

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	2251162006	Tạ Văn Hiếu	64CNTT2

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

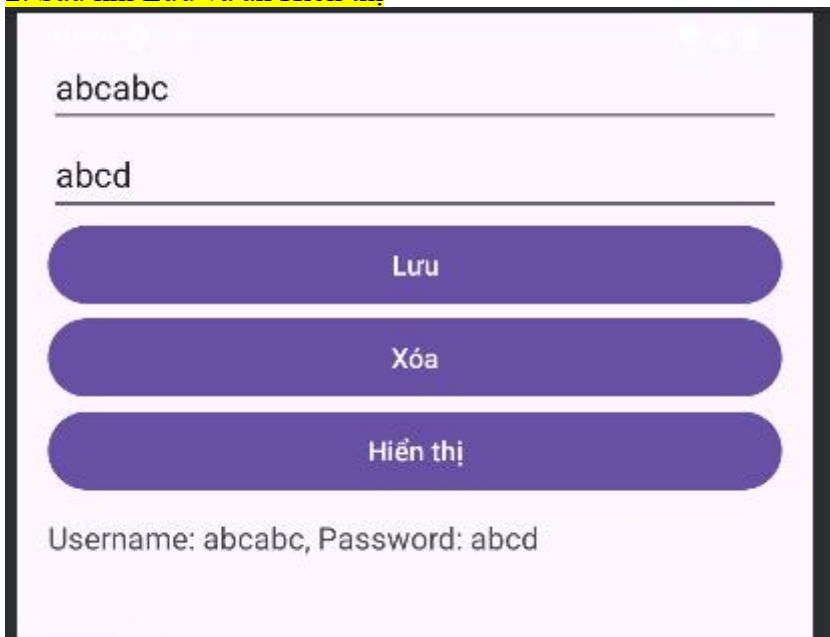
<<Sinh viên chụp Ảnh màn hình kết quả và mã nguồn chính tại đây>>

1. Nhập dữ liệu



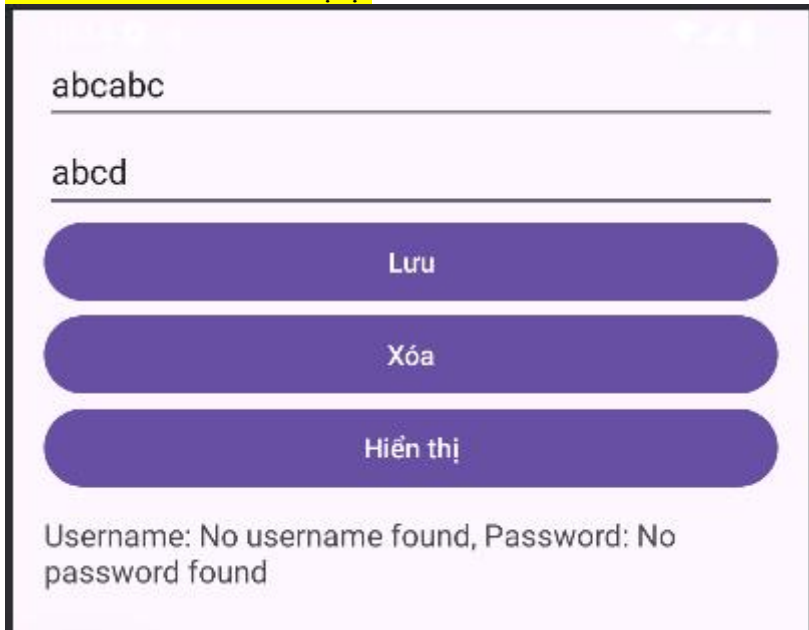
The screenshot shows a login interface with a light purple background. At the top right, there are three small white heart icons. Below them are two input fields. The first field contains the text 'abcabc' and the second field contains 'abcd'. Below the input fields are three rounded rectangular buttons stacked vertically, labeled 'Lưu', 'Xóa', and 'Hiển thị' from top to bottom. At the bottom of the form, there is a text label 'Thông tin sẽ hiển thị ở đây'.

