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	2151060213	Nguyễn Khắc Toàn	63CNTT2

Hà Nội, năm 2025

BÀI TẬP 1: SHARED PREFERENCE

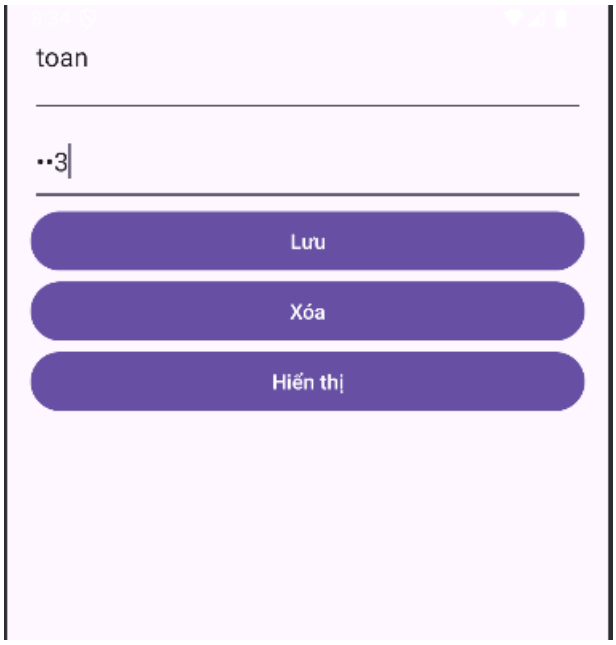
Mục tiêu:

- 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.

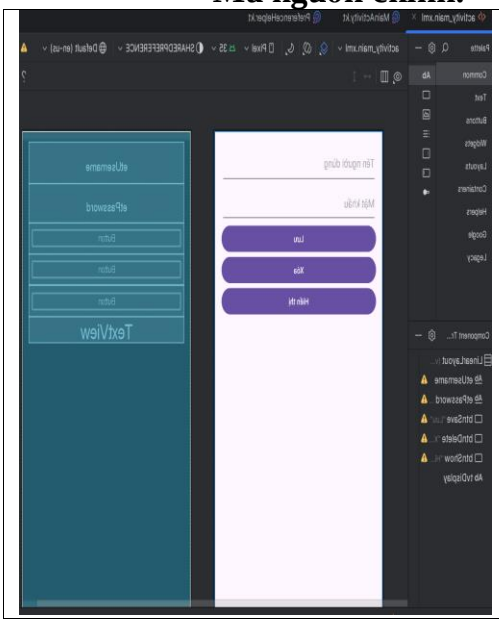
Yêu cầu:

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.

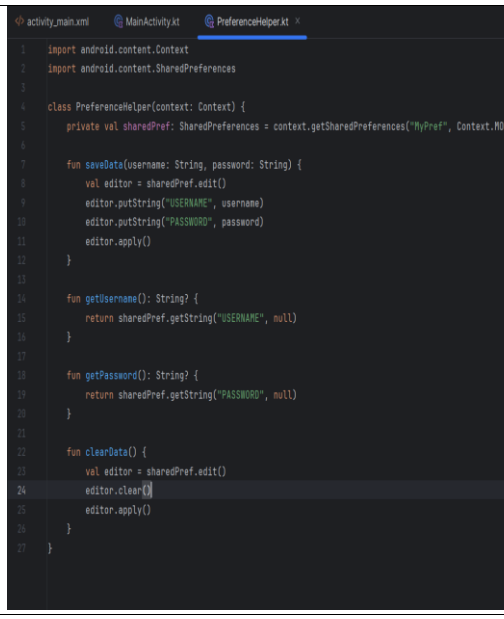
4. Kết quả



Mã nguồn chính:



```
class MainActivity : AppCompatActivity() {

    private lateinit var etUsername: EditText
    private lateinit var etPassword: EditText
    private lateinit var btnSave: Button
    private lateinit var btnDelete: Button
    private lateinit var btnShow: Button
    private lateinit var tvDisplay: TextView
    private lateinit var preferenceHelper: PreferenceHelper

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

        etUsername = findViewById(R.id.etUsername)
        etPassword = findViewById(R.id.etPassword)
        btnSave = findViewById(R.id.btnSave)
        btnDelete = findViewById(R.id.btnDelete)
        btnShow = findViewById(R.id.btnShow)
        tvDisplay = findViewById(R.id.tvDisplay)

        preferenceHelper = PreferenceHelper( context: this)

        btnSave.setOnClickListener {
            val username = etUsername.text.toString()
            val password = etPassword.text.toString()
            if (username.isNotEmpty() && password.isNotEmpty()) {
                preferenceHelper.saveData(username, password)
                Toast.makeText( context: this,  text: "Đã lưu dữ liệu", Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText( context: this,  text: "Vui lòng nhập đầy đủ thông tin", Toast.LENGTH_SHORT).show()
            }
```



