**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 | 2251061771 | Bùi Thị Hảo | 64CNTT1 |

**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 file activity\_main:** dưới đây là mã nguồn dùng để tạo các editText (tên người dùng , mật khẩu) và các button (xóa, lưu, hiển thị)

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <EditText  android:id="@+id/edtName"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="text"  android:text="Tên người dùng " />   <EditText  android:id="@+id/edtPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="text"  android:minHeight="48dp"  android:text="Mật khẩu " />   <Button  android:id="@+id/btnHienThi"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Hiển thị " />   <Button  android:id="@+id/btnLuu"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Lưu" />   <Button  android:id="@+id/btnXoa"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Xóa" />  </LinearLayout> |

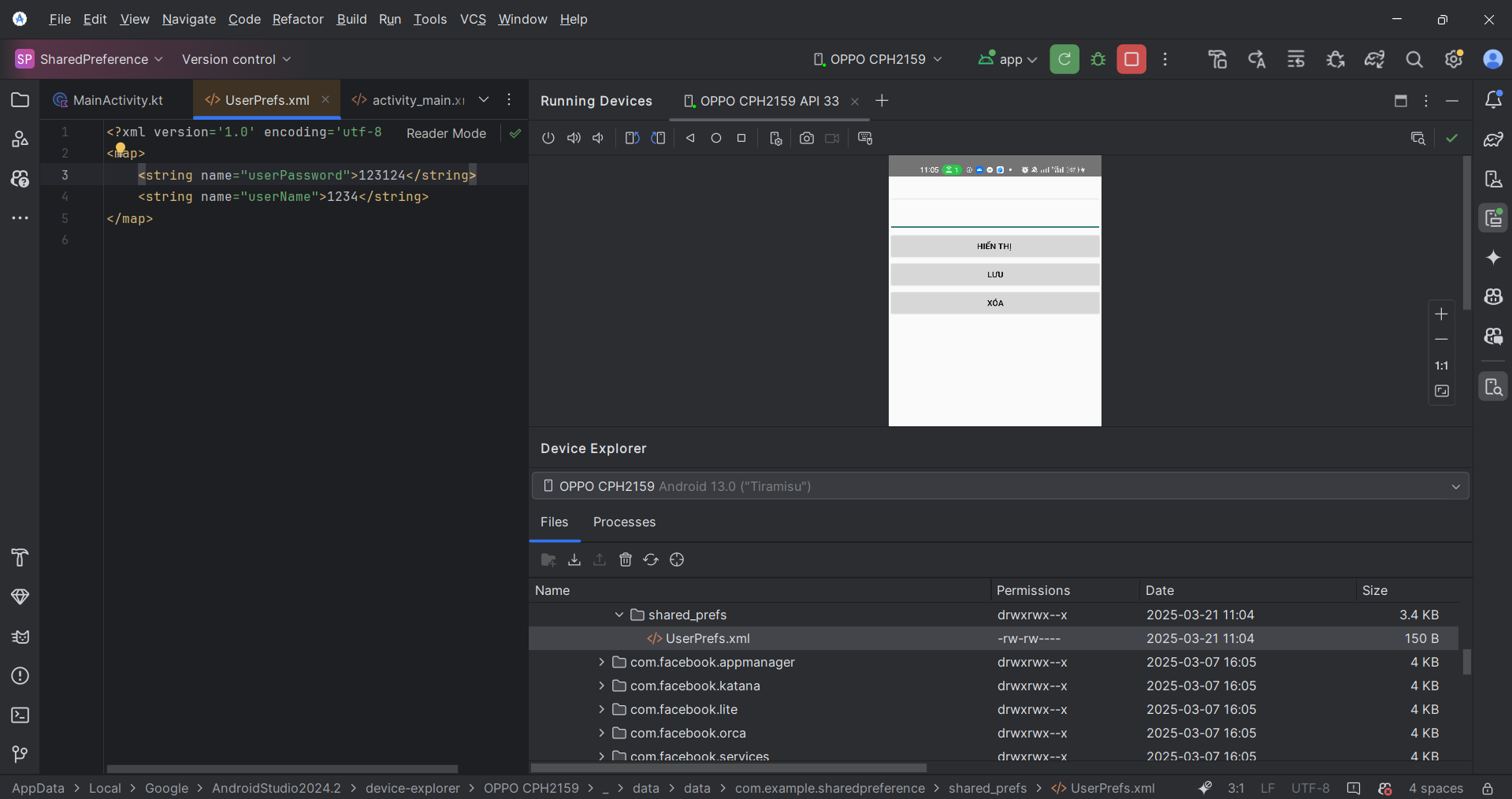


**Mã nguồn của file PreferenceHelper:** Tạo một lớp helper **PreferenceHelper** để quản lý Shared Preference

|  |
| --- |
| package com.example.sharedpreference  import android.content.Context import android.content.SharedPreferences  class PreferenceHelper(context:Context) {  private val sharedPreferences:SharedPreferences = context.getSharedPreferences("UserPrefs",Context.*MODE\_PRIVATE*)  fun saveUser(name:String,password:String){  sharedPreferences.edit().*apply* **{** putString("userName", name)  putString("userPassword", password)  apply()  **}** }  fun getUserName():String{  return sharedPreferences.getString("userName","")?:""  }  fun getPassword():String{  return sharedPreferences.getString("userPassword","")?:""   }  fun clearData(){  sharedPreferences.edit().clear().apply()  } } |

**Mã nguồn của file MainActivity:** để xử lý sự kiện “Lưu” “Hiển thị” “Xóa” vào SharePreferences

|  |
| --- |
