

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	2251172530	Nguyễn Tiến Trọng	64KTPM3

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:

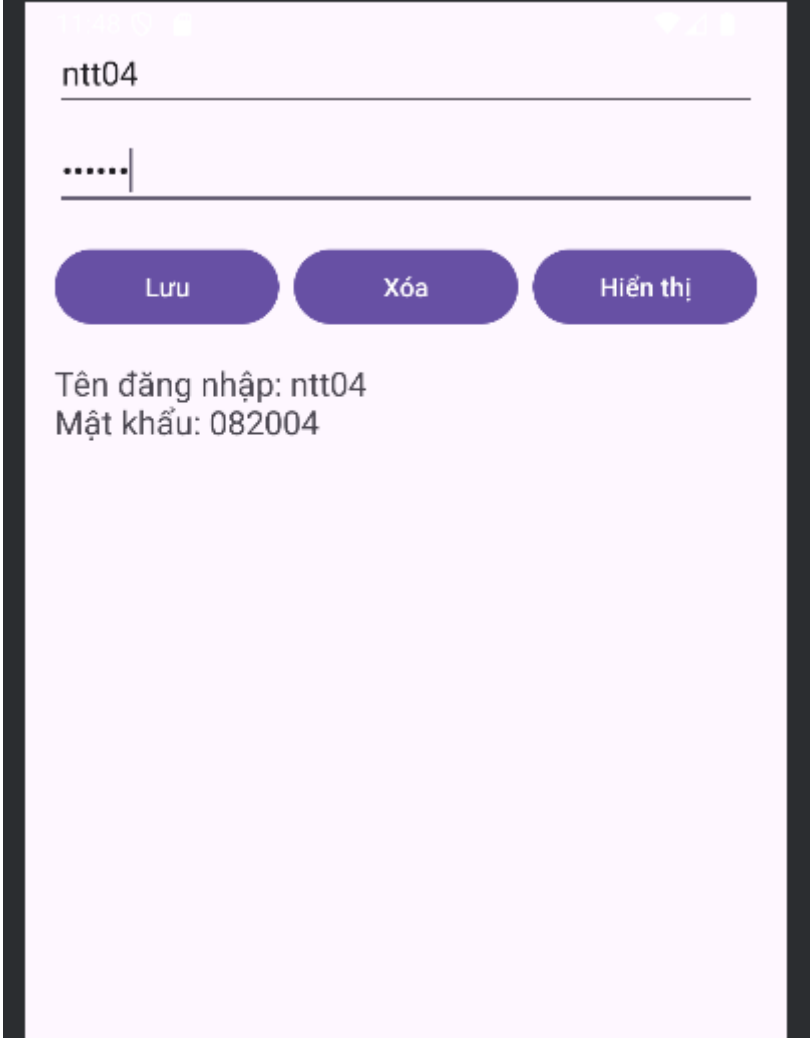
- Tạo một lớp helper **PreferenceHelper** để quản lý Shared Preference.
- Khi người dùng nhấn nút "Lưu", lưu tên người dùng và mật khẩu vào Shared Preference.
- Khi người dùng nhấn nút "Xóa", xóa dữ liệu đã lưu trong Shared Preference.
- Khi người dùng nhấn nút "Hiển thị", đọc dữ liệu từ Shared Preference 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>>



The screenshot shows a mobile application interface with a light purple background. At the top, there is a status bar with the time 10:00 and battery level 21%. Below the status bar, there are two input fields. The first field contains the text "ntt04". The second field contains six dots, indicating a password. Below the input fields, there are three buttons: "Lưu" (Save), "Xóa" (Delete), and "Hiển thị" (Show). Below the buttons, there is a text label "Tên đăng nhập: ntt04" and a text label "Mật khẩu: 082004".

```

package com.edu.sharedprefsdemo

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

class PreferenceHelper(context: Context) {

    private val PREFS_NAME = "MyPrefs"
    private val sharedPref: SharedPreferences = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE)

    fun save(KEY_NAME: String, text: String) {
        val editor: SharedPreferences.Editor = sharedPref.edit()
        editor.putString(KEY_NAME, text)
        editor.apply()
    }

    fun getValueString(KEY_NAME: String): String? {
        return sharedPref.getString(KEY_NAME, defValue: null)
    }

    fun clearSharedPreference() {
        val editor: SharedPreferences.Editor = sharedPref.edit()
        editor.clear()
        editor.apply()
    }

    fun removeValue(KEY_NAME: String) {
        val editor: SharedPreferences.Editor = sharedPref.edit()
        editor.remove(KEY_NAME)
        editor.apply()
    }
}

```

```

package com.edu.sharedprefsdemo

import ...

class MainActivity : AppCompatActivity() {

    private lateinit var usernameEditText: EditText
    private lateinit var passwordEditText: EditText
    private lateinit var saveButton: Button
    private lateinit var deleteButton: Button
    private lateinit var displayButton: Button
    private lateinit var displayTextView: TextView
    private lateinit var preferenceHelper: PreferenceHelper

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

        usernameEditText = findViewById(R.id.usernameEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        saveButton = findViewById(R.id.saveButton)
        deleteButton = findViewById(R.id.deleteButton)
        displayButton = findViewById(R.id.displayButton)
        displayTextView = findViewById(R.id.displayTextView)
        preferenceHelper = PreferenceHelper(context: this)

        saveButton.setOnClickListener {
            val username = usernameEditText.text.toString()
            val password = passwordEditText.text.toString()
            preferenceHelper.save(KEY_NAME: "username", username)
            preferenceHelper.save(KEY_NAME: "password", password)
        }
    }
}