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

class PreferenceHelper(context: Context) {
    private val sharedPref: SharedPreferences = context.getSharedPreferences("MyPref", Context.MO

    fun saveData(username: String, password: String) {
        val editor = sharedPref.edit()
        editor.putString("USERNAME", username)
        editor.putString("PASSWORD", password)
        editor.apply()
    }

    fun getUsername(): String? {
        return sharedPref.getString("USERNAME", null)
    }

    fun getPassword(): String? {
        return sharedPref.getString("PASSWORD", null)
    }

    fun clearData() {
        val editor = sharedPref.edit()
        editor.clear()
        editor.apply()
    }
}
```

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".

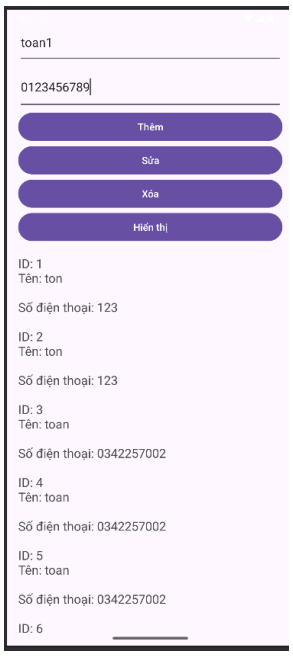
2. Sử dụng SQLite:

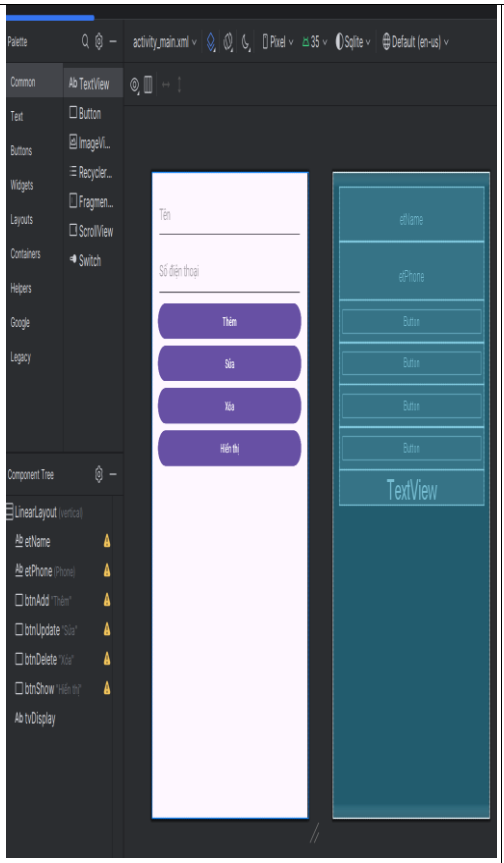
- Tạo một lớp helper để quản lý cơ sở dữ liệu SQLite.
- Tạo bảng dữ liệu với hai cột: tên và số điện thoại.
- Viết các hàm để thêm, sửa, xóa dữ liệu từ cơ sở dữ liệu.
- Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ cơ sở dữ liệu và hiển thị lên màn hình.

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






```
class MainActivity : AppCompatActivity() {
    private lateinit var etPhone: EditText
    private lateinit var btnAdd: Button
    private lateinit var btnUpdate: Button
    private lateinit var btnDelete: Button
    private lateinit var btnShow: Button
    private lateinit var tvDisplay: TextView
    private lateinit var dbHelper: DatabaseHelper

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

        etName = findViewById(R.id.etName)
        etPhone = findViewById(R.id.etPhone)
        btnAdd = findViewById(R.id.btnAdd)
        btnUpdate = findViewById(R.id.btnUpdate)
        btnDelete = findViewById(R.id.btnDelete)
        btnShow = findViewById(R.id.btnShow)
        tvDisplay = findViewById(R.id.tvDisplay)

        dbHelper = DatabaseHelper( context: this)

        btnAdd.setOnClickListener {
            val name = etName.text.toString()
            val phone = etPhone.text.toString()
            if (name.isNotEmpty() && phone.isNotEmpty()) {
                val result = dbHelper.addContact(name, phone)
                if (result != -1L) {
                    Toast.makeText( context: this, text: "Đã thêm liên hệ", Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText( context: this, text: "Lỗi khi thêm liên hệ", Toast.LENGTH_SHORT).show
```