2. Sau khi Lưu và ấn Hiển thị



This screenshot shows the same login interface as the previous one, but with the result of the 'Lưu' and 'Hiển thị' actions. The input fields still contain 'abcabc' and 'abcd'. The buttons 'Lưu', 'Xóa', and 'Hiển thị' are still present. However, the text label at the bottom now displays 'Username: abcabc, Password: abcd'.

3. Sau khi xóa và hiển thị lại



The screenshot shows a web form with a light purple background. At the top right, there are three small, faint icons. Below them, there are two text input fields. The first field contains the text "abcabc" and the second field contains "abcd". Below the input fields are three purple buttons with white text: "Lưu" (Save), "Xóa" (Delete), and "Hiển thị" (Display). At the bottom of the form, there is a message: "Username: No username found, Password: No password found".

4. Mã nguồn

```
activity_main.xml MainActivity.kt x
1 package com.example.bth4
2
3 import android.os.Bundle
4 import android.widget.Button
5 import android.widget.EditText
6 import android.widget.TextView
7 import androidx.appcompat.app.AppCompatActivity
8 import android.content.SharedPreferences
9
10 class MainActivity : AppCompatActivity() {
11
12     private lateinit var edtUsername: EditText
13     private lateinit var edtPassword: EditText
14     private lateinit var btnSave: Button
15     private lateinit var btnDelete: Button
16     private lateinit var btnShow: Button
17     private lateinit var txtDisplay: TextView
18     private lateinit var sharedPreferences: SharedPreferences
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         setContentView(R.layout.activity_main)
23
24         edtUsername = findViewById(R.id.edtUsername)
25         edtPassword = findViewById(R.id.edtPassword)
26         btnSave = findViewById(R.id.btnSave)
27         btnDelete = findViewById(R.id.btnDelete)
28         btnShow = findViewById(R.id.btnShow)
29         txtDisplay = findViewById(R.id.txtDisplay)
30
31         sharedPreferences = getSharedPreferences("Users", MODE_PRIVATE)
32
33         btnSave.setOnClickListener {
34             val username = edtUsername.text.toString()
35             val password = edtPassword.text.toString()
36             with(sharedPreferences.edit()) {
37                 putString("USERNAME", username)
38                 putString("PASSWORD", password)
39                 apply()
40             }
41         }
42
43         btnDelete.setOnClickListener {
44             with(sharedPreferences.edit()) {
45                 clear()
46                 apply()
47             }
48         }
49
50         btnShow.setOnClickListener {
51             val username = sharedPreferences.getString("USERNAME", "No username found")
52             val password = sharedPreferences.getString("PASSWORD", "No password found")
53             txtDisplay.text = "Username: $username Password: $password"
54         }
55     }
56 }
```

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. Nhập dữ liệu và thêm

asddfvs

124135263

Thêm

Sửa

Xóa

Hiển thị



Thêm thành công

2. Đặt id tự động tăng và hiển thị các bản ghi (đã xóa bản ghi id=1)

The screenshot displays a mobile application interface with a light pink background. At the top, there are two input fields: the first contains the text "asddfvs" and the second contains "124135263". Below these fields are four purple rounded rectangular buttons stacked vertically, labeled "Thêm", "Sửa", "Xóa", and "Hiển thị" from top to bottom. Under the buttons, there is a list of four text entries, each representing a contact record: "ID: 2, Name: Hieu, Phone: 098765765", "ID: 3, Name: abcd, Phone: 09876", "ID: 4, Name: ashdlaj, Phone: 2356363445", and "ID: 5, Name: asddfvs, Phone: 124135263". The interface is framed by a dark border, and a small portion of a bottom navigation bar is visible at the very bottom.

3. Sửa id=2

The screenshot shows a mobile app interface with a light pink background. At the top, there are two input fields: the first contains 'Ta Van Hie' and the second contains '124135263'. Below these fields are four purple rounded rectangular buttons stacked vertically, labeled 'Thêm', 'Sửa', 'Xóa', and 'Hiện thị'. At the bottom, there is a list of four contact entries, each showing an ID, a Name, and a Phone number. The first entry is 'ID: 2, Name: Ta Van Hieu, Phone: 124135263'. A small white circular button with a black 'x' icon is positioned to the left of the first entry in the list.

