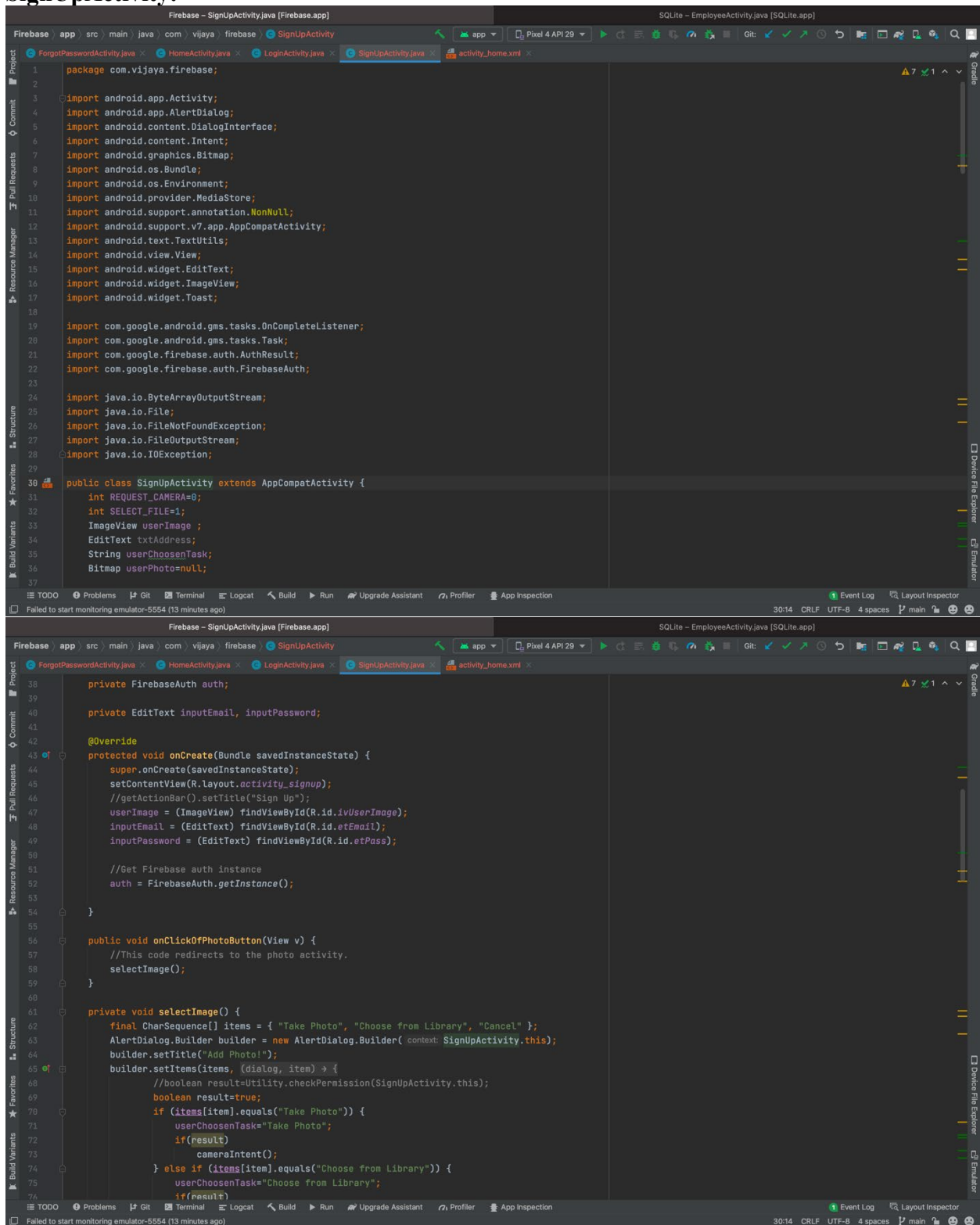



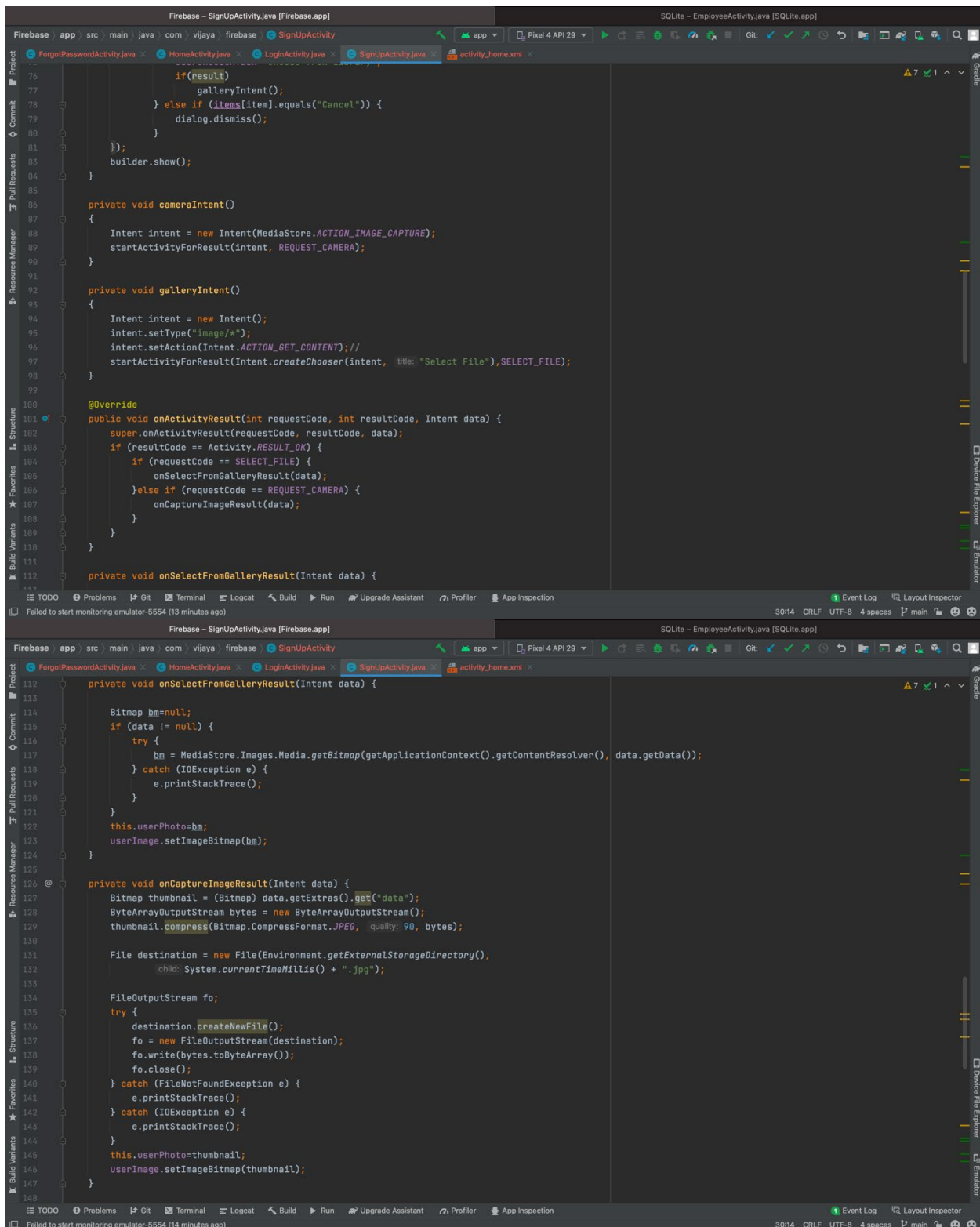
LoginActivity:

```
1 package com.vijaya.firebase;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.support.annotation.NonNull;
6 import android.support.v7.app.AppCompatActivity;
7 import android.text.TextUtils;
8 import android.view.View;
9 import android.widget.AutoCompleteTextView;
10 import android.widget.EditText;
11 import android.widget.Toast;
12
13 import com.google.android.gms.tasks.OnCompleteListener;
14 import com.google.android.gms.tasks.Task;
15 import com.google.firebase.auth.AuthResult;
16 import com.google.firebase.auth.FirebaseAuth;
17
18 /**
19  * A login screen that offers login via email/password.
20  */
21 public class LoginActivity extends AppCompatActivity {
22
23     /**
24      * Id to identity READ_CONTACTS permission request.
25      */
26     private static final int REQUEST_READ_CONTACTS = 0;
27
28     /**
29      * A dummy authentication store containing known user names and passwords.
30      * TODO: remove after connecting to a real authentication system.
31      */
32     private static final String[] DUMMY_CREDENTIALS = new String[]{
33         "foo@example.com:hello", "bar@example.com:world"
34     };
35
36     private FirebaseAuth auth;
37
38     // UI references.
39     private AutoCompleteTextView mEmailView;
40     private EditText mPasswordView;
41     private View mProgressView;
42     private View mLoginFormView;
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46
47         //Get Firebase auth instance
48         auth = FirebaseAuth.getInstance();
49         super.onCreate(savedInstanceState);
50         setContentView(R.layout.activity_login);
51         // Set up the login form.
52         mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
53         mPasswordView = (EditText) findViewById(R.id.password);
54
55     }
56
57     /**
58      * Attempts to sign in or register the account specified by the login form.
59      * If there are form errors (invalid email, missing fields, etc.), the
60      * errors are presented and no actual login attempt is made.
61      */
62     public void attemptLogin(View v) {
63
64         // Reset errors.
65         mEmailView.setError(null);
66         mPasswordView.setError(null);
67
68         // Store values at the time of the login attempt.
69         String email = mEmailView.getText().toString();
70         String password = mPasswordView.getText().toString();
71
72         // Check for a valid password, if the user entered one.
73         if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
```



SignUpActivity:






```

    Firebase - SignUpActivity.java [Firebase.app]
    SQLite - EmployeeActivity.java [SQLite.app]

    app src main java com vijaya firebase SignUpActivity
    ForgotPasswordActivity.java HomeActivity.java LoginActivity.java SignUpActivity.java activity_home.xml

    149 public void onSignUp(View v)
    150 {
    151     //add the sign up details to firebase
    152     String email = inputEmail.getText().toString().trim();
    153     String password = inputPassword.getText().toString().trim();
    154
    155     if (TextUtils.isEmpty(email)) {
    156         Toast.makeText(getApplicationContext(), text: "Enter email address!", Toast.LENGTH_SHORT).show();
    157         return;
    158     }
    159
    160     if (TextUtils.isEmpty(password)) {
    161         Toast.makeText(getApplicationContext(), text: "Enter password!", Toast.LENGTH_SHORT).show();
    162         return;
    163     }
    164
    165     if (password.length() < 6) {
    166         Toast.makeText(getApplicationContext(), text: "Password too short, enter minimum 6 characters!", Toast.LENGTH_SHORT).show();
    167         return;
    168     }
    169
    170     if (true) {
    171         Toast.makeText(getApplicationContext(), text: "inside firebase sign up ", Toast.LENGTH_SHORT).show();
    172     }
    173     //create user
    174     auth.createUserWithEmailAndPassword(email, password)
    175     .addOnCompleteListener( activity: SignUpActivity.this, (task) -> {
    176         Toast.makeText( context: SignUpActivity.this, text: "createUserWithEmail:onComplete:" + task.isSuccessful(), Toast.LENGTH_SHORT).show();
    177         // If sign in fails, display a message to the user. If sign in succeeds
    178         // the auth state listener will be notified and logic to handle the
    179         // signed in user can be handled in the listener.
    180         if (!task.isSuccessful()) {
    181             Toast.makeText( context: SignUpActivity.this, text: "Authentication failed." + task.getException(),
    182                 Toast.LENGTH_SHORT).show();
    183         } else {
    184             startActivity(new Intent( packageContext: SignUpActivity.this, LoginActivity.class));
    185             finish();
    186         }
    187     });
    188 }
    189 }
    190 }
    191 }
    192 }
    193 }
    194 }
    195 }
    196 }
    197 }

    TODO Problems Git Terminal Logcat Build Run Upgrade Assistant Profiler App Inspection
    Failed to start monitoring emulator-5554 (14 minutes ago)
    30:14 CRLF UTF-8 4 spaces P main
```

```

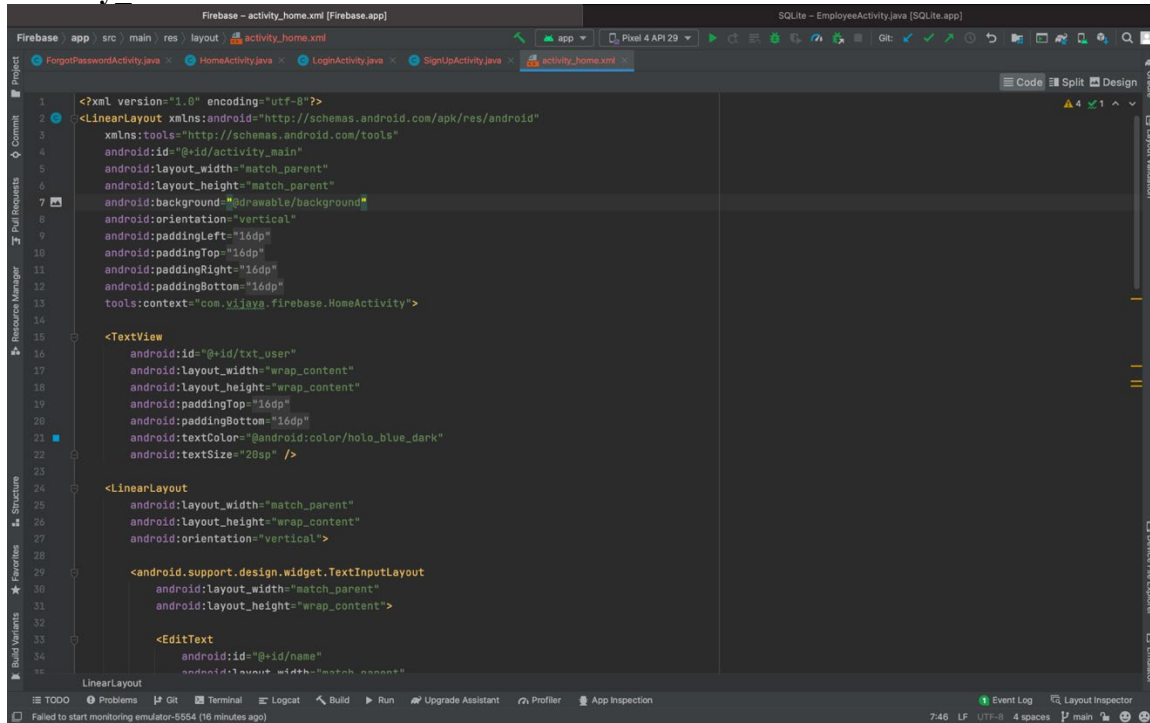
    Firebase - SignUpActivity.java [Firebase.app]
    SQLite - EmployeeActivity.java [SQLite.app]

    app src main java com vijaya firebase SignUpActivity
    ForgotPasswordActivity.java HomeActivity.java LoginActivity.java SignUpActivity.java activity_home.xml

    163 }
    164 }
    165 }
    166 }
    167 }
    168 }
    169 }
    170 }
    171 }
    172 }
    173 }
    174 }
    175 }
    176 }
    177 }
    178 }
    179 }
    180 }
    181 }
    182 }
    183 }
    184 }
    185 }
    186 }
    187 }
    188 }
    189 }
    190 }
    191 }
    192 }
    193 }
    194 }
    195 }
    196 }
    197 }

    TODO Problems Git Terminal Logcat Build Run Upgrade Assistant Profiler App Inspection
    Failed to start monitoring emulator-5554 (14 minutes ago)
    30:14 CRLF UTF-8 4 spaces P main
```

Layout: Activity home:



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    android:orientation="vertical"
    android:paddingLeft="16dp"
    android:paddingTop="16dp"
    android:paddingRight="16dp"
    android:paddingBottom="16dp"
    tools:context="com.yijaya.firebase.HomeActivity">

    <TextView
        android:id="@+id/txt_user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="16dp"
        android:paddingBottom="16dp"
        android:textColor="@android:color/holo_blue_dark"
        android:textSize="20sp" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

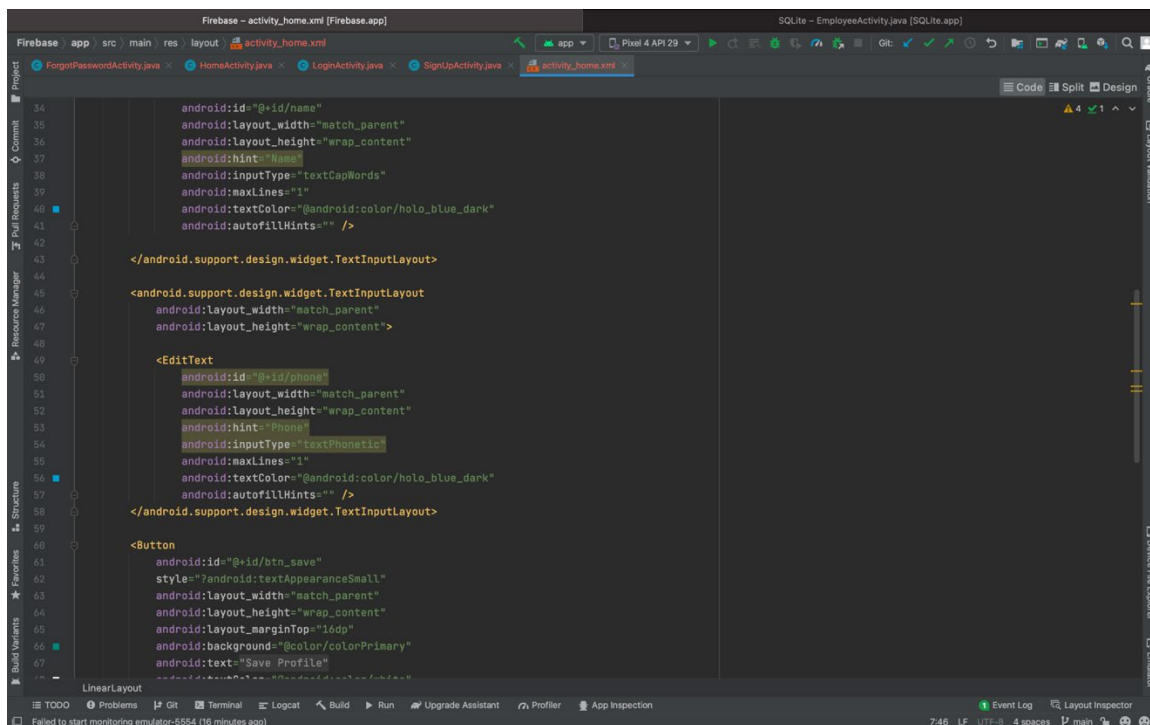
        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <EditText
                android:id="@+id/name"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:inputType="text" />

        </android.support.design.widget.TextInputLayout>

    </LinearLayout>

</LinearLayout>
```



```

        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/hint_name"
        android:inputType="textCapWords"
        android:maxLines="1"
        android:textColor="@android:color/holo_blue_dark"
        android:autofillHints="" />

    </android.support.design.widget.TextInputLayout>

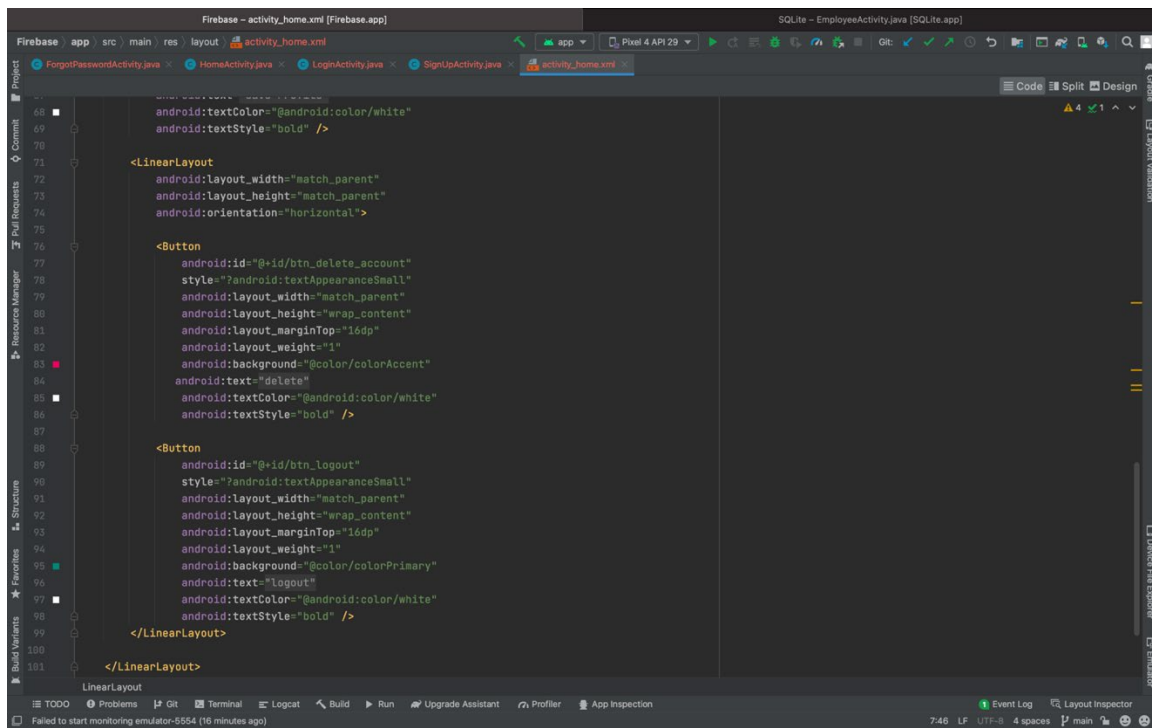
    <android.support.design.widget.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <EditText
            android:id="@+id/phone"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/hint_phone"
            android:inputType="textPhonetic"
            android:maxLines="1"
            android:textColor="@android:color/holo_blue_dark"
            android:autofillHints="" />

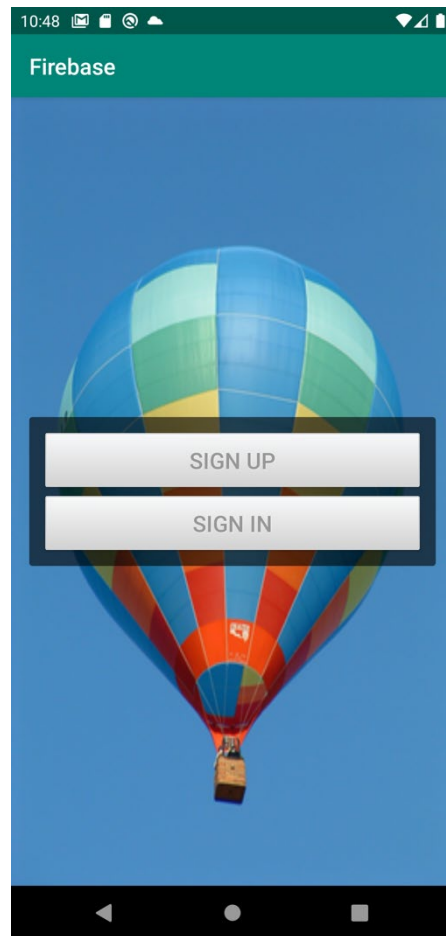
    </android.support.design.widget.TextInputLayout>

    <Button
        android:id="@+id/btn_save"
        style="?android:textAppearanceSmall"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:background="@color/colorPrimary"
        android:text="Save Profile"
        android:textColor="@color/white" />

</LinearLayout>
```



OUTPUT:



Firestore

Abhinay

Yadav

Take Picture

yabhinay404@gail.com

.....

CREATE ACCOUNT

Sign in

Email

yabhinay404@gail.com

Password

.....

SIGN IN OR REGISTER

[Forgot Password ?](#)

[Register Now](#)



Abhinay, 123456789

Name

Phone

UPDATE

DELETE

LOGOUT