| package com.example.sharedpreference  import android.os.Bundle import android.util.Log import android.widget.Button import android.widget.EditText import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.activity.enableEdgeToEdge import androidx.compose.foundation.layout.fillMaxSize import androidx.compose.foundation.layout.padding import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.runtime.Composable import androidx.compose.ui.Modifier import androidx.compose.ui.tooling.preview.Preview import com.example.sharedpreference.ui.theme.SharedPreferenceTheme  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_main*)   //anh xa cac doi tuong trong layout  val edtName = findViewById<EditText>(R.id.*edtName*)  val edtPassword = findViewById<EditText>(R.id.*edtPassword*)  val btnHienThi = findViewById<Button>(R.id.*btnHienThi*)  val btnLuu = findViewById<Button>(R.id.*btnLuu*)  val btnXoa = findViewById<Button>(R.id.*btnXoa*)   val preferenceHelper = PreferenceHelper(this)   //Su kien Hien thi khi nhan nut Hien thi  btnHienThi.setOnClickListener **{** val name= preferenceHelper.getUserName()  val password = preferenceHelper.getPassword()  // In ra logcat để kiểm tra  Log.d("SharedPreferences", "Name: $name, Password: $password")  edtName.setText(name)  edtPassword.setText(password)  **}** //Su kien Luu khi nhan nut Luu  btnLuu.setOnClickListener **{** val name = edtName.*text*.toString()  val password = edtPassword.*text*.toString()  Log.d("SharedPreferences", "Lưu: Name = $name, Password = $password")  preferenceHelper.saveUser(name, password)  edtName.setText("")  edtPassword.setText("")  **}** //Su kien Xoa khi nhan nut Xoa  btnXoa.setOnClickListener**{** Log.d("SharedPreferences", "Xóa dữ liệu")  preferenceHelper.clearData()  edtName.setText("")  edtPassword.setText("")  **}** }  } |

Khi người dùng nhấn nút "Lưu", lưu tên người dùng và mật khẩu vào Shared Preference. 

Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ Shared Preference và hiển thị lên màn hình.

**A screenshot of a computer

AI-generated content may be incorrect.**

Khi người dùng nhấn nút "Xóa", xóa dữ liệu đã lưu trong Shared Preference.

A screenshot of a computer

AI-generated content may be incorrect.