Ta Van Hie

124135263

Thêm

Sửa

Xóa

Hiện thị

ID: 2, Name: Ta Van Hieu, Phone: 124135263
ID: 3, Name: abcd, Phone: 09876
ID: 4, Name: ashdlaj, Phone: 2356363445
ID: 5, Name: asddfvs, Phone: 124135263

4. Xóa id=2

The screenshot shows the same mobile app interface as in the previous image. The 'Xóa' button is now highlighted with a dark purple background, indicating it has been selected. The other buttons and input fields remain the same. The list of contact entries at the bottom is also the same, but the first entry 'ID: 2, Name: Ta Van Hieu, Phone: 124135263' is no longer visible, suggesting it has been successfully deleted.

Ta Van Hieu

124135263

Thêm

Sửa

Xóa

Hiện thị

ID: 3, Name: abcd, Phone: 09876
ID: 4, Name: ashdlaj, Phone: 2356363445
ID: 5, Name: asddfvs, Phone: 124135263

5. File database

```
class DBHelper(context: Context) : SQLiteOpenHelper(context, name: "ContactsDB", factory: null, version: 1) {
    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(sql: "CREATE TABLE Contacts (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, phone TEXT)")
    }

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

    fun insertContact(name: String, phone: String): Boolean {
        val db = writableDatabase
        val values = ContentValues()
        values.put("name", name)
        values.put("phone", phone)
        val result = db.insert(table: "Contacts", nullColumnHack: null, values)
        db.close()
        return result != -1L
    }

    fun updateContact(id: Int, name: String, phone: String) {
        val db = writableDatabase
        val values = ContentValues()
        values.put("name", name)
        values.put("phone", phone)
        db.update(table: "Contacts", values, whereClause: "id=?", arrayOf(id.toString()))
        db.close()
    }

    fun deleteContact(id: Int) {
        val db = writableDatabase
        db.delete(table: "Contacts", whereClause: "id=?", arrayOf(id.toString()))
        db.close()
    }
}
```

```
fun getAllContacts(): String {
    val db = readableDatabase
    val cursor = db.rawQuery(sql: "SELECT * FROM Contacts", selectionArgs: null)
    val result = StringBuilder()
    while (cursor.moveToNext()) {
        result.append("ID: ").append(cursor.getInt(columnIndex: 0)).append(", Name: ").append(cursor.getString(columnIndex: 1))
        .append(", Phone: ").append(cursor.getString(columnIndex: 2)).append("\n")
    }
    cursor.close()
    db.close()
    return result.toString()
}
```

6. File Main

```

dbHelper = DBHelper(context, this)
nameInput = findViewById(R.id.nameInput)
phoneInput = findViewById(R.id.phoneInput)
resultView = findViewById(R.id.resultView)
btnAdd = findViewById(R.id.addButton)
btnEdit = findViewById(R.id.updateButton)
btnDelete = findViewById(R.id.deleteButton)
btnShow = findViewById(R.id.displayButton)

btnAdd.setOnClickListener {
    val success = dbHelper.insertContact(nameInput.text.toString(), phoneInput.text.toString())
    if (success) {
        Toast.makeText(context, this, text: "Thêm thành công", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(context, this, text: "Thêm thất bại", Toast.LENGTH_SHORT).show()
    }
}

btnEdit.setOnClickListener {
    dbHelper.updateContact(id: 2, nameInput.text.toString(), phoneInput.text.toString())
    displayContacts()
}

btnDelete.setOnClickListener {
    dbHelper.deleteContact(id: 2)
    displayContacts()
}

btnShow.setOnClickListener {
    displayContacts()
}

private fun displayContacts() {
    resultView.text = dbHelper.getAllContacts()
}

```

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.

