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	2251061731	Lường Văn Cương	64CNTT2

Hà Nội, năm 2025

BÀI TẬP 1: SHARED PREFERENCE

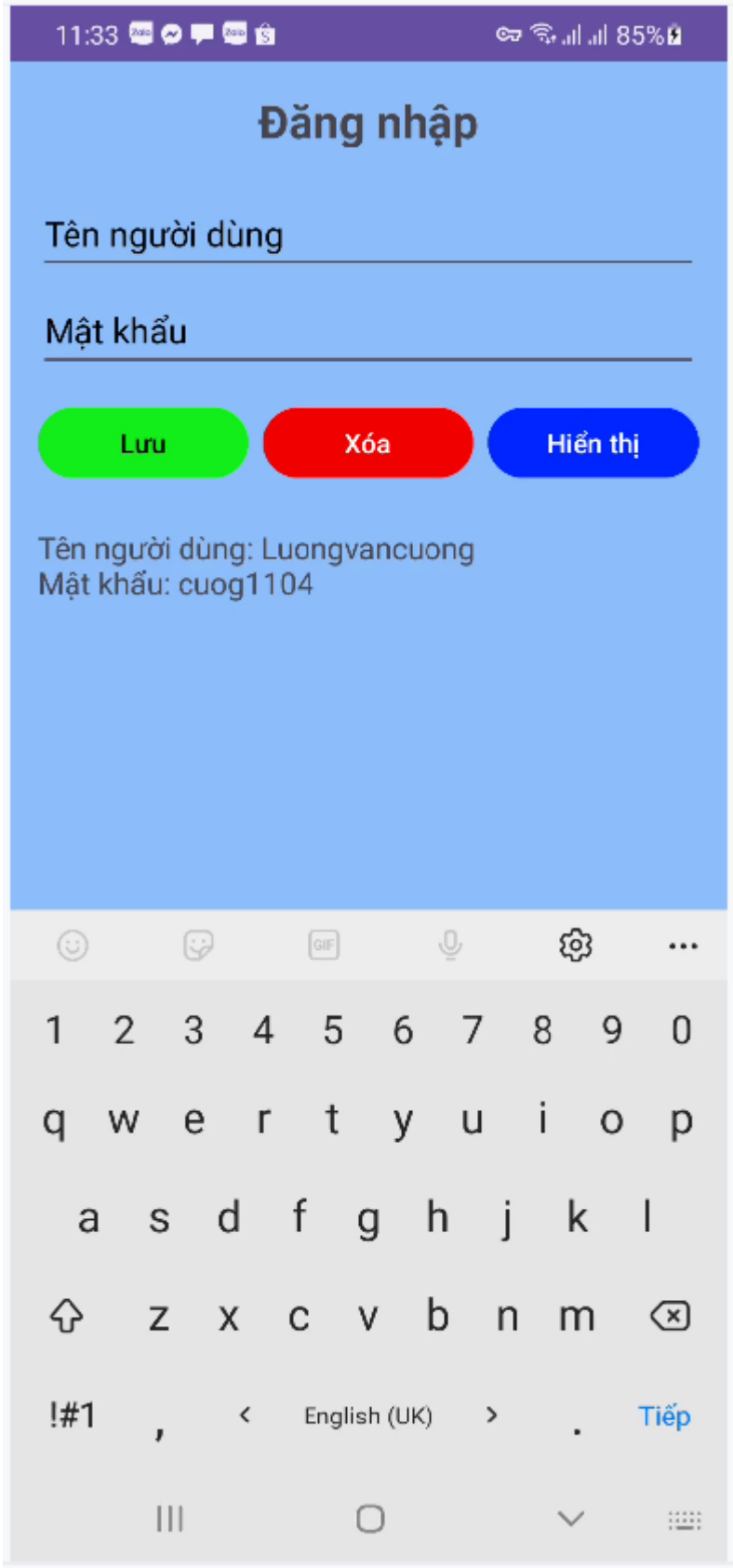
Mục tiêu:

- 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


11:33
85%
Đăng nhập
Tên người dùng
Mật khẩu
Lưu
Xóa
Hiển thị
Tên người dùng: Luongvancuong
Mật khẩu: cuog1104
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
!#1
English (UK)
Tiếp

```
//Code của lớp MainActivity.kt
package vn.edu.tlu.ex1
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
class MainActivity : AppCompatActivity() {
    private lateinit var etUsername: EditText
    private lateinit var etPassword: EditText
    private lateinit var btnSave: Button
    private lateinit var btnClear: Button
    private lateinit var btnShow: Button
    private lateinit var tvResult: TextView
    private lateinit var preferenceHelper: PreferenceHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // Khởi tạo views
        etUsername = findViewById(R.id.etUsername)
        etPassword = findViewById(R.id.etPassword)
        btnSave = findViewById(R.id.btnSave)
        btnClear = findViewById(R.id.btnClear)
        btnShow = findViewById(R.id.btnShow)
        tvResult = findViewById(R.id.tvResult)
        // Khởi tạo PreferenceHelper
        preferenceHelper = PreferenceHelper(this)
        // Xử lý sự kiện khi nhấn nút Lưu
        btnSave.setOnClickListener {
            val username = etUsername.text.toString()
            val password = etPassword.text.toString()
```

```
            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Vui lòng nhập đầy đủ thông tin",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            preferenceHelper.saveUserCredentials(username, password)
            Toast.makeText(this, "Đã lưu thông tin thành công",
Toast.LENGTH_SHORT).show()
            // Xóa trường nhập sau khi lưu
            etUsername.setText("")
            etPassword.setText("")
        }
        // Xử lý sự kiện khi nhấn nút Xóa
        btnClear.setOnClickListener {
            preferenceHelper.clearUserCredentials()
            tvResult.text = ""
            Toast.makeText(this, "Đã xóa thông tin", Toast.LENGTH_SHORT).show()
        }
        // Xử lý sự kiện khi nhấn nút Hiển thị
        btnShow.setOnClickListener {
            val (username, password) = preferenceHelper.getUserCredentials()
            if (username != null && password != null) {
                tvResult.text = "Tên người dùng: $username\nMật khẩu: $password"
            } else {
                tvResult.text = "Không có thông tin người dùng"
            }
        }
    }
}
```

Code Hàm lớp Helper

```kotlin
package vn.edu.tlu.ex1
import android.content.Context
import android.content.SharedPreferences
class PreferenceHelper(context: Context) {
    private val PREF_NAME = "UserPrefs"
    private val KEY_USERNAME = "username"
    private val KEY_PASSWORD = "password"
    private val sharedPreferences: SharedPreferences = context.getSharedPreferences(PREF_NAME, Context.MODE_PRIVATE)
    fun saveUserCredentials(username: String, password: String) {
        val editor = sharedPreferences.edit()
        editor.putString(KEY_USERNAME, username)
        editor.putString(KEY_PASSWORD, password)
        editor.apply()
    }
    fun getUserCredentials(): Pair<String?, String?> {
        val username = sharedPreferences.getString(KEY_USERNAME, null)
        val password = sharedPreferences.getString(KEY_PASSWORD, null)
        return Pair(username, password)
    }
    fun clearUserCredentials() {
        val editor = sharedPreferences.edit()
        editor.remove(KEY_USERNAME)
        editor.remove(KEY_PASSWORD)
        editor.apply()
    }
}
```

