

BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI



BÀI TẬP THỰC HÀNH SỐ 4
PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG
NỘI DUNG BỔ SUNG: ỨNG DỤNG VỚI CSDL

STT	Mã sinh viên	Họ và tên	Lớp
1	2251061863	Lê Hà Phương	64CNTT1

Hà Nội, năm 2025

BÀI TẬP 1: SHARED PREFERENCE

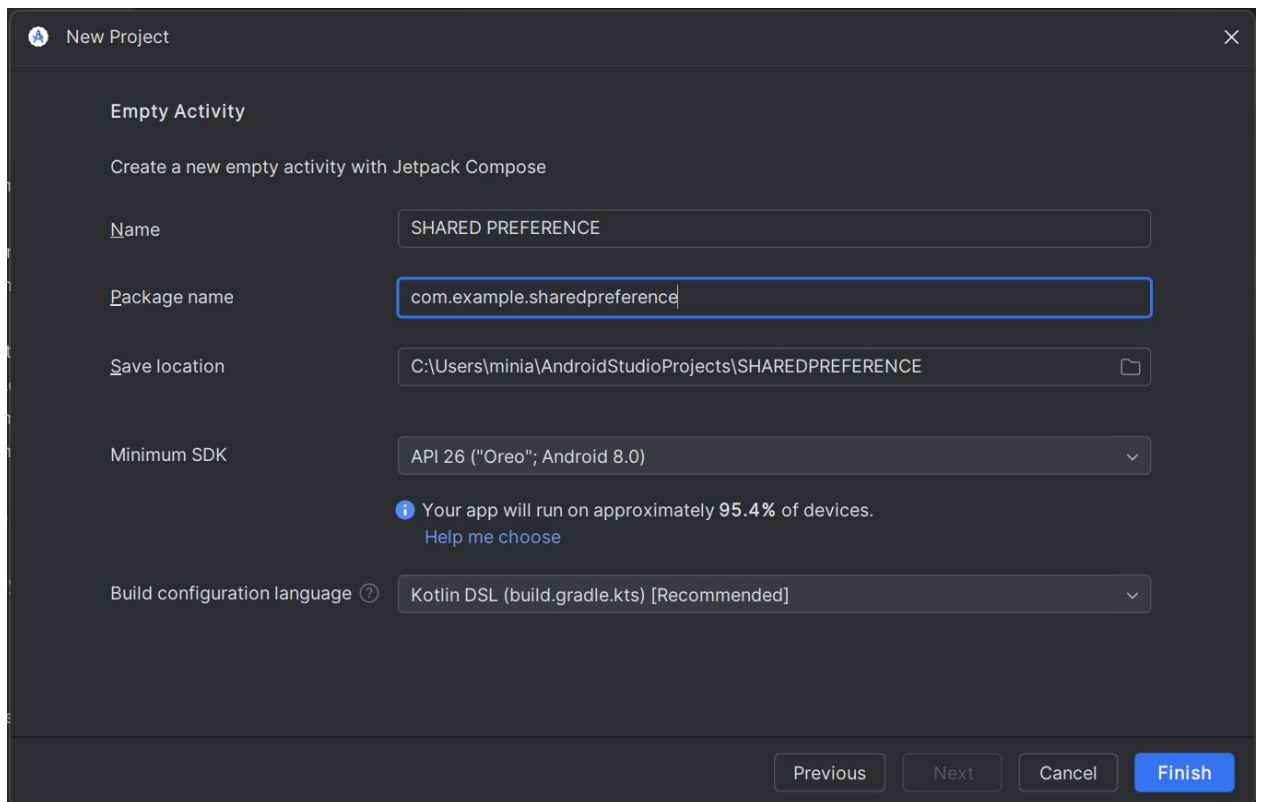
Mục tiêu:

- Hiểu cách sử dụng Shared Preference để lưu trữ dữ liệu cục bộ trong ứng dụng Android.
- Thực hành lưu trữ và đọc dữ liệu từ Shared Preference.

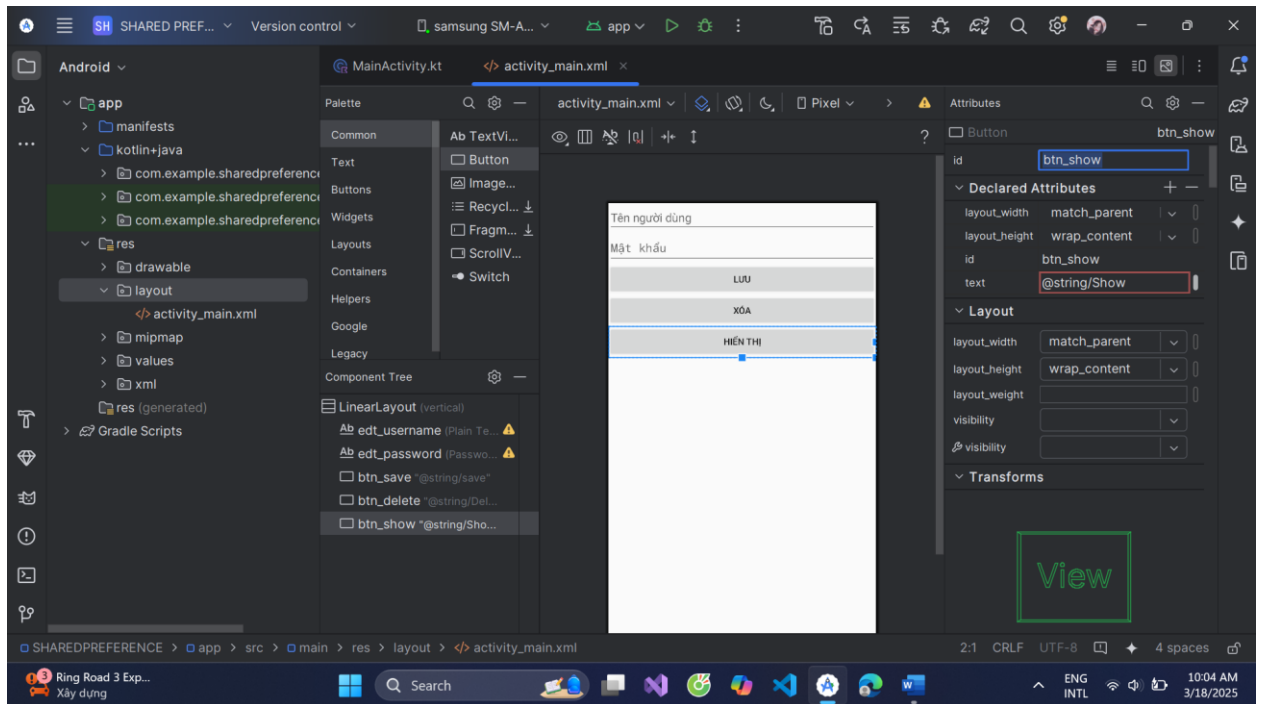
Yêu cầu:

1. Tạo ứng dụng mới:

- Tạo một dự án Android mới bằng Kotlin.



- Thiết kế giao diện người dùng với hai trường nhập (EditText) cho tên người dùng và mật khẩu, và ba nút bấm: "Lưu", "Xóa", và "Hiển thị".



2. Sử dụng Shared Preference:

- Tạo một lớp helper **PreferenceHelper** để quản lý Shared Preference.
- Khi người dùng nhấn nút "Lưu", lưu tên người dùng và mật khẩu vào Shared Preference.
- Khi người dùng nhấn nút "Xóa", xóa dữ liệu đã lưu trong Shared Preference.
- Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ Shared Preference và hiển thị lên màn hình.

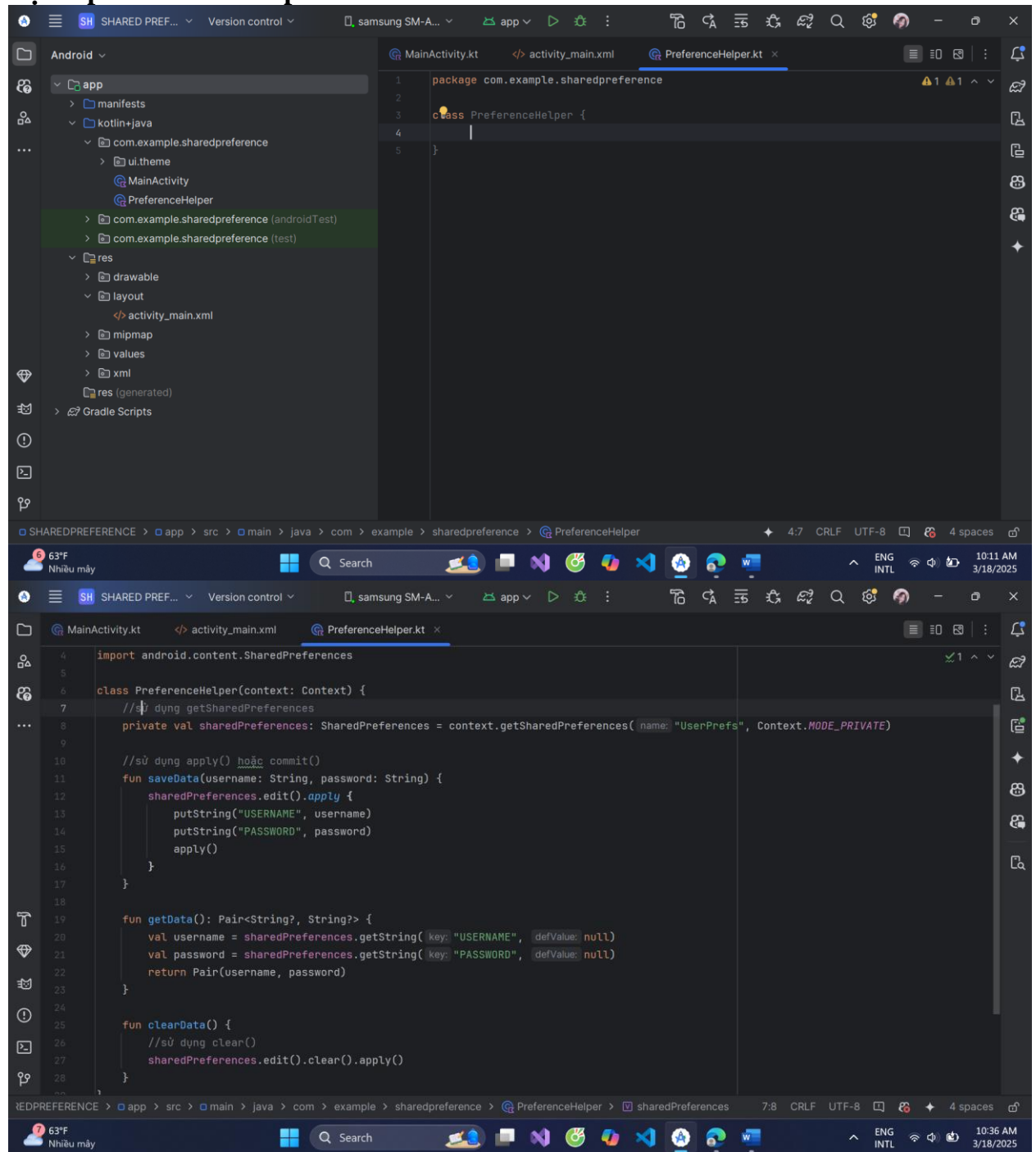
3. Thực hành:

- Viết mã Kotlin để thực hiện các chức năng trên.
- Sử dụng `getSharedPreferences` để truy cập Shared Preference và `edit()` để lưu dữ liệu.
- Sử dụng `commit()` hoặc `apply()` để lưu thay đổi.

