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	2251061740	Phạm Tiến Đat	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>>

```
package com.example.sharedpreference.ui.theme

import android.content.Context
import android.content.SharedPreferences
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken

// Định nghĩa class User thay vì dùng Pair<String, String>
data class User(val username: String, val password: String)

class PreferenceHelper(context: Context) {
    companion object {
        private const val PREF_NAME = "UserPrefs"
        private const val KEY_USER_LIST = "USER_LIST"
    }

    private val sharedPreferences: SharedPreferences =
        context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE)
    private val gson = Gson()

    // Lấy danh sách user từ SharedPreferences
    fun getUserDataList(): List<User> {
        val jsonString = sharedPreferences.getString(KEY_USER_LIST, defValue: null) ?: return emptyList()
        val type = object : TypeToken<List<User>>() {}.type
        return gson.fromJson(jsonString, type)
    }

    // Lưu user vào SharedPreferences, kiểm tra trùng lặp trước khi thêm
    fun saveUserData(username: String, password: String): Boolean {
        val userList = getUserDataList().toMutableList()

        // Kiểm tra nếu username đã tồn tại
        if (userList.any { it.username == username }) {
            return false
        }

        userList.add(User(username, password)) // Thêm user mới

        val jsonString = gson.toJson(userList) // Chuyển danh sách thành JSON
        sharedPreferences.edit().putString(KEY_USER_LIST, jsonString).apply()
        return true
    }

    // Xóa toàn bộ danh sách
    fun clearUserData() {
        sharedPreferences.edit().remove(KEY_USER_LIST).apply()
    }
}
```

```
    // Xử lý sự kiện nút "Lưu"
    buttonSave.setOnClickListener {
        val username = editTextUsername.text.toString().trim()
        val password = editTextPassword.text.toString().trim()

        if (username.isEmpty() || password.isEmpty()) {
            Toast.makeText( context: this,  text: "Vui lòng nhập đầy đủ thông tin!", Toast.LENGTH_SHORT).show()
        } else {
            val isSaved = preferenceHelper.saveUserData(username, password)
            if (isSaved) {
                Toast.makeText( context: this,  text: "Lưu thành công!", Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText( context: this,  text: "Tên người dùng đã tồn tại!", Toast.LENGTH_SHORT).show()
            }
        }
    }

    // Xử lý sự kiện nút "Hiển thị"
    buttonShow.setOnClickListener {
        val userList = preferenceHelper.getUserDataList()
        if (userList.isEmpty()) {
            textViewDisplay.text = "Chưa có dữ liệu!"
        } else {
            val userText = userList.joinToString( separator: "\n") { "👤 ${it.username} - 🔒 ${it.password}" }
            textViewDisplay.text = userText
        }
    }
```

```
package com.example.sharedpreference                                    ⚠1 ✓3

import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import android.widget.EditText
import com.example.sharedpreference.ui.theme.PreferenceHelper

class MainActivity : AppCompatActivity() {
    private lateinit var editTextUsername: EditText
    private lateinit var editTextPassword: EditText
    private lateinit var buttonSave: Button
    private lateinit var buttonDelete: Button
    private lateinit var buttonShow: Button
    private lateinit var textViewDisplay: TextView
    private lateinit var preferenceHelper: PreferenceHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Ánh xạ UI
        editTextUsername = findViewById(R.id.editTextUsername)
        editTextPassword = findViewById(R.id.editTextPassword)
        buttonSave = findViewById(R.id.buttonSave)
        buttonDelete = findViewById(R.id.buttonDelete)
        buttonShow = findViewById(R.id.buttonShow)
        textViewDisplay = findViewById(R.id.textViewDisplay)

        // Khởi tạo SharedPreferences Helper
```

```
            } else {
                val userText = userList.joinToString( separator: "\n") { "👤 ${it.username} - 🔒 ${it.password}" }
                textViewDisplay.text = userText
            }
        }

        // Xử lý sự kiện nút "Xóa"
        buttonDelete.setOnClickListener {
            preferenceHelper.clearUserData()
            textViewDisplay.text = "" // Xóa luôn nội dung hiển thị
            Toast.makeText( context: this,  text: "Xóa dữ liệu thành công!", Toast.LENGTH_SHORT).show()
        }
    }
}
```