**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**

**Mã nguồn của file activity\_main:** nhằm thiết kế giao diện người dùng với 2 trường nhập (EditText) cho tên và số điện thoại, và bốn nút bấm: “Thêm”, “Sửa”, “Xóa”, và “Hiện thị”.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:id="@+id/LinearLayout"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <EditText  android:id="@+id/edtName"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="text"  android:text="Tên" />   <EditText  android:id="@+id/edtPhone"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="text"  android:minHeight="48dp"  android:text="SDT" />   <Button  android:id="@+id/btnHienthi"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Hiển thị " />   <Button  android:id="@+id/btnThem"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Thêm " />   <Button  android:id="@+id/btnSua"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Sửa" />   <Button  android:id="@+id/BtnXoa"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Xóa " />   <ListView  android:id="@+id/lstView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" /> </LinearLayout> |

A screenshot of a computer

AI-generated content may be incorrect.

**Mã nguồn của file MainActivity:** nhằm để thực hiện các sự kiện **thêm, sửa, xóa, và hiển thị** CSDL từ bảng ra ListView trên màn hình giao diện.

|  |
| --- |
| package com.example.reviewsqlites  import android.content.ContentValues import android.database.Cursor import android.database.sqlite.SQLiteDatabase import android.os.Bundle import android.util.Log import android.view.View import android.widget.ArrayAdapter import android.widget.Button import android.widget.EditText import android.widget.ListView import android.widget.Toast import androidx.activity.ComponentActivity  class MainActivity : ComponentActivity() {  private lateinit var edtname: EditText  private lateinit var edtphone: EditText  private lateinit var btnThem: Button  private lateinit var btnSua: Button  private lateinit var btnXoa: Button  private lateinit var btnHienthi: Button  private lateinit var lstView: ListView  private lateinit var databaseHelper: DatabaseHelper  private lateinit var myadapter: ArrayAdapter<String>   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*main\_activity*)   //mylist = ArrayList()  edtname = findViewById(R.id.*edtName*)  edtphone = findViewById(R.id.*edtPhone*)  btnHienthi = findViewById(R.id.*btnHienthi*)  btnThem = findViewById(R.id.*btnThem*)  btnSua = findViewById(R.id.*btnSua*)  btnXoa = findViewById(R.id.*BtnXoa*)  lstView = findViewById(R.id.*lstView*)   //khoi tao database helper  databaseHelper = DatabaseHelper(this)  //gan adapter cho listview  myadapter = ArrayAdapter<String>(this, android.R.layout.*simple\_list\_item\_1*, databaseHelper.getAllUsers())  lstView.*adapter* = myadapter    myadapter = ArrayAdapter<String>(this, android.R.layout.*simple\_list\_item\_1*, ArrayList())  lstView.setAdapter(myadapter)   //xu ly su kien them  btnThem.setOnClickListener(View.OnClickListener() **{** val name = edtname.getText().toString();  val phone = edtphone.getText().toString();  if (name.*isNotEmpty*() && phone.*isNotEmpty*()){  if (databaseHelper.insertUser(name, phone)) {  Toast.makeText(this@MainActivity, "Thêm thành công", Toast.*LENGTH\_SHORT*).show()  loaddata()  } else {  Toast.makeText(this@MainActivity, "Thêm thất bại", Toast.*LENGTH\_SHORT*).show()  }  }else{  Toast.makeText(this@MainActivity, "Vui lòng nhập đủ thông tin", Toast.*LENGTH\_SHORT*).show()  }   **}**)  //xu ly su kien xoa  btnXoa.setOnClickListener(View.OnClickListener() **{** val name = edtname.getText().toString();  if (name.*isNotEmpty*()){  if (databaseHelper.deleteUser(name)) {  Toast.makeText(this@MainActivity, "Xóa thành công", Toast.*LENGTH\_SHORT*).show()  loaddata()  } else {  Toast.makeText(this@MainActivity, "Xóa thất bại", Toast.*LENGTH\_SHORT*).show()  }  }else{  Toast.makeText(this@MainActivity, "Vui lòng nhập đủ thông tin", Toast.*LENGTH\_SHORT*).show()  }  **}**)  //xu ly su kien sua  btnSua.setOnClickListener(View.OnClickListener() **{** val name = edtname.getText().toString();  val phone = edtphone.getText().toString();  if (name.*isNotEmpty*() && phone.*isNotEmpty*()) {  if (databaseHelper.updateUser(name, phone)) {  Toast.makeText(this@MainActivity, "Sửa thành công", Toast.*LENGTH\_SHORT*).show()  loaddata()  } else {  Toast.makeText(this@MainActivity, "Sửa thất bại", Toast.*LENGTH\_SHORT*).show()  }  } else{  Toast.makeText(this@MainActivity, "Vui lòng nhập đủ thông tin", Toast.*LENGTH\_SHORT*).show()  }   **}**)  //xu ly su kien hien thi  btnHienthi.setOnClickListener(View.OnClickListener()**{** loaddata()  **}**)  }  private fun loaddata(){  val datalist = databaseHelper.getAllUsers()  myadapter.clear()  myadapter.addAll(datalist)  myadapter.notifyDataSetChanged()  } } |