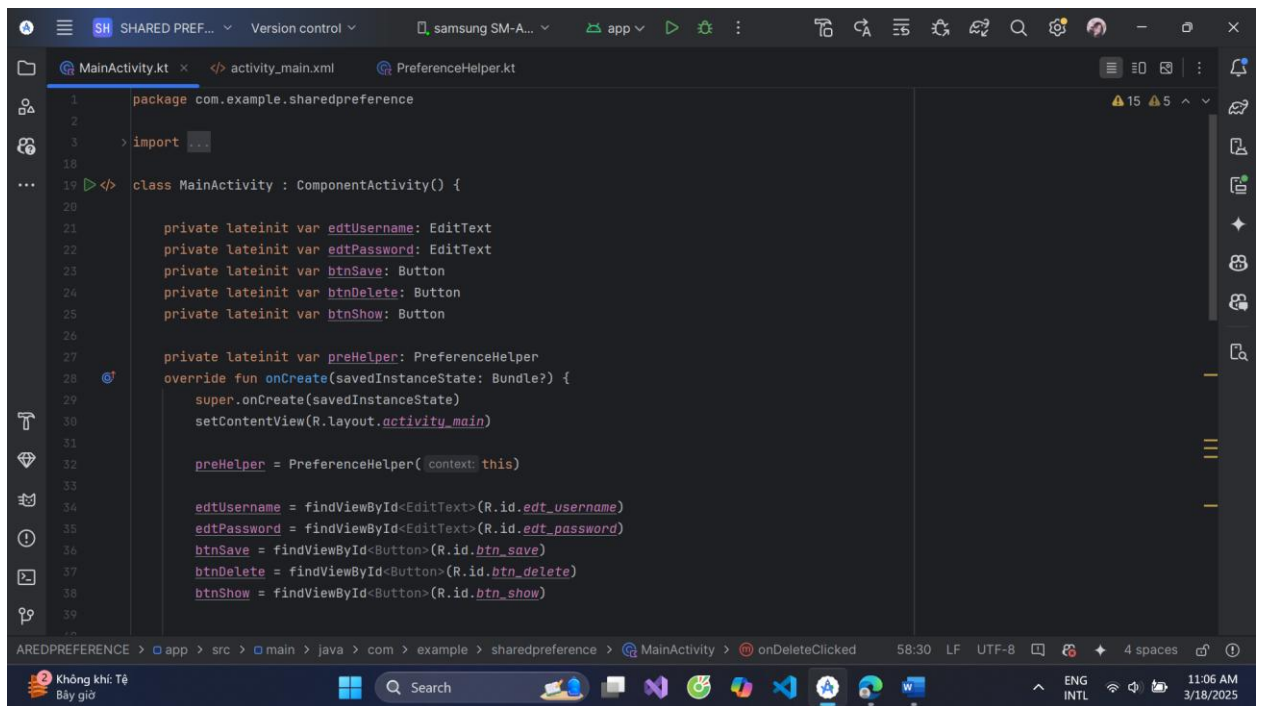
4. Kết quả

<<Sinh viên chụp Ảnh màn hình kết quả và mã nguồn chính tại đây>>

Tạo lớp PreferenceHelper

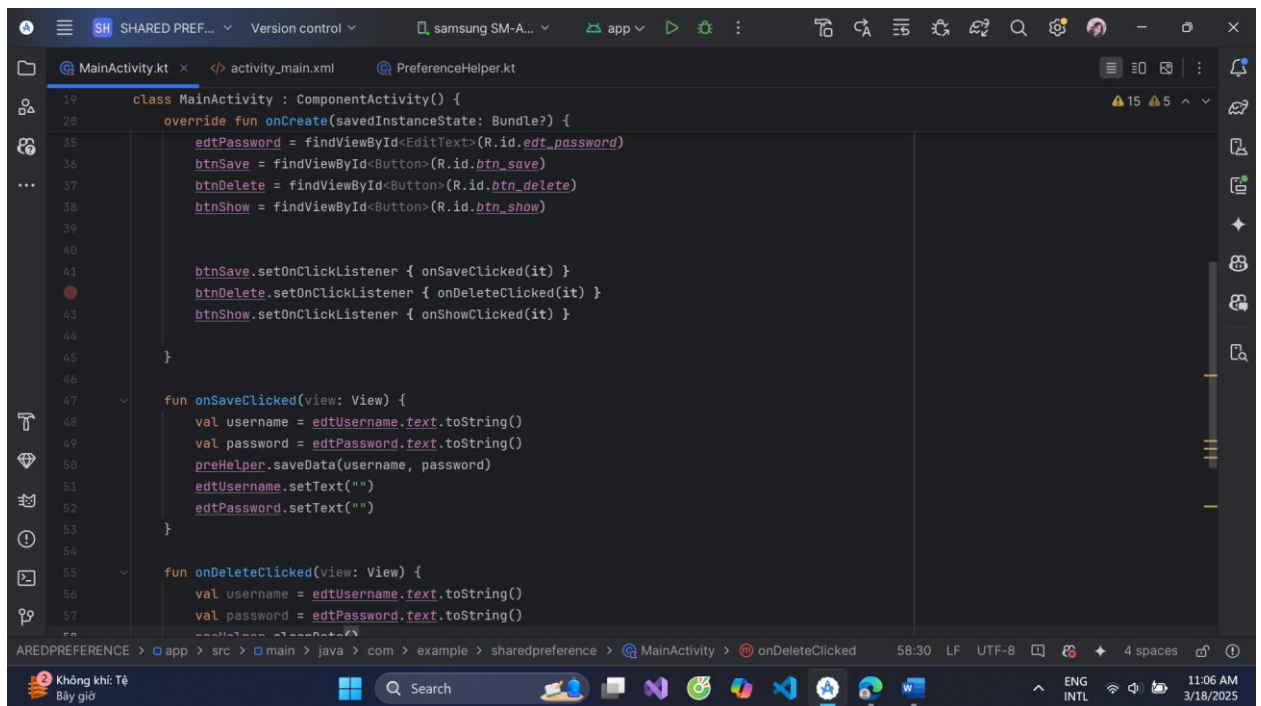


MainActivity:



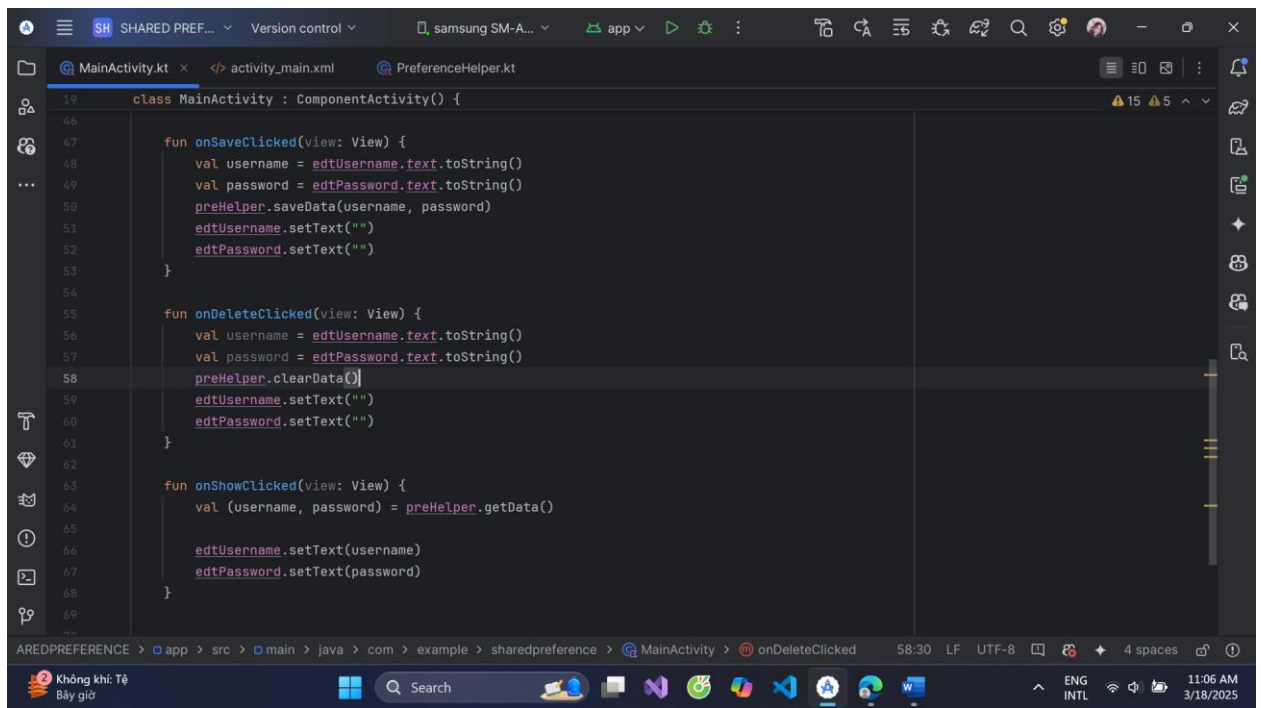
This screenshot shows the initial setup of MainActivity.kt in an Android Studio IDE. The code includes the package declaration, imports for SharedPreferences and PreferenceHelper, and the MainActivity class with its onCreate method. Fields for EditTexts, Buttons, and PreferenceHelper are declared.

```
1 package com.example.sharedpreference
2
3 import androidx.appcompat.app.AppCompatActivity
4 import androidx.appcompat.widget.Toolbar
5 import androidx.preference.PreferenceHelper
6
7 class MainActivity : AppCompatActivity() {
8
9     private lateinit var edtUsername: EditText
10     private lateinit var edtPassword: EditText
11     private lateinit var btnSave: Button
12     private lateinit var btnDelete: Button
13     private lateinit var btnShow: Button
14
15     private lateinit var preHelper: PreferenceHelper
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         setContentView(R.layout.activity_main)
20
21         preHelper = PreferenceHelper(context = this)
22
23         edtUsername = findViewById<EditText>(R.id.edt_username)
24         edtPassword = findViewById<EditText>(R.id.edt_password)
25         btnSave = findViewById<Button>(R.id.btn_save)
26         btnDelete = findViewById<Button>(R.id.btn_delete)
27         btnShow = findViewById<Button>(R.id.btn_show)
28     }
29 }
```



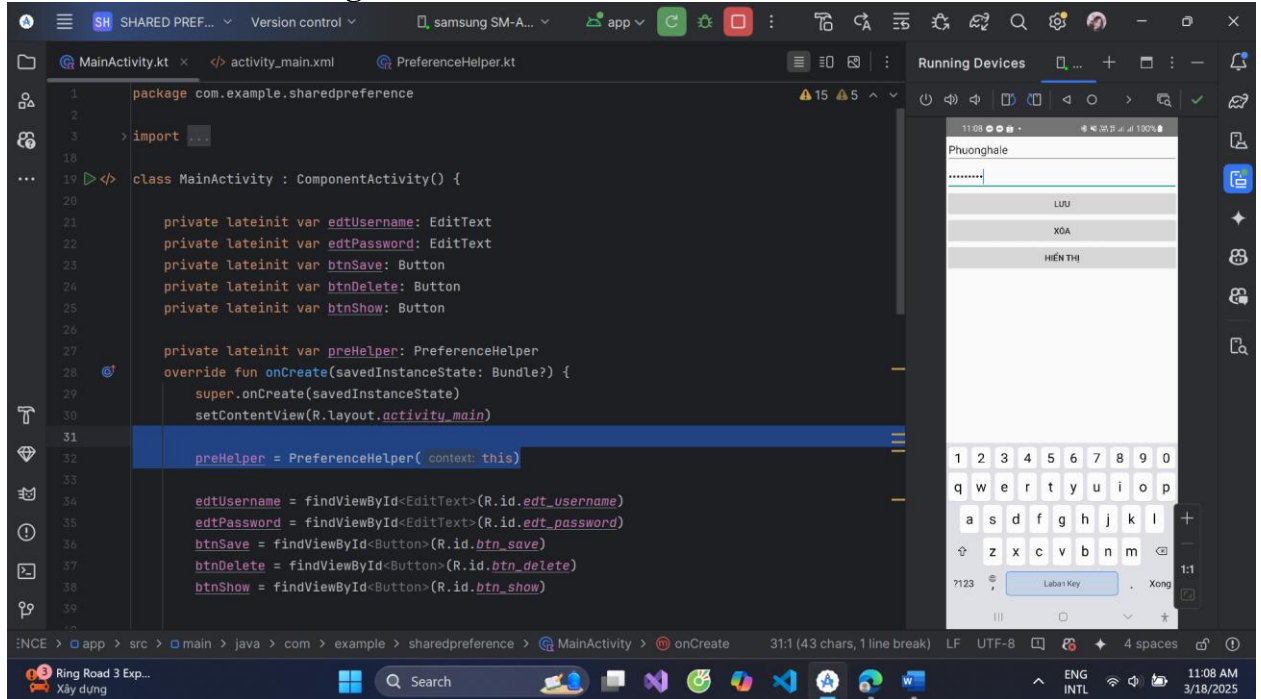
This screenshot shows the continuation of MainActivity.kt, adding click listeners for the buttons and implementing the onSaveClicked and onDeleteClicked methods. The onSaveClicked method saves the username and password to SharedPreferences and clears the input fields. The onDeleteClicked method clears the input fields.

```
29
30
31 btnSave.setOnClickListener { onSaveClicked(it) }
32 btnDelete.setOnClickListener { onDeleteClicked(it) }
33 btnShow.setOnClickListener { onShowClicked(it) }
34
35 fun onSaveClicked(view: View) {
36     val username = edtUsername.text.toString()
37     val password = edtPassword.text.toString()
38     preHelper.saveData(username, password)
39     edtUsername.setText("")
40     edtPassword.setText("")
41 }
42
43 fun onDeleteClicked(view: View) {
44     val username = edtUsername.text.toString()
45     val password = edtPassword.text.toString()
46     preHelper.deleteData()
47 }
```