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>>


Quản Lý Danh Bạ
Nhập tên
Nhập số điện thoại
THÊM
SỬA
XÓA
HIỂN THỊ
dat: 123
datbeo: 123123123123123

```kotlin
package com.example.sqlite

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var etName: EditText
    private lateinit var etPhone: EditText
    private lateinit var tvResult: TextView
    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Khởi tạo các view
        etName = findViewById(R.id.etName)
        etPhone = findViewById(R.id.etPhone)
        tvResult = findViewById(R.id.tvResult)

        // Khởi tạo DatabaseHelper
        dbHelper = DatabaseHelper( context: this)

        val btnAdd = findViewById<Button>(R.id.btnAdd)
        val btnUpdate = findViewById<Button>(R.id.btnUpdate)
        val btnDelete = findViewById<Button>(R.id.btnDelete)
```

```
val btnAdd = findViewById<Button>(R.id.btnAdd)
val btnUpdate = findViewById<Button>(R.id.btnUpdate)
val btnDelete = findViewById<Button>(R.id.btnDelete)
val btnShow = findViewById<Button>(R.id.btnShow)

// Xử lý sự kiện nút Thêm
btnAdd.setOnClickListener {
    val name = etName.text.toString()
    val phone = etPhone.text.toString()
    if (name.isNotEmpty() && phone.isNotEmpty()) {
        if (dbHelper.addContact(name, phone)) {
            clearInputs()
            tvResult.text = "Thêm liên hệ thành công"
        } else {
            tvResult.text = "Thêm thất bại (tên đã tồn tại)"
        }
    } else {
        tvResult.text = "Vui lòng điền đầy đủ thông tin"
    }
}

// Xử lý sự kiện nút Sửa
btnUpdate.setOnClickListener {
    val name = etName.text.toString()
    val phone = etPhone.text.toString()
    if (name.isNotEmpty() && phone.isNotEmpty()) {
        if (dbHelper.updateContact(name, phone)) {
```

```
// Xử lý sự kiện nút Sửa
btnUpdate.setOnClickListener {
    val name = etName.text.toString()
    val phone = etPhone.text.toString()
    if (name.isNotEmpty() && phone.isNotEmpty()) {
        if (dbHelper.updateContact(name, phone)) {
            clearInputs()
            tvResult.text = "Cập nhật liên hệ thành công"
        } else {
            tvResult.text = "Không tìm thấy liên hệ"
        }
    } else {
        tvResult.text = "Vui lòng điền đầy đủ thông tin"
    }
}

// Xử lý sự kiện nút Xóa
btnDelete.setOnClickListener {
    val name = etName.text.toString()
    if (name.isNotEmpty()) {
        if (dbHelper.deleteContact(name)) {
            clearInputs()
            tvResult.text = "Xóa liên hệ thành công"
        } else {
            tvResult.text = "Không tìm thấy liên hệ"
        }
    } else {
```

```
                tvResult.text = "Không tìm thấy liên hệ"
            }
        } else {
            tvResult.text = "Vui lòng nhập tên để xóa"
        }
    }

    // Xử lý sự kiện nút Hiển thị
    btnShow.setOnClickListener {
        val contacts = dbHelper.getAllContacts()
        if (contacts.isEmpty()) {
            tvResult.text = "Không có liên hệ nào"
        } else {
            val contactsText = contacts.joinToString( separator: "\n") { "${it.first}: ${it.second}" }
            tvResult.text = contactsText
        }
    }
}

private fun clearInputs() {
    etName.text.clear()
    etPhone.text.clear()
}
}
```

```
package com.example.sqlite

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 = "ContactsDB"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "contacts"
        private const val COLUMN_NAME = "name"
        private const val COLUMN_PHONE = "phone"
    }

    override fun onCreate(db: SQLiteDatabase) {
        // Tạo bảng contacts với cột name (khóa chính) và phone
        val createTableQuery = """
            CREATE TABLE $TABLE_NAME (
                $COLUMN_NAME TEXT PRIMARY KEY,
                $COLUMN_PHONE TEXT
            )
        """.trimIndent()
        db.execSQL(createTableQuery)
    }
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        // Xóa bảng cũ và tạo lại nếu có cập nhật version
        db.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    // Hàm thêm dữ liệu
    fun addContact(name: String, phone: String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NAME, name)
            put(COLUMN_PHONE, phone)
        }
        val result = db.insert(TABLE_NAME,  nullColumnHack: null, values)
        db.close()
        return result != -1L // Trả về true nếu thêm thành công
    }

    // Hàm cập nhật dữ liệu
    fun updateContact(name: String, phone: String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_PHONE, phone)
        }
        val result = db.update(TABLE_NAME, values,  whereClause: "$COLUMN_NAME = ?", arrayOf(name))
        db.close()
```

```kotlin
// Hàm cập nhật dữ liệu
fun updateContact(name: String, phone: String): Boolean {
    val db = writableDatabase
    val values = ContentValues().apply {
        put(COLUMN_PHONE, phone)
    }
    val result = db.update(TABLE_NAME, values, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
    return result > 0 // Trả về true nếu cập nhật thành công
}

// Hàm xóa dữ liệu
fun deleteContact(name: String): Boolean {
    val db = writableDatabase
    val result = db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
    return result > 0 // Trả về true nếu xóa thành công
}

// Hàm lấy tất cả dữ liệu
fun getAllContacts(): List<Pair<String, String>> {
    val contactsList = mutableListOf<Pair<String, String>>()
    val db = readableDatabase
    val cursor = db.rawQuery( sql: "SELECT * FROM $TABLE_NAME", selectionArgs: null)
    if (cursor.moveToFirst()) {
        do {
            val name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME))
            val phone = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_PHONE))
            contactsList.add(Pair(name, phone))
```

```kotlin
    val result = db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME = ?", arrayOf(name))
    db.close()
    return result > 0 // Trả về true nếu xóa thành công
}

// Hàm lấy tất cả dữ liệu
fun getAllContacts(): List<Pair<String, String>> {
    val contactsList = mutableListOf<Pair<String, String>>()
    val db = readableDatabase
    val cursor = db.rawQuery( sql: "SELECT * FROM $TABLE_NAME", selectionArgs: null)
    if (cursor.moveToFirst()) {
        do {
            val name = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NAME))
            val phone = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN
            contactsList.add(Pair(name, phone))
        } while (cursor.moveToNext())
    }
    cursor.close()
    db.close()
    return contactsList
}
}
```

```
private const final val COLUMN_NAME: String
com.example.sqlite.DatabaseHelper.Companion
SQLite.app.main
```