**Mã nguồn của file DatabaseHelper:** dùng để tạo cơ sở dữ liệu, tạo bảng, các hàm thêm xóa, sửa và hiển thị lên màn hình

|  |
| --- |
| package com.example.reviewsqlites  import android.content.ContentValues import android.content.Context import android.database.Cursor import android.database.sqlite.SQLiteDatabase import android.database.sqlite.SQLiteOpenHelper  class DatabaseHelper(context: Context) : SQLiteOpenHelper(context, "qlnguoidung", null, 2) {   override fun onCreate(db: SQLiteDatabase) {  // Tạo bảng "tbnguoidung " với hai cột: name (tên) và phone (số điện thoại)  val createTable = """  CREATE TABLE tbnguoidung (  name TEXT NOT NULL PRIMARY KEY,  phone TEXT NOT NULL  )  """.*trimIndent*()  db.execSQL(createTable) //thuc thi cau lenh sql  }   override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  db.execSQL("DROP TABLE IF EXISTS tbnguoidung")  onCreate(db)  }   // Thêm dữ liệu vào bảng tbnguoidung  fun insertUser(name: String, phone: String): Boolean {  val db = this.*writableDatabase* //writableDatabase: cho phép thao tác đọc và ghi với dữ liệu: insert, update, delete  val values = ContentValues().*apply* **{** put("name", name)  put("phone", phone)  **}** val result = db.insert("tbnguoidung", null, values)  db.close()  return result != -1L // Trả về true nếu thêm thành công  }   // Cập nhật số điện thoại dựa trên tên  fun updateUser(name: String, newPhone: String): Boolean {  val db = this.*writableDatabase* val values = ContentValues().*apply* **{** put("phone", newPhone)  **}** val result = db.update("tbnguoidung", values, "name=?", *arrayOf*(name))  db.close()  return result > 0 // Trả về true nếu cập nhật thành công  }   // Xóa liên hệ dựa trên tên  fun deleteUser(name: String): Boolean {  val db = this.*writableDatabase* val result = db.delete("tbnguoidung", "name=?", *arrayOf*(name))  db.close()  return result > 0 // Trả về true nếu xóa thành công  }   // Lấy tất cả dữ liệu từ bảng contacts  fun getAllUsers(): List<String> {  val db = *readableDatabase* val list = *mutableListOf*<String>()  val cursor: Cursor = db.query("tbnguoidung", null, null, null, null, null, null)   while (cursor.moveToNext()) {  val name = cursor.getString(0)  val phone = cursor.getString(1)  list.add("$name - $phone")  }  cursor.close()  return list  } } |