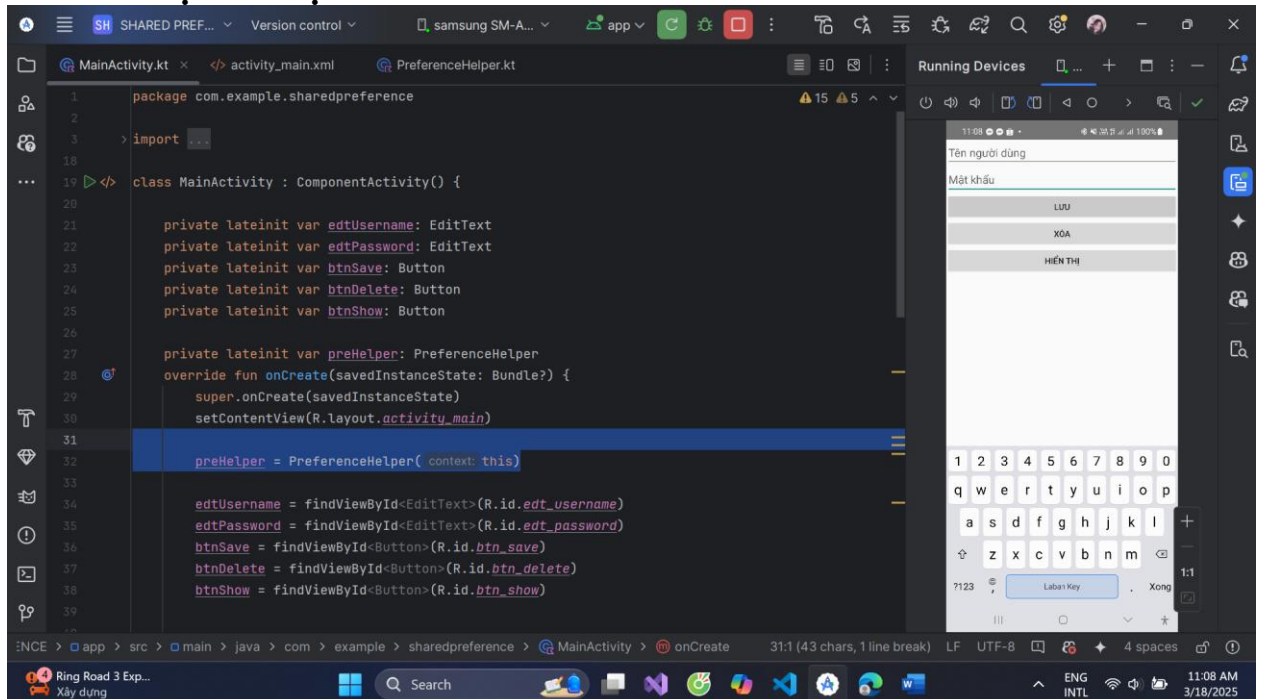
Kết quả:

1. Điền thông tin:



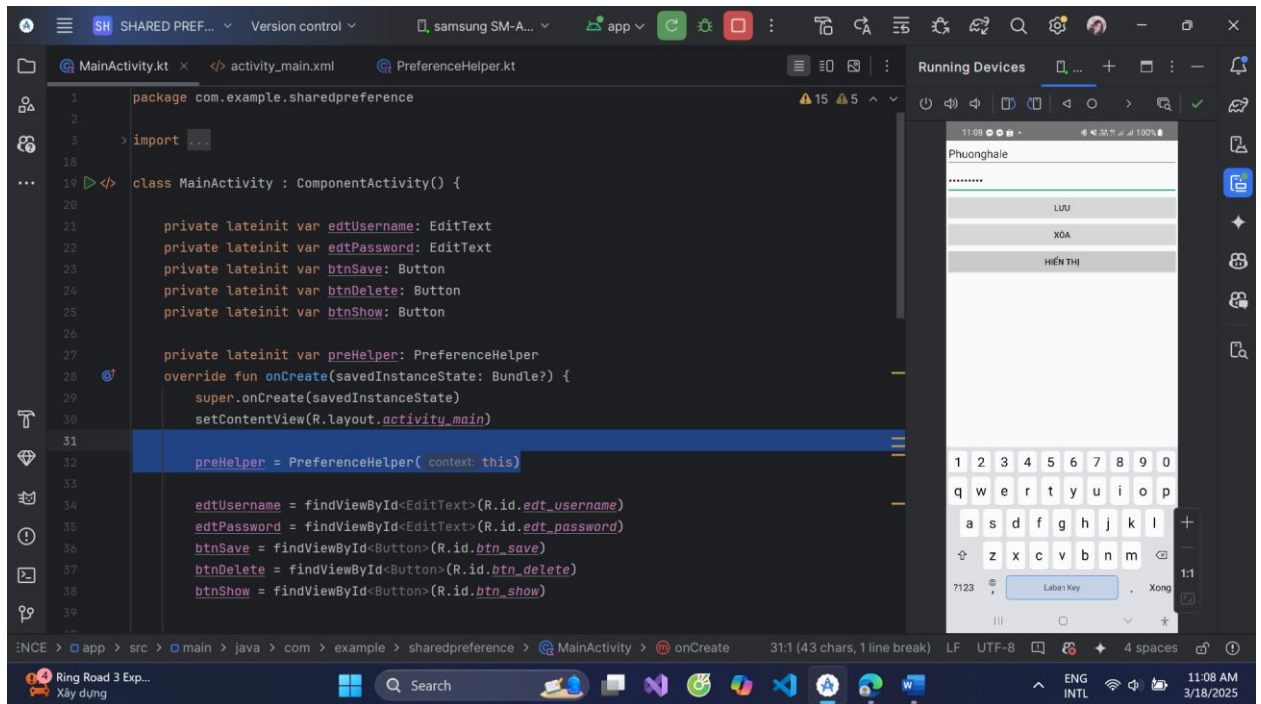
2. Ấn nút lưu:

- Dữ liệu đã được lưu và clear các editText



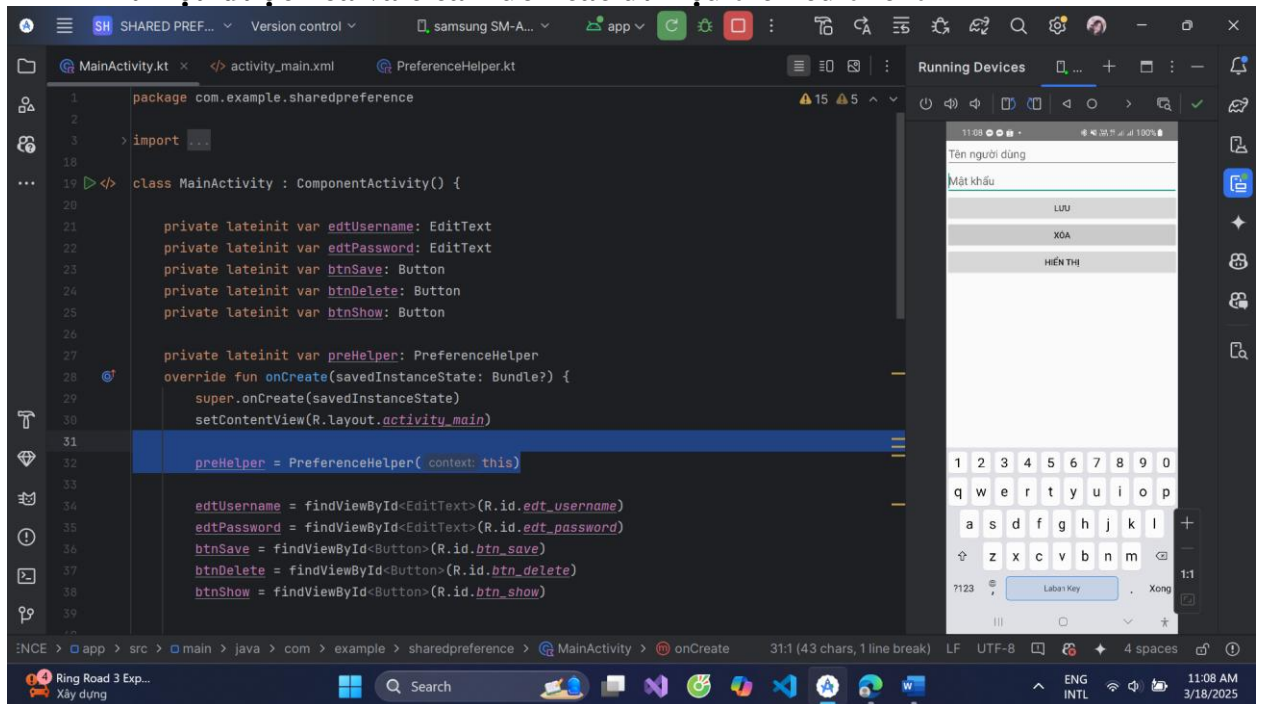
3. Ấn nút hiển thị

- Thông tin sẽ hiển thị lại lên trên editText sau khi lấy dữ liệu

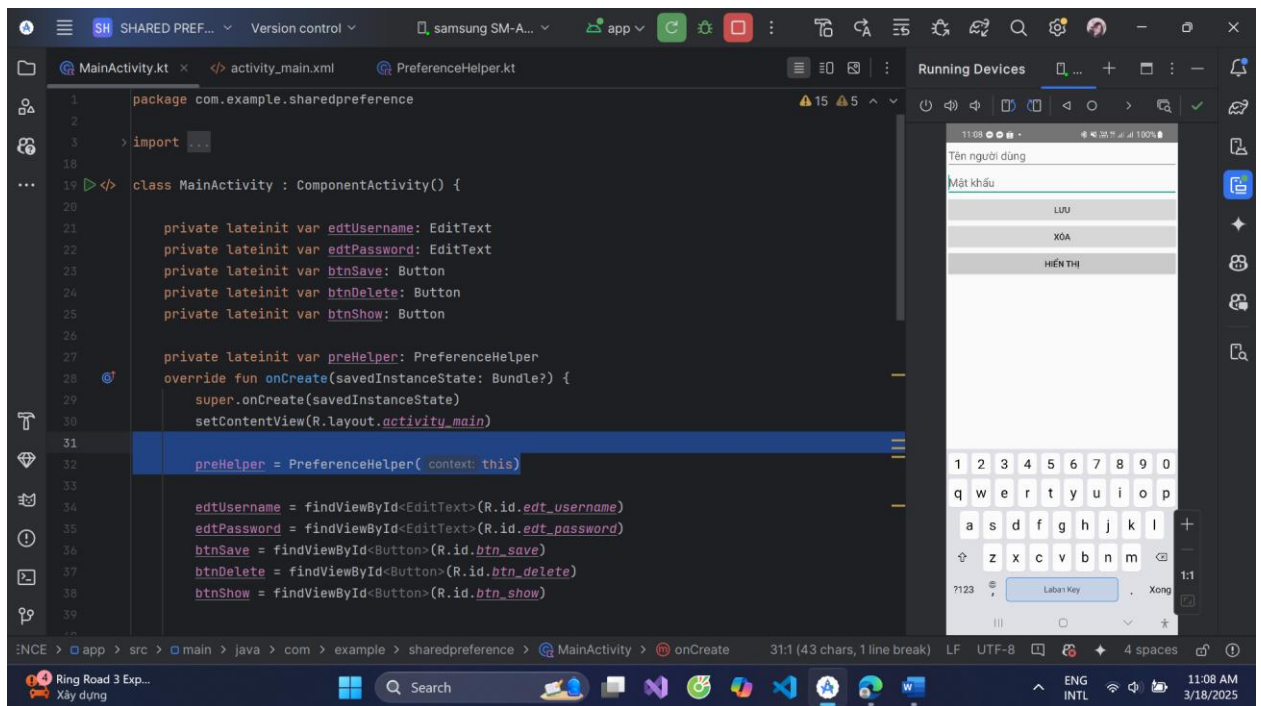


4. Ấn Xóa:

- Dữ liệu được xóa và clear luôn các dữ liệu trên editText



Kiểm tra dữ liệu còn không - Ấn nút Hiển thị lại một lần nữa:



- Không có gì hiển thị lên các editText nữa
⇒ Dữ liệu đã được xóa hoàn toàn

BÀI TẬP 2: SQLite

Mục tiêu:

- Hiểu cách sử dụng SQLite để lưu trữ dữ liệu trong ứng dụng Android.
- Thực hành tạo cơ sở dữ liệu SQLite, thêm, sửa, xóa dữ liệu.