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.

BÁO CÁO: HỆ SINH THÁI FIREBASE

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

Firebase là một nền tảng phát triển ứng dụng toàn diện được cung cấp bởi Google, giúp các nhà phát triển xây dựng, quản lý và mở rộng ứng dụng một cách nhanh chóng và hiệu quả. Nó cung cấp một bộ công cụ và dịch vụ tích hợp, từ xác thực người dùng, cơ sở dữ liệu, lưu trữ, đến các chức năng nâng cao như gửi thông báo đẩy và tích hợp trí tuệ nhân tạo.

Firebase ra đời vào năm 2011, ban đầu là một công ty khởi nghiệp do James Tamplin và Andrew Lee sáng lập, tập trung vào cơ sở dữ liệu thời gian thực. Năm 2014, Google mua lại Firebase và tích hợp nó vào hệ sinh thái của mình, mở rộng các dịch vụ để hỗ trợ phát triển ứng dụng di động và web. Kể từ đó, Firebase đã trở thành một công cụ phổ biến, được sử dụng rộng rãi bởi các nhà phát triển nhờ tính linh hoạt, khả năng mở rộng và tích hợp chặt chẽ với các sản phẩm khác của Google như Google Cloud Platform.

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

Firebase Authentication: Xác thực người dùng

Firebase Authentication cung cấp một giải pháp đơn giản để quản lý danh tính người dùng trong ứng dụng. Nó hỗ trợ nhiều phương thức xác thực như email/mật khẩu, số điện thoại, tài khoản Google, Facebook, Twitter, GitHub, và thậm chí cả xác thực ẩn danh. Dịch vụ này tích hợp SDK dễ sử dụng, cho phép

nhà phát triển thêm tính năng đăng nhập/đăng ký chỉ với vài dòng mã. Ngoài ra, nó đảm bảo tính bảo mật cao với các tiêu chuẩn mã hóa hiện đại.

Firebase Realtime Database và Cloud Firestore: Cơ sở dữ liệu thời gian thực và NoSQL

- Firebase Realtime Database: Là cơ sở dữ liệu NoSQL dựa trên đám mây, lưu trữ dữ liệu dưới dạng JSON và đồng bộ hóa dữ liệu theo thời gian thực giữa các thiết bị. Nó phù hợp cho các ứng dụng cần cập nhật dữ liệu tức thì, như ứng dụng trò chuyện hoặc bảng xếp hạng trực tuyến.
- Cloud Firestore: Là phiên bản nâng cấp của Realtime Database, cung cấp khả năng mở rộng tốt hơn, truy vấn mạnh mẽ hơn và hỗ trợ dữ liệu có cấu trúc phức tạp. Firestore hoạt động ngoại tuyến, cho phép người dùng truy cập dữ liệu ngay cả khi không có kết nối mạng, và đồng bộ hóa khi kết nối được khôi phục.

Firebase Cloud Functions: Chạy mã backend serverless

Firebase Cloud Functions cho phép nhà phát triển viết và triển khai mã backend mà không cần quản lý máy chủ. Các hàm này được kích hoạt bởi các sự kiện trong Firebase (như thay đổi dữ liệu, xác thực người dùng) hoặc thông qua yêu cầu HTTP. Đây là giải pháp lý tưởng để xử lý logic phức tạp, tích hợp với API bên thứ ba hoặc tự động hóa tác vụ.

