

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	2151062699	Đỗ Phạm Hoàng Anh	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:

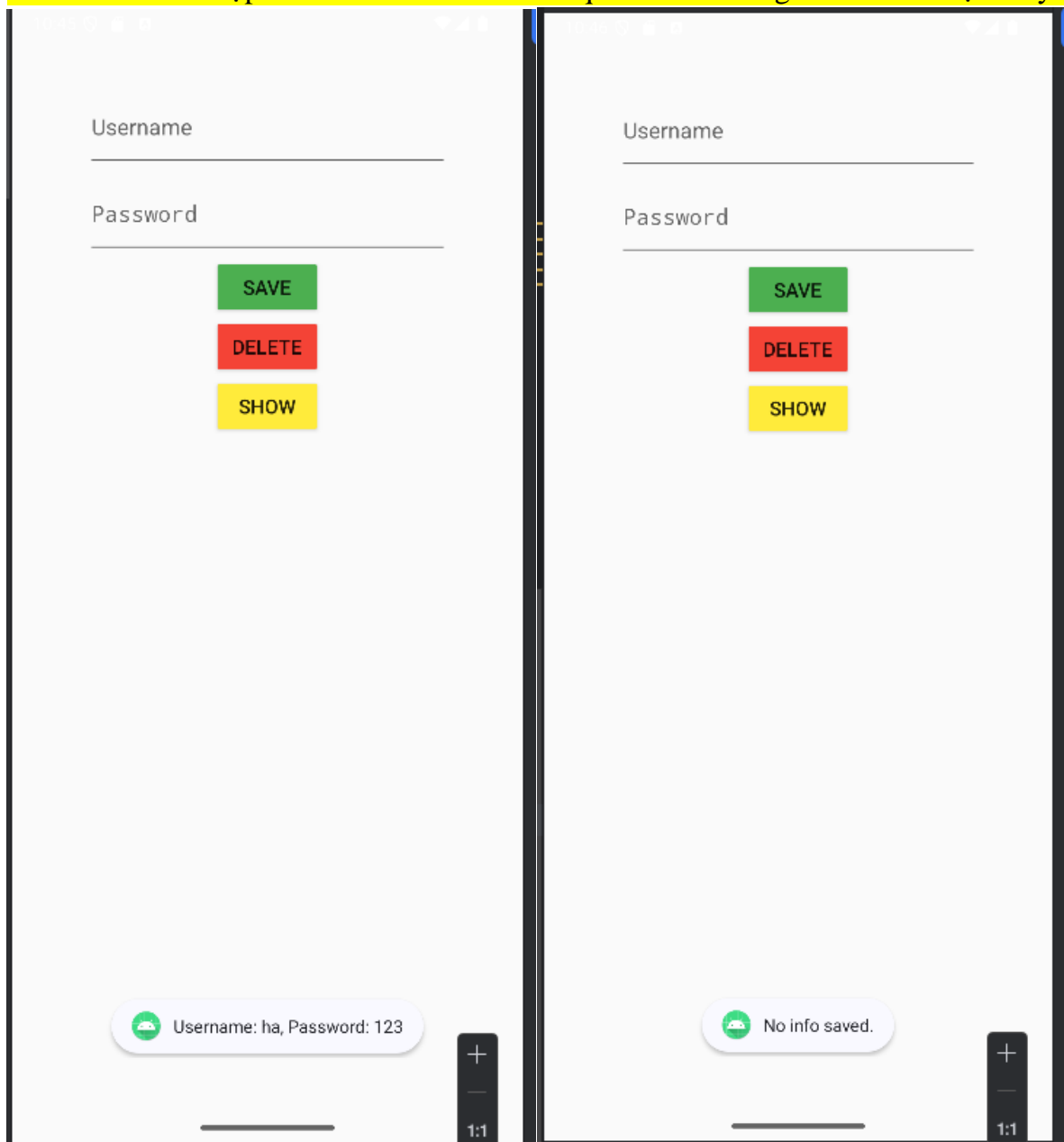
- Tạo một lớp helper **PreferenceHelper** để quản lý Shared Preference.
- Khi người dùng nhấn nút "Lưu", lưu tên người dùng và mật khẩu vào Shared Preference.
- Khi người dùng nhấn nút "Xóa", xóa dữ liệu đã lưu trong Shared Preference.
- Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ Shared Preference 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 package com.example.btth04
2
3 > import ...
4
20
21 private lateinit var editTextUsername: EditText
22 private lateinit var editTextPassword: EditText
23 private lateinit var buttonSave: Button
24 private lateinit var buttonDelete: Button
25 private lateinit var buttonShow: Button
26
27 private lateinit var preferenceHelper: PreferenceHelper
28
29 class MainActivity : AppCompatActivity() {
30     override fun onCreate(savedInstanceState: Bundle?) {
31         super.onCreate(savedInstanceState)
32         setContentView(R.layout.activity_main)
33         preferenceHelper = PreferenceHelper(context = this)
34
35         editTextUsername = findViewById<EditText>(R.id.editTextUsername)
36         editTextPassword = findViewById(R.id.editTextPassword)
37         buttonSave = findViewById<Button>(R.id.buttonSave)
38         buttonDelete = findViewById(R.id.buttonDelete)
