**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG**

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

A blue and white logo

AI-generated content may be incorrect.

**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 | 2251061769 | Nguyễn Duy Hanh | 64CNTT2 |

**Hà Nội, năm 2025**

**BÀI TẬP 1: SHARED PREFERENCE**

**Mục tiêu:**

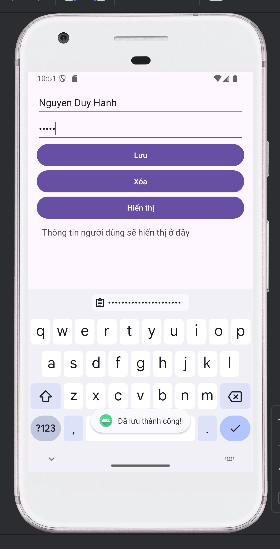
* Hiểu cách sử dụng Shared Preference để lưu trữ dữ liệu cục bộ trong ứng dụng Android.
* Thực hành lưu trữ và đọc dữ liệu từ Shared Preference.

**Yêu cầu:**

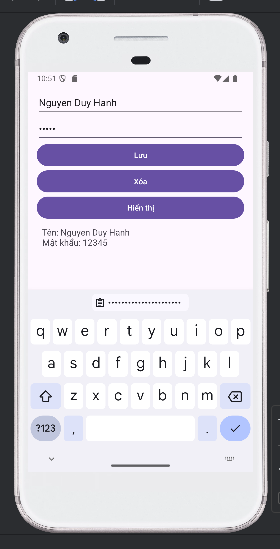
1. **Tạo ứng dụng mới:**
   * Tạo một dự án Android mới bằng Kotlin.
   * Thiết kế giao diện người dùng với hai trường nhập (EditText) cho tên người dùng và mật khẩu, và ba nút bấm: "Lưu", "Xóa", và "Hiển thị".
2. **Sử dụng Shared Preference:**
   * Tạo một lớp helper **PreferenceHelper** để quản lý 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>>

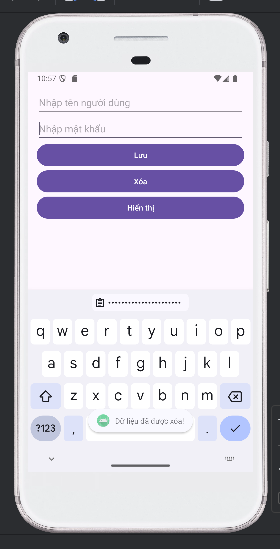
Lưu:



Hiển thi:



Xóa:



CODE:

PreferenceHelper:

package com.example.sharepreference  
  
import android.content.Context  
import android.content.SharedPreferences  
  
class PreferenceHelper(context: Context) {  
 private val sharedPreferences: SharedPreferences =  
 context.getSharedPreferences("UserPrefs", Context.*MODE\_PRIVATE*)  
  
 fun saveUserData(username: String, password: String) {  
 sharedPreferences.edit().*apply* **{**  
putString("USERNAME", username)  
 putString("PASSWORD", password)  
 apply() // Hoặc dùng commit() nếu cần lưu ngay lập tức  
 **}**  
}  
  
 fun getUserData(): Pair<String?, String?> {  
 val username = sharedPreferences.getString("USERNAME", null)  
 val password = sharedPreferences.getString("PASSWORD", null)  
 return Pair(username, password)  
 }  
  
 fun clearUserData() {  
 sharedPreferences.edit().clear().apply()  
 }  
}

MainActivity:

package com.example.sharepreference  
  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.TextView  
import android.widget.Toast  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
  
  
class MainActivity : AppCompatActivity() {  
 private lateinit var preferenceHelper: PreferenceHelper  
 private lateinit var edtUsername: EditText  
 private lateinit var edtPassword: EditText  
 private lateinit var txtResult: TextView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
 // Khởi tạo Shared Preferences Helper  
 preferenceHelper = PreferenceHelper(this)  
  
 // Ánh xạ View  
 edtUsername = findViewById(R.id.*edtUsername*)  
 edtPassword = findViewById(R.id.*edtPassword*)  
 txtResult = findViewById(R.id.*txtResult*)  
 val btnSave = findViewById<Button>(R.id.*btnSave*)  
 val btnShow = findViewById<Button>(R.id.*btnShow*)  
 val btnClear = findViewById<Button>(R.id.*btnClear*)  
  