Nguồn: Tham khảo chatGPT

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 vào năm 2014. Ban đầu, Firebase được thành lập vào năm 2011 bởi James Tamplin và Andrew Lee với mục tiêu cung cấp dịch vụ cơ sở dữ liệu thời gian thực giúp các nhà phát triển dễ dàng đồng bộ dữ liệu giữa các ứng dụng. Sau khi được Google mua lại, Firebase đã phát triển mạnh mẽ và trở thành một hệ sinh thái toàn diện, cung cấp nhiều dịch vụ hỗ trợ cho các nhà phát triển ứng dụng, bao gồm xác thực người dùng, cơ sở dữ liệu, lưu trữ tệp tin, thông báo đẩy và nhiều công cụ hỗ trợ khác.

Mô tả chi tiết từng dịch vụ chính của Firebase

1. Firebase Authentication Firebase Authentication giúp các nhà phát triển dễ dàng quản lý xác thực người dùng trong ứng dụng của họ. Dịch vụ này hỗ trợ nhiều phương thức đăng nhập,

bao gồm:

- Đăng nhập bằng email/mật khẩu
- Đăng nhập bằng số điện thoại
- Đăng nhập bằng các nền tảng bên thứ ba như Google, Facebook, Twitter, Apple
- Sử dụng Anonymous Authentication để tạo phiên tạm thời

Firebase Authentication giúp cải thiện bảo mật với các tính năng như xác thực hai yếu tố (2FA) và quản lý phiên đăng nhập.

2. Firebase Realtime Database và Cloud Firestore

Firebase Realtime Database là một cơ sở dữ liệu NoSQL thời gian thực, cho phép dữ liệu được đồng bộ hóa giữa các thiết bị ngay lập tức. Nó phù hợp cho các ứng dụng như chat, theo dõi vị trí và các ứng dụng yêu cầu dữ liệu theo thời gian thực.

Cloud Firestore là một cơ sở dữ liệu NoSQL tiên tiến hơn, hỗ trợ truy vấn mạnh mẽ hơn, khả năng mở rộng tốt hơn và tích hợp chặt chẽ với các dịch vụ khác của Firebase.

3. Firebase Cloud Functions Firebase Cloud Functions cho phép chạy mã backend serverless dựa trên các sự kiện xảy ra trong ứng dụng. Các nhà phát triển có thể sử dụng dịch vụ này để:

Xử lý xác thực người dùng

Gửi thông báo tự động

Tích hợp với các API bên ngoài

Thực hiện các thao tác trên cơ sở dữ liệu

4. Firebase Cloud Messaging (FCM) Firebase Cloud Messaging là một dịch vụ thông báo đẩy giúp gửi tin nhắn đến các thiết bị di động và trình duyệt web. Nó hỗ trợ:

Gửi thông báo cá nhân hóa đến người dùng

Gửi tin nhắn nhóm theo chủ đề

Gửi thông báo dựa trên điều kiện cụ thể

5. Firebase Storage Firebase Storage cung cấp giải pháp lưu trữ tệp tin trên đám mây, giúp các nhà phát triển lưu trữ và chia sẻ hình ảnh, video, âm thanh hoặc tài liệu. Firebase Storage tích hợp với Firebase Authentication để đảm bảo quyền truy cập an toàn.

6. Firebase Machine Learning (ML) Firebase ML giúp tích hợp trí tuệ nhân tạo vào ứng dụng mà không cần kiến thức chuyên sâu về AI. Dịch vụ này hỗ trợ:

Nhận diện văn bản (Text Recognition)

Phát hiện đối tượng trong hình ảnh (Object Detection)

Dịch ngôn ngữ (Language Translation)

Các mô hình tùy chỉnh sử dụng TensorFlow Lite

Lợi ích và ứng dụng của Firebase trong phát triển ứng dụng

Tiết kiệm thời gian và công sức: Firebase cung cấp nhiều dịch vụ sẵn có giúp các nhà phát triển không cần phải xây dựng backend từ đầu.

Tích hợp dễ dàng: Firebase hoạt động tốt trên cả Android, iOS và Web, giúp đồng bộ dữ liệu nhanh chóng giữa các nền tảng.