39         buttonShow = findViewById(R.id.buttonShow)
40
41         buttonSave.setOnClickListener {
42             saveInfo()
43         }
44         buttonDelete.setOnClickListener {
45             deleteInfo()
46         }
47         buttonShow.setOnClickListener {
48             showInfo()
49         }
50     }
51
52     private fun saveInfo() {
53         val username = editTextUsername.text.toString()
54         val password = editTextPassword.text.toString()
55         if (username.isEmpty() && password.isEmpty()) {
56             Toast.makeText(context = this, text = "Nhập tài khoản và mật khẩu", Toast.LENGTH_SHORT).show()
57
58             preferenceHelper.saveInfo(username, password)
59             Toast.makeText(context = this, text = "Lưu thành công", Toast.LENGTH_SHORT).show()
60         }
61     }
62     private fun deleteInfo() {
63         preferenceHelper.deleteInfo()
64         editTextUsername.text.clear()
65         editTextPassword.text.clear()
66     }
67     private fun showInfo() {
68         val username = preferenceHelper.getUsername()
69         val password = preferenceHelper.getPassword()
70
71         if (username != null && password != null) {
72             Toast.makeText(context = this, text = "Username: $username, Password: $password", Toast.LENGTH_LONG).show()
73         } else {
74             Toast.makeText(context = this, text = "No info saved.", Toast.LENGTH_SHORT).show()
75         }
76     }
77 }

```

```

package com.example.bth04

import android.content.Context
import android.content.SharedPreferences

class PreferenceHelper(context: Context) {
    private val PREF_NAME = "Info"
    private val KEY_USERNAME = "username"
    private val KEY_PASSWORD = "password"

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

    fun saveInfo(username: String, password: String){
        val editor = sharedPreferences.edit()
        editor.putString(KEY_USERNAME, username)
        editor.putString(KEY_PASSWORD, password)
        editor.apply()
    }

    fun deleteInfo(){
        val editor = sharedPreferences.edit()
        editor.clear()
        editor.apply()
    }

    fun getUsername(): String? {
        return sharedPreferences.getString(KEY_USERNAME, defValue: null)
    }

    fun getPassword(): String? {
        return sharedPreferences.getString(KEY_PASSWORD, defValue: null)
    }
}

```

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".

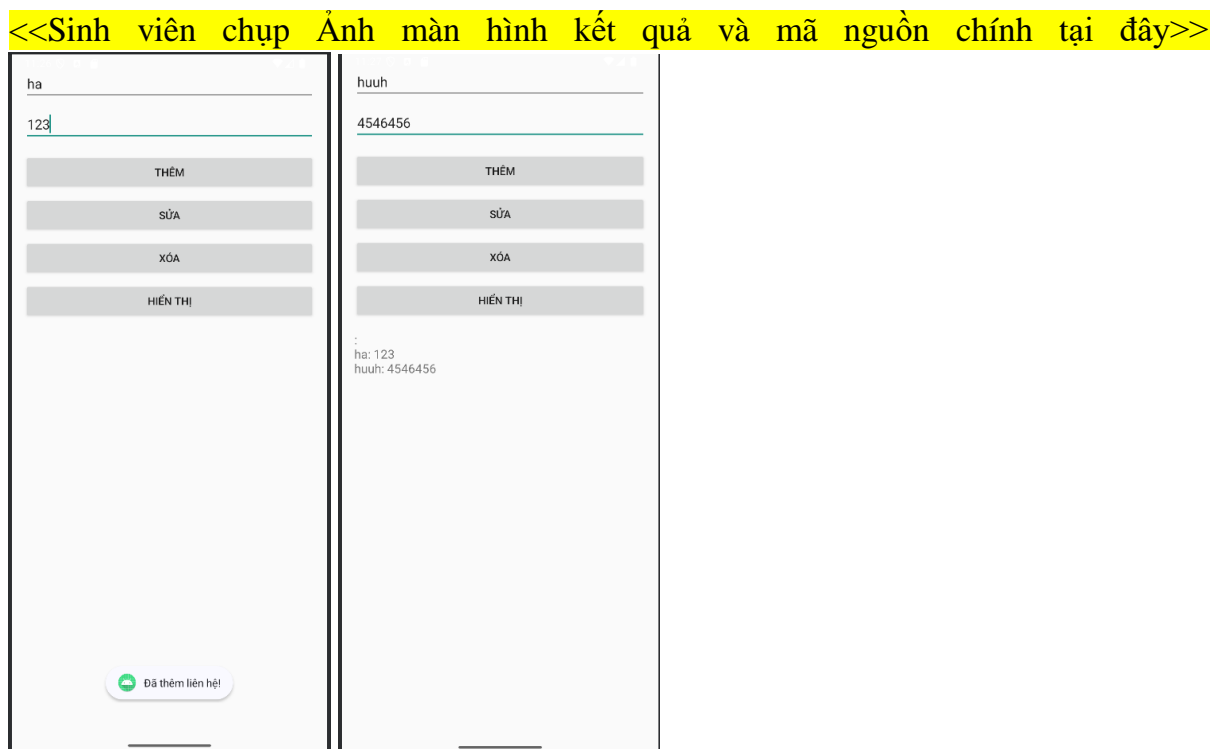
2. Sử dụng SQLite:

- 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



huuh

4546456


THÊM

SỬA

XÓA

HIỂN THỊ

:
ha: 123
huuh: 4546456

 Đã xóa liên hệ!

huuh

4546456

THÊM

SỬA

XÓA

HIỂN THỊ

:
ha: 123

```
MainActivity.kt DatabaseHelper.kt activity_main.xml
1 package com.example.btt402
2
3 > import ...
4
5
6
7
8
9
10 class MainActivity : Activity() {
11
12     lateinit var editTextName: EditText
13     lateinit var editTextPhone: EditText
14     lateinit var buttonAdd: Button
15     lateinit var buttonUpdate: Button
16     lateinit var buttonDelete: Button
17     lateinit var buttonShow: Button
18     lateinit var textViewDisplay: TextView
19     lateinit var dbHelper: DatabaseHelper
20
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContentView(R.layout.activity_main)
24
25         editTextName = findViewById(R.id.editTextName)
26         editTextPhone = findViewById(R.id.editTextPhone)
27         buttonAdd = findViewById(R.id.buttonAdd)
28         buttonUpdate = findViewById(R.id.buttonUpdate)
29         buttonDelete = findViewById(R.id.buttonDelete)
30         buttonShow = findViewById(R.id.buttonShow)
31         textViewDisplay = findViewById(R.id.textViewDisplay)
32         dbHelper = DatabaseHelper(context = this)
33
34         buttonAdd.setOnClickListener {
35             dbHelper.addContact(editTextName.text.toString(), editTextPhone.text.toString())
36             Toast.makeText(context = this, text = "Đã thêm liên hệ!", Toast.LENGTH_SHORT).show()
37         }
38
39         buttonUpdate.setOnClickListener {
40             dbHelper.updateContact(editTextName.text.toString(), editTextPhone.text.toString())
41             Toast.makeText(context = this, text = "Đã sửa liên hệ!", Toast.LENGTH_SHORT).show()
42         }
43
44         buttonDelete.setOnClickListener {
45             dbHelper.deleteContact(editTextName.text.toString())
46             Toast.makeText(context = this, text = "Đã xóa liên hệ!", Toast.LENGTH_SHORT).show()
47         }
48
49         buttonShow.setOnClickListener {
50             val contacts = dbHelper.getAllContacts()
51             textViewDisplay.text = contacts.joinToString(separator = "\n") { "${it.first}: ${it.second}" }
52         }
53     }
54 }
```



```

1 package com.example.bth402
2
3 > import ...
4
5
6
7
8 class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION) {
9
10     companion object {
11         const val DATABASE_NAME = "contacts.db"
12         const val DATABASE_VERSION = 1
13         const val TABLE_NAME = "contacts"
14         const val COLUMN_NAME = "name"
15         const val COLUMN_PHONE = "phone"
16     }
17
18     override fun onCreate(db: SQLiteDatabase) {
19         val createTable = "CREATE TABLE $TABLE_NAME ($COLUMN_NAME TEXT, $COLUMN_PHONE TEXT)"
20         db.execSQL(createTable)
21     }
22
23     override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
24         db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
25         onCreate(db)
26     }
27
28     fun addContact(name: String, phone: String) {
29         val db = writableDatabase
30         val values = ContentValues().apply {
31             put(COLUMN_NAME, name)
32             put(COLUMN_PHONE, phone)
33         }
34         db.insert(TABLE_NAME, nullColumnHack: null, values)
35         db.close()
36     }
37
38     fun updateContact(name: String, phone: String) {
39         val db = writableDatabase
40         val values = ContentValues().apply {
41             put(COLUMN_PHONE, phone)
42         }
43         db.update(TABLE_NAME, values, whereClause: "$COLUMN_NAME=?", arrayOf(name))
44         db.close()
45     }
46
47     fun deleteContact(name: String) {
48         val db = writableDatabase
49         db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME=?", arrayOf(name))
50         db.close()
51     }
52
53     fun getAllContacts(): List<Pair<String, String>> {
54         val db = readableDatabase
55         val cursor = db.query(TABLE_NAME, columns: null, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null)
56         val contacts = mutableListOf<Pair<String, String>>()
57         with(cursor) {
58             while (moveToNext()) {
59                 val name = getString(getColumnIndexOrThrow(COLUMN_NAME))
60                 val phone = getString(getColumnIndexOrThrow(COLUMN_PHONE))
61                 contacts.add(name to phone)
62             }
63         }
64         cursor.close()
65         db.close()
66         return contacts
67     }
68 }

```

```

<?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">

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Tên"
        android:inputType="textPersonName"
        android:layout_marginBottom="8dp" />

    <EditText
        android:id="@+id/editTextPhone"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Số điện thoại"
        android:inputType="phone"
        android:layout_marginBottom="16dp" />

    <Button
        android:id="@+id/buttonAdd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Thêm" />

    <Button
        android:id="@+id/buttonUpdate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sửa"
        android:layout_marginTop="8dp" />

    <Button
        android:id="@+id/buttonDelete"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Xóa"
        android:layout_marginTop="8dp" />

    <Button
        android:id="@+id/buttonShow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hiển thị"
        android:layout_marginTop="8dp" />

    <TextView
        android:id="@+id/textViewDisplay"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text=""
        android:layout_marginTop="16dp"
        android:textSize="16sp" />
</LinearLayout>

```

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

Firebase là một nền tảng phát triển ứng dụng di động và web được Google cung cấp. Được thành lập vào năm 2011 bởi Andrew Lee và James Tamplin, Firebase ban đầu là một dịch vụ đồng bộ hóa thời gian thực để hỗ trợ các ứng dụng. Sau khi được Google mua lại vào năm 2014, Firebase đã mở rộng thành một bộ công cụ toàn diện giúp các nhà phát triển tạo, quản lý và cải thiện ứng dụng của mình một cách hiệu quả.

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

Firestore Authentication

- Cho phép xác thực người dùng thông qua nhiều phương thức như email/password, số điện thoại, tài khoản Google, Facebook, hoặc Twitter.
- Dễ dàng tích hợp với các ứng dụng Android, iOS, và web.

Firebase Realtime Database và Cloud Firestore

- *Realtime Database*: Cơ sở dữ liệu NoSQL cho phép đồng bộ hóa dữ liệu trực tiếp giữa người dùng và thiết bị.
- *Cloud Firestore*: Một phiên bản nâng cấp của Realtime Database, hỗ trợ các truy vấn phức tạp và tích hợp tốt với các dịch vụ khác của Google.

Firebase Cloud Functions

- Cho phép viết mã backend serverless, chạy trên cơ sở hạ tầng đám mây của Google.
- Thường được dùng để xử lý các sự kiện (ví dụ: gửi email khi người dùng đăng ký).

Firebase Cloud Messaging (FCM)

- Hỗ trợ gửi thông báo đẩy (push notifications) đến các thiết bị người dùng.
- Hoạt động hiệu quả trên nhiều nền tảng như Android, iOS, và web.

Firebase Storage

- Cung cấp dịch vụ lưu trữ tệp tin như hình ảnh, video, và tài liệu.
- Tối ưu hóa tốc độ tải xuống và bảo mật thông qua Google Cloud Storage.

Firebase Machine Learning (ML)

- Tích hợp các mô hình AI như nhận diện khuôn mặt, dịch văn bản, và phân tích hình ảnh.
- Giúp nhà phát triển xây dựng các ứng dụng thông minh mà không cần kiến thức chuyên sâu về học máy.

3. Lợi ích và ứng dụng của Firebase

Lợi ích:

- Tiết kiệm thời gian: Firebase cung cấp giải pháp toàn diện giúp nhà phát triển tập trung vào chức năng ứng dụng.
- Dễ tích hợp: Hỗ trợ đa nền tảng với các SDK và API dễ sử dụng.
- Bảo mật: Google cung cấp khả năng mã hóa mạnh mẽ và kiểm soát truy cập.

Ứng dụng:

- Firebase thường được sử dụng để xây dựng ứng dụng thời gian thực như trò chuyện trực tuyến, chia sẻ dữ liệu, hoặc ứng dụng học tập.
- Với Firebase ML, các ứng dụng như nhận diện hình ảnh hoặc dịch thuật trở nên phổ biến.

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).

```
jl/android/maven2/com/google/android/gms/play-services-measurement-impl/22.4.0/play-services-measurement-impl-22.4.0.aar, took 6 s 45 ms
jl/android/maven2/com/google/android/gms/play-services-measurement/22.4.0/play-services-measurement-22.4.0-javadoc.jar, took 1 s 389 ms
jl/android/maven2/com/google/android/gms/play-services-measurement-impl/22.4.0/play-services-measurement-impl-22.4.0-javadoc.jar, took 177 ms
jl/android/maven2/com/google/android/gms/play-services-measurement-api/22.4.0/play-services-measurement-api-22.4.0-javadoc.jar, took 1 s 705 ms

used in this build, making it incompatible with Gradle 9.0.

' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.

to https://docs.gradle.org/8.10.2/userguide/command\_line\_interface.html#sec:command\_line\_warnings in the Gradle documentation.
```

BTTH4 ▾

Authentication

Users Sign-in method Templates Usage Settings Extensions

Sign-in providers

Add new provider

Provider

Status

✉ Email/Password

✓ Enabled

👤 Anonymous

✓ Enabled

🔗 <https://btth4-3ce65-default-rtdb.asia-southeast1.firebaseio.com>

<https://btth4-3ce65-default-rtdb.asia-southeast1.firebaseio.com/>: null

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:

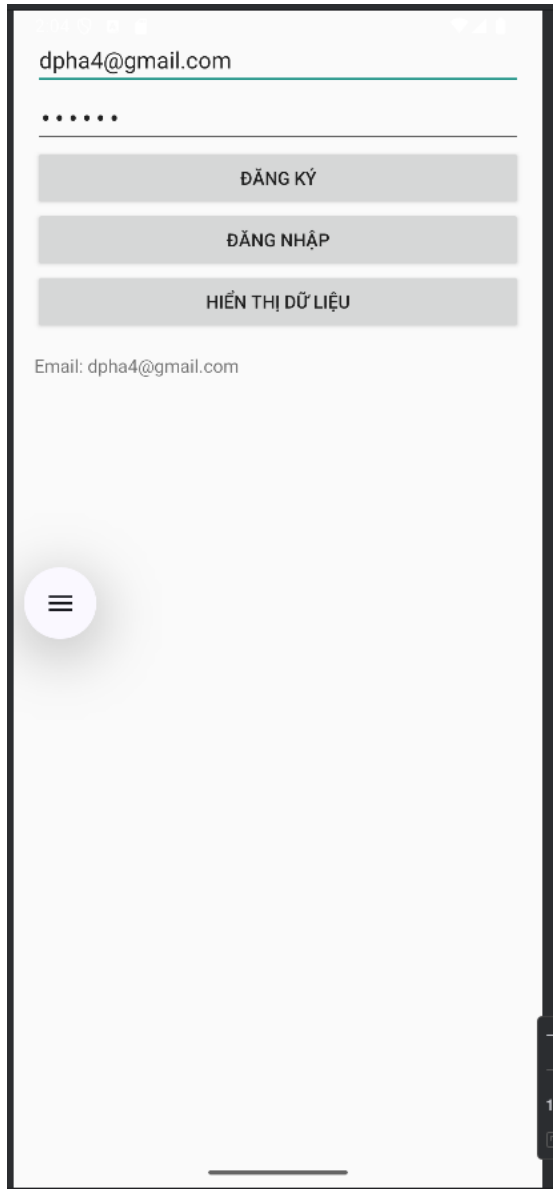
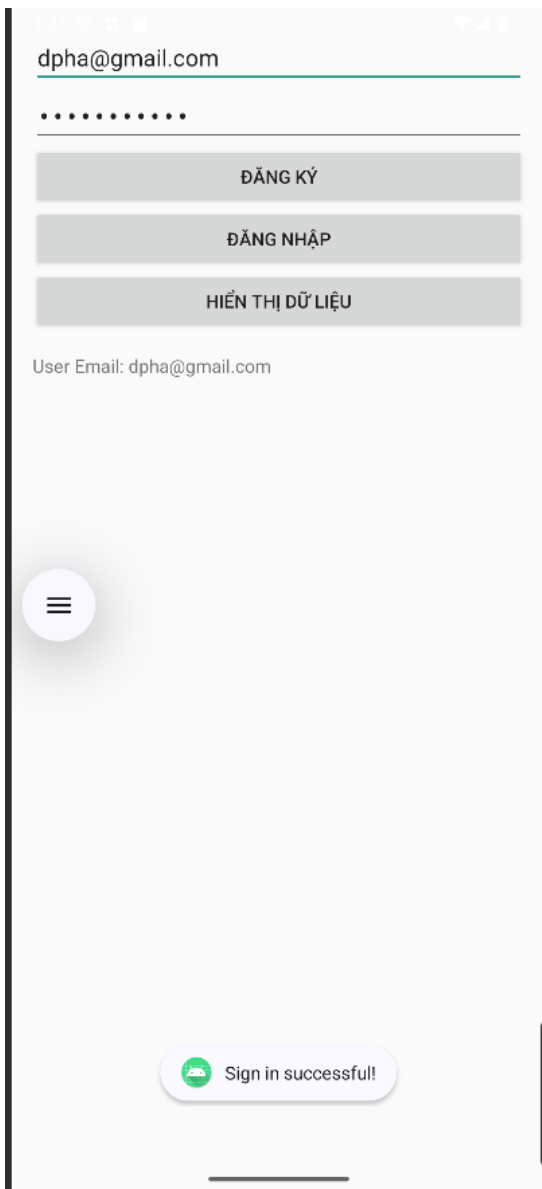
- Sử dụng Firebase Authentication để cho phép người dùng đăng ký và đăng nhập bằng email và mật khẩu.
- Viết mã để xử lý các sự kiện đăng ký và đăng nhập thành công hoặc thất bại.

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>>



```
package com.example.bth403

> import ...

class MainActivity : Activity() {

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

    private lateinit var editTextEmail: EditText
    private lateinit var editTextPassword: EditText
    private lateinit var buttonSignUp: Button
    private lateinit var buttonSignIn: Button
    private lateinit var buttonShowData: Button
    private lateinit var textViewData: TextView

    companion object {
        private const val TAG = "MainActivity"
    }

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

        auth = FirebaseAuth
        database = FirebaseDatabase.reference

        editTextEmail = findViewById(R.id.editTextEmail)
        editTextPassword = findViewById(R.id.editTextPassword)
        buttonSignUp = findViewById(R.id.buttonSignUp)
        buttonSignIn = findViewById(R.id.buttonSignIn)
        buttonShowData = findViewById(R.id.buttonShowData)
        textViewData = findViewById(R.id.textViewData)

        buttonSignUp.setOnClickListener { signUpUser() }
        buttonSignIn.setOnClickListener { signInUser() }
        buttonShowData.setOnClickListener { showData() }
    }

    private fun signUpUser() {
        val email = editTextEmail.text.toString().trim()
    }
}
```