 // Xử lý sự kiện khi nhấn "Lưu"  
 btnSave.setOnClickListener **{**  
val username = edtUsername.*text*.toString()  
 val password = edtPassword.*text*.toString()  
  
 if (username.*isNotEmpty*() && password.*isNotEmpty*()) {  
 preferenceHelper.saveUserData(username, password)  
 Toast.makeText(this, "Đã lưu thành công!", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Vui lòng nhập đầy đủ thông tin!", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
  
// Xử lý sự kiện khi nhấn "Hiển thị"  
 btnShow.setOnClickListener **{**  
val (username, password) = preferenceHelper.getUserData()  
 txtResult.*text* = if (username != null && password != null) {  
 "Tên: $username\nMật khẩu: $password"  
 } else {  
 "Không có dữ liệu được lưu!"  
 }  
 **}**  
  
// Xử lý sự kiện khi nhấn "Xóa"  
 btnClear.setOnClickListener **{**  
preferenceHelper.clearUserData()  
 txtResult.*text* = ""  
 edtUsername.*text*.clear()  
 edtPassword.*text*.clear()  
 Toast.makeText(this, "Dữ liệu đã được xóa!", Toast.*LENGTH\_SHORT*).show()  
 **}**  
}  
}

**Giao diện:**

<?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:id="@+id/main"  
 android:orientation="vertical"  
 android:layout\_marginTop="16dp"  
 android:padding="16dp">  
  
 <EditText  
 android:id="@+id/edtUsername"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Nhập tên người dùng" />  
  
 <EditText  
 android:id="@+id/edtPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Nhập mật khẩu"  
 android:inputType="textPassword" />  
  
 <Button  
 android:id="@+id/btnSave"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Lưu" />  
  
 <Button  
 android:id="@+id/btnClear"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Xóa" />  
  
 <Button  
 android:id="@+id/btnShow"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Hiển thị" />  
  
 <TextView  
 android:id="@+id/txtResult"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Thông tin người dùng sẽ hiển thị ở đây"  
 android:textSize="16sp"  
 android:padding="10dp" />  
</LinearLayout>

**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.

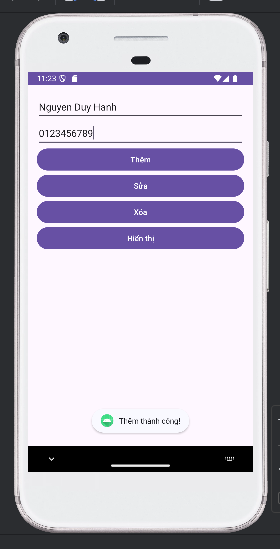
**Yêu cầu:**

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>>

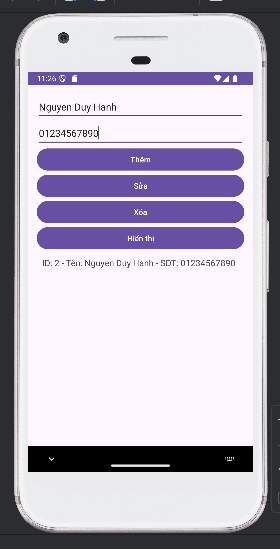
Thêm:



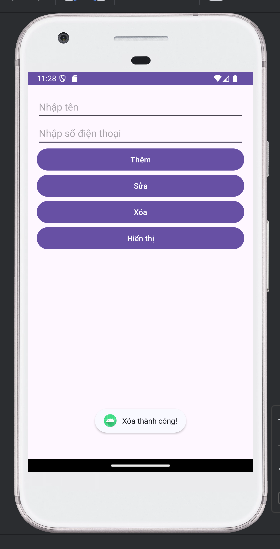
Sửa:



**Hiển thị:**



Xóa:



CODE:

Helper:

package com.example.usesqlite  
  
import android.content.ContentValues  
import android.content.Context  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
  
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, "ContactDB", null, 1) {  
  
 override fun onCreate(db: SQLiteDatabase) {  
 val createTable = "CREATE TABLE Contacts (ID INTEGER PRIMARY KEY AUTOINCREMENT, Name TEXT, Phone TEXT)"  
 db.execSQL(createTable)  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
 db.execSQL("DROP TABLE IF EXISTS Contacts")  
 onCreate(db)  
 }  
  