Yêu cầu:

1. Tạo ứng dụng mới:

- Tạo một dự án Android mới bằng Kotlin.
- Thiết kế giao diện người dùng với hai trường nhập (EditText) cho tên và số điện thoại, và bốn nút bấm: "Thêm", "Sửa", "Xóa", và "Hiển thị".

2. Sử dụng SQLite:

- Tạo một lớp helper để quản lý cơ sở dữ liệu SQLite.
- Tạo bảng dữ liệu với hai cột: tên và số điện thoại.
- Viết các hàm để thêm, sửa, xóa dữ liệu từ cơ sở dữ liệu.
- Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ cơ sở dữ liệu và hiển thị lên màn hình.

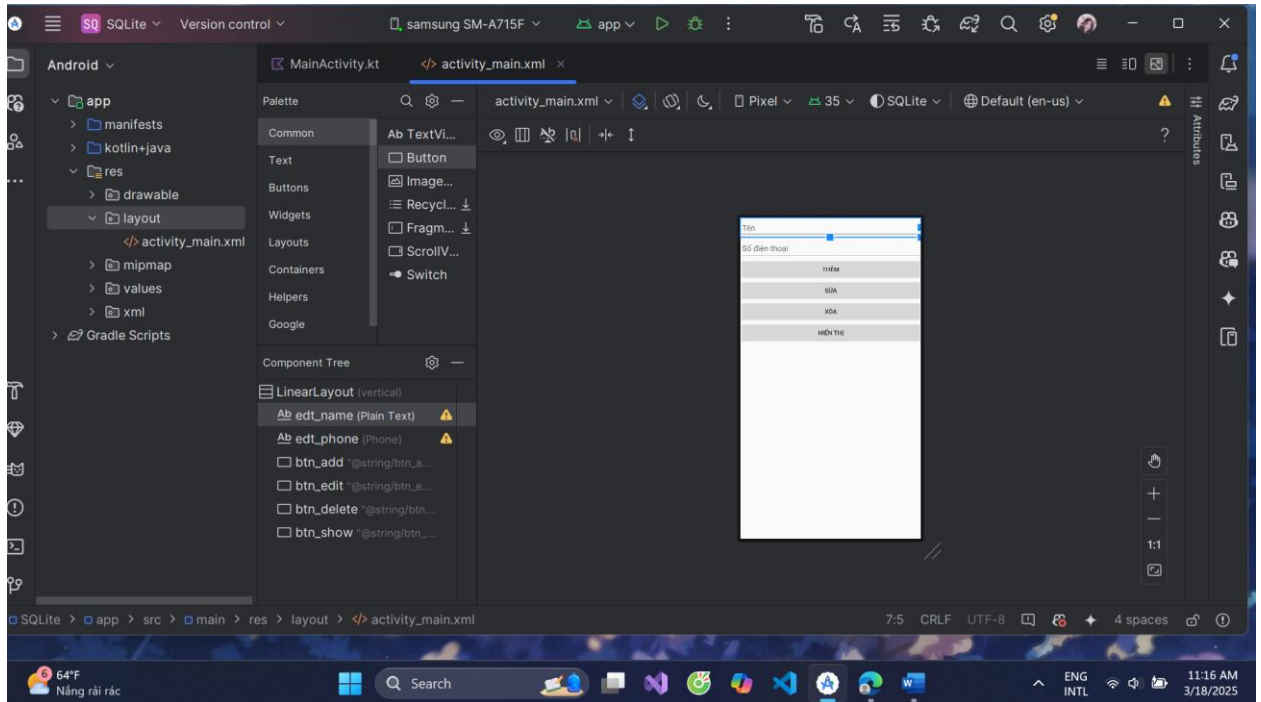
3. Thực hành:

- Viết mã Kotlin để thực hiện các chức năng trên.
- Sử dụng SQLiteOpenHelper để tạo và quản lý cơ sở dữ liệu.