```

```

saveButton = findViewById(R.id.saveButton)
deleteButton = findViewById(R.id.deleteButton)
displayButton = findViewById(R.id.displayButton)
displayTextView = findViewById(R.id.displayTextView)
preferenceHelper = PreferenceHelper(context: this)

saveButton.setOnClickListener {
    val username = usernameEditText.text.toString()
    val password = passwordEditText.text.toString()
    preferenceHelper.save(KEY_NAME: "username", username)
    preferenceHelper.save(KEY_NAME: "password", password)
}

deleteButton.setOnClickListener {
    preferenceHelper.clearSharedPreferences()
    displayTextView.text = ""
}

displayButton.setOnClickListener {
    val username = preferenceHelper.getValueString(KEY_NAME: "username")
    val password = preferenceHelper.getValueString(KEY_NAME: "password")
    displayTextView.text = "Tên đăng nhập: $username\nMật khẩu: $password"
}
}
}

```

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>>

The screenshot shows a mobile application interface with a dark gray background. At the top, there is a status bar showing the time 11:59. Below the status bar, there are two input fields: "Tên" (Name) and "Số điện thoại" (Phone number). Below these fields are two buttons: "Thêm" (Add) and "Hiển thị" (Display). Below the buttons, there are two lines of text: "Tên: ntt, SĐT: 02432543" and "Tên: qwwr, SĐT: 0242353". Below the text, there is a light blue rounded rectangle containing the title "Chỉnh sửa/Xóa" (Edit/Delete). Inside this rectangle, there are two input fields: "Tên" (Name) with the value "ntt" and "Số điện thoại" (Phone number) with the value "02432543". Below the input fields are two buttons: "Xóa" (Delete) and "Lưu" (Save).

DBHelper:

```

package com.vinnorman.sqlitetest

> import ...

class DBHelper(val context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null, DATABASE_VERSION)

    companion object {
        private const val DATABASE_NAME = "ContactDB.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "contacts"
        private const val COLUMN_NAME = "name"
        private const val COLUMN_PHONE = "phone"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = ("CREATE TABLE " + TABLE_NAME + "("
            + COLUMN_NAME + " TEXT," + COLUMN_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()
        values.put(COLUMN_NAME, name)
        values.put(COLUMN_PHONE, phone)
        val result = db.insert(TABLE_NAME, nullColumnHack: null, values)

    fun updateContact(oldName: String, newName: String, newPhone: String): Int {
        val db = this.writableDatabase
        val values = ContentValues()
        values.put(COLUMN_NAME, newName) // Update name
        values.put(COLUMN_PHONE, newPhone)
        val result = db.update(TABLE_NAME, values, whereClause: "$COLUMN_NAME=?", arrayOf(oldName))
        db.close()
        return result
    }

    fun deleteContact(name: String): Int {
        val db = this.writableDatabase
        val result = db.delete(TABLE_NAME, whereClause: "$COLUMN_NAME=?", arrayOf(name))
        db.close()
        return result
    }

    fun getAllContacts(): Cursor {
        val db = this.readableDatabase
        return db.rawQuery("SELECT * FROM $TABLE_NAME", selectionArgs: null)
    }
}

```

MainActivity:

```

class MainActivity : ComponentActivity() {

    private lateinit var dbHelper: DBHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        dbHelper = DBHelper(context = this)

        setContent {
            SQLiteTestTheme {
                MainScreen(dbHelper)
            }
        }
    }
}

@Composable
fun MainScreen(dbHelper: DBHelper) {
    var name by remember { mutableStateOf(value: "") }
    var phone by remember { mutableStateOf(value: "") }
    var contacts by remember { mutableStateOf(emptyList<Contact>()) }
    var showDialog by remember { mutableStateOf(value: false) }
    var selectedContact by remember { mutableStateOf<Contact?>(value: null) }
    var editName by remember { mutableStateOf(value: "") }
    var editPhone by remember { mutableStateOf(value: "") }
    var showContacts by remember { mutableStateOf(value: false) } // Thêm state để kiểm soát hiển thị danh sách

    fun refreshContacts() {
        contacts = getContacts(dbHelper)
    }

    fun refreshContacts() {
        contacts = getContacts(dbHelper)
    }
}

Surface(modifier = Modifier.fillMaxSize(), color = MaterialTheme.colorScheme.background) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        OutlinedTextField(
            value = name,
            onChange = { name = it },
            label = { Text(text: "Tên") },
            modifier = Modifier.fillMaxWidth(),
            shape = RoundedCornerShape(8.dp)
        )
        OutlinedTextField(
            value = phone,
            onChange = { phone = it },
            label = { Text(text: "Số điện thoại") },
            modifier = Modifier.fillMaxWidth(),
            shape = RoundedCornerShape(8.dp)
        )
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.spacedBy(8.dp)
        ) {

```



```

Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.spacedBy(8.dp)
) {
    Button(
        onClick = {
            addContact(dbHelper, name, phone)
            name = ""
            phone = ""
            Toast.makeText(dbHelper.context, text: "Thêm thành công", Toast.LENGTH_SHORT).show()
        },
        modifier = Modifier.weight(1f),
        shape = RoundedCornerShape(8.dp)
    ) {
        Text(text: "Thêm")
    }
    Button(
        onClick = {
            refreshContacts()
            showContacts = true // Hiển thị danh sách khi nhấn "Hiển thị"
        },
        modifier = Modifier.weight(1f),
        shape = RoundedCornerShape(8.dp)
    ) {
        Text(text: "Hiển thị")
    }
}
}