Khả năng mở rộng: Firebase có thể hỗ trợ từ ứng dụng nhỏ đến các hệ thống lớn với hàng triệu người dùng.

Bảo mật mạnh mẽ: Firebase cung cấp các công cụ bảo mật như xác thực người dùng, quản lý quyền truy cập dữ liệu và bảo vệ thông tin người dùng.

Hỗ trợ phân tích dữ liệu: Firebase Analytics giúp theo dõi hành vi người dùng, từ đó tối ưu hóa trải nghiệm và chiến lược phát triển sản phẩm.

Kết luận Firebase là một nền tảng mạnh mẽ hỗ trợ phát triển ứng dụng với nhiều dịch vụ hữu ích. Nó giúp giảm thiểu chi phí phát triển, cải thiện bảo mật và tối ưu hóa hiệu suất ứng dụng. Firebase phù hợp với cả cá nhân và doanh nghiệp muốn xây dựng các ứng dụng di động và web hiện đại.

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.

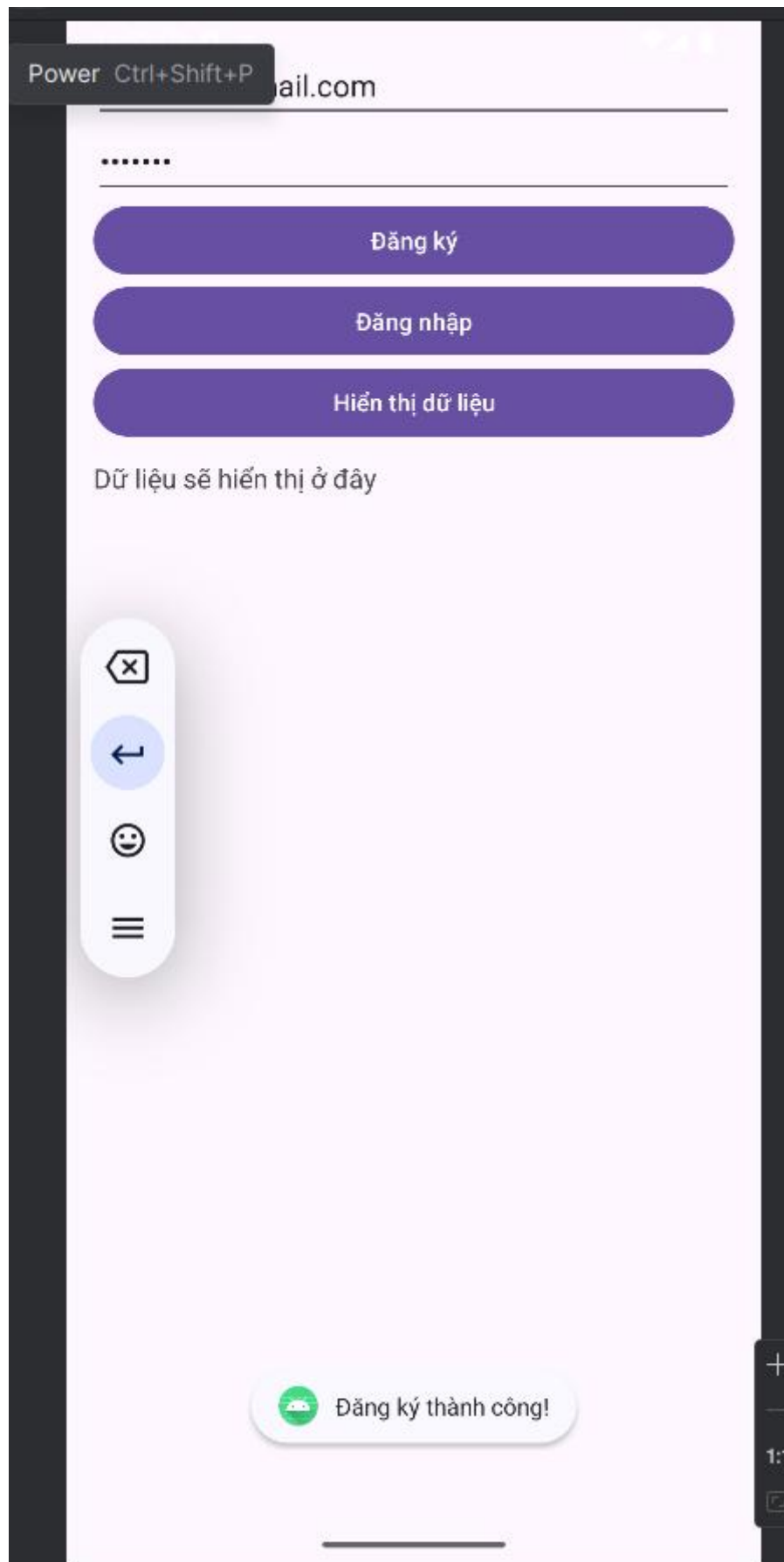
3. Tích hợp Firebase Realtime Database:

- 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. Đăng ký và Đăng nhập



tvh2003@gmail.com

.....

Đăng ký

Đăng nhập

Hiển thị dữ liệu

Dữ liệu sẽ hiển thị ở đây



Đăng nhập thành công!

2. Show dữ liệu

The screenshot shows a mobile application interface with a light pink background. At the top, there is a status bar with the time 12:11 and battery level 92%. Below the status bar, the email address abc03@gmail.com is displayed in a dark blue font. Underneath the email address is a password field with a black outline and a vertical cursor, containing six dots. Below the password field are three rounded rectangular buttons with a dark blue background and white text. The buttons are labeled "Đăng ký", "Đăng nhập", and "Hiển thị dữ liệu". Below the buttons, there is a block of text containing a long alphanumeric string "8e7gVmP0kiWTbvUtMjqWKNbxcq72:", followed by "tvh03@gmail.com", and then "knftXhZX0YVOrYSvIKoJGUYaEjK2: abc03@gmail.com".

3. Code Main

```

auth = FirebaseAuth.getInstance()

val etEmail = findViewById<EditText>(R.id.etEmail)
val etPassword = findViewById<EditText>(R.id.etPassword)
val btnRegister = findViewById<Button>(R.id.btnRegister)
val btnLogin = findViewById<Button>(R.id.btnLogin)
val btnShowData = findViewById<Button>(R.id.btnShowData)
val tvData = findViewById<TextView>(R.id.tvData)

btnRegister.setOnClickListener {
    val email = etEmail.text.toString()
    val password = etPassword.text.toString()
    if (email.isEmpty() || password.isEmpty()) {
        showToast( msg: "Nhập đầy đủ thông tin!")
        return@setOnClickListener
    }
    auth.createUserWithEmailAndPassword(email, password)
        .addOnSuccessListener {
            saveUser(email)
            showToast( msg: "Đăng ký thành công!")
        }
        .addOnFailureListener { showToast( msg: "Đăng ký thất bại!") }
}

```

```

btnLogin.setOnClickListener {
    val email = etEmail.text.toString()
    val password = etPassword.text.toString()
    auth.signInWithEmailAndPassword(email, password)
        .addOnSuccessListener { showToast(msg: "Đăng nhập thành công!") }
        .addOnFailureListener { showToast(msg: "Đăng nhập thất bại!") }
}

btnShowData.setOnClickListener {
    database.child(pathString: "users").get()
        .addOnSuccessListener { snapshot ->
            tvData.text = snapshot.children.joinToString(separator: "\n") {
                it.key + ": " + it.child(path: "email").value.toString()
            }
        }
        .addOnFailureListener { showToast(msg: "Lỗi khi tải dữ liệu!") }
}

private fun saveUser(email: String) {
    auth.currentUser?.uid?.let { uid ->
        database.child(pathString: "users").child(uid).setValue(mapOf("email" to email))
    }
}

private fun showToast(msg: String) {
    Toast.makeText(context: this, msg, Toast.LENGTH_SHORT).show()
}

```