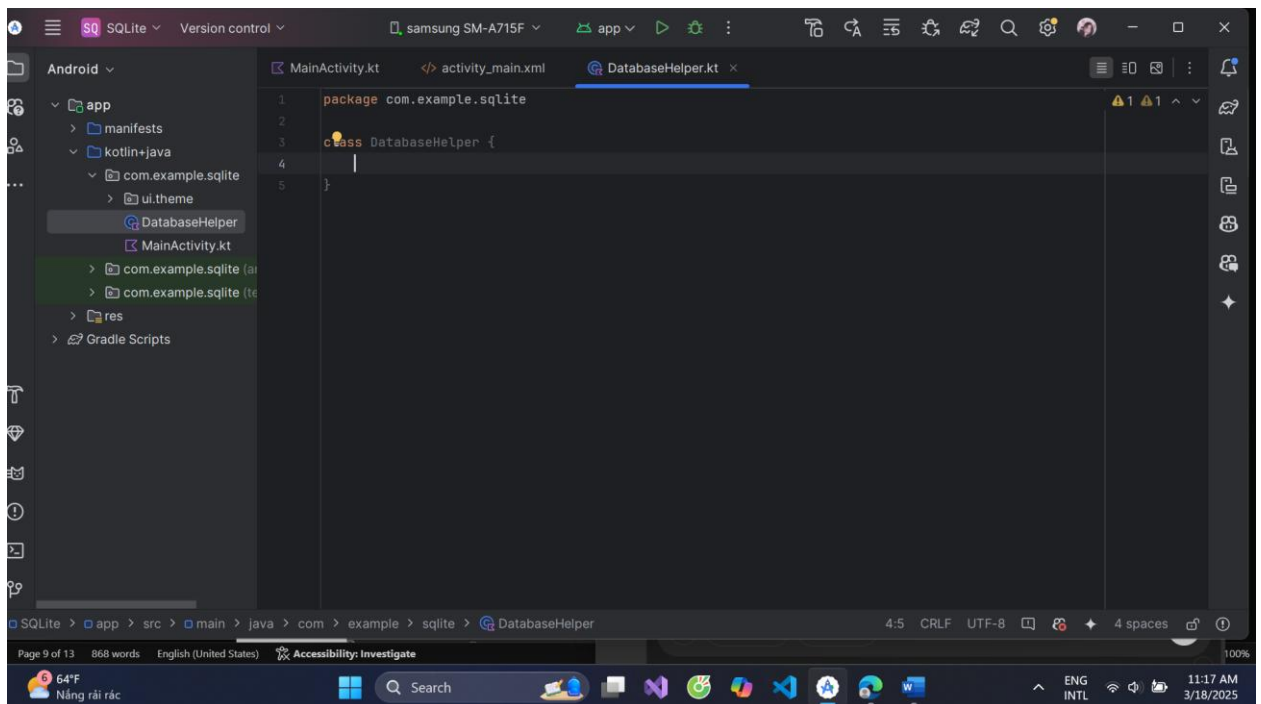
4. Kết quả

<<Sinh viên chụp Ảnh màn hình kết quả và mã nguồn chính tại đây>>

1. Tạo giao diện:



2. Tạo lớp SQLite Helper:



```
package com.example.sqlite

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_NAME = "mydatabase.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "nguoidung"
        private const val COLUMN_NAME = "name"
        private const val COLUMN_PHONE = "phone"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTableQuery = "CREATE TABLE $TABLE_NAME ($COLUMN_NAME TEXT PRIMARY KEY, $COLUMN_PHONE TEXT)"
        db?.execSQL(createTableQuery)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }
}
```

```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {

    override fun onCreate(db: SQLiteDatabase?) {
        val createTableQuery = "CREATE TABLE $TABLE_NAME ($COLUMN_NAME TEXT PRIMARY KEY, $COLUMN_PHONE TEXT)"
        db?.execSQL(createTableQuery)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun addData(name: String, phone: String): Boolean {
        val db = writableDatabase

        // Kiểm tra xem name đã tồn tại chưa
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE $COLUMN_NAME = ?", arrayOf(name))
        if (cursor.count > 0) {
            cursor.close()
            db.close()
            return false // Trả về false nếu đã tồn tại
        }
        cursor.close()

        val contentValues = ContentValues()
        contentValues.put(COLUMN_NAME, name)
        contentValues.put(COLUMN_PHONE, phone)
        db.insert(TABLE_NAME, null, contentValues)
        db.close()
        return true
    }
}
```

```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {
    fun addData(name: String, phone: String): Boolean {
        return false // Trả về false nếu đã tồn tại
    }
    cursor.close()

    val contentValues = ContentValues()
    contentValues.put(COLUMN_NAME, name)
    contentValues.put(COLUMN_PHONE, phone)

    val result = db.insert(TABLE_NAME, nullColumnHack: null, contentValues)
    db.close()

    return result != -1
}

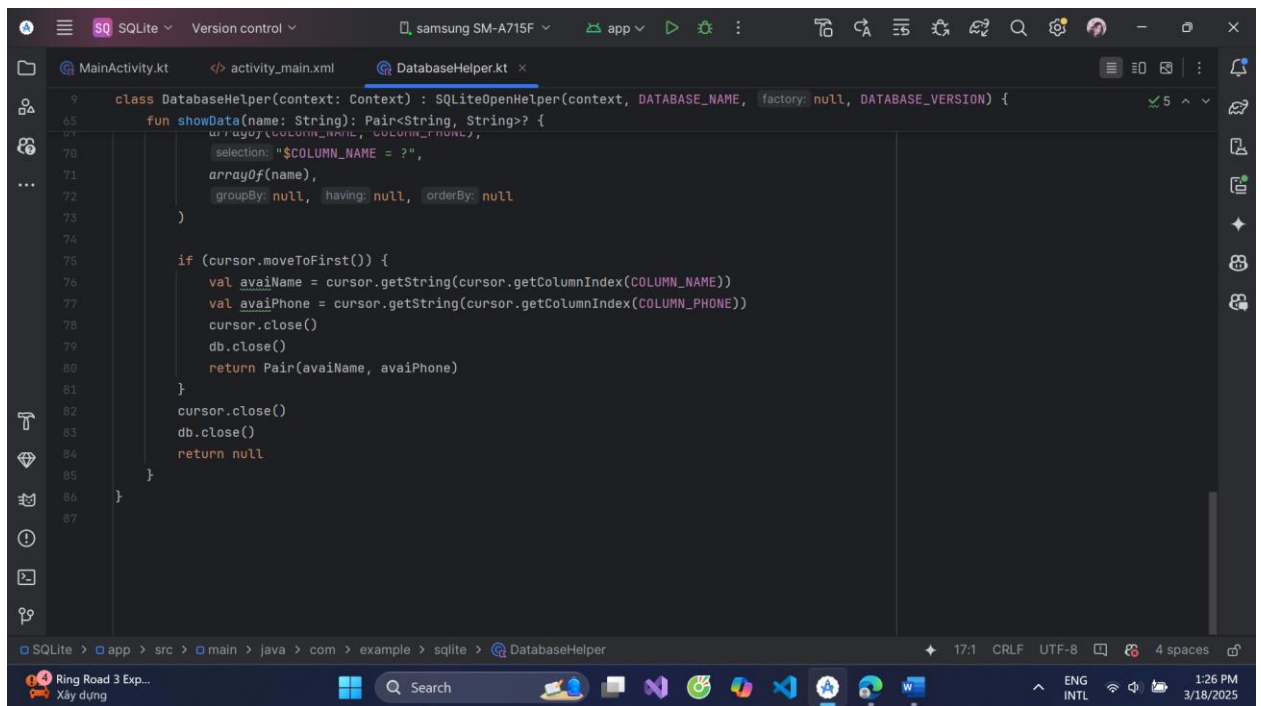
fun editData(name: String, phone: String) {
    val db = writableDatabase
    val contentValues = ContentValues()
    contentValues.put(COLUMN_PHONE, phone)
    db.update(TABLE_NAME, contentValues, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
}

fun deleteData(name: String) {
    val db = writableDatabase
    db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
}
```

```
fun deleteData(name: String) {
    val db = writableDatabase
    db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
}

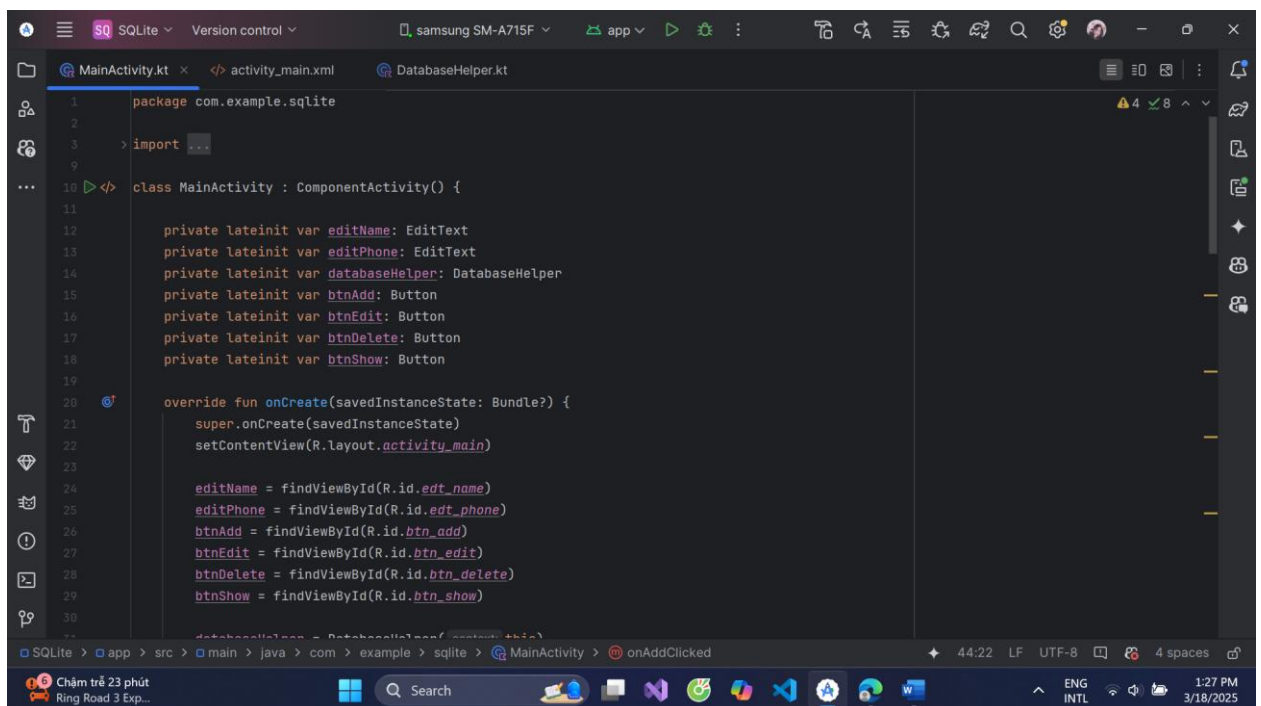
@SuppressLint("Recycle", "Range")
fun showData(name: String): Pair<String, String>? {
    val db = this.readableDatabase
    val cursor = db.query(
        TABLE_NAME,
        arrayOf(COLUMN_NAME, COLUMN_PHONE),
        selection: "$COLUMN_NAME = ?",
        arrayOf(name),
        groupBy: null, having: null, orderBy: null
    )

    if (cursor.moveToFirst()) {
        val avaiName = cursor.getString(cursor.getColumnIndex(COLUMN_NAME))
        val avaiPhone = cursor.getString(cursor.getColumnIndex(COLUMN_PHONE))
        cursor.close()
        db.close()
        return Pair(avaiName, avaiPhone)
    }
}
```



```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {  
    fun showData(name: String): Pair<String, String>? {  
        val cursor = db.query(DATABASE_NAME, COLUMN_NAME, COLUMN_PHONE, selection: "$COLUMN_NAME = ?", arrayOf(name),  
            orderBy: null, having: null, null)  
        if (cursor.moveToFirst()) {  
            val avaiName = cursor.getString(cursor.getColumnIndex(COLUMN_NAME))  
            val avaiPhone = cursor.getString(cursor.getColumnIndex(COLUMN_PHONE))  
            cursor.close()  
            db.close()  
            return Pair(avaiName, avaiPhone)  
        }  
        cursor.close()  
        db.close()  
        return null  
    }  
}
```

3. MainActivity:



```
package com.example.sqlite  
  
import androidx.appcompat.app.AppCompatActivity  
  
class MainActivity : AppCompatActivity() {  
    private lateinit var editName: EditText  
    private lateinit var editPhone: EditText  
    private lateinit var databaseHelper: DatabaseHelper  
    private lateinit var btnAdd: Button  
    private lateinit var btnEdit: Button  
    private lateinit var btnDelete: Button  
    private lateinit var btnShow: Button  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        editName = findViewById(R.id.edt_name)  
        editPhone = findViewById(R.id.edt_phone)  
        btnAdd = findViewById(R.id.btn_add)  
        btnEdit = findViewById(R.id.btn_edit)  
        btnDelete = findViewById(R.id.btn_delete)  
        btnShow = findViewById(R.id.btn_show)  
  
        databaseHelper = DatabaseHelper(this)  
    }  
}
```



```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        btnShow = findViewById(R.id.btn_show)

        databaseHelper = DatabaseHelper(context: this)

        btnAdd.setOnClickListener { onAddClicked(it) }
        btnEdit.setOnClickListener { onEditClicked(it) }
        btnDelete.setOnClickListener { onDeleteClicked(it) }
        btnShow.setOnClickListener { onShowClicked(it) }
    }

    fun onAddClicked(view: View) {
        val name = editName.text.toString().trim()
        val phone = editPhone.text.toString().trim()

        val result = databaseHelper.addData(name, phone)
        if (result) {
            Toast.makeText(context: this, text: "Thêm dữ liệu thành công!", Toast.LENGTH_SHORT).show()
            editName.setText("")
            editPhone.setText("")
        } else {
            Toast.makeText(context: this, text: "Thêm dữ liệu thất bại! Tên đã tồn tại.", Toast.LENGTH_SHORT).show()
        }
    }
}
```

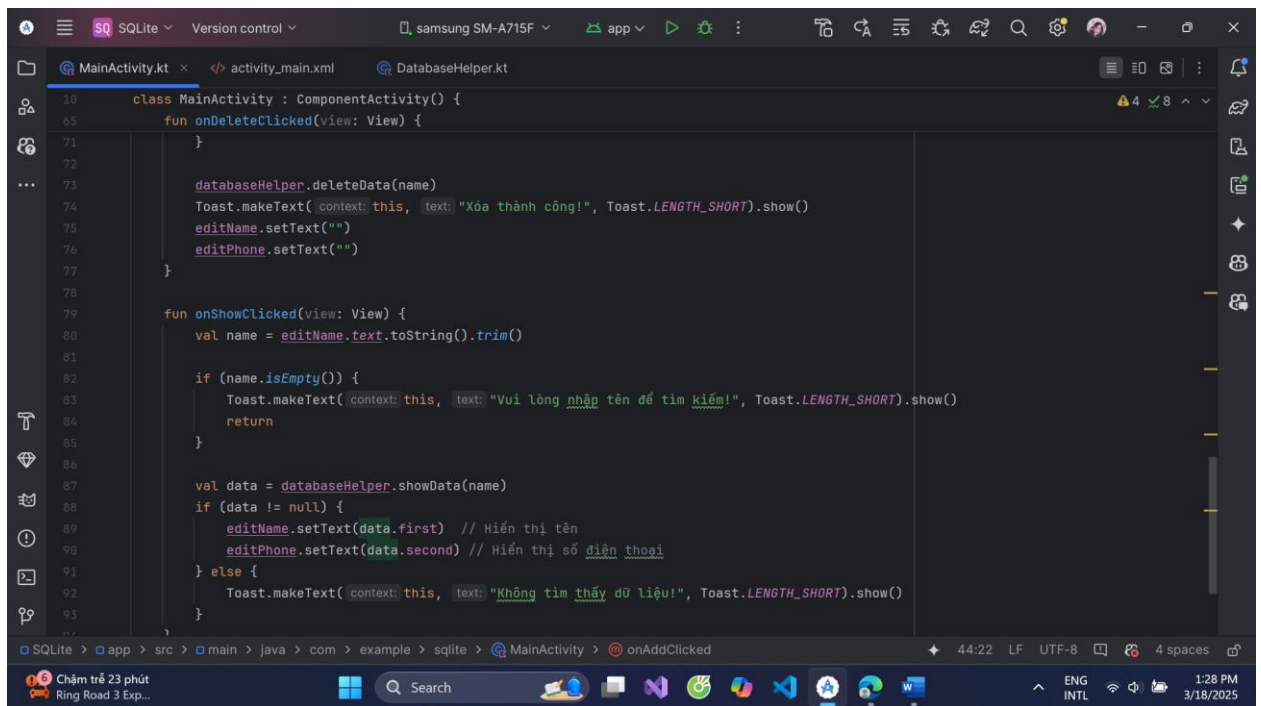
```
class MainActivity : AppCompatActivity() {
    fun onEditClicked(view: View) {
        val name = editName.text.toString().trim()
        val phone = editPhone.text.toString().trim()

        databaseHelper.editData(name, phone)
        Toast.makeText(context: this, text: "Cập nhật thành công!", Toast.LENGTH_SHORT).show()
        editName.setText("")
        editPhone.setText("")
    }

    fun onDeleteClicked(view: View) {
        val name = editName.text.toString().trim()

        if (name.isEmpty()) {
            Toast.makeText(context: this, text: "Vui lòng nhập tên để xóa!", Toast.LENGTH_SHORT).show()
            return
        }

        databaseHelper.deleteData(name)
        Toast.makeText(context: this, text: "Xóa thành công!", Toast.LENGTH_SHORT).show()
        editName.setText("")
        editPhone.setText("")
    }
}
```