Firebase Cloud Messaging (FCM): Gửi thông báo đẩy

FCM là dịch vụ gửi thông báo đẩy miễn phí, giúp nhà phát triển gửi tin nhắn đến người dùng trên các nền tảng Android, iOS và web. Thông báo có thể là thông tin cập nhật, khuyến mãi hoặc nhắc nhở, với khả năng tùy chỉnh cao. FCM hỗ trợ gửi thông báo đến từng cá nhân, nhóm người dùng hoặc toàn bộ người dùng ứng dụng.

Firebase Storage: Lưu trữ tệp tin trên đám mây

Firebase Storage cung cấp không gian lưu trữ đám mây để lưu tệp tin như hình ảnh, video, âm thanh hoặc tài liệu. Nó được tích hợp với Firebase Authentication để đảm bảo quyền truy cập an toàn và hỗ trợ tải lên/tải xuống tệp với hiệu suất cao, ngay cả khi mạng không ổn định. Dịch vụ này rất hữu ích cho các ứng dụng như mạng xã hội hoặc ứng dụng chỉnh sửa ảnh.

Firebase Machine Learning (ML): Tích hợp trí tuệ nhân tạo vào ứng dụng

Firebase ML cung cấp các công cụ để tích hợp trí tuệ nhân tạo và học máy vào ứng dụng mà không cần kiến thức chuyên sâu về AI. Nó bao gồm các mô hình sẵn có như nhận diện văn bản, phân loại hình ảnh, và phát hiện đối tượng. Ngoài ra, nhà phát triển có thể sử dụng mô hình tùy chỉnh được huấn luyện bằng

TensorFlow Lite, giúp ứng dụng thông minh hơn và cá nhân hóa trải nghiệm người dùng.

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

Lợi ích của Firebase

- Tích hợp dễ dàng: Firebase cung cấp SDK đa nền tảng (Android, iOS, web) và tài liệu hướng dẫn chi tiết, giúp nhà phát triển bắt đầu nhanh chóng.
- Tiết kiệm thời gian và chi phí: Với các dịch vụ serverless như Cloud Functions và cơ sở dữ liệu sẵn có, nhà phát triển không cần xây dựng backend từ đầu hoặc quản lý máy chủ.
- Khả năng mở rộng: Firebase tự động mở rộng theo lưu lượng truy cập, phù hợp cho cả ứng dụng nhỏ và lớn.
- Hỗ trợ thời gian thực: Các tính năng như Realtime Database và FCM giúp ứng dụng phản hồi nhanh chóng với người dùng.
- Bảo mật: Firebase tích hợp các tiêu chuẩn bảo mật cao, từ xác thực đến mã hóa dữ liệu.

Ứng dụng của Firebase

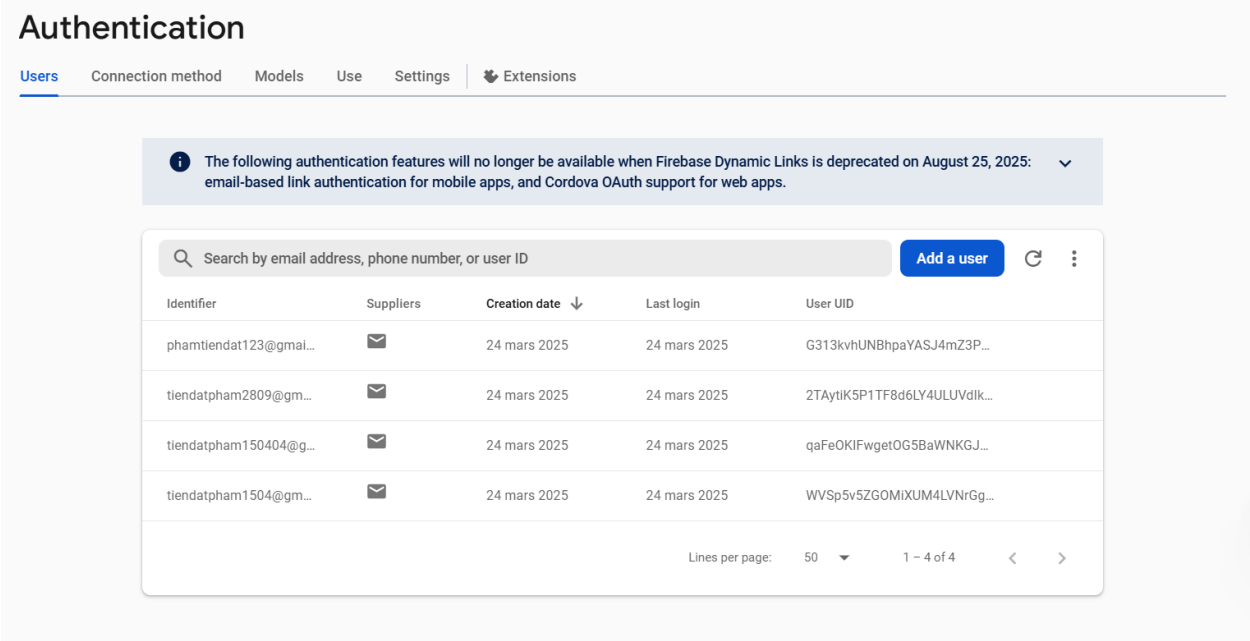
- Ứng dụng trò chuyện: Sử dụng Realtime Database để đồng bộ tin nhắn tức thì và FCM để thông báo tin nhắn mới.
- Mạng xã hội: Kết hợp Storage để lưu ảnh/video, Authentication để quản lý người dùng, và Firestore để lưu bài đăng.
- Ứng dụng thương mại điện tử: Cloud Functions xử lý thanh toán, FCM gửi thông báo khuyến mãi, và ML phân tích hành vi người dùng.
- Trò chơi trực tuyến: Realtime Database lưu điểm số, Authentication xác thực người chơi, và Storage lưu tài nguyên trò chơi.