```
class DatabaseHelper(context: Context) : SQLiteOpenHel
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVer
        db?.execSQL( sql: "DROP TABLE IF EXISTS $TABLE_N
        onCreate(db)
    }

    fun addContact(name: String, phone: String): Long
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put(COLUMN_NAME, name)
        contentValues.put(COLUMN_PHONE, phone)
        return db.insert(TABLE_NAME,  nullColumnHack: nul
    }

    fun updateContact(id: Int, name: String, phone: St
        val db = this.writableDatabase
        val contentValues = ContentValues()
        contentValues.put(COLUMN_NAME, name)
        contentValues.put(COLUMN_PHONE, phone)
        return db.update(TABLE_NAME, contentValues,  w
    }

    fun deleteContact(id: Int): Int {
        val db = this.writableDatabase
        return db.delete(TABLE_NAME,  whereClause: "$COLU
    }

    fun getContacts(): Cursor {
        val db = this.readableDatabase
        return db.rawQuery( sql: "SELECT * FROM $TABLE_N
```

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.

1. Giới thiệu tổng quan về Firebase

Firebase là nền tảng phát triển ứng dụng di động và web do Google cung cấp, giúp các lập trình viên xây dựng ứng dụng nhanh chóng mà không cần quá nhiều công sức trong việc thiết lập backend. Firebase cung cấp nhiều dịch vụ khác nhau như cơ sở dữ liệu thời gian thực, xác thực người dùng, lưu trữ đám mây, nhắn tin đẩy và các công cụ hỗ trợ trí tuệ nhân tạo.

Firebase được ra mắt vào năm 2011 và được Google mua lại vào năm 2014, từ đó phát triển thành một hệ sinh thái mạnh mẽ hỗ trợ lập trình viên trong việc phát triển ứng dụng.

2. Mô tả các dịch vụ chính của Firebase

2.1 Firebase Authentication

Dịch vụ xác thực giúp quản lý đăng nhập người dùng thông qua email/password, số điện thoại, và các nền tảng mạng xã hội như Google,

Facebook, Twitter... Điều này giúp lập trình viên dễ dàng triển khai hệ thống đăng nhập mà không cần tự xây dựng từ đầu.

2.2 Firebase Realtime Database và Cloud Firestore

- **Realtime Database: Cơ sở dữ liệu NoSQL, cho phép đồng bộ dữ liệu theo thời gian thực giữa các client.**
- **Cloud Firestore: Phiên bản nâng cấp của Realtime Database, hỗ trợ mạnh mẽ hơn trong việc truy vấn và xử lý dữ liệu.**

2.3 Firebase Cloud Functions

Dịch vụ này cho phép chạy các đoạn mã backend trên cloud mà không cần quản lý server. Cloud Functions thường được sử dụng để xử lý các sự kiện tự động như gửi thông báo khi có dữ liệu mới, xác thực người dùng, hoặc tích hợp với các dịch vụ bên thứ ba.

2.4 Firebase Cloud Messaging (FCM)

Cung cấp tính năng gửi thông báo đẩy đến thiết bị di động hoặc trình duyệt web. FCM thường được dùng để gửi thông báo từ ứng dụng đến người dùng, giúp nâng cao trải nghiệm và mức độ tương tác.

2.5 Firebase Storage

Dịch vụ lưu trữ dữ liệu trên đám mây, phù hợp để lưu trữ các tệp tin như hình ảnh, video, âm thanh... Firebase Storage có tích hợp sẵn tính năng bảo mật và quyền truy cập.

2.6 Firebase Machine Learning (ML)

Tích hợp trí tuệ nhân tạo vào ứng dụng một cách dễ dàng, giúp thực hiện các tác vụ như nhận diện hình ảnh, xử lý văn bản, dịch ngôn ngữ, phân tích dữ liệu... mà không cần kiến thức chuyên sâu về AI.

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

Firebase mang lại nhiều lợi ích quan trọng trong quá trình phát triển ứng dụng:

- **Tăng tốc phát triển: Các dịch vụ có sẵn giúp lập trình viên tập trung vào phát triển tính năng thay vì quản lý backend.**
- **Hỗ trợ đa nền tảng: Firebase hỗ trợ cả Android, iOS và Web, giúp ứng dụng có thể hoạt động trên nhiều thiết bị.**
- **Bảo mật tốt: Firebase tích hợp các cơ chế bảo mật mạnh mẽ như xác thực người dùng, kiểm soát truy cập dữ liệu.**

- **Khả năng mở rộng linh hoạt: Hệ thống được xây dựng trên hạ tầng của Google, có thể mở rộng tùy theo nhu cầu của ứng dụng.**
- **Hỗ trợ mạnh mẽ cho ứng dụng thời gian thực: Các dịch vụ như Realtime Database và Cloud Firestore giúp xây dựng ứng dụng chat, hệ thống theo dõi đơn hàng, hoặc bất kỳ ứng dụng nào yêu cầu đồng bộ dữ liệu nhanh.**

4. Kết luận

Firebase là một hệ sinh thái mạnh mẽ giúp lập trình viên phát triển ứng dụng một cách nhanh chóng, an toàn và hiệu quả. Với nhiều dịch vụ hữu ích, Firebase phù hợp cho cả dự án nhỏ lẫn ứng dụng có quy mô lớn. Việc tích hợp Firebase vào ứng dụng sẽ giúp cải thiện trải nghiệm người dùng và tối ưu hóa quá trình phát triển phần mềm.

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>>