BÀI TẬP 2: SQLite

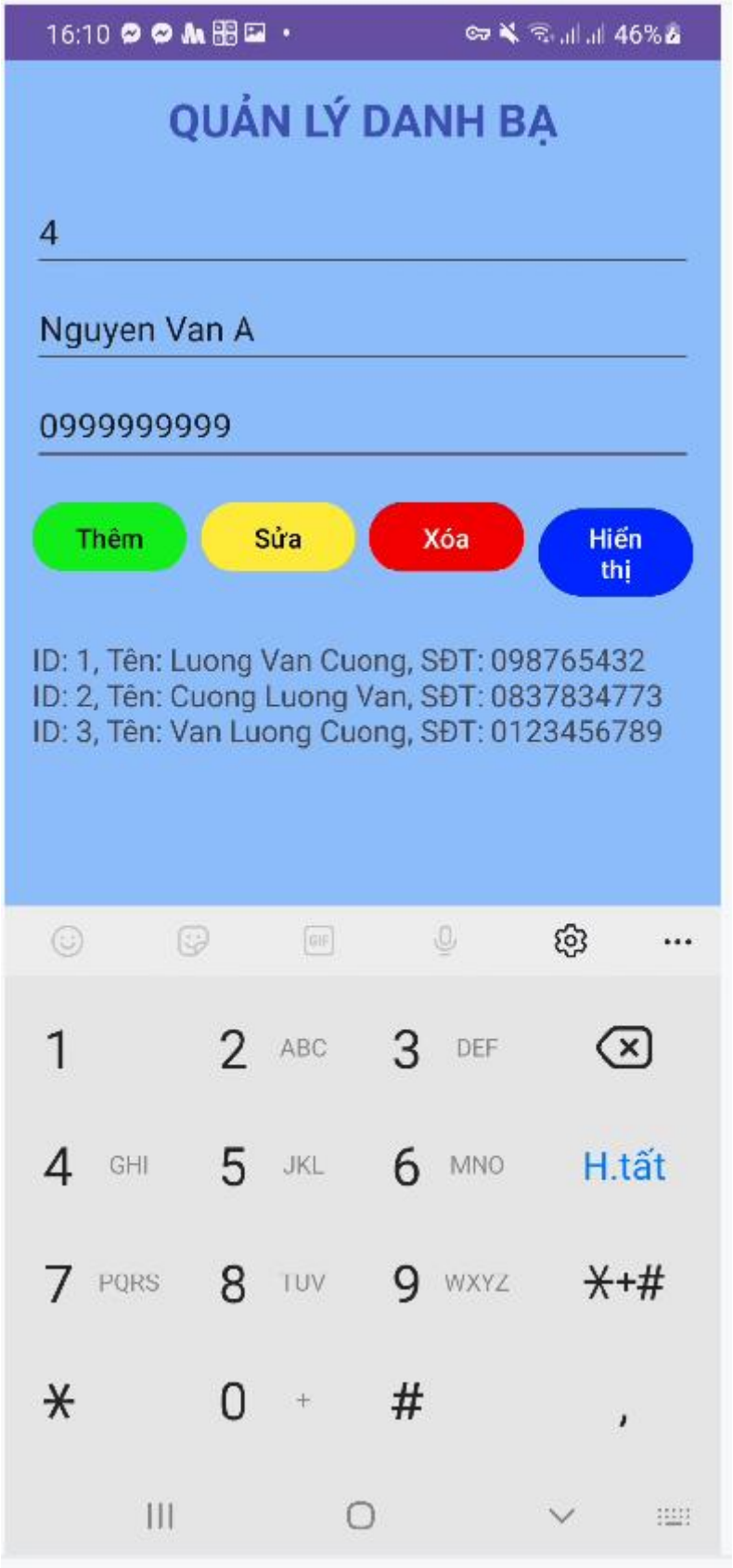
Mục tiêu:

- 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


16:10
46%
QUẢN LÝ DANH BẠ
4
Nguyen Van A
0999999999
Thêm
Sửa
Xóa
Hiển thị
ID: 1, Tên: Luong Van Cuong, SĐT: 098765432
ID: 2, Tên: Cuong Luong Van, SĐT: 0837834773
ID: 3, Tên: Van Luong Cuong, SĐT: 0123456789
H.tất

Lớp Helper

```
//Lớp Helper.kt
package vn.edu.tlu.ex2
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,
null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_NAME = "ContactDB"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "contacts"
        private const val KEY_ID = "id"
        private const val KEY_NAME = "name"
        private const val KEY_PHONE = "phone"
    }
    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = """
            CREATE TABLE $TABLE_NAME (
                $KEY_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $KEY_NAME TEXT,
                $KEY_PHONE TEXT
            )
        """
        db?.execSQL(createTable)
    }
    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int)
{
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }
```

```
    fun addContact(name: String, phone: String): Long {
        val db = this.writableDatabase
        val values = ContentValues().apply {
            put(KEY_NAME, name)
            put(KEY_PHONE, phone)
        }
        val id = db.insert(TABLE_NAME, null, values)
        db.close()
        return id
    }
    fun updateContact(id: Long, name: String, phone: String): Int {
        val db = this.writableDatabase
        val values = ContentValues().apply {
            put(KEY_NAME, name)
            put(KEY_PHONE, phone)
        }
        val rowsAffected = db.update(TABLE_NAME, values, "$KEY_ID = ?",
arrayOf(id.toString()))
        db.close()
        return rowsAffected
    }
    fun deleteContact(id: Long): Int {
        val db = this.writableDatabase
        val rowsAffected = db.delete(TABLE_NAME, "$KEY_ID = ?",
arrayOf(id.toString()))
        db.close()
        return rowsAffected
    }
    fun getAllContacts(): List<String> {
        val contactList = mutableListOf<String>()
        val db = this.readableDatabase
```

```
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val id = cursor.getLong(cursor.getColumnIndexOrThrow(KEY_ID))
                val name =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_NAME))
                val phone =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_PHONE))
                contactList.add("ID: $id, Tên: $name, SĐT: $phone")
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return contactList
    }
}
```