```

```

if (showContacts) { // Chỉ hiển thị danh sách nếu showContacts là true
    LazyColumn {
        items(contacts) { contact ->
            Text(
                text = "Tên: ${contact.name}, SĐT: ${contact.phone}",
                modifier = Modifier
                    .fillMaxWidth()
                    .clickable {
                        selectedContact = contact
                        editName = contact.name
                        editPhone = contact.phone
                        showDialog = true
                    }
                    .padding(8.dp)
            )
        }
    }
}
}
}
}
}

```

```

if (showDialog) {
    AlertDialog(
        onDismissRequest = { showDialog = false },
        title = { Text(text: "Chỉnh sửa/Xóa") },
        text = {
            Column {
                OutlinedTextField(
                    value = editName,
                    onValueChange = { editName = it },
                    label = { Text(text: "Tên") },
                    modifier = Modifier.fillMaxWidth()
                )
                OutlinedTextField(
                    value = editPhone,
                    onValueChange = { editPhone = it },
                    label = { Text(text: "Số điện thoại") },
                    modifier = Modifier.fillMaxWidth()
                )
            }
        },
        confirmButton = {
            TextButton(
                onClick = {
                    selectedContact?.let {
                        updateContact(dbHelper, it.name, editName, editPhone)
                        refreshContacts()
                        showDialog = false
                    }
                }
            )
        },
        dismissButton = {
            TextButton(
                onClick = {
                    refreshContacts()
                    showDialog = false
                }
            )
        },
        textButtons = {
            TextButton(
                onClick = {
                    refreshContacts()
                    showDialog = false
                }
            )
        },
        text = {
            Text(text: "Xóa")
        }
    )
}

data class Contact(val name: String, val phone: String)

fun getContacts(dbHelper: DBHelper): List<Contact> {
    val cursor = dbHelper.getAllContacts()
    val contacts = mutableListOf<Contact>()
    if (cursor.moveToFirst()) {
        do {
            val name = cursor.getString(cursor.getColumnIndex(columnName: "name"))
            val phone = cursor.getString(cursor.getColumnIndex(columnName: "phone"))
            contacts.add(Contact(name, phone))
        } while (cursor.moveToNext())
    }
    return contacts
}

```

```
fun addContact(dbHelper: DBHelper, name: String, phone: String) {
    val result = dbHelper.addContact(name, phone)
}

fun updateContact(dbHelper: DBHelper, oldName: String, name: String, phone: String) {
    val result = dbHelper.updateContact(oldName, name, phone)
    if (result > 0) {
        Toast.makeText(dbHelper.context, text: "Cập nhật thành công", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(dbHelper.context, text: "Cập nhật thất bại", Toast.LENGTH_SHORT).show()
    }
}

fun deleteContact(dbHelper: DBHelper, name: String) {
    val result = dbHelper.deleteContact(name)
    if (result > 0) {
        Toast.makeText(dbHelper.context, text: "Xóa thành công", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(dbHelper.context, text: "Xóa thất bại", Toast.LENGTH_SHORT).show()
    }
}
```

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

Giới thiệu tổng quan về Firebase và lịch sử phát triển

Firebase là nền tảng phát triển ứng dụng được Google phát triển, cung cấp các công cụ và dịch vụ để xây dựng ứng dụng web và di động một cách nhanh chóng, hiệu quả. Firebase bắt đầu vào năm 2011 dưới dạng một startup chuyên cung cấp dịch vụ cơ sở dữ liệu thời gian thực. Sau khi được Google mua lại vào năm 2014, Firebase đã phát triển thành một hệ sinh thái toàn diện, hỗ trợ cả frontend lẫn backend, với khả năng mở rộng cao và tích hợp chặt chẽ với hạ tầng Google Cloud Platform.

Mô tả chi tiết các dịch vụ chính của Firebase

- *Firebase Authentication*

Dịch vụ xác thực người dùng hỗ trợ nhiều phương thức như email/password, số điện thoại, tài khoản Google, Facebook, Twitter,... Giúp quản lý người dùng an toàn và dễ dàng mà không cần xây dựng hệ thống xác thực riêng.

- ***Firebase Realtime Database và Cloud Firestore***

1. **Realtime Database:** Cơ sở dữ liệu NoSQL lưu trữ dữ liệu dưới dạng JSON, đồng bộ hóa theo thời gian thực đến tất cả client.
2. **Cloud Firestore:** Phiên bản cải tiến, linh hoạt hơn, hỗ trợ truy vấn mạnh mẽ, cấu trúc dữ liệu theo dạng tài liệu (document) và bộ sưu tập (collection), đồng bộ thời gian thực, tích hợp tốt với Google Cloud.

- ***Firebase Cloud Functions***

Cho phép chạy mã backend (Node.js) dưới dạng serverless khi xảy ra sự kiện từ Firebase (như ghi dữ liệu, đăng ký người dùng) hoặc từ HTTP request. Giúp mở rộng ứng dụng mà không cần quản lý máy chủ.

- ***Firebase Cloud Messaging (FCM)***

Hệ thống gửi thông báo đẩy miễn phí đến thiết bị Android, iOS và trình duyệt web. Hỗ trợ thông báo theo chủ đề, đến từng thiết bị hoặc nhóm thiết bị.

- ***Firebase Storage***

Cung cấp giải pháp lưu trữ tệp tin (hình ảnh, video, tài liệu...) trên đám mây với khả năng mở rộng và bảo mật cao, tích hợp sẵn với Firebase Authentication để kiểm soát truy cập.