|  |  |
| --- | --- |
|  |  |
|  |  |

**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. Tổng quan về Firebase và lịch sử phát triển**  Firebase là một nền tảng phát triển ứng dụng di động và web được Google cung cấp, giúp lập trình viên xây dựng, quản lý và mở rộng ứng dụng một cách nhanh chóng mà không cần lo lắng về cơ sở hạ tầng backend.  Firebase ban đầu được thành lập vào năm 2011 bởi công ty Firebase, Inc. với mục tiêu cung cấp dịch vụ đồng bộ hóa dữ liệu thời gian thực cho các ứng dụng. Năm 2014, Google mua lại Firebase và từ đó mở rộng thành một bộ công cụ toàn diện cho phát triển ứng dụng, tích hợp chặt chẽ với Google Cloud Platform.  **2. Các dịch vụ chính của Firebase**  a. Firebase Authentication  Firebase Authentication cung cấp các phương thức đăng nhập an toàn và dễ dàng như:   * Đăng nhập bằng email/mật khẩu * Đăng nhập bằng Google, Facebook, Twitter, GitHub * Đăng nhập bằng số điện thoại (OTP) * Xác thực bằng Anonymous Login (đăng nhập ẩn danh)   b. Firebase Realtime Database & Firestore   * Firebase Realtime Database: Cơ sở dữ liệu NoSQL lưu trữ và đồng bộ hóa dữ liệu theo thời gian thực. * Cloud Firestore: Một phiên bản nâng cấp của Realtime Database với nhiều tính năng mạnh mẽ hơn như truy vấn linh hoạt, mở rộng quy mô tốt hơn.   c. Firebase Cloud Messaging (FCM)  FCM giúp gửi thông báo đẩy (push notifications) miễn phí trên các nền tảng Android, iOS và Web.  d. Firebase Hosting  Dịch vụ hosting mạnh mẽ cho các ứng dụng web tĩnh với khả năng tích hợp dễ dàng với các công nghệ như React, Angular và Vue.  e. Firebase Storage  Lưu trữ và quản lý các tập tin (hình ảnh, video, tài liệu,...) với độ bảo mật cao.  f. Firebase Analytics  Cung cấp báo cáo chi tiết về hành vi người dùng giúp lập trình viên tối ưu trải nghiệm ứng dụng.  g. Firebase Remote Config  Cho phép thay đổi cấu hình ứng dụng trong thời gian thực mà không cần phát hành bản cập nhật.  h. Firebase Performance Monitoring  Theo dõi hiệu suất ứng dụng, đo lường thời gian tải, truy vấn mạng để tối ưu tốc độ.  i. Firebase Crashlytics  Dịch vụ phát hiện và báo cáo lỗi giúp cải thiện độ ổn định của ứng dụng.  **3. Lợi ích và ứng dụng của Firebase trong phát triển ứng dụng**  a. Lợi ích của Firebase   * Tiết kiệm thời gian phát triển: Firebase cung cấp nhiều dịch vụ backend sẵn có, giúp lập trình viên tập trung vào phát triển tính năng chính. * Khả năng mở rộng tốt: Firebase sử dụng hạ tầng của Google Cloud, hỗ trợ ứng dụng phát triển mạnh mẽ mà không cần quản lý máy chủ. * Dễ dàng tích hợp: Firebase hỗ trợ nhiều ngôn ngữ và framework như Java, Kotlin, Swift, Flutter, React Native. * Bảo mật cao: Firebase cung cấp các cơ chế bảo mật mạnh mẽ như xác thực người dùng và kiểm soát quyền truy cập dữ liệu.   b. Ứng dụng của Firebase   * Ứng dụng chat, mạng xã hội: Nhờ vào Firebase Realtime Database và Firestore, các ứng dụng chat có thể đồng bộ dữ liệu tức thì. * Thương mại điện tử: Firebase hỗ trợ lưu trữ sản phẩm, xử lý thanh toán, theo dõi hành vi người dùng qua Firebase Analytics. * Ứng dụng học tập trực tuyến: Firebase giúp quản lý nội dung, gửi thông báo và đồng bộ dữ liệu bài giảng. * Game mobile: Firebase giúp lưu trữ tiến trình người chơi, gửi thông báo và giám sát lỗi trong game. * Firebase là một công cụ mạnh mẽ giúp lập trình viên xây dựng ứng dụng nhanh chóng và hiệu quả. Với hệ sinh thái đa dạng, Firebase đang trở thành lựa chọn hàng đầu cho nhiều dự án ứng dụng di động và web hiện nay. |