4. Kết luận

Firebase là một hệ sinh thái mạnh mẽ, cung cấp các công cụ toàn diện để phát triển ứng dụng hiện đại. Với khả năng tích hợp dễ dàng, tính linh hoạt và hỗ trợ từ Google, Firebase đã trở thành lựa chọn hàng đầu của nhiều nhà phát triển. Việc hiểu và áp dụng các dịch vụ chính của Firebase không chỉ giúp tối ưu hóa quá trình phát triển mà còn nâng cao trải nghiệm người dùng trong các ứng dụng thực tế.

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>>

    ManiActivity.Main

```kotlin
package com.example.myapp

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import java.text.SimpleDateFormat
import java.util.Date

class MainActivity : AppCompatActivity() {

    private lateinit var emailEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var registerButton: Button
    private lateinit var loginButton: Button
    private lateinit var showDataButton: Button
    private lateinit var logoutButton: Button
    private lateinit var userDataTextView: TextView

    private lateinit var auth: FirebaseAuth
    private lateinit var database: DatabaseReference
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Tắt cache cục bộ để kiểm tra
    FirebaseDatabase.getInstance().setPersistenceEnabled(false)

    // Liên kết giao diện
    emailEditText = findViewById(R.id.emailEditText)
    passwordEditText = findViewById(R.id.passwordEditText)
    registerButton = findViewById(R.id.registerButton)
    loginButton = findViewById(R.id.loginButton)
    showDataButton = findViewById(R.id.showDataButton)
    logoutButton = findViewById(R.id.logoutButton)
    userDataTextView = findViewById(R.id.userDataTextView)

    // Khởi tạo Firebase
    auth = FirebaseAuth.getInstance()
    database = FirebaseDatabase.getInstance().reference.child("users")

    // Xử lý nút Đăng ký
    registerButton.setOnClickListener {
        val email = emailEditText.text.toString().trim()
        val password = passwordEditText.text.toString().trim()

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Vui lòng nhập email và mật khẩu", Toast.LENGTH_SHORT).show()
        } else if (!isValidEmail(email)) {
            Toast.makeText(this, "Email không hợp lệ", Toast.LENGTH_SHORT).show()
```

```
        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Vui lòng nhập email và mật khẩu", Toast.LENGTH_SHORT).show()
        } else if (!isValidEmail(email)) {
            Toast.makeText(this, "Email không hợp lệ", Toast.LENGTH_SHORT).show()
        } else if (password.length < 6) {
            Toast.makeText(this, "Mật khẩu phải có ít nhất 6 ký tự", Toast.LENGTH_SHORT).show()
        } else {
            registerUser(email, password)
        }
    }

    // Xử lý nút Đăng nhập
    loginButton.setOnClickListener {
        val email = emailEditText.text.toString().trim()
        val password = passwordEditText.text.toString().trim()

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(this, "Vui lòng nhập email và mật khẩu", Toast.LENGTH_SHORT).show()
        } else {
            loginUser(email, password)
        }
    }
```

```
    // Xử lý nút Hiển thị dữ liệu
    showDataButton.setOnClickListener {
        val currentUser = auth.currentUser
        if (currentUser != null) {
            fetchUserData(currentUser.uid)
        } else {
            Toast.makeText(this, "Vui lòng đăng nhập trước", Toast.LENGTH_SHORT).show()
        }
    }

    // Xử lý nút Đăng xuất
    logoutButton.setOnClickListener {
        auth.signOut()
        userDataTextView.text = "Dữ liệu người dùng sẽ hiển thị ở đây"
        Toast.makeText(this, "Đã đăng xuất", Toast.LENGTH_SHORT).show()
    }
}

// Kiểm tra định dạng email
private fun isValidEmail(email: String): Boolean {
    return android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()
}
```

```
    // Đăng ký người dùng
    private fun registerUser(email: String, password: String) {
        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task: com.google.android.gms.tasks.Task<com.google.firebase.auth.AuthResult> ->
                if (task.isSuccessful) {
                    val userId = auth.currentUser?.uid
                    if (userId != null) {
                        saveUserData(userId, email)
                    } else {
                        Toast.makeText(this, "Không tìm thấy UID người dùng", Toast.LENGTH_SHORT).show()
                    }
                    Toast.makeText(this, "Đăng ký thành công!", Toast.LENGTH_SHORT).show()
                    clearInputs()
                } else {
                    val errorMessage = task.exception?.message ?: "Đăng ký thất bại"
                    Toast.makeText(this, errorMessage, Toast.LENGTH_SHORT).show()
                }
            }
    }
```