 // Thêm dữ liệu  
 fun insertData(name: String, phone: String): Boolean {  
 val db = this.*writableDatabase*  
val values = ContentValues()  
 values.put("Name", name)  
 values.put("Phone", phone)  
 val result = db.insert("Contacts", null, values)  
 return result != -1L  
 }  
  
 // Cập nhật dữ liệu  
 fun updateData(name: String, phone: String): Boolean {  
 val db = this.*writableDatabase*  
val values = ContentValues()  
 values.put("Phone", phone)  
 val result = db.update("Contacts", values, "Name=?", *arrayOf*(name))  
 return result > 0  
 }  
  
 // Xóa dữ liệu  
 fun deleteData(name: String): Boolean {  
 val db = this.*writableDatabase*  
val result = db.delete("Contacts", "Name=?", *arrayOf*(name))  
 return result > 0  
 }  
  
 // Lấy dữ liệu  
 fun getAllData(): String {  
 val db = this.*readableDatabase*  
val cursor = db.rawQuery("SELECT \* FROM Contacts", null)  
 val result = StringBuilder()  
  
 if (cursor.moveToFirst()) {  
 do {  
 val id = cursor.getInt(0)  
 val name = cursor.getString(1)  
 val phone = cursor.getString(2)  
 result.append("ID: $id - Tên: $name - SĐT: $phone\n")  
 } while (cursor.moveToNext())  
 }  
 cursor.close()  
 return result.toString()  
 }  
}

MainActivity:

package com.example.usesqlite  
  
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 helper: DatabaseHelper  
 private lateinit var edtName: EditText  
 private lateinit var edtPhone: EditText  
 private lateinit var txtResult: TextView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 // Khởi tạo database helper  
 helper = DatabaseHelper(this)  
  
 // Ánh xạ view  
 edtName = findViewById(R.id.*edtName*)  
 edtPhone = findViewById(R.id.*edtPhone*)  
 txtResult = findViewById(R.id.*txtResult*)  
 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 thêm dữ liệu  
 btnAdd.setOnClickListener **{**  
val name = edtName.*text*.toString()  
 val phone = edtPhone.*text*.toString()  
 if (name.*isNotEmpty*() && phone.*isNotEmpty*()) {  
 val isInserted = helper.insertData(name, phone)  
 showToast(if (isInserted) "Thêm thành công!" else "Lỗi khi thêm!")  
 } else {  
 showToast("Vui lòng nhập đủ thông tin!")  
 }  
 **}**  
  
// Xử lý sửa dữ liệu  
 btnUpdate.setOnClickListener **{**  
val name = edtName.*text*.toString()  
 val phone = edtPhone.*text*.toString()  
 val isUpdated = helper.updateData(name, phone)  
 showToast(if (isUpdated) "Cập nhật thành công!" else "Không tìm thấy tên!")  
 **}**  
  
// Xử lý xóa dữ liệu  
 btnDelete.setOnClickListener **{**  
val name = edtName.*text*.toString()  
 val isDeleted = helper.deleteData(name)  
 txtResult.*text* = ""  
 edtName.*text*.clear()  
 edtPhone.*text*.clear()  
 showToast(if (isDeleted) "Xóa thành công!" else "Không tìm thấy tên!")  
 **}**  
  
// Xử lý hiển thị dữ liệu  
 btnShow.setOnClickListener **{**  
txtResult.*text* = helper.getAllData()  
 **}**  
}  
  
 private fun showToast(message: String) {  
 Toast.makeText(this, message, Toast.*LENGTH\_SHORT*).show()  
 }  
}

Giao diện:

<?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/edtName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Nhập tên" />  
  
 <EditText  
 android:id="@+id/edtPhone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Nhập số điện thoại"  
 android:inputType="phone" />  
  
 <Button  
 android:id="@+id/btnAdd"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Thêm" />  
  
 <Button  
 android:id="@+id/btnUpdate"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Sửa" />  
  
 <Button  
 android:id="@+id/btnDelete"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Xóa" />  
  
 <Button  
 android:id="@+id/btnShow"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Hiển thị" />  
  