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>>

**Mã nguỗn của file main\_activity.xml : thiết kế giao diện in ra danh sách các email**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <ListView  android:id="@+id/lstView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" /> </LinearLayout> |

**A screenshot of a computer

AI-generated content may be incorrect.**

**Mã nguồn của file MainActivity.kt : để đọc ra danh sách các email từ firebase realtime database ra màn hình**

|  |
| --- |
| package com.example.reviewfirebaserealtimedatabase  import android.annotation.SuppressLint import android.os.Bundle import android.util.Log import android.widget.ArrayAdapter import android.widget.Button import android.widget.EditText import android.widget.ListView import android.widget.Toast import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.activity.enableEdgeToEdge import androidx.compose.foundation.layout.fillMaxSize import androidx.compose.foundation.layout.padding import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.runtime.Composable import androidx.compose.ui.Modifier import androidx.compose.ui.tooling.preview.Preview import com.example.reviewfirebaserealtimedatabase.ui.theme.ReviewFirebaseRealtimeDatabaseTheme import com.google.firebase.auth.FirebaseAuth import com.google.firebase.database.DataSnapshot import com.google.firebase.database.DatabaseError import com.google.firebase.database.DatabaseReference import com.google.firebase.database.FirebaseDatabase import com.google.firebase.database.ValueEventListener  class MainActivity : ComponentActivity() {  private lateinit var listViewUsers: ListView  private lateinit var userList: ArrayList<String>  private lateinit var arrayAdapter: ArrayAdapter<String>  private lateinit var database: DatabaseReference   @SuppressLint("MissingInflatedId")  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*main\_activity*)  listViewUsers = findViewById(R.id.*lstView*)  userList = ArrayList()  arrayAdapter = ArrayAdapter(this, android.R.layout.*simple\_list\_item\_1*, userList)  listViewUsers.*adapter* = arrayAdapter   database = FirebaseDatabase.getInstance().getReference("Users")  loadUserData()  }  private fun loadUserData() {  database.addValueEventListener(object : ValueEventListener {  override fun onDataChange(snapshot: DataSnapshot) {  userList.clear()  for (userSnapshot in snapshot.*children*) {  val user = userSnapshot.getValue(Users::class.*java*)  user?.email?.*let* { userList.add(it) }  }  arrayAdapter.notifyDataSetChanged()  }   override fun onCancelled(error: DatabaseError) {  Toast.makeText(this@MainActivity, "Lỗi tải dữ liệu!", Toast.*LENGTH\_SHORT*).show()  }  })  }   } |

**Mã nguồn của file dangnhap.xml: thiết kế giao diện đăng nhập khi ấn nút Login thì chuyển sang giao diện khác. Khi chúng ta nhập đầy đủ các trường thông tin nếu ta nhấn vào Display thì hiện ra giao diện main\_activity.xml và nếu ta ấn vào Register thì sẽ hiện ra giao diện dangki.xml.**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <TextView  android:id="@+id/txtEmail"  android:layout\_width="match\_parent"  android:layout\_height="35dp"  android:text="Email" />   <EditText  android:id="@+id/edtEmail"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:hint="@string/google\_storage\_bucket"  android:inputType="textEmailAddress"  android:minHeight="48dp" />   <TextView  android:id="@+id/txtPassword"  android:layout\_width="match\_parent"  android:layout\_height="37dp"  android:text="Password" />   <EditText  android:id="@+id/edtPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="textPassword"  android:minHeight="48dp"  android:text="Password" />   <Button  android:id="@+id/btnDangNhap"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Login" />  <Button  android:id="@+id/btnDangKi"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Register" />   <Button  android:id="@+id/btnHienthi"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Display" /> </LinearLayout> |