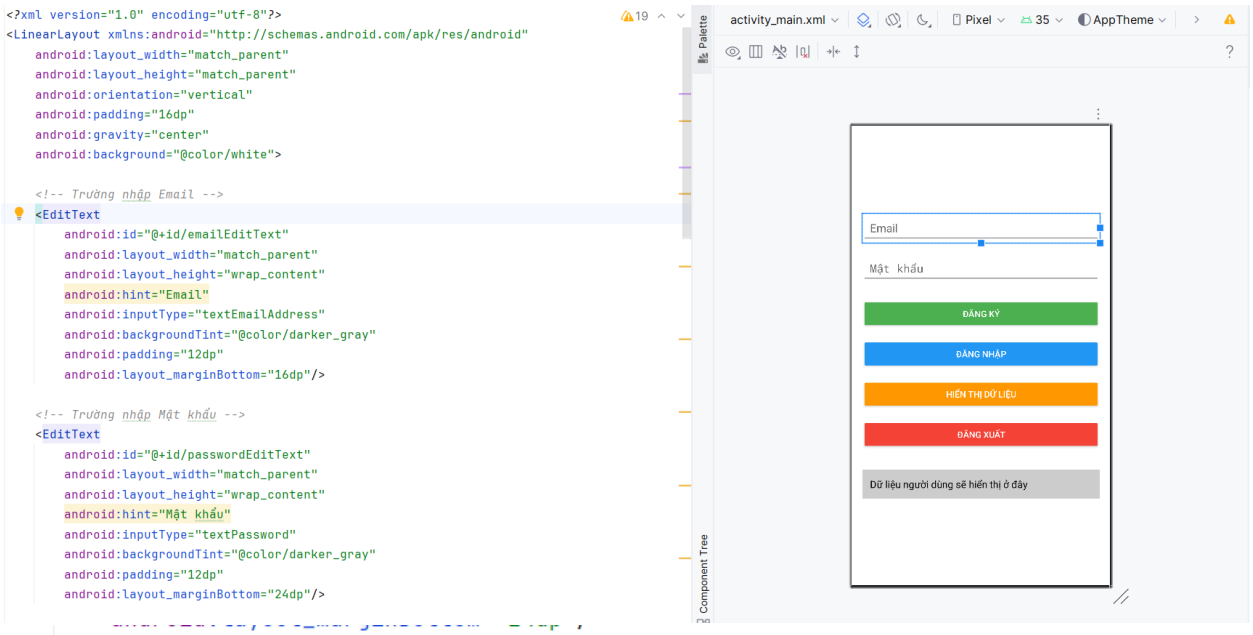
```
// Đăng nhập người dùng
private fun loginUser(email: String, password: String) {
    auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task: com.google.android.gms.tasks.Task<com.google.firebase.auth.AuthResult> ->
            if (task.isSuccessful) {
                Toast.makeText(this, "Đăng nhập thành công!", Toast.LENGTH_SHORT).show()
                val userId = auth.currentUser?.uid
                if (userId != null) {
                    saveUserData(userId, email)
                } else {
                    Toast.makeText(this, "Không tìm thấy UID người dùng", Toast.LENGTH_SHORT).show()
                }
                clearInputs()
            } else {
                val errorMessage = task.exception?.message ?: "Đăng nhập thất bại"
                Toast.makeText(this, errorMessage, Toast.LENGTH_SHORT).show()
            }
        }
}

// Lưu dữ liệu người dùng vào Realtime Database
private fun saveUserData(userId: String, email: String) {
    val user = mapOf(
        "email" to email,
        "lastLogin" to System.currentTimeMillis().toString()
    )
    database.child(userId).setValue(user)
```

```
            }
    }

    // Lưu dữ liệu người dùng vào Realtime Database
    private fun saveUserData(userId: String, email: String) {
        val user = mapOf(
            "email" to email,
            "lastLogin" to System.currentTimeMillis().toString()
        )
        database.child(userId).setValue(user)
            .addOnCompleteListener { task: com.google.android.gms.tasks.Task<Void> ->
                if (task.isSuccessful) {
                    Toast.makeText(this, "Lưu dữ liệu thành công", Toast.LENGTH_SHORT).show()
                    // Kiểm tra trạng thái đồng bộ với server
                    database.database.getReference(".info/connected").addValueEventListener(object : ValueEventListener {
                        override fun onDataChange(snapshot: DataSnapshot) {
                            val connected = snapshot.getValue(Boolean::class.java) ?: false
                            if (connected) {
                                Toast.makeText(this@MainActivity, "Đã đồng bộ với server", Toast.LENGTH_SHORT).show()
                            } else {
                                Toast.makeText(this@MainActivity, "Chưa đồng bộ, kiểm tra kết nối mạng", Toast.LENGTH_SHORT).show()
                            }
                        }

                        override fun onCancelled(error: DatabaseError) {