- ***Firebase Machine Learning (ML)***

Tích hợp các tính năng trí tuệ nhân tạo vào ứng dụng, gồm ML Kit (có sẵn nhiều API như nhận diện văn bản, khuôn mặt, dịch ngôn ngữ...) và khả năng triển khai mô hình tùy chỉnh.

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

- ***Lợi ích:***

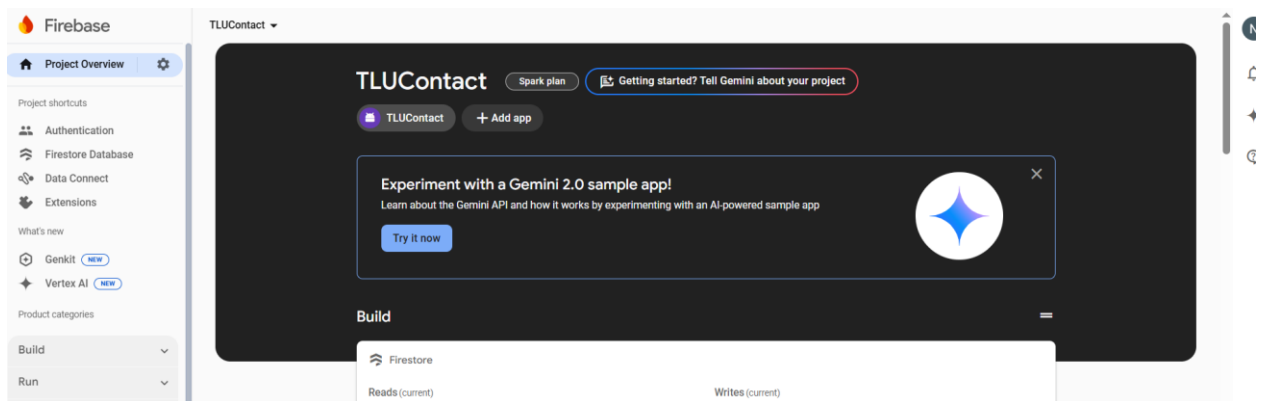
- Giảm thời gian phát triển: Cung cấp giải pháp backend hoàn chỉnh.
- Dễ dàng mở rộng và tích hợp với Google Cloud.
- Đồng bộ hóa dữ liệu thời gian thực.
- Không cần quản lý hạ tầng (serverless).
- Bảo mật và đáng tin cậy với Google Cloud.

- ***Ứng dụng thực tế:***

- Ứng dụng chat, mạng xã hội.
- Ứng dụng thương mại điện tử.
- Game mobile cần đồng bộ dữ liệu.
- Hệ thống thông báo, cảnh báo.

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.

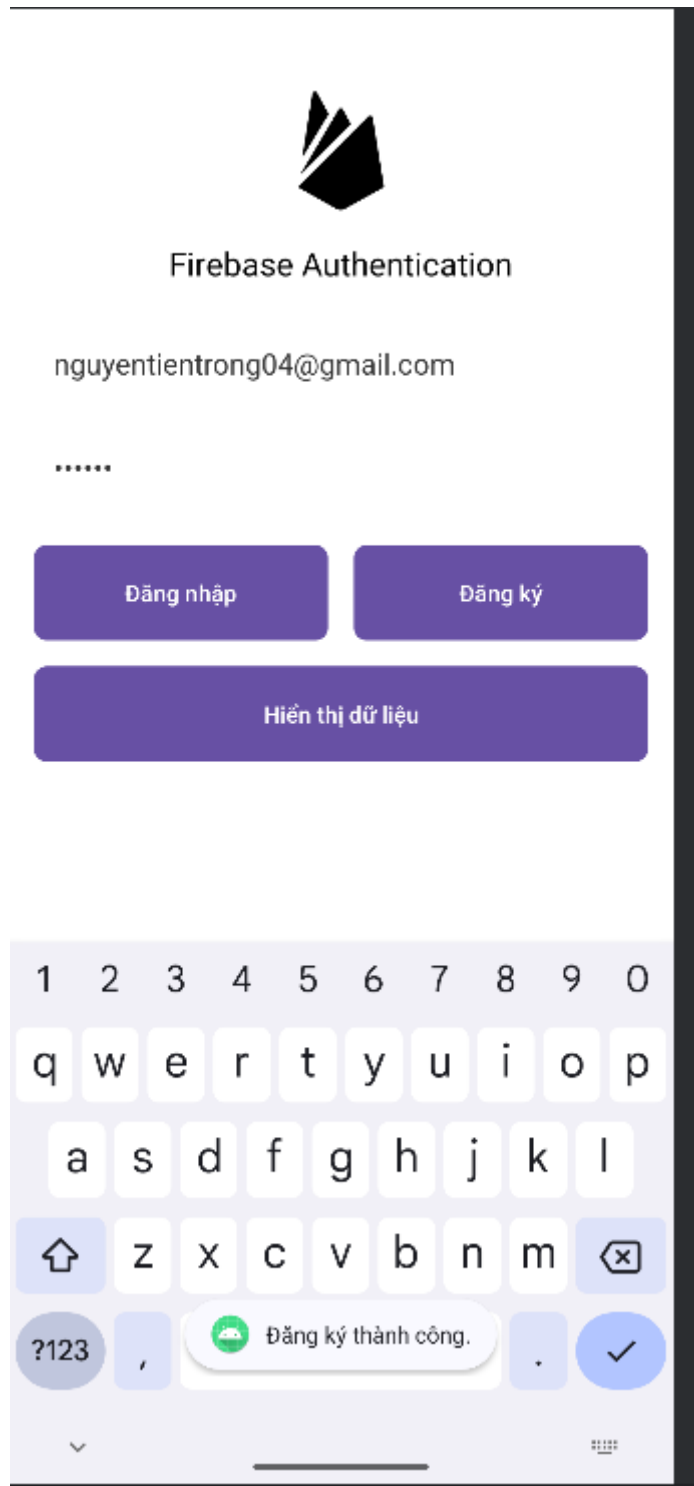
3. Tích hợp Firebase Realtime Database:

- 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>>



Thông tin người dùng

Thông tin tài khoản

Email: nguyentientrong04@gmail.com

UID: jvuACywmS3bsTVGVJE68rqZlouQ2

Thời gian đăng nhập: 24/03/2025 23:38:32

Dữ liệu từ Database

createdAt: 24/03/2025 23:38:08

lastLogin: 24/03/2025 23:38:32

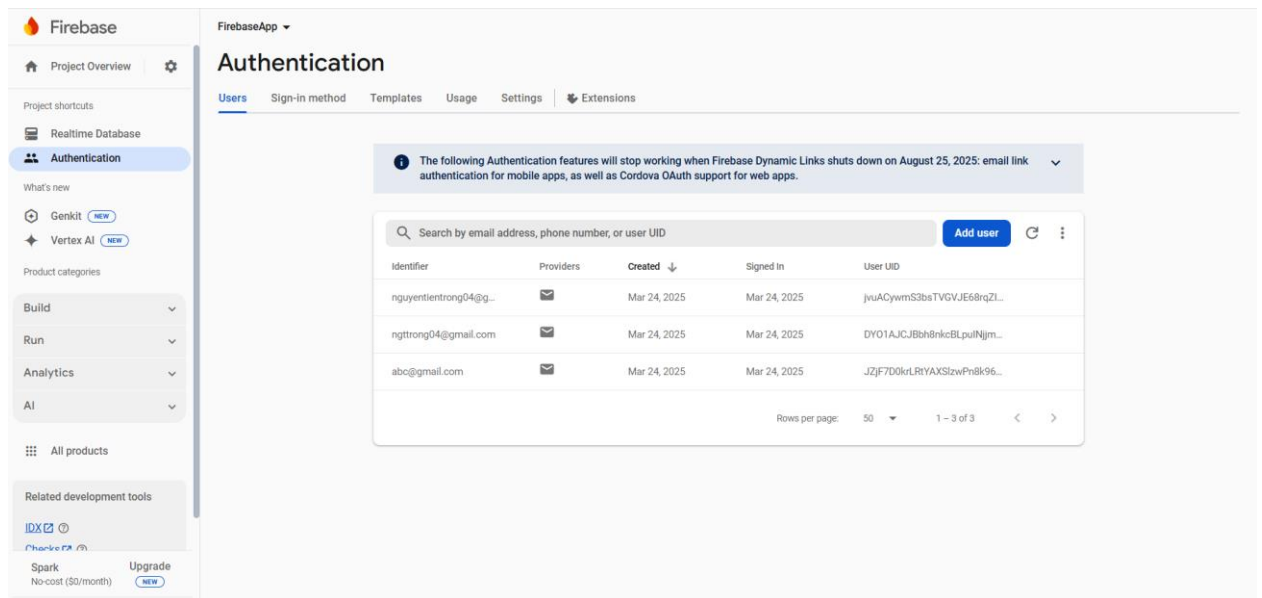
email: nguyentientrong04@gmail.com

Đăng xuất



Đăng nhập thành công.

Dữ liệu từ sau khi đăng kí từ firebase:



File UserInfoActivity.kt:

```

package com.edu.firebaseapp

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale

class UserInfoActivity : AppCompatActivity() {

    private lateinit var emailTextView: TextView
    private lateinit var uidTextView: TextView
    private lateinit var loginTimeTextView: TextView
    private lateinit var createdAtTextView: TextView
    private lateinit var lastLoginTextView: TextView
    private lateinit var emailDatabaseTextView: TextView
    private lateinit var logoutButton: Button

    private lateinit var auth: FirebaseAuth
    private lateinit var database: FirebaseDatabase

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

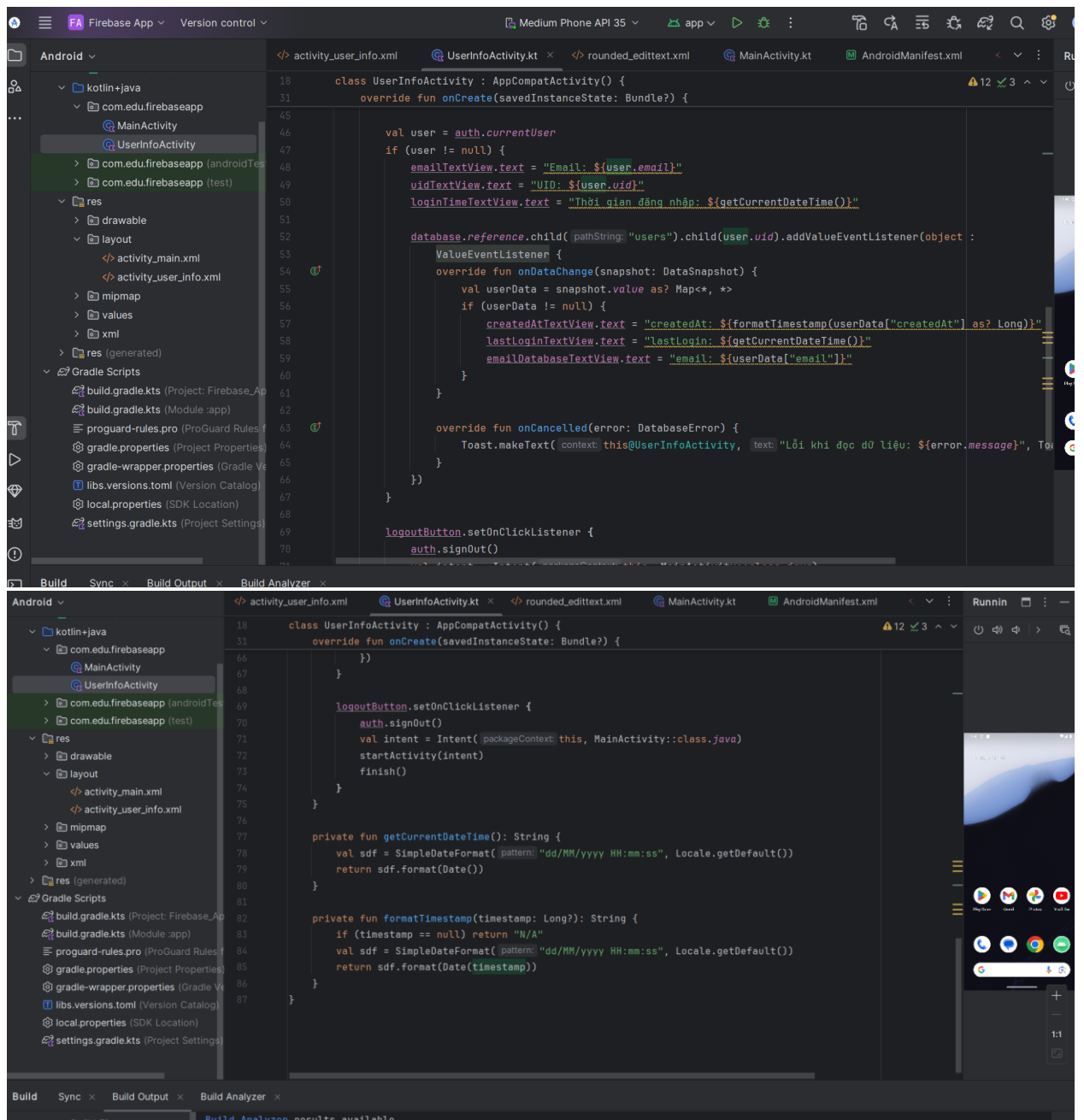
        emailTextView = findViewById(R.id.emailTextView)
        uidTextView = findViewById(R.id.uidTextView)
        loginTimeTextView = findViewById(R.id.loginTimeTextView)
        createdAtTextView = findViewById(R.id.createdAtTextView)
        lastLoginTextView = findViewById(R.id.lastLoginTextView)
        emailDatabaseTextView = findViewById(R.id.emailDatabaseTextView)