**A screenshot of a computer

AI-generated content may be incorrect.**

**Mã nguồn của file LoginAcctivity.kt: là mã nguồn xử lý sự kiện đăng nhập, đăng kí, và hiển thị dữ liệu lên màn hình.**

|  |
| --- |
| package com.example.reviewfirebaserealtimedatabase  import android.content.Intent import android.os.Bundle import android.text.TextUtils import android.util.Log import android.widget.Button import android.widget.EditText import android.widget.Toast import androidx.activity.ComponentActivity import com.google.firebase.auth.FirebaseAuth import com.google.firebase.database.DataSnapshot import com.google.firebase.database.DatabaseError import com.google.firebase.database.DatabaseReference import com.google.firebase.database.FirebaseDatabase import com.google.firebase.database.ValueEventListener  class LoginActivity: ComponentActivity() {  private lateinit var database: FirebaseDatabase  private lateinit var myRef: DatabaseReference  private lateinit var edtemail: EditText  private lateinit var edtpassword: EditText  private lateinit var btnDangNhap: Button  private lateinit var btnDangKi: Button  private lateinit var btnHienthi: Button  private lateinit var mAuth: FirebaseAuth  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*dangnhap*)  mAuth = FirebaseAuth.getInstance()  edtemail = findViewById(R.id.*edtEmail*)  edtpassword = findViewById(R.id.*edtPassword*)  btnDangNhap = findViewById(R.id.*btnDangNhap*)  btnDangKi = findViewById(R.id.*btnDangKi*)  btnHienthi = findViewById(R.id.*btnHienthi*)  btnDangNhap.setOnClickListener {  login()  }  btnDangKi.setOnClickListener {  register()   }  btnHienthi.setOnClickListener {  onclickPushData()  val i = Intent(this@LoginActivity, MainActivity::class.*java*)  startActivity(i)  }   }   private fun onclickPushData() {  val email = edtemail.*text*.toString().*trim*()   if (email.*isEmpty*()) {  Toast.makeText(this, "Vui lòng nhập email!", Toast.*LENGTH\_SHORT*).show()  return  }   val database = FirebaseDatabase.getInstance()  val myRef = database.getReference("Users")   val userId = myRef.push().*key* // Tạo một ID ngẫu nhiên  val user = Users(email)   if (userId != null) {  myRef.child(userId).setValue(user)  .addOnSuccessListener {  Toast.makeText(this, "Ghi dữ liệu thành công!", Toast.*LENGTH\_SHORT*).show()  }  .addOnFailureListener { e ->  Toast.makeText(this, "Lỗi khi ghi dữ liệu: ${e.message}", Toast.*LENGTH\_SHORT*).show()  }  }  }   private fun login() {  val email = edtemail.getText().toString()  val pass = edtpassword.getText().toString()  if (TextUtils.isEmpty(email)) {  Toast.makeText(this, "vui long nhap email!!!", Toast.*LENGTH\_SHORT*).show()  return  }  if (TextUtils.isEmpty(pass)) {  Toast.makeText(this, "vui long nhap password!!!", Toast.*LENGTH\_SHORT*).show()  return  }  mAuth.signInWithEmailAndPassword(email, pass).addOnCompleteListener { task ->  if (task.*isSuccessful*) {  Toast.makeText(this, "Dang nhap thanh cong!!!", Toast.*LENGTH\_SHORT*).show()  val i = Intent(this@LoginActivity, MainActivity::class.*java*)  startActivity(i)  finish()  } else {  Toast.makeText(this@LoginActivity, "Dang nhap that bai!!!", Toast.*LENGTH\_SHORT*).show()  }   }  }   private fun register() {  val i = Intent(this@LoginActivity, RegisterActivity::class.*java*)  startActivity(i)  }  } |