                            Toast.makeText(this@MainActivity, "Lỗi kiểm tra kết nối: ${error.message}", Toast.LENGTH_SHORT).show()
                        }
                    })
                } else {
```

```
                    }

                    override fun onCancelled(error: DatabaseError) {
                        Toast.makeText(this@MainActivity, "Lỗi kiểm tra kết nối: ${error.message}", Toast.LENGTH_SHORT).show()
                    }
                })
            } else {
                val errorMessage = task.exception?.message ?: "Lưu dữ liệu thất bại"
                Toast.makeText(this, errorMessage, Toast.LENGTH_SHORT).show()
            }
        }
    }

    // Đọc dữ liệu từ Realtime Database
    private fun fetchUserData(userId: String) {
        database.child(userId).addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    val email = snapshot.child("email").getValue(String::class.java)
                    val lastLogin = snapshot.child("lastLogin").getValue(String::class.java)
                    // Định dạng thời gian
                    val formattedDate = lastLogin?.let {
                        val date = Date(it.toLong())
                        SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date)
                    } ?: "Không xác định"
                    val message = "Email: $email\nLần đăng nhập cuối: $formattedDate"
                    userDataTextView.text = message
```

```
    // Đọc dữ liệu từ Realtime Database
    private fun fetchUserData(userId: String) {
        database.child(userId).addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()) {
                    val email = snapshot.child("email").getValue(String::class.java)
                    val lastLogin = snapshot.child("lastLogin").getValue(String::class.java)
                    // Định dạng thời gian
                    val formattedDate = lastLogin?.let {
                        val date = Date(it.toLong())
                        SimpleDateFormat("dd/MM/yyyy HH:mm:ss").format(date)
                    } ?: "Không xác định"
                    val message = "Email: $email\nLần đăng nhập cuối: $formattedDate"
                    userDataTextView.text = message
                } else {
                    userDataTextView.text = "Không tìm thấy dữ liệu"
                }
            }

            override fun onCancelled(error: DatabaseError) {
                userDataTextView.text = "Lỗi: ${error.message}"
            }
        })
    }

    // Xóa nội dung EditText
    private fun clearInputs() {
        emailEditText.text.clear()
        passwordEditText.text.clear()
    }
```