```

class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
    }

    private fun signUpUser() {
        val email = editTextEmail.text.toString().trim()
        val password = editTextPassword.text.toString().trim()
        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(context: this, text: "Vui lòng nhập email và mật khẩu", Toast.LENGTH_SHORT).s
            return
        }

        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    Log.d(TAG, msg: "Đăng ký thành công")
                    val user = auth.currentUser
                    saveUserData(user?.uid, email)
                } else {
                    Log.e(TAG, msg: "Đăng ký thất bại: ${task.exception?.message}", task.exception)
                    Toast.makeText(context: this, text: "Đăng ký thất bại: ${task.exception?.message}",
                }
            }
    }

    private fun signInUser() {
        val email = editTextEmail.text.toString().trim()
        val password = editTextPassword.text.toString().trim()

        if (email.isEmpty() || password.isEmpty()) {
            Toast.makeText(context: this, text: "Vui lòng nhập email và mật khẩu", Toast.LENGTH_SHORT).s
            return
        }

        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    Log.d(TAG, msg: "Đăng nhập thành công")
                    val user = auth.currentUser
                } else {
                    Log.e(TAG, msg: "Đăng nhập thất bại: ${task.exception?.message}", task.exception)
                    Toast.makeText(context: this, text: "Đăng nhập thất bại: ${task.exception?.message}"

```

```

        val user = auth.currentUser
    } else {
        Log.e(TAG, msg: "Đăng nhập thất bại: ${task.exception?.message}", task.exception)
        Toast.makeText(context: this, text: "Đăng nhập thất bại: ${task.exception?.message}"
        }
    }
}

private fun saveUserData(userId: String?, email: String?) {
    if (userId == null || email == null) {
        Log.e(TAG, msg: "Không thể lưu dữ liệu: userId hoặc email null")
        Toast.makeText(context: this, text: "Không thể lưu dữ liệu người dùng", Toast.LENGTH_SHORT).
        return
    }

    val user = User(email)

    database.child(pathString: "users").child(userId).setValue(user)
        .addOnSuccessListener {
            Log.d(TAG, msg: "Lưu dữ liệu người dùng thành công")
            Toast.makeText(context: this, text: "Lưu dữ liệu người dùng thành công", Toast.LENGTH_SH
        }
        .addOnFailureListener { e ->
            Log.e(TAG, msg: "Lỗi khi lưu dữ liệu người dùng: ${e.message}", e)
            Toast.makeText(context: this, text: "Lỗi khi lưu dữ liệu người dùng: ${e.message}", Toas
        }
}

private fun showData() {
    val userId = auth.currentUser?.uid
    if (userId == null) {
        Log.e(TAG, msg: "Không thể hiển thị dữ liệu: Người dùng chưa đăng nhập")
        Toast.makeText(context: this, text: "Vui lòng đăng nhập để xem dữ liệu", Toast.LENGTH_SHORT)
        return
    }

    database.child(pathString: "users").child(userId).addValueEventListener(object : ValueEventListener
        override fun onDataChange(snapshot: DataSnapshot) {
            val user = snapshot.getValue(User::class.java)

```

```

val userId = auth.currentUser?.uid
if (userId == null) {
    Log.e(TAG, msg: "Không thể hiển thị dữ liệu: Người dùng chưa đăng nhập")
    Toast.makeText(context: this, text: "Vui lòng đăng nhập để xem dữ liệu", Toast.LENGTH_SHORT)
    return
}

database.child(pathString: "users").child(userId).addValueEventListener(object : ValueEventListener {
    override fun onDataChange(snapshot: DataSnapshot) {
        val user = snapshot.getValue(User::class.java)
        if (user != null) {
            textViewData.text = "Email: ${user.email}"
        } else {
            Log.w(TAG, msg: "Không tìm thấy dữ liệu người dùng")
            textViewData.text = "Không tìm thấy dữ liệu người dùng"
        }
    }

    override fun onCancelled(error: DatabaseError) {
        Log.e(TAG, msg: "Lỗi khi đọc dữ liệu: ${error.message}", error.toException())
        Toast.makeText(context: this@MainActivity, text: "Lỗi khi đọc dữ liệu: ${error.message}"
    }
})
}

data class User(val email: String? = null)

```