**Mã nguồn của file dangki.xml: thiết kế giao diện Đăng kí các trường thông tin email và password**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <TextView  android:id="@+id/txtEmail"  android:layout\_width="match\_parent"  android:layout\_height="35dp"  android:text="Email" />   <EditText  android:id="@+id/edtEmail"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:hint="@string/google\_storage\_bucket"  android:inputType="textEmailAddress"  android:minHeight="48dp" />   <TextView  android:id="@+id/txtPassword"  android:layout\_width="match\_parent"  android:layout\_height="37dp"  android:text="Password" />   <EditText  android:id="@+id/edtPassword"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:ems="10"  android:inputType="textPassword"  android:minHeight="48dp"  android:text="Password" />  <Button  android:id="@+id/btnDangKi"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="Register" /> </LinearLayout> |

A screenshot of a computer

AI-generated content may be incorrect.

**Mã nguồn của file RegisterAcctivity.kt: là mã nguồn xử lý sự kiện đăng kí.**

|  |
| --- |
| package com.example.reviewfirebaserealtimedatabase  import android.content.Intent import android.os.Bundle import android.text.TextUtils import android.widget.Button import android.widget.EditText import android.widget.Toast import androidx.activity.ComponentActivity import com.google.firebase.auth.FirebaseAuth import com.google.firebase.database.FirebaseDatabase  class RegisterActivity: ComponentActivity() {  private lateinit var edtemail: EditText  private lateinit var edtpassword: EditText  private lateinit var btnDangKi: Button  private lateinit var mAuth: FirebaseAuth  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*dangki*)   mAuth = FirebaseAuth.getInstance()   edtemail = findViewById(R.id.*edtEmail*)  edtpassword = findViewById(R.id.*edtPassword*)  btnDangKi = findViewById(R.id.*btnDangKi*)  btnDangKi.setOnClickListener {  register()   }   }   private fun register() {  val email = edtemail.getText().toString()  val pass = edtpassword.getText().toString()  if (TextUtils.isEmpty(email)) {  Toast.makeText(this, "vui long nhap email!!!", Toast.*LENGTH\_SHORT*).show()  return  }  if (TextUtils.isEmpty(pass)) {  Toast.makeText(this, "vui long nhap password!!!", Toast.*LENGTH\_SHORT*).show()  return  }  mAuth.createUserWithEmailAndPassword(email, pass).addOnCompleteListener { task ->  if (task.*isSuccessful*) {  Toast.makeText(this, "Tao tai khoan thanh cong!", Toast.*LENGTH\_SHORT*).show()  val user = mAuth.*currentUser* user?.*let* {  saveUserToDatabase(it.*uid*, email) // Lưu vào Firebase Realtime Database  // Thực hiện các thao tác khác sau khi đăng ký thành công   }  val i = Intent(this@RegisterActivity, LoginActivity::class.*java*)  startActivity(i)  finish()  } else {  Toast.makeText(this@RegisterActivity, "Dang ki that bai!!!", Toast.*LENGTH\_SHORT*).show()  }   }   }  private fun saveUserToDatabase(uid: String, email: String) {  val database = FirebaseDatabase.getInstance().getReference("Users")  val user = Users(email) // Đảm bảo class Users có constructor mặc định   database.child(uid).setValue(user).addOnCompleteListener { task ->  if (task.*isSuccessful*) {  Toast.makeText(this, "Lưu thông tin người dùng thành công!", Toast.*LENGTH\_SHORT*).show()  } else {  Toast.makeText(this, "Lỗi khi lưu dữ liệu!", Toast.*LENGTH\_SHORT*).show()  }  }  } } |

**Mã nguồn file Users.kt:biểu diễn đối tượng như 1 data model**

|  |
| --- |
| package com.example.reviewfirebaserealtimedatabase   class Users {  var email: String? = null   constructor() {}  constructor(email: String?) {  this.email = email  } } |

**Kết quả chạy chương trình:**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Kết quả tích hợp với firebase realtime database:**

A screenshot of a computer

AI-generated content may be incorrect.

**Kết quả tích hợp với firebase authentication:**

A screenshot of a computer

AI-generated content may be incorrect.