

**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	2151062863	Đào Quang Tân	63CNTT4

**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".

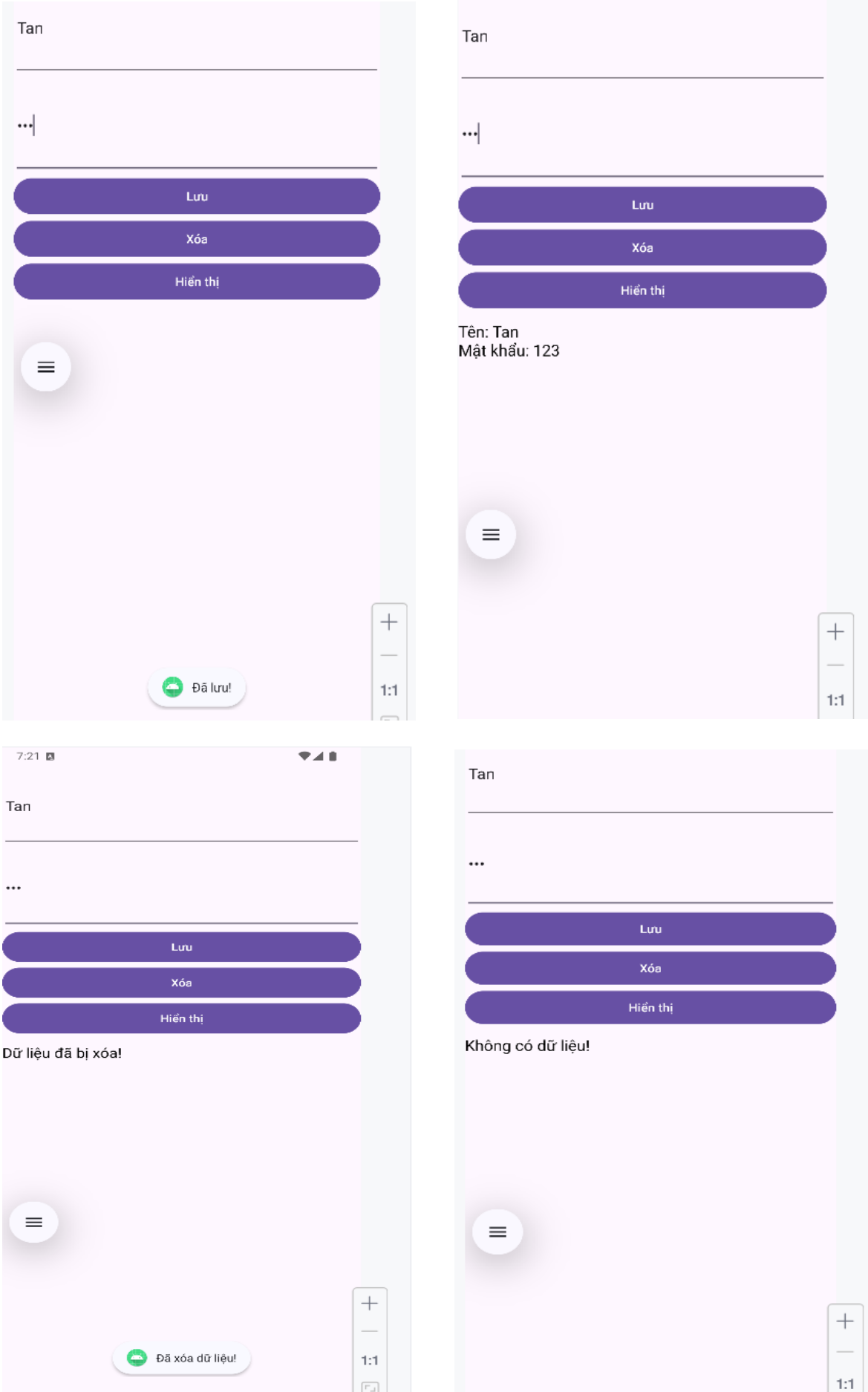
#### 2. Sử dụng Shared Preference:

- 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



```

class MainActivity : AppCompatActivity() {
    private lateinit var preferenceHelper: PreferenceHelper
    private lateinit var edtUsername: EditText
    private lateinit var edtPassword: EditText
    private lateinit var btnSave: Button
    private lateinit var btnDelete: Button
    private lateinit var btnShow: Button
    private lateinit var tvResult: TextView

    @SuppressWarnings("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }

        // Ánh xạ View từ layout
        edtUsername = findViewById(R.id.editUsername)
        edtPassword = findViewById(R.id.editPassword)
        btnSave = findViewById(R.id.btnSave)
        btnDelete = findViewById(R.id.btnDelete)
        btnShow = findViewById(R.id.btnShow)
        tvResult = findViewById(R.id.tvResult)

        // Khởi tạo PreferenceHelper

        preferenceHelper = PreferenceHelper(context: this)

        // Xử lý nút "Lưu"
        btnSave.setOnClickListener {
            val username = edtUsername.text.toString().trim()
            val password = edtPassword.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 {
                preferenceHelper.saveUser(username, password)
                Toast.makeText(context: this, text: "Đã lưu!", Toast.LENGTH_SHORT).show()
            }
        }

        // Xử lý nút "Xóa"
        btnDelete.setOnClickListener {
            preferenceHelper.deleteUser()
            tvResult.text = "Dữ liệu đã bị xóa!"
            Toast.makeText(context: this, text: "Đã xóa dữ liệu!", Toast.LENGTH_SHORT).show()
        }

        // Xử lý nút "Hiển thị"
        btnShow.setOnClickListener {
            val (username, password) = preferenceHelper.getUser()
            if (username != null && password != null) {
                tvResult.text = "Tên: $username\nMật khẩu: $password"
            } else {
                tvResult.text = "Không có dữ liệu!"
            }
        }
    }
}

```

## 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".

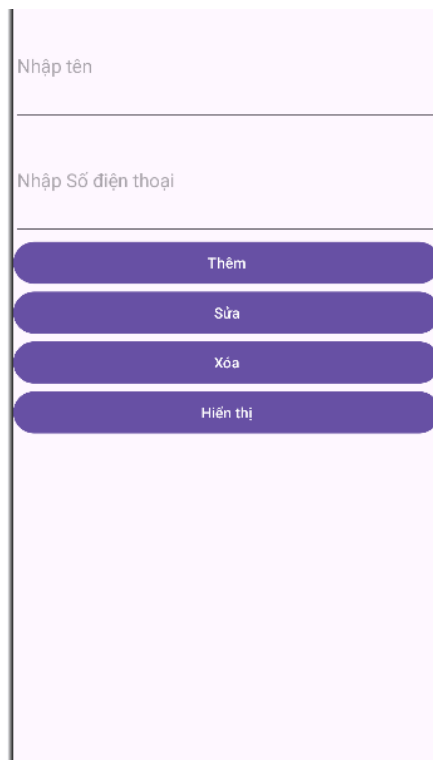
#### 2. Sử dụng SQLite:

- 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



The screenshot shows a mobile application interface with a light purple background. At the top, there is a text input field labeled "Nhập tên". Below it is another text input field labeled "Nhập Số điện thoại". Underneath these fields are four rounded rectangular buttons stacked vertically, each with a different label: "Thêm", "Sửa", "Xóa", and "Hiển thị". The bottom half of the screen is a large, empty rectangular area, likely intended for displaying the data retrieved from the SQLite database.

```

class MainActivity : AppCompatActivity() {

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

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }

        // Ánh xạ View từ layout
        edtName = findViewById(R.id.editUsername)
        edtPhone = findViewById(R.id.editPhone)
        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
        databaseHelper = DatabaseHelper(context: this)

        // Xử lý nút "Thêm"
        btnAdd.setOnClickListener {
            val name = edtName.text.toString()
            val phone = edtPhone.text.toString()
            if (name.isNotEmpty() && phone.isNotEmpty()) {
                if (databaseHelper.insertContact(name, phone)) {
                    Toast.makeText(context: this, text: "Thêm thành công!", Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(context: this, text: "Thêm thất bại!", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(context: this, text: "Vui lòng nhập đầy đủ thông tin!", Toast.LENGTH_SHORT).show()
            }
        }

        // Xử lý nút "Sửa"
        btnUpdate.setOnClickListener {
            val name = edtName.text.toString()
            val phone = edtPhone.text.toString()
            if (name.isNotEmpty() && phone.isNotEmpty()) {
                if (databaseHelper.updateContact(name, phone)) {
                    Toast.makeText(context: this, text: "Cập nhật thành công!", Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(context: this, text: "Không tìm thấy tên này!", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(context: this, text: "Vui lòng nhập đầy đủ thông tin!", Toast.LENGTH_SHORT).show()
            }
        }

        // Xử lý nút "Xóa"
        btnDelete.setOnClickListener {
            val name = edtName.text.toString()
            if (name.isNotEmpty()) {
                if (databaseHelper.deleteContact(name)) {
                    Toast.makeText(context: this, text: "Xóa thành công!", Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(context: this, text: "Xóa thất bại!", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(context: this, text: "Vui lòng nhập đầy đủ thông tin!", Toast.LENGTH_SHORT).show()
            }
        }
    }
}