        logoutButton = findViewById(R.id.logoutButton)

        auth = FirebaseAuth.getInstance()
        database = FirebaseDatabase.getInstance()

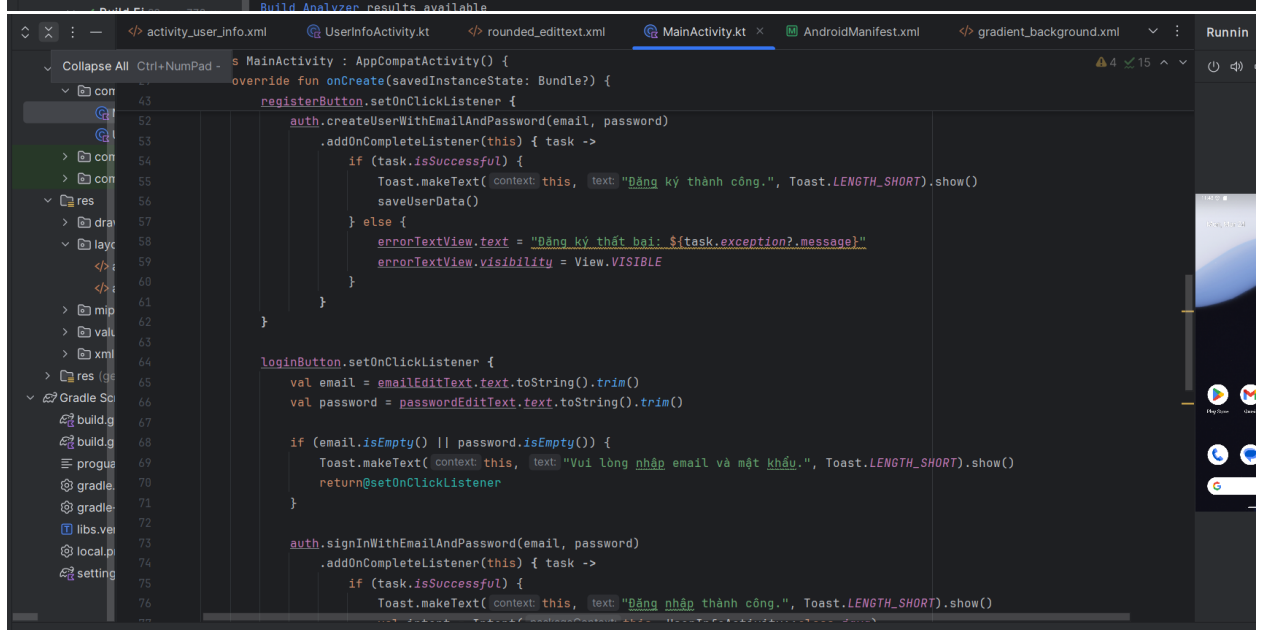
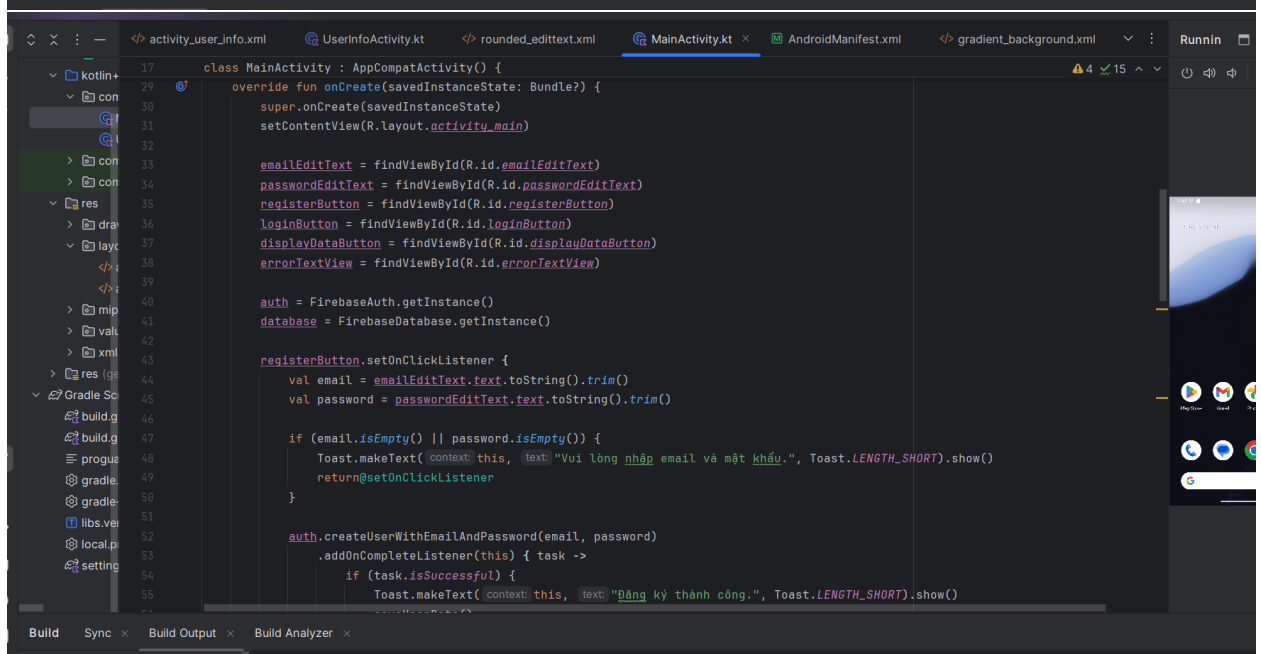
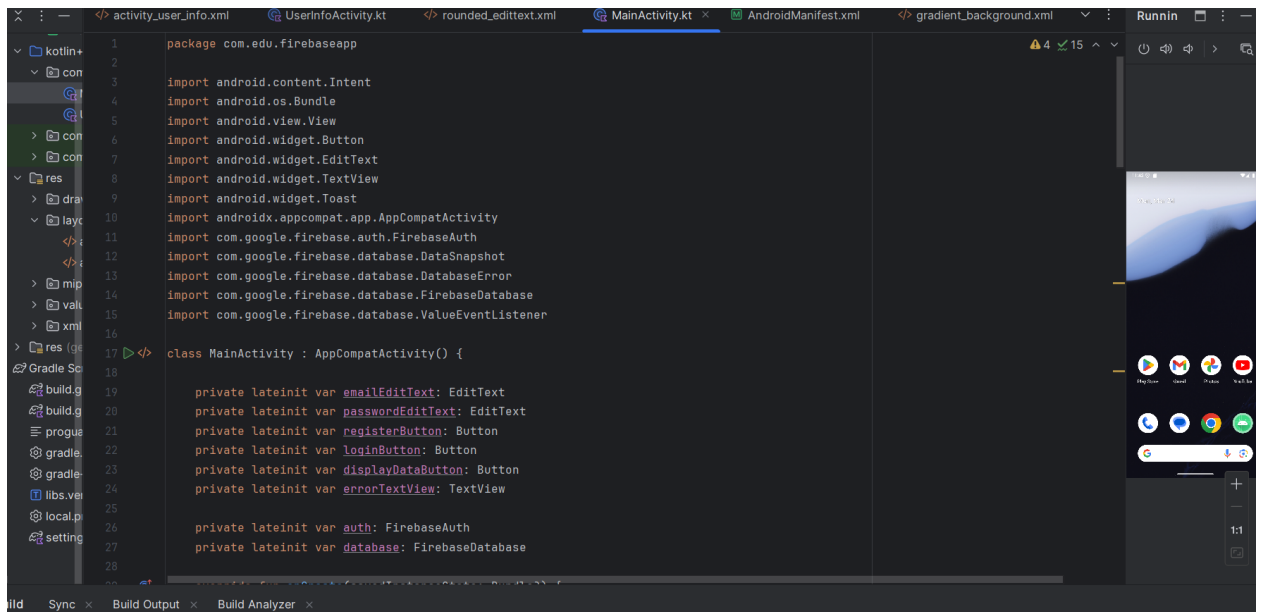
        val user = auth.currentUser
        if (user != null) {
            emailTextView.text = "Email: ${user.email}"
            uidTextView.text = "UID: ${user.uid}"
            loginTimeTextView.text = "Thời gian đăng nhập: ${getCurrentDateTime()}"

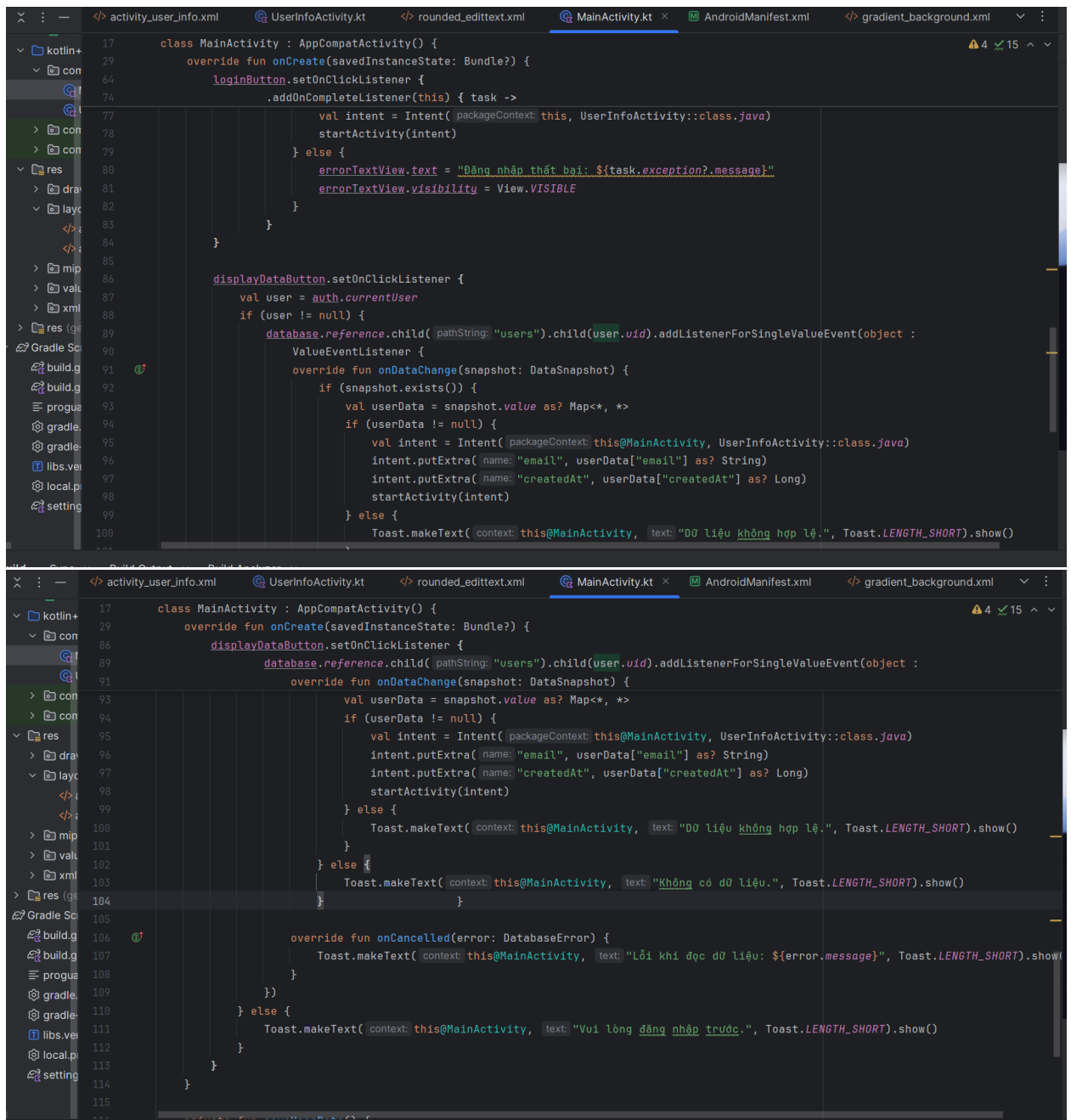
            database.reference.child(pathString: "users").child(user.uid).addValueEventListener(object :
                ValueEventListener {
                    override fun onDataChange(snapshot: DataSnapshot) {
                        val userData = snapshot.value as? Map<*, *>
                    }
                })
        }
    }
}

```



File MainActivity.kt





```

17      class MainActivity : AppCompatActivity() {
18          override fun onCreate(savedInstanceState: Bundle?) {
19              displayDataButton.setOnClickListener {
20                  database.reference.child(pathString: "users").child(user.uid).addListenerForSingleValueEvent(object :
21                      ValueEventListener() {
22                          override fun onDataChange(snapshot: DataSnapshot) {
23                              // Handle data change
24                          }
25                          override fun onCancelled(error: DatabaseError) {
26                              // Handle error
27                          }
28                      })
29              }
30          }
31          private fun saveUserData() {
32              val user = auth.currentUser
33              if (user != null) {
34                  val userData = hashMapOf(
35                      "email" to user.email,
36                      "createdAt" to System.currentTimeMillis()
37                  )
38                  database.reference.child(pathString: "users").child(user.uid).setValue(userData)
39              }
40          }
41      }
42  }

```