 <TextView  
 android:id="@+id/txtResult"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp"  
 android:padding="10dp" />  
</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 và lịch sử phát triển.   - Firebase là nền tảng phát triển ứng dụng đa năng của di động và website.  - Tiền thân của nền tảng Firebase chính là Envolve. Envolve đến với người dùng dưới mô hình startup, họ cung cấp cho người dùng những API để dễ dàng tích hợp tính năng chat vào trang web. Thế nhưng, người dùng lại sử dụng Envolve để truyền dữ liệu ứng dụng chứ không đơn thuần là nhắn tin trò chuyện. Chính điều này đã khiến các nhà phát triển Envolve quyết định tách riêng chat system và the real-time architecture.  - Và năm 2012, Firebase ra đời dưới dạng Backend-as-a-Service với chức năng thời gian thực. Sau khi được Google mua lại vào năm 2014 thì Firebase nhanh chóng phát triển thành nền tảng phát triển ứng dụng đa năng của di động và website như ngày nay. Nền tảng này là sự kết hợp giữa cloud với hệ thống máy chủ của Google để tập trung chính cho 2 đối tượng là:   * **Develop & test your app**: phát triển và thử nghiệm các ứng dụng được thiết kế. * **Grow & engage your audience**: phân tích dữ liệu và tối ưu hóa trải nghiệm với người dùng.   - Firebase cung cấp cho chúng ta những API đơn giản, mạnh mẽ và đa nền tảng trong việc quản lý, sử dụng database, bởi vậy giờ đây chúng ta chỉ cần gọi API và phần server đã có Firebase lo!  **2. Những Service nổi bật**  **2.1 Real-time Database** Dịch vụ Realtime database cho phép người dùng lưu trữ và đồng bộ dữ liệu theo thời gian thực. Dịch vụ này được lưu trữ trực tiếp trên iCloud. Trong trường hợp thiết bị của bạn ngoại tuyến thì chúng sẽ sử dụng tới bộ nhớ của thiết bị và tự động đồng bộ lên server khi thiết bị online. Do đó bạn hoàn toàn có thể yên tâm về độ tương tác.  2.2 Authentication  Dịch vụ Authentication cung cấp cho ứng dụng của bạn một số phương pháp xác thực thông qua email, mật khẩu, số điện thoải, tài khoản Google, tài khoản Facebook… Với tính năng này, người dùng sẽ dễ dàng xây dựng login mà không cần sử dụng dữ liệu đăng ký riêng.  2.3 Firebase cloud messaging  Firebase được xây dựng với tính năng nguyên thủy là trò chuyện. Bởi thế Firebase cloud messaging (FCM) là tính năng cơ bản nhất của nền tảng này, nó cho phép người dùng xây dựng ứng dụng chat và đẩy thông báo tới nhiều thiết bị khác nhau như web, Android, iOS… Điểm nổi bật của dịch vụ này là hầu như không có bất kỳ mã hóa nào liên quan! FCM được tích hợp hoàn toàn với Firebase Analytics, mang đến cho bạn sự tương tác chi tiết (detailed engagement) và theo dõi chuyển đổi (conversion tracking) trong quá trình sử dụng.  Hơn thế, dịch vụ FCM còn giúp bạn tùy chọn thời điểm gửi tin là ngay lập tức hoặc vào thời điểm tương lai theo múi giờ địa phương của người dùng.  2.3 Firebase database query  Một trong những dịch vụ nổi bật của Firebase chính là nó có thể giúp bạn đơn giản hóa quá trình lấy dữ liệu thay vì phải thông qua các câu lệnh SQL phức tạp. Tuy nhiên việc thiết lập ngôn ngữ truy vấn tại Firebase khá khó khăn, để làm được nó thì bạn cần xây dựng một database thật chính xác nhé.  2.4 Remote config  Remote Config giúp bạn làm 2 nhiệm vụ cơ bản là:   * Cập nhật các ứng dụng một cách nhanh chóng và dễ dàng mà không cần xuất bản bản dựng mới lên app/play store. * Bạn có thể dễ dàng cài đặt phân đoạn hoạt động trong ứng dụng dựa trên yêu cầu thiết bị hoặc người dùng đang sử dụng nó.   