```

```

// Xử lý nút "Xóa"
btnDelete.setOnClickListener {
    val name = edtName.text.toString()
    if (name.isNotEmpty()) {
        if (databaseHelper.deleteContact(name)) {
            Toast.makeText(context: this, text: "Xóa thành công!", Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(context: this, text: "Không tìm thấy tên này!", Toast.LENGTH_SHORT).show()
        }
    } else {
        Toast.makeText(context: this, text: "Vui lòng nhập tên để xóa!", Toast.LENGTH_SHORT).show()
    }
}

// Xử lý nút "Hiển thị"
btnShow.setOnClickListener {
    tvResult.text = databaseHelper.getAllContacts()
}
}

```

## BÀI TẬP 3: HỆ SINH THÁI FIREBASE

### Mục tiêu:

- Hiểu rõ về các dịch vụ chính của Firebase.
- Biết cách tích hợp Firebase vào dự án phát triển ứng dụng.

### Yêu cầu:

#### 1. Tìm hiểu các dịch vụ chính của Firebase:

- Firebase Authentication: Xác thực người dùng.
- Firebase Realtime Database và Cloud Firestore: Cơ sở dữ liệu thời gian thực và NoSQL.
- Firebase Cloud Functions: Chạy mã backend serverless.
- Firebase Cloud Messaging (FCM): Gửi thông báo đẩy.
- Firebase Storage: Lưu trữ tệp tin trên đám mây.
- Firebase Machine Learning (ML): Tích hợp trí tuệ nhân tạo vào ứng dụng.

#### 2. Viết báo cáo:

- Giới thiệu tổng quan về Firebase và lịch sử phát triển.
- Mô tả chi tiết từng dịch vụ chính của Firebase.
- Thảo luận về lợi ích và ứng dụng của Firebase trong phát triển ứng dụng.

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

### **- Tổng quan:**

Firebase là một nền tảng phát triển ứng dụng do Google cung cấp, được thiết kế để hỗ trợ 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 dựa trên đám mây, giúp đơn giản hóa các tác vụ phức tạp như lưu trữ dữ liệu, xác thực người dùng, thông báo đẩy và phân tích hiệu suất ứng dụng.

### **- Lịch sử phát triển:**

Firebase ban đầu được thành lập vào năm 2011 bởi James Tamplin và Andrew Lee dưới dạng một công ty khởi nghiệp độc lập với tên gọi "Envolv" – một dịch vụ cung cấp API trò chuyện thời gian thực. Tuy nhiên, họ nhận thấy rằng các nhà phát triển sử dụng công nghệ của mình không chỉ để truyền tin nhắn mà còn để đồng bộ hóa dữ liệu ứng dụng. Từ đó, Firebase được tái định hướng để trở thành một nền tảng cơ sở dữ liệu thời gian thực. Năm 2014, Google mua lại Firebase và tích hợp nó vào hệ sinh thái của mình. Kể từ đó, Firebase đã không ngừng mở rộng với nhiều dịch vụ mới, từ cơ sở dữ liệu đến phân tích, máy học và lưu trữ đám mây, trở thành một trong những công cụ phổ biến nhất cho phát triển ứng dụng di động và web.