Lớp MainActivity

```
package vn.edu.tlu.ex2

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

class MainActivity : AppCompatActivity() {

    private lateinit var etId: EditText

    private lateinit var etName: EditText

    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 tvResult: TextView

    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        // Khởi tạo views

        etId = findViewById(R.id.etId)

        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)

        tvResult = findViewById(R.id.tvResult)
```

```
        // Khởi tạo DatabaseHelper
        dbHelper = DatabaseHelper(this)
        // Xử lý nút Thêm
        btnAdd.setOnClickListener {
            val name = etName.text.toString().trim()
            val phone = etPhone.text.toString().trim()
            if (name.isEmpty() || phone.isEmpty()) {
                Toast.makeText(this, "Vui lòng nhập đầy đủ thông tin",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            val id = dbHelper.addContact(name, phone)
            if (id > 0) {
                Toast.makeText(this, "Đã thêm liên hệ",
Toast.LENGTH_SHORT).show()
                clearInputs()
            }
        }
        // Xử lý nút Sửa
        btnUpdate.setOnClickListener {
            val idText = etId.text.toString().trim()
            val name = etName.text.toString().trim()
            val phone = etPhone.text.toString().trim()
            if (idText.isEmpty() || name.isEmpty() || phone.isEmpty()) {
                Toast.makeText(this, "Vui lòng nhập ID và đầy đủ thông tin",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            val id = idText.toLongOrNull()
            if (id == null) {
```

```
                Toast.makeText(this, "ID không hợp lệ",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            val rowsAffected = dbHelper.updateContact(id, name, phone)
            if (rowsAffected > 0) {
                Toast.makeText(this, "Đã cập nhật liên hệ",
Toast.LENGTH_SHORT).show()
                clearInputs()
            } else {
                Toast.makeText(this, "Không tìm thấy liên hệ với ID: $id",
Toast.LENGTH_SHORT).show()
            }
        }
        // Xử lý nút Xóa
        btnDelete.setOnClickListener {
            val idText = etId.text.toString().trim()

            if (idText.isEmpty()) {
                Toast.makeText(this, "Vui lòng nhập ID để xóa",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            val id = idText.toLongOrNull()
            if (id == null) {
                Toast.makeText(this, "ID không hợp lệ",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            val rowsAffected = dbHelper.deleteContact(id)
            if (rowsAffected > 0) {
```

```
                Toast.makeText(this, "Đã xóa liên hệ", Toast.LENGTH_SHORT).show()
                clearInputs()
            } else {
                Toast.makeText(this, "Không tìm thấy liên hệ với ID: $id",
Toast.LENGTH_SHORT).show()
            }
        }
        // Xử lý nút Hiển thị
        btnShow.setOnClickListener {
            val contacts = dbHelper.getAllContacts()
            if (contacts.isEmpty()) {
                tvResult.text = "Không có liên hệ nào"
            } else {
                tvResult.text = contacts.joinToString("\n")
            }
        }
    }
    private fun clearInputs() {
        etId.setText("")
        etName.setText("")
        etPhone.setText("")
    }
}
```

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.

Giới thiệu tổng quan về Firebase và lịch sử phát triển:
- Tổng quan: Firebase là một nền tảng phát triển ứng dụng di động và web được Google cung cấp, giúp các lập trình viên xây dựng, quản lý và mở rộng ứng dụng một cách nhanh chóng mà không cần tập trung quá nhiều vào hạ tầng phía sau. Firebase cung cấp các dịch vụ như cơ sở dữ liệu thời gian thực, xác thực người dùng, lưu trữ tệp, thông báo đẩy, phân tích ứng dụng và nhiều công cụ hỗ trợ khác.
- Lịch sử phát triển:
+ 2011: Firebase được thành lập bởi James Tamplin và Andrew Lee với tên ban đầu là Envolve, cung cấp một dịch vụ chat thời gian thực dành cho website.

+ 2012: Công ty nhận thấy rằng nhiều lập trình viên sử dụng Envolve không chỉ để chat mà còn để đồng bộ hóa dữ liệu theo thời gian thực giữa các ứng dụng. Vì vậy, họ quyết định chuyển hướng thành một nền tảng cung cấp cơ sở dữ liệu thời gian thực.

+ 2014: Firebase chính thức được Google mua lại, giúp nền tảng này phát triển mạnh mẽ hơn với nhiều tính năng mở rộng.

+ 2016: Firebase được tích hợp sâu vào hệ sinh thái của Google, mở rộng từ một dịch vụ cơ sở dữ liệu trở thành một nền tảng phát triển ứng dụng toàn diện, bao gồm xác thực, phân tích, lưu trữ, hosting, v.v.

+ 2020 - Nay: Firebase tiếp tục cải tiến, tích hợp với Google Cloud, AI/ML và cung cấp nhiều công cụ hỗ trợ cho lập trình viên.

Mô tả chi tiết từng dịch vụ chính của Firebase:

- Cơ sở dữ liệu và lưu trữ:

+ Realtime Database: Là cơ sở dữ liệu **NoSQL** lưu trữ dữ liệu dưới dạng **JSON** và đồng bộ theo **thời gian thực** giữa các client.

+ Cloud Firestore: Là phiên bản nâng cấp của Realtime Database, hỗ trợ lưu trữ dữ liệu theo dạng **document** (tài liệu) và **collection** (bộ sưu tập).

+ Cloud Storage : Cung cấp khả năng **lưu trữ và chia sẻ tệp (hình ảnh, video, tài liệu, v.v.)** trên đám mây của Google.

- Xác thực và bảo mật:

+ Firebase Authentication: Dịch vụ xác thực người dùng hỗ trợ nhiều phương thức

+ Firebase Security Rules: Hệ thống bảo mật giúp kiểm soát quyền truy cập dữ liệu trong Realtime Database, Firestore và Storage.

- Phân tích và tối ưu hóa ứng dụng:

+ Google Analytics for Firebase: Cung cấp số liệu thống kê chi tiết về hành vi của người dùng trong ứng dụng.

+ Firebase Performance Monitoring: theo dõi và phân tích hiệu suất ứng dụng theo **thời gian thực**.

+ Firebase Crashlytics: Dịch vụ **phát hiện và báo cáo lỗi** trong ứng dụng.

- Dịch vụ đám mây và thông báo

+ Firebase Cloud Messaging (FCM): Dịch vụ gửi **thông báo đẩy miễn phí** đến người dùng trên Android, iOS và Web.

+ Firebase Remote Config: Cho phép **tùy chỉnh giao diện, chức năng ứng dụng** mà không cần cập nhật phiên bản mới.

+ Firebase Cloud Functions: Cho phép viết **code backend trên cloud** mà không cần server riêng.

- Công cụ phát triển và kiểm thử

+ Firebase App Distribution: Giúp phân phối **các bản thử nghiệm ứng dụng** cho tester trước khi phát hành chính thức.

+ Firebase Test Lab: Dịch vụ kiểm thử tự động ứng dụng Android và iOS trên nhiều thiết bị thật.

Lợi ích và ứng dụng

Lợi ích:

- Phát triển nhanh chóng, không cần quản lý backend
- Hỗ trợ đa nền tảng (Android, iOS, Web)
- Cơ sở dữ liệu thời gian thực và đồng bộ nhanh

- Bảo mật và xác thực người dùng dễ dàng
- Quản lý và phân tích người dùng hiệu quả
- Gửi thông báo đẩy miễn phí
- Kiểm thử và phát hiện lỗi tự động
- Triển khai và lưu trữ ứng dụng dễ dàng

Ứng dụng:

- Ứng dụng chat và nhắn tin
- Ứng dụng thương mại điện tử (E-commerce)
- Ứng dụng đặt xe, giao hàng (Grab, Gojek, Now, Baemin)
- Ứng dụng mạng xã hội (Social Network)
- Ứng dụng giáo dục (E-learning, Quiz, Thi thử trực tuyến)
- Ứng dụng theo dõi sức khỏe và thể thao
- Ứng dụng quản lý công việc (To-do list, Calendar, Notes)

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>>