Activity_main

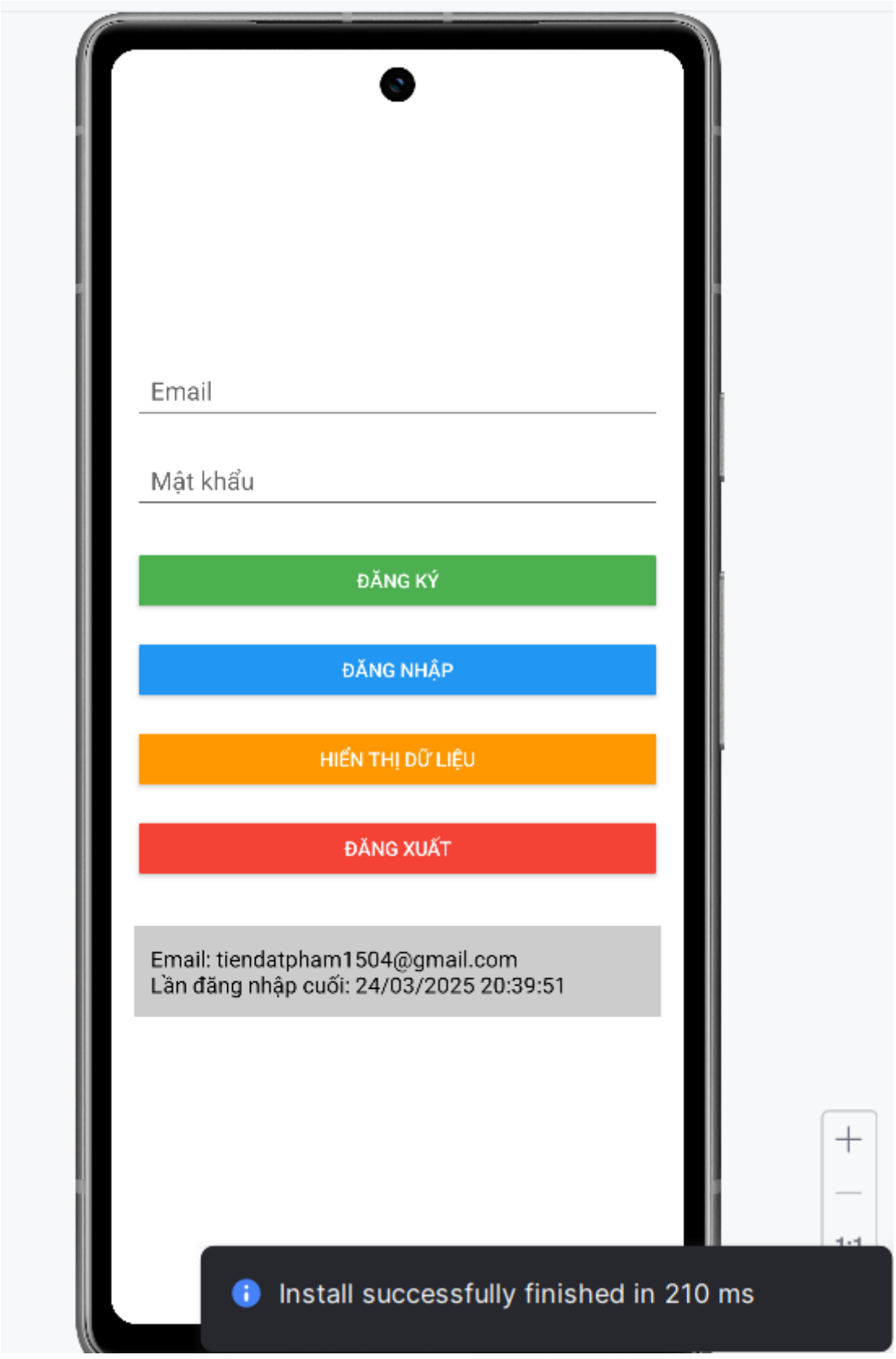



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:gravity="center"
    android:background="@color/white">

    <!-- Trường nhập Email -->
    <EditText
        android:id="@+id/emailEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:backgroundTint="@color/darker_gray"
        android:padding="12dp"
        android:layout_marginBottom="16dp"/>

    <!-- Trường nhập Mật khẩu -->
    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Mật khẩu"
        android:inputType="textPassword"
        android:backgroundTint="@color/darker_gray"
        android:padding="12dp"
        android:layout_marginBottom="24dp"/>
```

```
    <!-- Nút Đăng ký -->
    <Button
        android:id="@+id/registerButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Đăng ký"
        android:backgroundTint="@color/green"
        android:textColor="@color/white"
        android:padding="12dp"
        android:layout_marginBottom="16dp"/>

    <!-- Nút Đăng nhập -->
    <Button
        android:id="@+id/loginButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Đăng nhập"
        android:backgroundTint="@color/blue"
        android:textColor="@color/white"
        android:padding="12dp"
        android:layout_marginBottom="16dp"/>
```

```
        android:layout_marginBottom="16dp"/>

    <!-- Nút Hiển thị dữ liệu -->
    <Button
        android:id="@+id/showDataButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hiển thị dữ liệu"
        android:backgroundTint="@color/orange"
        android:textColor="@color/white"
        android:padding="12dp"
        android:layout_marginBottom="16dp"/>

    <!-- Nút Đăng xuất -->
    <Button
        android:id="@+id/logoutButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Đăng xuất"
        android:backgroundTint="@color/red"
        android:textColor="@color/white"
        android:padding="12dp"
        android:layout_marginBottom="16dp"/>

    <!-- TextView để hiển thị dữ liệu -->
```

```
    <!-- TextView để hiển thị dữ liệu -->
    <TextView
        android:id="@+id/userDataTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Dữ liệu người dùng sẽ hiển thị ở đây"
        android:textSize="16sp"
        android:textColor="@color/black"
        android:background="@color/light_gray"
        android:padding="12dp"
        android:layout_marginTop="16dp"/>

</LinearLayout>
```


Email
Mật khẩu
ĐĂNG KÝ
ĐĂNG NHẬP
HIỂN THỊ DỮ LIỆU
ĐĂNG XUẤT
Email: tiendatpham1504@gmail.com
Lần đăng nhập cuối: 24/03/2025 20:39:51
Install successfully finished in 210 ms