Để làm được điều đó, Firebase sẽ cài đặt các thông số bên máy chủ giúp người dùng cập nhật các dữ liệu ngay lập tức dù là thay đổi bố cục, bảng màu hay một phần cụ thể nào đó trong ứng dụng. Điều này hết sức hữu dụng bởi các ứng dụng thông thường sẽ bị mất hoàn toàn cài đặt nếu người dùng gỡ ứng dụng đi. Nếu muốn cài đặt trên nhiều ứng dụng khác nhau thì họ sẽ phải tự cấu hình thiết lập bằng tay trên từng thiết bị sao cho giống nhau. Quá trình này khá phức tạp và mất nhiều thời gian. **1. Lợi ích của Firebase****a) Dễ dàng tích hợp và sử dụng**  * Firebase cung cấp SDK cho nhiều nền tảng (**Android, iOS, Web**). * Hỗ trợ **Realtime Database** và **Firestore** giúp lưu trữ và đồng bộ dữ liệu theo thời gian thực. * Không cần thiết lập máy chủ riêng, giúp lập trình viên tập trung vào phát triển ứng dụng.   **b) Tiết kiệm thời gian và chi phí**  * Firebase cung cấp **các dịch vụ backend sẵn có**, giảm thiểu việc xây dựng hệ thống backend từ đầu. * **Miễn phí** với giới hạn nhất định và có thể nâng cấp lên các gói trả phí khi ứng dụng phát triển.   **c) Bảo mật tốt**  * Firebase **Authentication** hỗ trợ xác thực an toàn qua **Email/Password, Google, Facebook, Số điện thoại, v.v.**. * **Firebase Security Rules** giúp bảo vệ dữ liệu khỏi truy cập trái phép.   **d) Phân tích dữ liệu & tối ưu hiệu suất**  * Firebase **Analytics** giúp theo dõi hành vi người dùng, đo lường hiệu suất ứng dụng. * **Firebase Performance Monitoring** giúp tối ưu tốc độ load, giảm lag, crash.  **e) Hỗ trợ mở rộng và quy mô lớn**  * Firebase dễ dàng mở rộng khi ứng dụng phát triển, không cần quan tâm đến hạ tầng máy chủ. * Hỗ trợ **Cloud Functions**, giúp xử lý logic backend mà không cần server.   **2. Ứng dụng của Firebase trong phát triển ứng dụng****a) Xác thực và quản lý người dùng (Firebase Authentication)** - Firebase giúp xác thực người dùng dễ dàng qua:   * Email/Password * Google, Facebook, Apple, Twitter * Số điện thoại (OTP) * Anonymous login (đăng nhập ẩn danh)   **Ứng dụng:** Ứng dụng đăng nhập, mạng xã hội, thương mại điện tử. **b) Lưu trữ dữ liệu theo thời gian thực (Realtime Database & Firestore)** **- Firebase Realtime Database**:   * Dữ liệu được cập nhật **ngay lập tức** trên tất cả thiết bị. * Phù hợp cho ứng dụng **chat, theo dõi đơn hàng, hiển thị dữ liệu động**.   **- Cloud Firestore**:   * Lưu trữ dữ liệu có cấu trúc linh hoạt, hỗ trợ **truy vấn mạnh mẽ hơn**. * Phù hợp cho ứng dụng **quy mô lớn, nhiều dữ liệu**.   **Ứng dụng:** Ứng dụng chat, hệ thống quản lý đơn hàng, mạng xã hội. **c) Gửi thông báo đẩy (Firebase Cloud Messaging - FCM)**  * Firebase giúp gửi thông báo **miễn phí**, hỗ trợ **cả Android và iOS**. * Hỗ trợ thông báo nền (background) và foreground. * Có thể gửi **thông báo cá nhân hóa** dựa trên hành vi người dùng.   **Ứng dụng:** Ứng dụng tin tức, thương mại điện tử (thông báo đơn hàng, khuyến mãi). **d) Phân tích hành vi người dùng (Firebase Analytics)**  * Thu thập dữ liệu về **tương tác, hành vi người dùng, tỷ lệ giữ chân (retention rate)**. * Kết hợp với Google Ads để **tối ưu hóa chiến dịch quảng cáo**.   **Ứng dụng:** Ứng dụng di động cần theo dõi hành vi người dùng, tối ưu hóa trải nghiệm. **☁️e) Xử lý logic backend không cần server (Cloud Functions)**  * Xử lý các sự kiện như **gửi email xác nhận, cập nhật dữ liệu tự động**. * Giảm tải công việc cho ứng dụng, giúp app chạy nhanh hơn.   **📌 Ứng dụng:** Ứng dụng thương mại điện tử, hệ thống quản lý đơn hàng. **📦 f) Lưu trữ và chia sẻ file (Firebase Storage)**  * Lưu trữ **hình ảnh, video, tài liệu** với khả năng bảo mật cao. * Hỗ trợ **tối ưu hình ảnh, video**, giúp tải nhanh hơn.   **📌 Ứng dụng:** Ứng dụng mạng xã hội, lưu trữ đám mây. |