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

- **Firebase Realtime Database:** Đây là cơ sở dữ liệu NoSQL dựa trên đám mây, cho phép lưu trữ và đồng bộ hóa dữ liệu giữa các thiết bị theo thời gian thực. Dữ liệu được lưu dưới dạng JSON và tự động cập nhật khi có thay đổi, rất phù hợp cho các ứng dụng cần phản hồi nhanh như trò chuyện hoặc trò chơi trực tuyến.
- **Cloud Firestore:** Một phiên bản nâng cấp của Realtime Database, Firestore là cơ sở dữ liệu NoSQL có khả năng mở rộng cao, hỗ trợ truy vấn phức tạp hơn và tích hợp tốt với các ứng dụng lớn. Nó cung cấp tính năng đồng bộ hóa offline, giúp ứng dụng hoạt động ngay cả khi không có kết nối mạng.
- **Firebase Authentication:** Dịch vụ này cung cấp giải pháp xác thực người dùng dễ dàng với nhiều phương thức như email/mật khẩu, đăng nhập ẩn danh, hoặc thông qua các nhà cung cấp bên thứ ba như Google, Facebook, Twitter. Nó giúp tăng cường bảo mật và đơn giản hóa quản lý người dùng.
- **Cloud Functions:** Cho phép nhà phát triển chạy mã phía server mà không cần quản lý máy chủ. Cloud Functions được kích hoạt bởi các sự kiện từ Firebase hoặc các yêu cầu HTTP, rất hữu ích cho tự động hóa tác vụ như gửi email hoặc xử lý dữ liệu.



- **Firebase Hosting:** Một dịch vụ lưu trữ tĩnh nhanh chóng và an toàn, hỗ trợ triển khai các ứng dụng web với chứng chỉ SSL miễn phí và tích hợp CDN toàn cầu để tăng tốc độ tải.
- **Cloud Messaging (FCM):** Dịch vụ gửi thông báo đẩy miễn phí, cho phép gửi tin nhắn đến người dùng trên các nền tảng Android, iOS và web. Nó hỗ trợ cả thông báo cá nhân hóa và thông báo hàng loạt.
- **Firebase Analytics:** Công cụ phân tích mạnh mẽ, cung cấp thông tin chi tiết về hành vi người dùng, hiệu suất ứng dụng và các sự kiện tùy chỉnh, giúp nhà phát triển tối ưu hóa trải nghiệm người dùng.
- **Firebase Machine Learning:** Cung cấp các mô hình học máy sẵn có hoặc tùy chỉnh để tích hợp vào ứng dụng, như nhận diện văn bản, phân loại hình ảnh hoặc dự đoán hành vi người dùng.
- **Firebase App Distribution:** Hỗ trợ phân phối ứng dụng beta đến người thử nghiệm một cách nhanh chóng thông qua email hoặc liên kết, giúp thu thập phản hồi trước khi phát hành chính thức.

### **3. Thảo luận về lợi ích và ứng dụng của Firebase trong phát triển ứng dụng**

#### **Lợi ích của Firebase:**

- **Tăng tốc độ phát triển:** Firebase cung cấp các công cụ sẵn có, giảm thời gian cần thiết để xây dựng các tính năng như xác thực, cơ sở dữ liệu hay thông báo đẩy từ đầu.
- **Khả năng mở rộng:** Được hỗ trợ bởi cơ sở hạ tầng của Google, Firebase tự động mở rộng theo nhu cầu ứng dụng mà không cần nhà phát triển quản lý server.
- **Tích hợp dễ dàng:** Các SDK của Firebase hỗ trợ nhiều nền tảng (Android, iOS, web), giúp việc tích hợp trở nên liền mạch.
- **Hỗ trợ offline:** Với Firestore, ứng dụng vẫn hoạt động khi không có kết nối mạng, nâng cao trải nghiệm người dùng.
- **Miễn phí cho dự án nhỏ:** Firebase có gói miễn phí hào phóng, phù hợp với các nhà phát triển độc lập hoặc khởi nghiệp.

#### **Ứng dụng của Firebase:**

- **Ứng dụng trò chuyện:** Nhờ Realtime Database và Cloud Messaging, Firebase là lựa chọn lý tưởng cho các ứng dụng như WhatsApp hoặc Slack.
- **Ứng dụng thương mại điện tử:** Firestore hỗ trợ quản lý danh mục sản phẩm, trong khi Analytics giúp theo dõi hành vi mua sắm của người dùng.

- Trò chơi trực tuyến: Đồng bộ hóa dữ liệu thời gian thực và khả năng mở rộng của Firebase rất phù hợp cho các game đa người chơi.
- Ứng dụng mạng xã hội: Authentication và Hosting hỗ trợ xây dựng các nền tảng như Instagram hoặc Twitter với chi phí thấp.
- Dự án khởi nghiệp: Các công ty khởi nghiệp tận dụng Firebase để nhanh chóng đưa sản phẩm ra thị trường mà không cần đầu tư lớn vào cơ sở hạ tầng.

### **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**