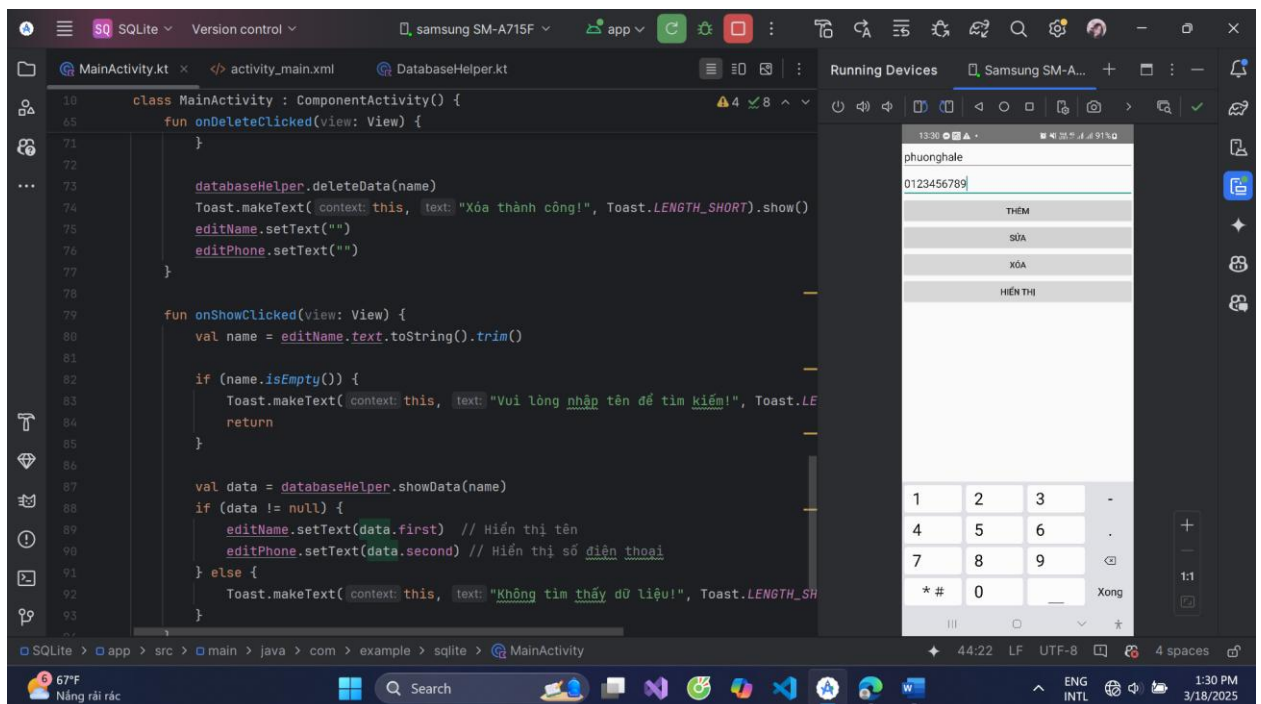



```
10 class MainActivity : AppCompatActivity() {
65     fun onDeleteClicked(view: View) {
71     }
72
73     databaseHelper.deleteData(name)
74     Toast.makeText(context, this, text: "Xóa thành công!", Toast.LENGTH_SHORT).show()
75     editName.setText("")
76     editPhone.setText("")
77 }
78
79 fun onShowClicked(view: View) {
80     val name = editName.text.toString().trim()
81
82     if (name.isEmpty()) {
83         Toast.makeText(context, this, text: "Vui lòng nhập tên để tìm kiếm!", Toast.LENGTH_SHORT).show()
84         return
85     }
86
87     val data = databaseHelper.showData(name)
88     if (data != null) {
89         editName.setText(data.first) // Hiển thị tên
90         editPhone.setText(data.second) // Hiển thị số điện thoại
91     } else {
92         Toast.makeText(context, this, text: "Không tìm thấy dữ liệu!", Toast.LENGTH_SHORT).show()
93     }
94 }
```

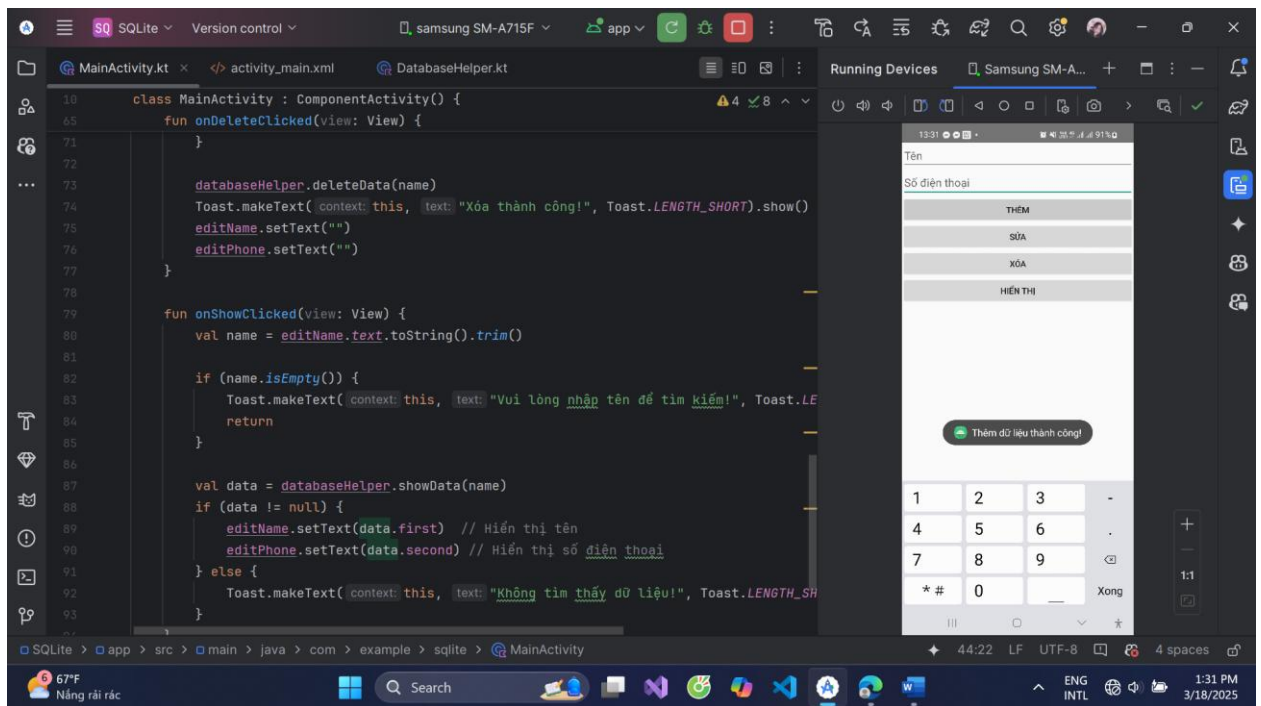
4. Kết quả:

4.1 Điền thông tin để thêm:

- Trong table em để name (tên) để làm khóa chính.



4.2 Ấn nút thêm:

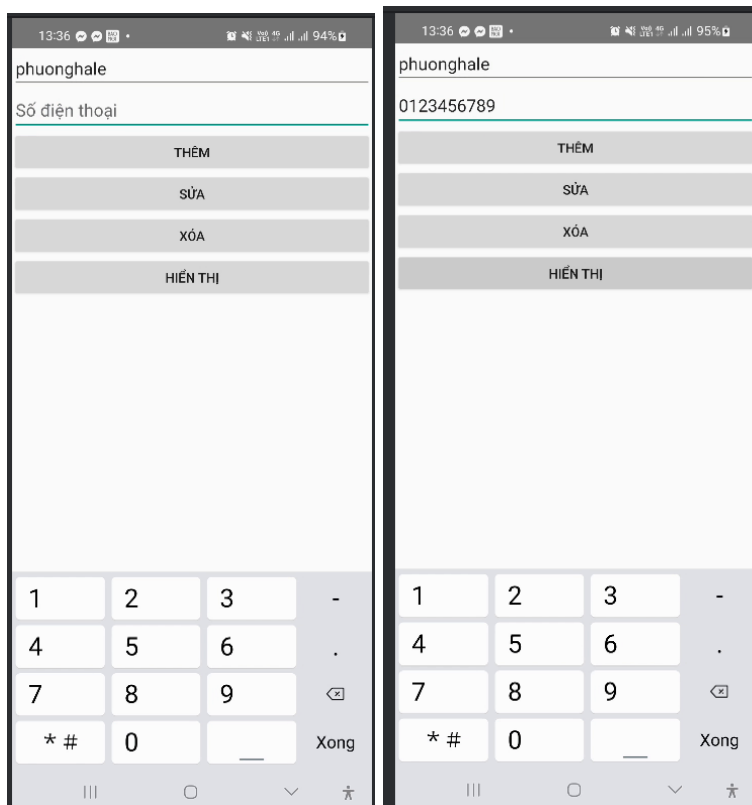


⇒ Dữ liệu đã được thêm thành công đồng thời clear dữ liệu trên EditText sau khi thêm

4.2 Ấn nút Hiển thị:

- Ở đây nút Hiển thị sẽ hoạt động như sau:

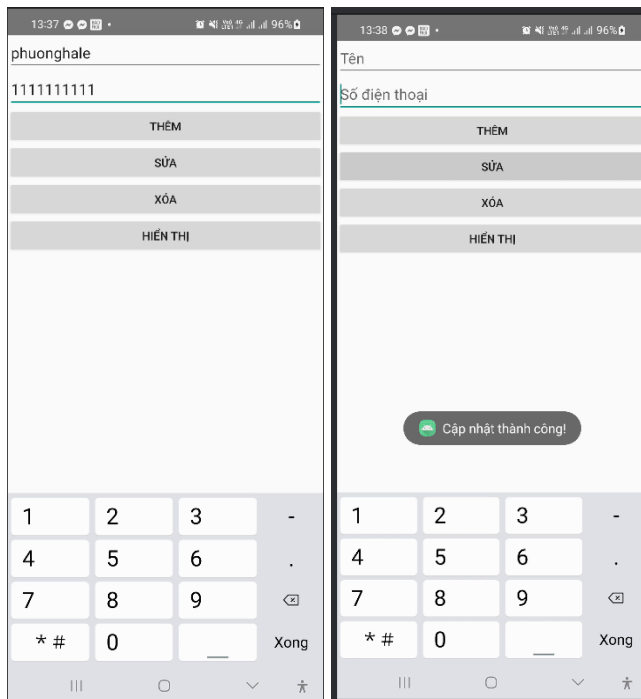
+ Vì ban đầu name là khóa chính cho nên name sẽ không thể sửa, em sẽ điền name vào phần tên để truy xuất dữ liệu trong csdl, sau đó ấn nút Hiển thị thì dữ liệu sẽ hiển thị lên các editText



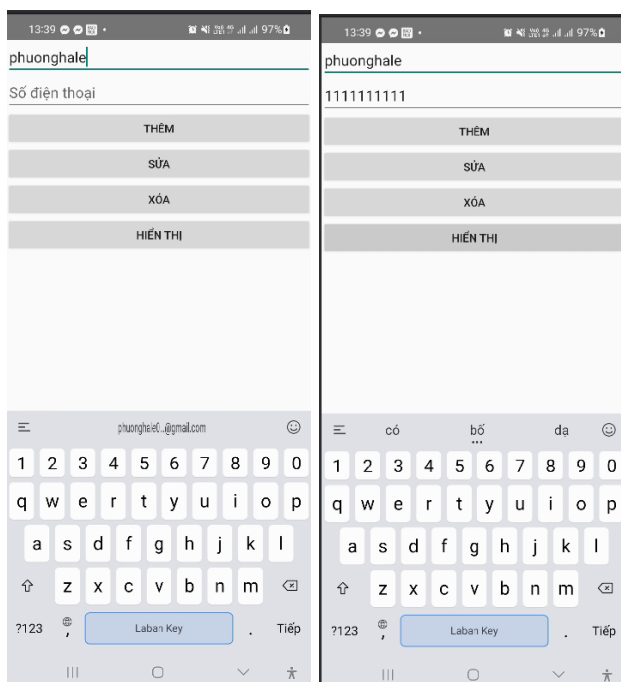
4.3 Nút sửa:

- Ở đây nút sửa sẽ hoạt động như sau:

+ Vì ban đầu name là khóa chính cho nên name sẽ không thể sửa, em sẽ điền name vào phần tên để truy xuất dữ liệu trong csdl, sau đó điền số điện thoại mới vào và ấn nút sửa thì dữ liệu sẽ được cập nhật theo khóa chính



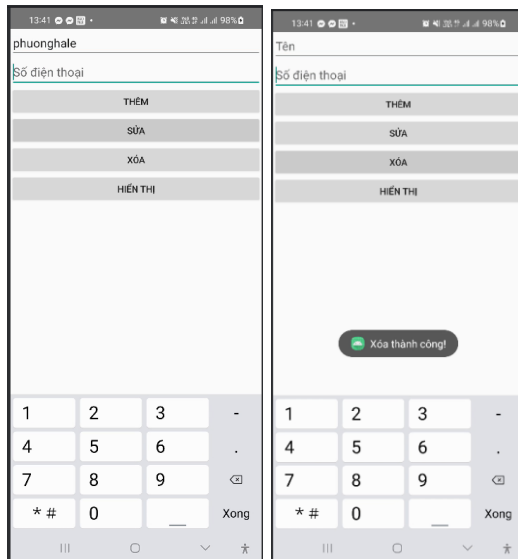
- Kiểm tra lại xem đã cập nhật thành công thực sự chưa em sẽ dùng nút Hiển thị:



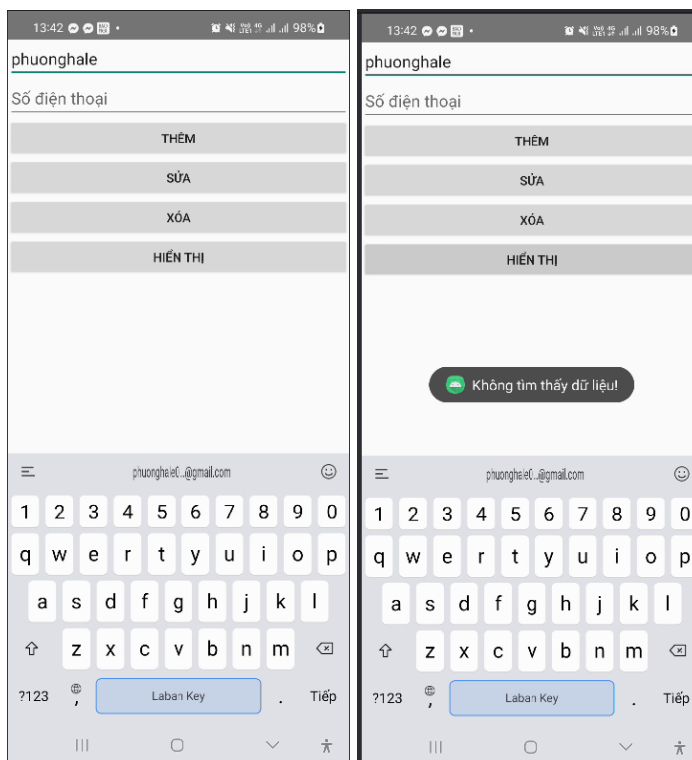
=> Vậy là nút sửa đã cập nhật thành công

4.4 Nút xóa:

- Tương tự như các nút Hiển thị và Sửa thì em cũng điền name để truy xuất dữ liệu
- Sau đó ấn nút xóa để xóa khỏi csdl theo khóa chính



- Kiểm tra lại bằng nút hiển thị:



⇒ Nút hiển thị không tìm được dữ liệu tức là đã xóa thành công

BÀI TẬP 3: HỆ SINH THÁI FIREBASE

Mục tiêu:

- Hiểu rõ về các dịch vụ chính của Firebase.
- Biết cách tích hợp Firebase vào dự án phát triển ứng dụng.

Yêu cầu:

1. Tìm hiểu các dịch vụ chính của Firebase:

- Firebase Authentication: Xác thực người dùng.
- Firebase Realtime Database và Cloud Firestore: Cơ sở dữ liệu thời gian thực và NoSQL.
- Firebase Cloud Functions: Chạy mã backend serverless.
- Firebase Cloud Messaging (FCM): Gửi thông báo đẩy.
- Firebase Storage: Lưu trữ tệp tin trên đám mây.
- Firebase Machine Learning (ML): Tích hợp trí tuệ nhân tạo vào ứng dụng.

2. Viết báo cáo:

- Giới thiệu tổng quan về Firebase và lịch sử phát triển.
- Mô tả chi tiết từng dịch vụ chính của Firebase.
- Thảo luận về lợi ích và ứng dụng của Firebase trong phát triển ứng dụng.

Nội dung báo cáo viết ở đây

1. Giới thiệu tổng quan về Firebase và lịch sử phát triển

Firebase là một nền tảng phát triển ứng dụng di động và web được Google mua lại và phát triển. Ban đầu, Firebase được thành lập vào năm 2011 bởi Envolv như một nền tảng nhắn tin thời gian thực. Sau đó, Firebase chuyển hướng sang cung cấp dịch vụ cơ sở dữ liệu thời gian thực và nhanh chóng được Google mua lại vào năm 2014. Kể từ đó, Firebase được mở rộng với nhiều tính năng hỗ trợ nhà phát triển.

2. Mô tả chi tiết từng dịch vụ chính của Firebase

Firebase cung cấp nhiều dịch vụ hữu ích cho nhà phát triển, bao gồm:

- Firebase Authentication: Hỗ trợ xác thực người dùng với các phương thức như email, Google, Facebook, v.v.
- Cloud Firestore: Cơ sở dữ liệu NoSQL thời gian thực, linh hoạt và mở rộng.

- Realtime Database: Lưu trữ dữ liệu thời gian thực và đồng bộ trên nhiều thiết bị.
- Firebase Cloud Messaging (FCM): Gửi thông báo push miễn phí tới các thiết bị.
- Firebase Hosting: Lưu trữ và triển khai các trang web tĩnh nhanh chóng.
- Firebase Crashlytics: Giúp phát hiện và theo dõi lỗi trong ứng dụng.
- Firebase Performance Monitoring: Theo dõi và tối ưu hiệu suất của ứng dụng.

3. Thảo luận về lợi ích và ứng dụng của Firebase trong phát triển ứng dụng

Firebase mang lại nhiều lợi ích cho nhà phát triển:

- Tiết kiệm thời gian: Firebase có sẵn các dịch vụ backend giúp nhà phát triển tập trung vào frontend.
- Dễ dàng tích hợp: Firebase hỗ trợ tích hợp với nhiều ngôn ngữ và nền tảng như Android, iOS, và web.
- Báo cáo và theo dõi tự động: Firebase cung cấp các công cụ giúp theo dõi lỗi, hiệu suất, và hành vi người dùng.
- Hỗ trợ quy mô linh hoạt: Firebase có thể phát triển từ dự án nhỏ đến các hệ thống lớn.

Nhờ những tính năng và lợi ích này, Firebase trở thành một lựa chọn lý tưởng cho nhà phát triển khi xây dựng và quản lý ứng dụng di động và web.

3. Thực hành:

- Tạo một dự án Firebase mới trên Firebase Console.
- Đăng ký ứng dụng Android vào dự án Firebase.
- Sử dụng ít nhất hai dịch vụ của Firebase trong dự án (ví dụ: Authentication và Realtime Database).

Bài tập cụ thể: Tích hợp Firebase Authentication và Realtime Database

Yêu cầu:

1. Tạo ứng dụng mới:

- Tạo một dự án Android mới bằng Kotlin.
- Thiết kế giao diện người dùng với hai trường nhập (EditText) cho email và mật khẩu, và ba nút bấm: "Đăng ký", "Đăng nhập", và "Hiển thị dữ liệu".

2. Tích hợp Firebase Authentication:

- Sử dụng Firebase Authentication để cho phép người dùng đăng ký và đăng nhập bằng email và mật khẩu.
- Viết mã để xử lý các sự kiện đăng ký và đăng nhập thành công hoặc thất bại.

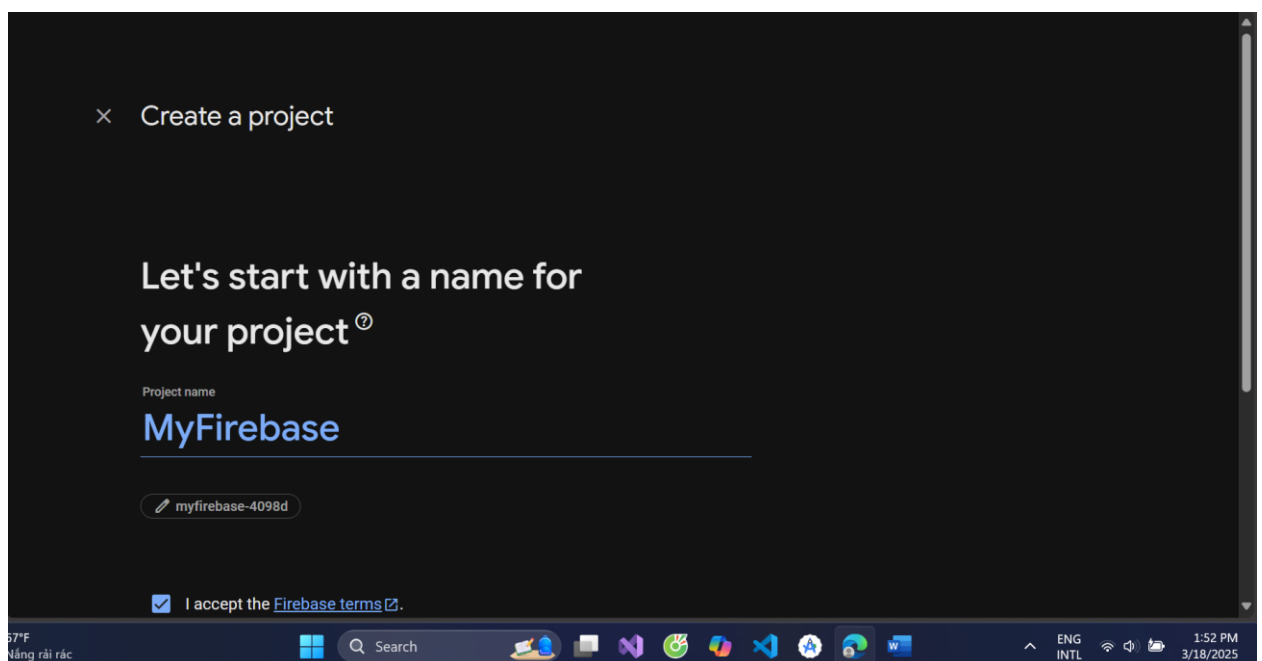
3. Tích hợp Firebase Realtime Database:

- Sau khi người dùng đăng nhập thành công, lưu trữ thông tin người dùng vào Firebase Realtime Database.
- Khi người dùng nhấn nút "Hiển thị dữ liệu", đọc dữ liệu từ Firebase Realtime Database và hiển thị lên màn hình.