1. **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**

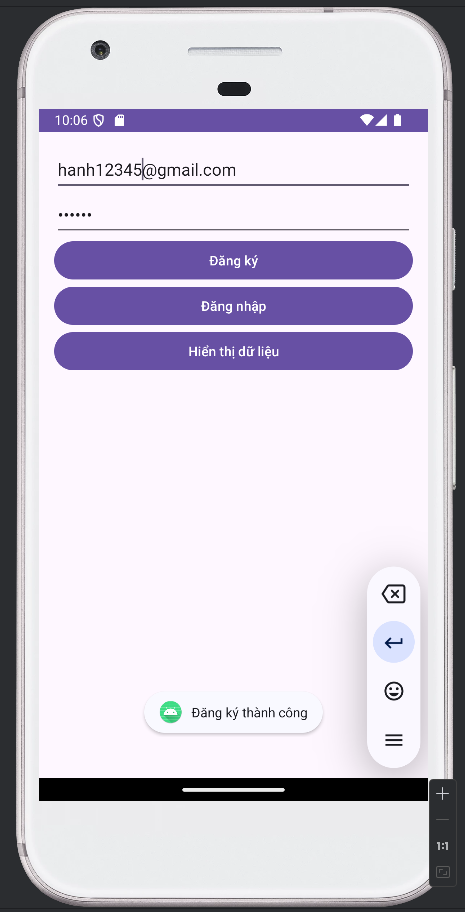
**Yêu cầu:**

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**

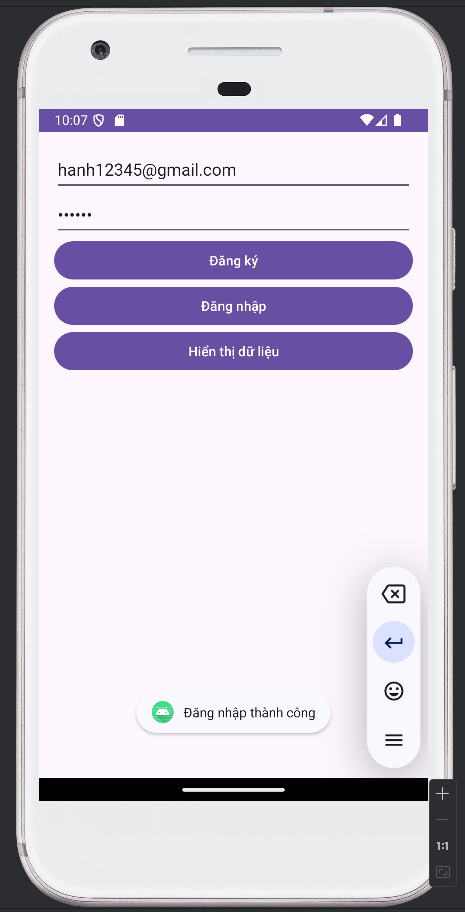
<<Sinh viên chụp Ảnh màn hình kết quả và mã nguồn chính tại đây>>

**Giao diện:**

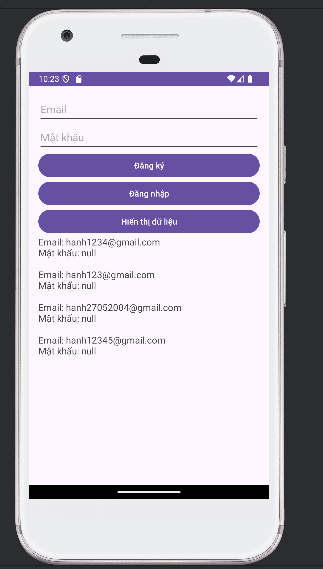
**Đăng Kí:**



Đăng Nhập:



Hiển thị dữ liệu:



CODE:

Giao diện:

<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/main"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <EditText  
 android:id="@+id/etEmail"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Email" />  
  
 <EditText  
 android:id="@+id/etPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:hint="Mật khẩu"  
 android:inputType="textPassword" />  
  
 <Button  
 android:id="@+id/btnRegister"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Đăng ký" />  
  
 <Button  
 android:id="@+id/btnLogin"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Đăng nhập" />  
  
 <Button  
 android:id="@+id/btnShowData"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Hiển thị dữ liệu" />  
  
 <TextView  
 android:id="@+id/tvUserData"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textSize="16sp" />  
</LinearLayout>

MainActivity:

package com.example.practiceusefirebase  
  
  
import android.os.Bundle  
import android.widget.\*  
import androidx.appcompat.app.AppCompatActivity  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.database.\*  
import com.google.firebase.database.DatabaseReference  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var auth: FirebaseAuth  
 private lateinit var database: DatabaseReference  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 auth = FirebaseAuth.getInstance()  
 database = FirebaseDatabase.getInstance().*reference*.child("users")  
  
 val etEmail = findViewById<EditText>(R.id.*etEmail*)  
 val etPassword = findViewById<EditText>(R.id.*etPassword*)  
 val btnRegister = findViewById<Button>(R.id.*btnRegister*)  
 val btnLogin = findViewById<Button>(R.id.*btnLogin*)  
 val btnShowData = findViewById<Button>(R.id.*btnShowData*)  
 val tvUserData = findViewById<TextView>(R.id.*tvUserData*)  
  
 // Xử lý đăng ký tài khoản  
 btnRegister.setOnClickListener **{**  
val email = etEmail.*text*.toString()  
 val password = etPassword.*text*.toString()  
  
 if (email.*isNotEmpty*() && password.*isNotEmpty*()) {  
 auth.createUserWithEmailAndPassword(email, password)  
 .addOnCompleteListener **{** task **->**  
if (task.*isSuccessful*) {  
 val userId = auth.*currentUser*?.*uid*  
val userData = *mapOf*("email" *to* email)  
 userId?.*let* **{** database.child(**it**).setValue(userData) **}**  
Toast.makeText(this, "Đăng ký thành công", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Đăng ký thất bại", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
} else {  
 Toast.makeText(this, "Vui lòng nhập email và mật khẩu", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
  
// Xử lý đăng nhập  
 btnLogin.setOnClickListener **{**  
val email = etEmail.*text*.toString()  
 val password = etPassword.*text*.toString()  
  
 if (email.*isNotEmpty*() && password.*isNotEmpty*()) {  
 auth.signInWithEmailAndPassword(email, password)  
 .addOnCompleteListener **{** task **->**  
if (task.*isSuccessful*) {  
 Toast.makeText(this, "Đăng nhập thành công", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Đăng nhập thất bại", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
} else {  
 Toast.makeText(this, "Vui lòng nhập email và mật khẩu", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**  
  
btnShowData.setOnClickListener **{**  
hienThiDuLieu(tvUserData)  
 **}**  
  
}  
  
 // Hàm đọc toàn bộ dữ liệu từ Firebase Realtime Database  
 private fun hienThiDuLieu(tvUserData: TextView) {  
 database.addListenerForSingleValueEvent(object : ValueEventListener {  
 override fun onDataChange(snapshot: DataSnapshot) {  
 val stringBuilder = StringBuilder()  
 for (userSnapshot in snapshot.*children*) {  
 val email = userSnapshot.child("email").getValue(String::class.*java*)  
 val password = userSnapshot.child("password").getValue(String::class.*java*)  
 stringBuilder.append("Email: $email\nMật khẩu: $password\n\n")  
 }  
 tvUserData.*text* = stringBuilder.toString() // Hiển thị dữ liệu lên TextView  
 }  
  
 override fun onCancelled(error: DatabaseError) {  
 Toast.makeText(this@MainActivity, "Lỗi đọc dữ liệu: ${error.*message*}", Toast.*LENGTH\_SHORT*).show()  
 }  
 })  
 }  
}