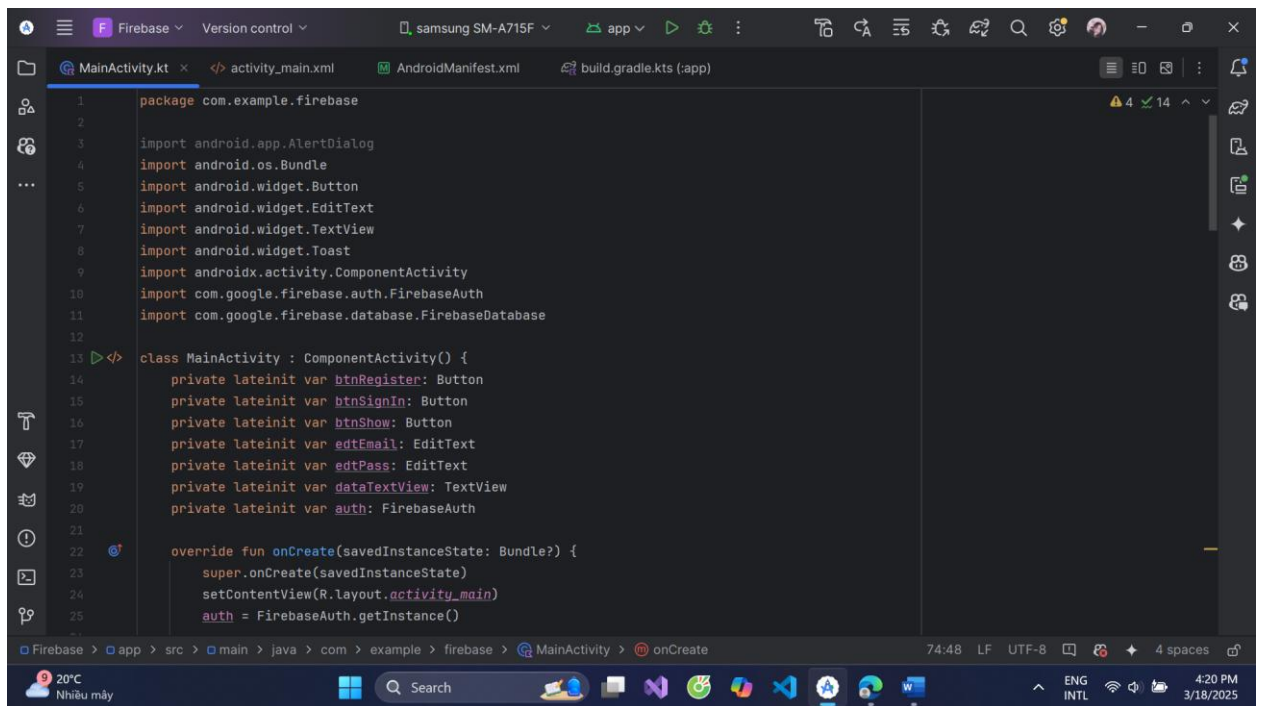
4. Kết quả

<<Sinh viên chụp Ảnh màn hình kết quả và mã nguồn chính tại đây>>

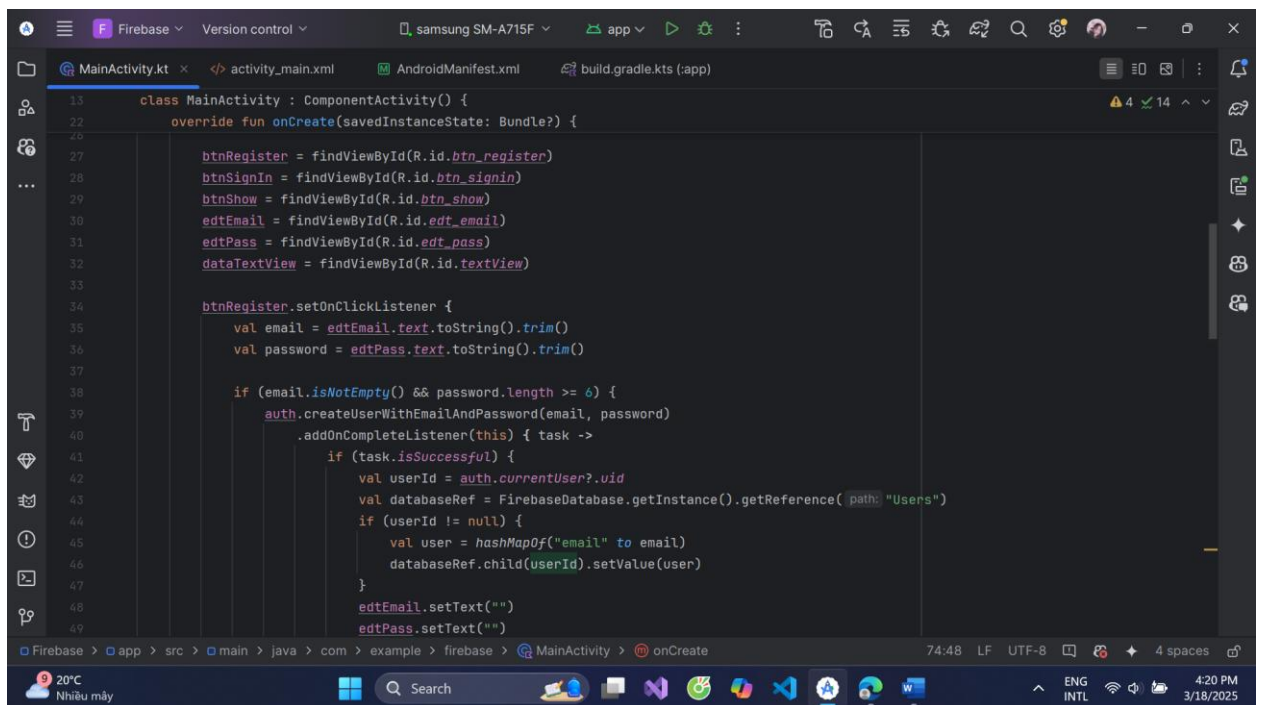
1. Tạo dự án Firebase:



2. MainActivity:



```
1 package com.example.firebase
2
3
4 import android.app.AlertDialog
5 import android.os.Bundle
6 import android.widget.Button
7 import android.widget.EditText
8 import android.widget.TextView
9 import android.widget.Toast
10 import androidx.activity.ComponentActivity
11 import com.google.firebase.auth.FirebaseAuth
12 import com.google.firebase.database.FirebaseDatabase
13
14 class MainActivity : ComponentActivity() {
15     private lateinit var btnRegister: Button
16     private lateinit var btnSignIn: Button
17     private lateinit var btnShow: Button
18     private lateinit var edtEmail: EditText
19     private lateinit var edtPass: EditText
20     private lateinit var dataTextView: TextView
21     private lateinit var auth: FirebaseAuth
22
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContentView(R.layout.activity_main)
26         auth = FirebaseAuth.getInstance()
27     }
28 }
```



```
27     btnRegister = findViewById(R.id.btn_register)
28     btnSignIn = findViewById(R.id.btn_signin)
29     btnShow = findViewById(R.id.btn_show)
30     edtEmail = findViewById(R.id.edt_email)
31     edtPass = findViewById(R.id.edt_pass)
32     dataTextView = findViewById(R.id.textview)
33
34     btnRegister.setOnClickListener {
35         val email = edtEmail.text.toString().trim()
36         val password = edtPass.text.toString().trim()
37
38         if (email.isNotEmpty() && password.length >= 6) {
39             auth.createUserWithEmailAndPassword(email, password)
40                 .addOnCompleteListener(this) { task ->
41                     if (task.isSuccessful) {
42                         val userId = auth.currentUser?.uid
43                         val databaseRef = FirebaseDatabase.getInstance().getReference("Users")
44                         if (userId != null) {
45                             val user = hashMapOf("email" to email)
46                             databaseRef.child(userId).setValue(user)
47                         }
48                         edtEmail.setText("")
49                         edtPass.setText("")
50                     }
51                 }
52         }
53     }
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        btnRegister.setOnClickListener {
            .addOnCompleteListener(this) { task ->
                val user = hashMapOf("email" to email)
                databaseRef.child(userId).setValue(user)
            }
            edtEmail.setText("")
            edtPass.setText("")
            Toast.makeText(context, this, text: "Đăng ký thành công!", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(
                context: this,
                text: "Đăng ký thất bại: ${task.exception?.message}",
                Toast.LENGTH_SHORT
            ).show()
        }
    } else {
        Toast.makeText(context, this, text: "Email không hợp lệ hoặc mật khẩu quá ngắn!", Toast.LENGTH_SHORT).show()
    }
}

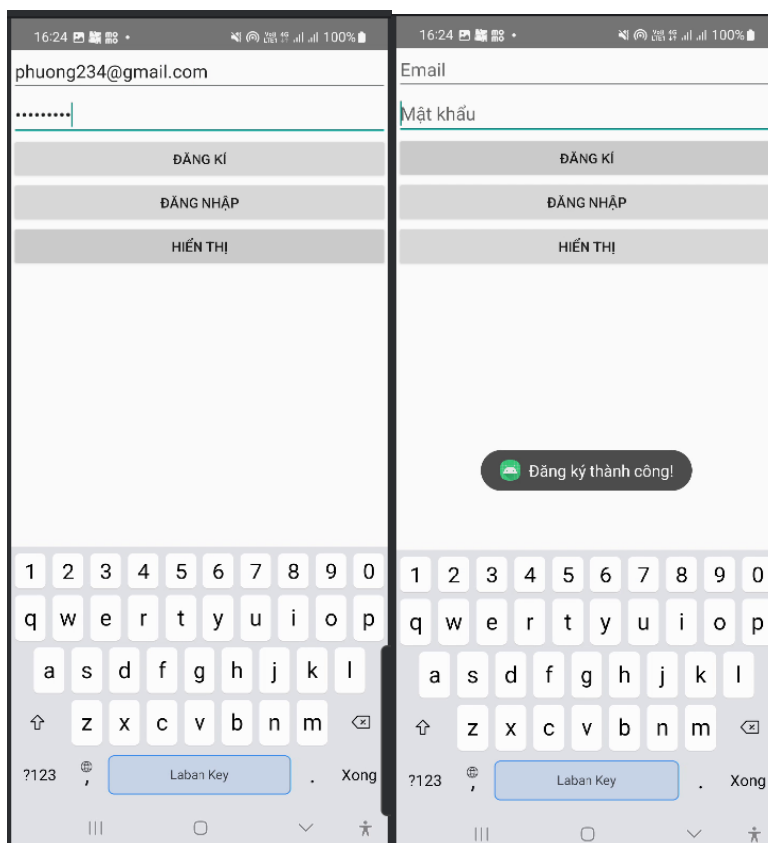
btnSignIn.setOnClickListener {
    val email = edtEmail.text.toString().trim()
    val password = edtPass.text.toString().trim()
    if (email.isNotEmpty() && password.isNotEmpty()) {
        auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                Toast.makeText(context, this, text: "Đăng nhập thành công!", Toast.LENGTH_SHORT).show()
                edtEmail.setText("")
                edtPass.setText("")
            } else {
                Toast.makeText(
                    context: this,
                    text: "Đăng nhập thất bại: ${task.exception?.message}",
                    Toast.LENGTH_SHORT
                ).show()
            }
        }
    } else {
        Toast.makeText(context, this, text: "Vui lòng nhập email và mật khẩu!", Toast.LENGTH_SHORT).show()
    }
}
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        btnSignIn.setOnClickListener {
            val email = edtEmail.text.toString().trim()
            val password = edtPass.text.toString().trim()
            if (email.isNotEmpty() && password.isNotEmpty()) {
                auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this) { task ->
                    if (task.isSuccessful) {
                        Toast.makeText(context, this, text: "Đăng nhập thành công!", Toast.LENGTH_SHORT).show()
                        edtEmail.setText("")
                        edtPass.setText("")
                    } else {
                        Toast.makeText(
                            context: this,
                            text: "Đăng nhập thất bại: ${task.exception?.message}",
                            Toast.LENGTH_SHORT
                        ).show()
                    }
                }
            } else {
                Toast.makeText(context, this, text: "Vui lòng nhập email và mật khẩu!", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

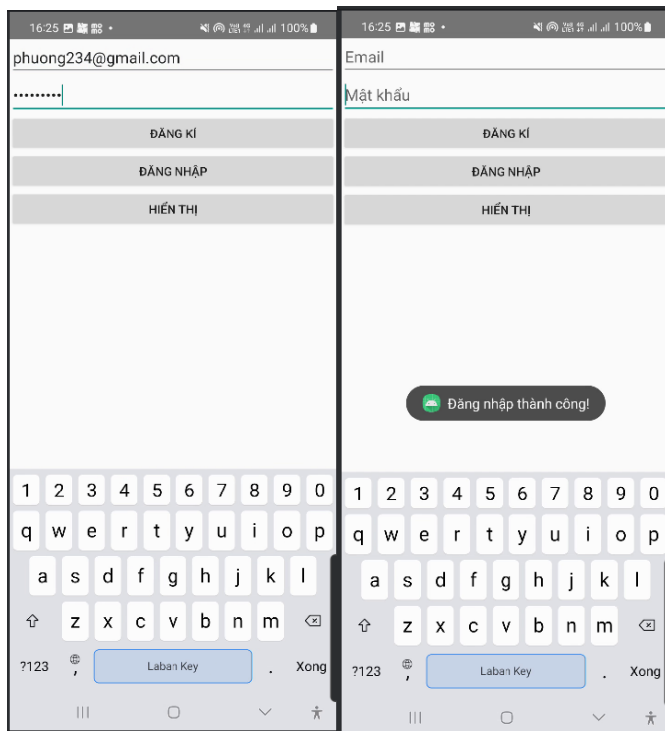
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        btnSignIn.setOnClickListener {
            // ...
        }
        btnShow.setOnClickListener {
            val email = edtEmail.text.toString().trim()

            if (email.isNotEmpty()) {
                val currentUser = auth.currentUser
                if (currentUser != null && currentUser.email == email) {
                    dataTextView.text = "UID: ${currentUser.uid}\nEmail: ${currentUser.email}"
                } else {
                    Toast.makeText(context, "Bạn chưa đăng nhập!", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(context, "Vui lòng nhập email!", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

3. Đăng kí:



4. Đăng nhập:



5. Hiển thị dữ liệu:

