

# BÁO CÁO THỰC HÀNH

Môn học: Bảo mật Web và Ứng dụng  
Lab 5: Basic Android Secure Programming

GVHD: Ngô Khánh Khoa

**Nhóm: 6**

## 1. THÔNG TIN CHUNG:

Lớp: NT213.P11.ANTT.2

STT	Họ và tên	MSSV	Email
1	Lại Quan Thiên	22521385	22521385@gm.uit.edu.vn
2	Mai Nguyễn Nam Phương	22521164	22521164@gm.uit.edu.vn
3	Hồ Diệp Huy	22520541	22520541@gm.uit.edu.vn
4	Nguyễn Phúc Nhi	22521041	22521041@gm.uit.edu.vn

## 2. NỘI DUNG THỰC HIỆN:<sup>1</sup>

STT	Nội dung	Tình Trạng	Thực hiện
1	Bài 1	100%	Nhóm 6
2	Bài 2	100%	Quan Thiên
3	Bài 3	100%	Nam Phương
4	Bài 4	100%	Diệp Huy
5	Bài 5	100%	Nam Phương
6	Bài 6	100%	Phúc Nhi

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

<sup>1</sup> Ghi nội dung công việc, các kịch bản trong bài Thực hành

# BÁO CÁO CHI TIẾT

Link Github chứa Source Code:

[https://github.com/WanThinnn/NT213\\_Web-and-application-security/tree/main/Labs/Lab\\_5/final\\_basic\\_app/final](https://github.com/WanThinnn/NT213_Web-and-application-security/tree/main/Labs/Lab_5/final_basic_app/final)

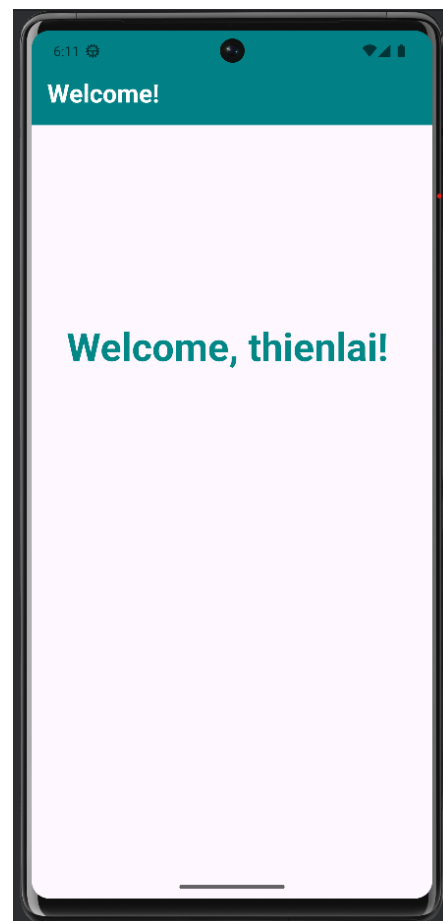
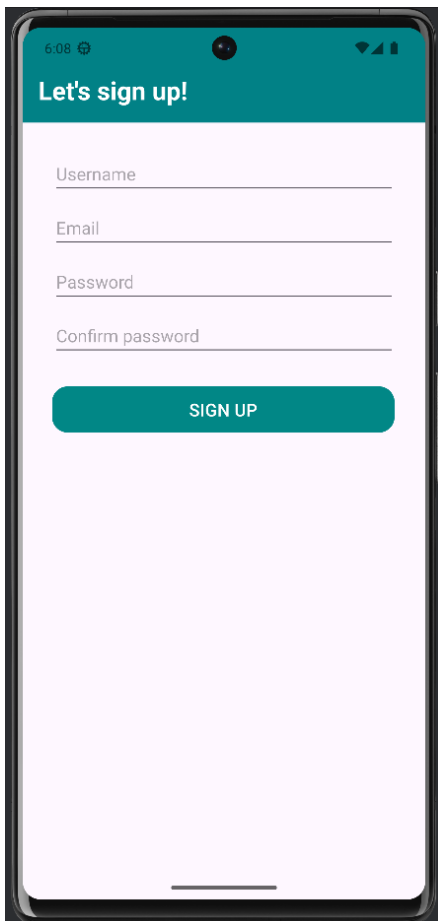
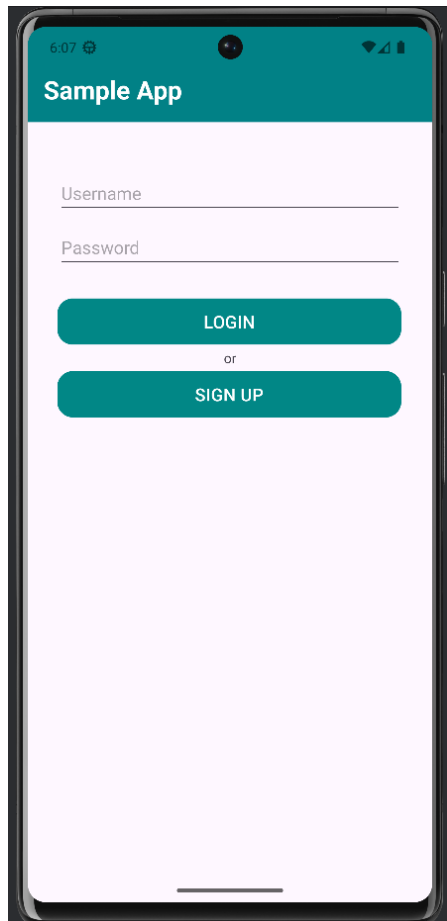
NT213_Web-and-application-security / Labs / Lab_5 / final_basic_app / final /			Add file ▾	⋮
WanThinnn add lab 5			9662661 · 1 minute ago	History
Name	Last commit message	Last commit date		
..				
backend	add lab 5	1 minute ago		
k/nt213_lab5	add lab 5	1 minute ago		

**Yêu cầu 2:** Sinh viên xây dựng ứng dụng Android gồm 3 giao diện chức năng chính:

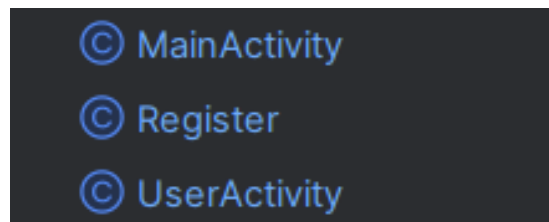
- Register - Đăng ký thông tin với ứng dụng (email, username, password).
- Login - Đăng nhập vào ứng dụng (username, password).
- Hiển thị thông tin người dùng (một lời chào có tên người dùng).

**Trả lời:**

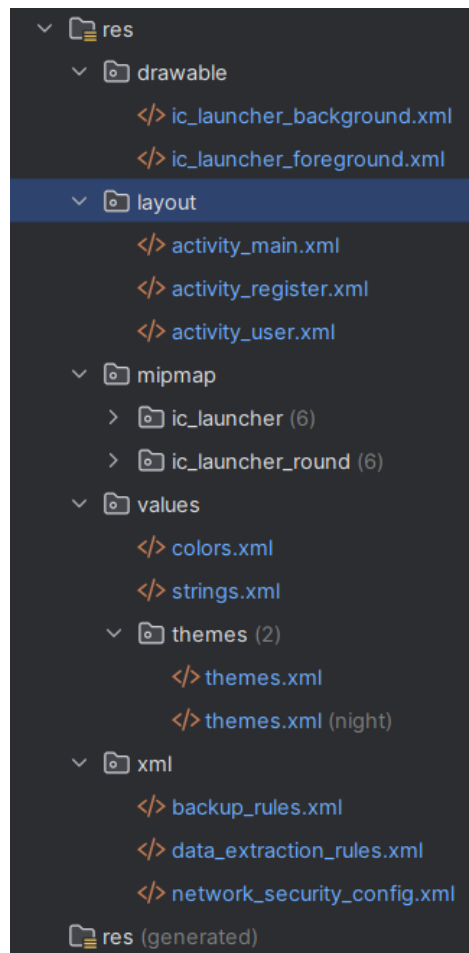
- Video demo chi tiết: <https://youtu.be/8V3RN7jI7o8>
- Giao diện chính của app:



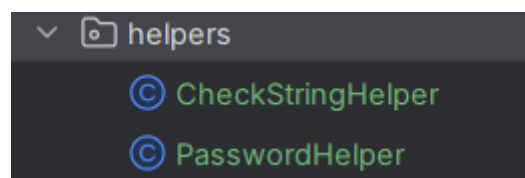
- Các Activity chính gồm:



- Một số file tùy chỉnh giao diện:



- Các module hỗ trợ cho app:





**Yêu cầu 3:** Sinh viên viết mã nguồn Java cho chức năng đăng nhập và đăng ký, sử dụng tập tin SQLiteConnector được giảng viên cung cấp để thực hiện kết nối đến cơ sở dữ liệu SQLite với các yêu cầu bên dưới.

- Video demo chi tiết: <https://youtu.be/msr-djKT4y0>
- Ta import SQLiteConnector để có thể sử dụng kết nối đến CSDL:

The first screenshot shows the MainActivity.java file with imports for Android and androidx classes, and the SQLiteConnector class from the example app.

```

3      import android.content.Intent;
4      import android.os.Bundle;
5      import android.view.View;
6      import android.widget.Button;
7      import android.widget.EditText;
8      import android.widget.Toast;
9
10
11     import androidx.activity.EdgeToEdge;
12     import androidx.appcompat.app.AppCompatActivity;
13
14     import com.example.app.sql.SQLiteConnector;
15     import com.example.app.entity.User;
16
  
```

The second screenshot shows the Register.java file with similar imports, including the SQLiteConnector class.

```

1      package com.example.app;
2
3      import android.content.Intent;
4      import android.os.Bundle;
5      import android.util.Log;
6      import android.widget.Button;
7      import android.widget.EditText;
8      import android.widget.Toast;
9
10     import androidx.activity.EdgeToEdge;
11     import androidx.appcompat.app.AppCompatActivity;
12
13     import com.example.app.sql.SQLiteConnector;
14
15     import com.example.app.entity.User;
  
```

- Sử dụng CSDL để lưu vào Local:

```

if (password_text.equals(matchPassword_text)) {
    User user = new User(name_text, email_text, password_text);
    SQLiteConnector db = new SQLiteConnector( context: Register.this);
    if (db.addUser(user)) {
        Toast.makeText( context: Register.this, text: "Add user successfully", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent( packageContext: Register.this, MainActivity.class);
        startActivity(intent);
    } else {
        Toast.makeText( context: Register.this, text: "Add user failed", Toast.LENGTH_SHORT).show();
    }
} else {
    Toast.makeText( context: Register.this, text: "Passwords do not match", Toast.LENGTH_SHORT).show();
}
  
```

```

sqlite> .table
android_metadata  user
sqlite> select * from user;
1|1|1|1
2|3|3|3
3|namphuong|namphuong|namphuong
4|qthhpn|qthhpn|qthhpn
  
```

- Sử dụng khi Login:

```
} else {  
    // Gọi phương thức checkUser để xác minh  
    boolean result = db.checkUser(username_text, password_text);  
    if (result) {  
        Toast.makeText(context: MainActivity.this, text: "Login successfully!", Toast.LENGTH_SHORT).show();  
        Intent intent = new Intent(packageContext: MainActivity.this, UserActivity.class);  
        intent.putExtra(name: "username", username.getText().toString());  
        startActivity(intent);  
        finish(); // Đóng LoginActivity  
    } else {  
        // Thông báo lỗi nếu đăng nhập thất bại  
        Toast.makeText(context: MainActivity.this, text: "Invalid Credentials", Toast.LENGTH_SHORT).show();  
    }  
}
```

**Yêu cầu 4: Điều chỉnh mã nguồn để password được lưu và kiểm tra dưới dạng mã hash thay vì plaintext.**

**Trả lời:**

- Video demo chi tiết: <https://youtu.be/SNdvDqRpHkk>
- Ta sử dụng thuật toán Bcrypt để bảo vệ mật khẩu:

```
private String hashPassword(String password) { 1 usage
    return BCrypt.hashpw(password, BCrypt.gensalt());
}
```

- Áp dụng vào app:

```
if (password_text.equals(matchPassword_text)) {
    String hashedPassword = hashPassword(password_text);
    User user = new User(name_text, email_text, hashedPassword);
}
```

- Ngoài ra, thêm một vài ràng buộc khi đăng ký tài khoản:

```
else {
    // Kiểm tra và chuẩn hóa tên
    if (!CheckStringHelper.validateNoWhitespaceOrDash(name_text)) {
        name.setError("Invalid username. Only letters and one space between words allowed.");
        name.requestFocus();
        return;
    }

    if (!CheckStringHelper.validateNoWhitespaceOrDash(email_text)) {
        email.setError("Invalid email. No whitespace or '-' allowed.");
        email.requestFocus();
        return;
    }

    if (!PasswordHelper.strength(matchPassword_text)) {
        password.setError("Weak password. Please use:\n" +
            "* both upper and lower case letters\n" +
            "* numbers\n" +
            "* special characters (e.g. !\"#$%&')\n" +
            "* from 10 to 128 characters sequence");
        password.requestFocus();
        return;
    }
}
```

- Kiểm tra:

```
sqlite> select * from user;
2|ThienLai|Thienlai159@gmail.com|$2a$10$Qi4Pm3V5pVue79illEtW6uX5TxTc6qnqG.9LTHuaL4RtJ0boN3FkS
```



**Yêu cầu 5: Tạo một cơ sở dữ liệu tương tự bên ngoài thiết bị, viết mã nguồn thực hiện kết nối đến CSDL này để truy vấn thay vì sử dụng SQLite.**

- Video demo chi tiết: <https://youtu.be/czhEhA8iwaw>
- Đầu tiên ta sẽ tạo một ứng dụng web cơ bản sử dụng Node.js và Express.js để quản lý đăng ký và đăng nhập người dùng.
- Máy chủ sẽ lắng nghe các yêu cầu HTTP trên cổng 8080 và sử dụng 2 API Post cho đăng nhập và đăng ký

```
const express = require('express');
const bodyParser = require('body-parser');
const bcrypt = require('bcrypt');

const app = express();
const PORT = 8080;
```

- + Import các thư viện cần thiết (express, bcrypt, body-parser)
- + Tạo một ứng dụng Express và đặt cổng lắng nghe là 8080

```
// Lưu trữ dữ liệu người dùng tạm thời
const users = [];
```

- + Nhóm sẽ sử dụng 1 mảng để tạm thời mô phỏng lưu cơ sở dữ liệu cho ứng dụng (dữ liệu sẽ mất khi server khởi động lại)

```
// API kiểm tra (health check)
app.get('/', (req, res) => {
    res.send('Server is running!');
});
```

- + Ngoài ra ta còn có thêm 1 API để kiểm tra trạng thái của server

```
// API đăng ký người dùng
app.post('/api/v1/user/register', (req, res) => {
  const { name, email, password } = req.body;

  if (!name || !email || !password) {
    return res.status(400).json({ message: 'All fields are required!' });
  }

  // Kiểm tra email đã tồn tại
  const existingUser = users.find(user => user.email === email);
  if (existingUser) {
    return res.status(400).json({ message: 'Email is already registered!' });
  }

  // Thêm người dùng mới
  const newUser = { name, email, password };
  users.push(newUser);

  // Hiển thị thông tin người dùng trong CMD
  console.log('New user added:', newUser);

  return res.status(201).json({ message: 'User registered successfully!', user: newUser });
});
```

+ Thực hiện tạo API cho phép người dùng gửi thông tin để đăng kí tài khoản mới. Cụ thể

- Lấy name, email, password từ body của yêu cầu
- Kiểm tra xem các trường có đủ hay không. Nếu thiếu, trả lỗi 400 Bad Request.
- Kiểm tra email có bị trùng với email đã tồn tại trong mảng users không.
- Nếu hợp lệ, thêm người dùng mới vào mảng users.
- In thông tin người dùng mới trong console và trả phản hồi thành công kèm thông tin

```
app.post('/api/v1/user/login', async (req, res) => {
  const { name, password } = req.body;

  const user = users.find(u => u.name === name);
  if (user && await bcrypt.compare(password, user.password)) {
    res.status(200).json({ name: user.name, email: user.email });
  } else {
    res.status(401).json({ message: 'Invalid username or password' });
  }
});
```

+ Thực hiện tạo API cho phép người dùng đăng nhập vào tài khoản. Cụ thể:

- Lấy name và password từ body của yêu cầu.
- Tìm người dùng có tên trùng với name.
- Dùng bcrypt.compare để kiểm tra mật khẩu (so sánh mật khẩu nhập vào với mật khẩu đã lưu).



- Nếu thông tin chính xác, trả về phản hồi thành công với name và email.
- Nếu thông tin không chính xác, trả lỗi 401 Unauthorized

- Tiếp theo sẽ là setting ở trong Android Studio
- Đầu tiên là ở lớp Register.java để xử lý đăng kí

```
//Call api register
btn_register.setOnClickListener(v -> {
    String name_text = name.getText().toString();
    String email_text = email.getText().toString();
    String password_text = password.getText().toString();
    String matchPassword_text = matchPassword.getText().toString();
    if (email_text.isEmpty() || password_text.isEmpty() || matchPassword_text.isEmpty()) {
        Toast.makeText(context: Register.this, text: "Please enter all the fields", Toast.LENGTH_SHORT).show();
    }
});
```

- + Gán sự kiện cho button Register
- + Lấy dữ liệu từ các trường nhập dữ liệu ở giao diện
- + Thực hiện kiểm tra nếu trường dữ liệu nào trống thì sẽ hiển thị thông báo và yêu cầu người dùng nhập đầy đủ

```
if (password_text.equals(matchPassword_text)) {
    String hashedPassword = hashPassword(password_text);
    User user = new User(name_text, email_text, hashedPassword);

    //Retrofit
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://10.0.2.2:8080/") // IP máy tính
        .addConverterFactory(GsonConverterFactory.create())
        .build();
    // Tạo interface API
    ApiService apiService = retrofit.create(ApiService.class);
}
```

- + Nếu các điều kiện tạo tài khoản thỏa mãn yêu cầu thì ta sẽ tạo đối tượng User mới, sau đó thiết lập Retrofit để gọi API
- baseUrl: địa chỉ server ( ở đây 10.0.2.2 đại diện cho localhost trên trình giả lập Android)
- GsonConverterFactory.create(): Dùng để chuyển đổi dữ liệu JSON giữa client và server
- + retrofit.create(ApiService.class): Tạo một đối tượng interface ApiService, chứa định nghĩa các API sẽ được gọi

```
// Gọi API
Call<User> call = apiService.registerUser(user);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        if (response.isSuccessful() && response.body() != null) {
            Toast.makeText(context: Register.this, text: "User registered successfully!", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(packageContext: Register.this, MainActivity.class);
            startActivity(intent);
        } else {
            Toast.makeText(context: Register.this, text: "Passwords do not match", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {
        // Xử lý lỗi khi không thể kết nối hoặc lỗi khác
        Toast.makeText(context: Register.this, text: "Error: " + t.getMessage(), Toast.LENGTH_SHORT).show();
        Log.e(tag: "RegisterActivity", msg: "Error: " + t.getMessage());
    }
});
} else {
    Toast.makeText(context: Register.this, text: "Passwords do not match", Toast.LENGTH_SHORT).show();
}
```

+ Gửi yêu cầu API đăng kí

+ Đồng thời xử lý các phản hồi từ server (tùy theo kết quả trả về là phản hồi thành công hay gặp các lỗi không thể kết nối đến server)

- Tiếp theo là lớp MainActivity.java để xử lý đăng nhập

```
login.setOnClickListener(v->{
    String usernameText = username.getText().toString();
    String passwordText = password.getText().toString();

    if (usernameText.isEmpty() || passwordText.isEmpty()) {
        Toast.makeText(context: MainActivity.this, text: "Please enter all the fields", Toast.LENGTH_SHORT).show();
        return;
    }

    User user = new User();
    user.setName(usernameText);
    user.setPassword(passwordText);
```

+ Tương tự như ở trên ta sẽ tạo sự kiện cho button login

+ Lấy dữ liệu từ người dùng nhập vào ở giao diện

+ Kiểm tra dữ liệu đầu vào, nếu bất kì trường nào bị bỏ trống thì hiển thị thông báo và yêu cầu điền đầy đủ

+ Tạo đối tượng User mới rồi gán các giá trị username và password mà người dùng nhập vào trước đó cho đối tượng này

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("http://10.0.2.2:8080/") // Địa chỉ máy chủ
    .addConverterFactory(GsonConverterFactory.create())
    .build();

ApiService apiService = retrofit.create(ApiService.class);
```

+ Thiết lập Retrofit để gọi API

- baseUrl: Địa chỉ server (10.0.2.2 đại diện cho máy tính localhost khi chạy trên trình giả lập Android).
- GsonConverterFactory.create(): Sử dụng Gson để chuyển đổi dữ liệu JSON từ/đến server.

+ retrofit.create(ApiService.class): Tạo đối tượng ApiService, chứa định nghĩa các API

```
Call<User> call = apiService.loginUser(user);
call.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {
        if (response.isSuccessful() && response.body() != null) {
            Toast.makeText(context: MainActivity.this, text: "Login successfully!", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(context: MainActivity.this, UserActivity.class);
            intent.putExtra(name: "username", username.getText().toString());
            startActivity(intent);
            finish(); // Đóng LoginActivity
        } else {
            Toast.makeText(context: MainActivity.this, text: "Invalid username or password", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onFailure(Call<User> call, Throwable t) {
        Toast.makeText(context: MainActivity.this, text: "Error: " + t.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
});
```

+ Gửi yêu cầu API đăng nhập với thông tin được nhập

+ Xử lý phản hồi từ server (thành công hoặc thất bại)

- Ngoài ra còn 1 điều cần lưu ý đó là ta nên tạo tệp network-security-config.xml để định cấu hình bảo mật mạng trong ứng dụng Android, với định dạng như sau thì tệp sẽ cho phép kết nối không bảo mật (cleartext traffic) với địa chỉ 10.0.2.2 và cung cấp quyền truy cập cục bộ (localhost) trên máy tính khi chạy ứng dụng trong môi trường giả lập Android

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <domain-config cleartextTrafficPermitted="true">
    <domain includeSubdomains="true">10.0.2.2</domain>
  </domain-config>
</network-security-config>
```

Kết quả: Cơ sở dữ liệu mô phỏng đã lưu các thông tin tài khoản người dùng

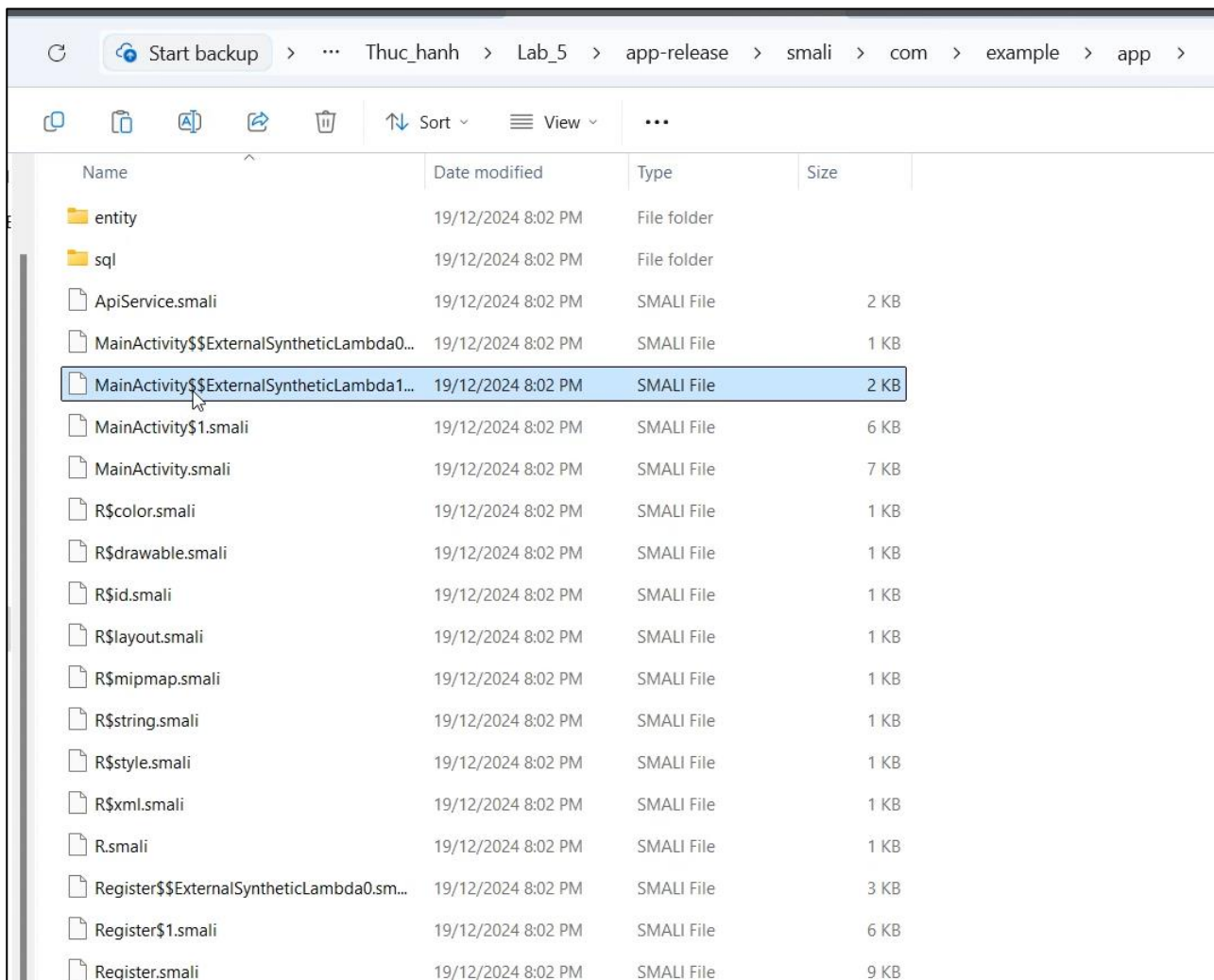
```
C:\Code\Bảo mật web và ứng dụng - NT213.P11.ANTT\LAB\W5\final\backend>node server.js
Server is running on http://localhost:8080
New user added: {
  name: '1',
  email: '1',
  password: '$2a$10$3mL1pHx1v2SWjb2NhI130u0C1Ka88opcZME/KPwY5p0ucG1aDZfr6'
}
New user added: {
  name: '3',
  email: '3',
  password: '$2a$10$hUFCVrQRKp8wxU2IfrahMeMjzh1Pvb0nq8iQdtyVqfNrzi7A6VPMa'
}
New user added: {
  name: '4',
  email: '4',
  password: '$2a$10$TnG2X2vNwzW5KeCHcFfm80FqJhAE/4vthixRGKDikrmVdnqXoY6e'
}
```

**Yêu cầu 6:** Với ứng dụng đã xây dựng, tìm hiểu và sử dụng công cụ ProGuard để tối ưu hóa mã nguồn. Trình bày khác biệt trước và sau khi sử dụng?

- Video demo chi tiết: <https://youtu.be/l-T9MG8Dsd4>
- Mở tệp build.gradle của module app và bật ProGuard cho bản phát hành (release build)
- Sử dụng cấu hình mặc định của ProGuard nên sẽ không có chỉnh sửa gì thêm trong tệp tin proguard-rules.pro

```
buildTypes {  
    release {  
        isMinifyEnabled = false  
        proguardFiles(  
            getDefaultProguardFile( name: "proguard-android-optimize.txt"),  
            "proguard-rules.pro"  
        )  
    }  
}
```

- Sử dụng Apktool kiểm tra thì ta thấy mã nguồn không được bảo vệ:

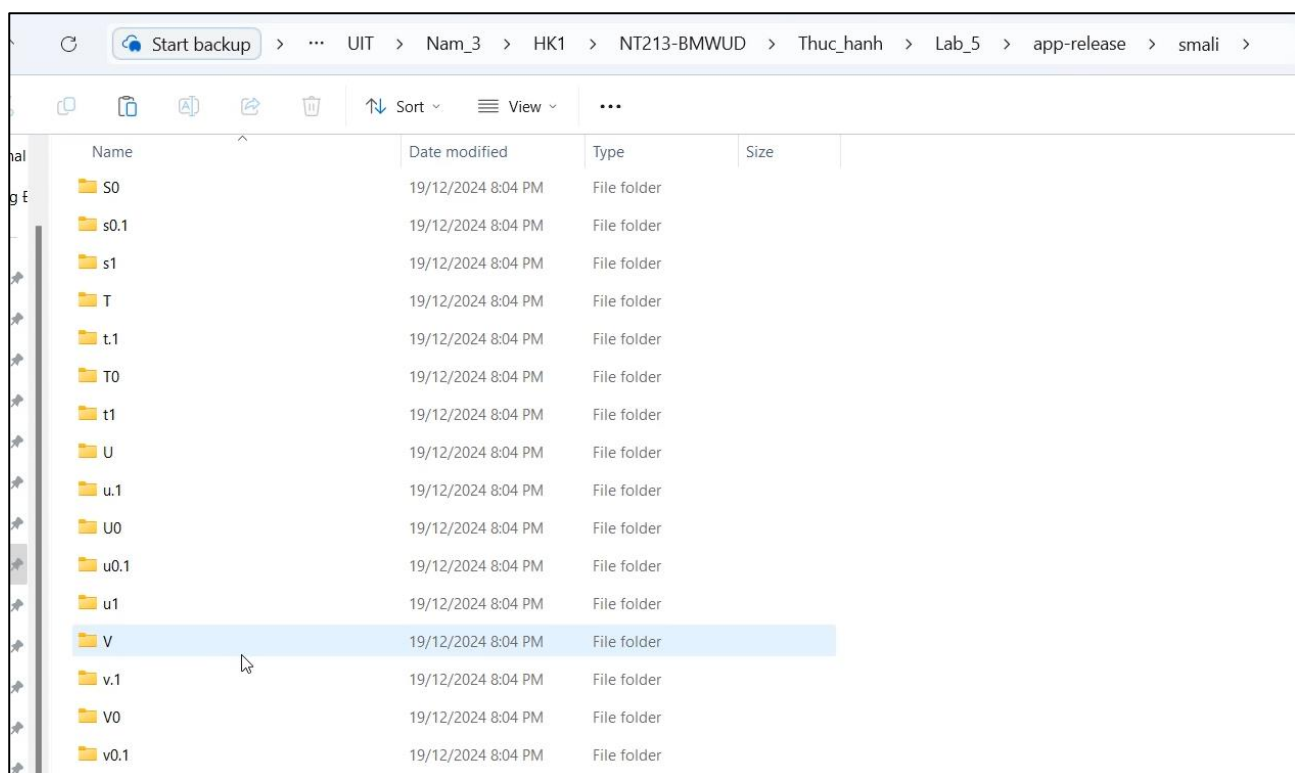


Start backup > ... Thuc_hanh > Lab_5 > app-release > smali > com > example > app >				
Name	Date modified	Type	Size	
entity	19/12/2024 8:02 PM	File folder		
sql	19/12/2024 8:02 PM	File folder		
ApiService.smali	19/12/2024 8:02 PM	SMALI File	2 KB	
MainActivity\$\$ExternalSyntheticLambda0...	19/12/2024 8:02 PM	SMALI File	1 KB	
MainActivity\$\$ExternalSyntheticLambda1...	19/12/2024 8:02 PM	SMALI File	2 KB	
MainActivity\$1.smali	19/12/2024 8:02 PM	SMALI File	6 KB	
MainActivity.smali	19/12/2024 8:02 PM	SMALI File	7 KB	
R\$color.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$drawable.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$id.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$layout.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$mipmap.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$string.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$style.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R\$xml.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
R.smali	19/12/2024 8:02 PM	SMALI File	1 KB	
Register\$\$ExternalSyntheticLambda0.sm...	19/12/2024 8:02 PM	SMALI File	3 KB	
Register\$1.smali	19/12/2024 8:02 PM	SMALI File	6 KB	
Register.smali	19/12/2024 8:02 PM	SMALI File	9 KB	

- Khi ta thiết lập giá trị minifyEnabled bằng true, mã nguồn của ứng dụng lúc này khi build bản release version sẽ được tối ưu, chống dịch ngược,...

```
buildTypes {  
    release {  
        isMinifyEnabled = true  
        proguardFiles(  
            getDefaultProguardFile( name: "proguard-android-optimize.txt"),  
            "proguard-rules.pro"  
        )  
    }  
}
```

- Kiểm tra lại lần nữa:



Name	Date modified	Type	Size
S0	19/12/2024 8:04 PM	File folder	
s0.1	19/12/2024 8:04 PM	File folder	
s1	19/12/2024 8:04 PM	File folder	
T	19/12/2024 8:04 PM	File folder	
t.1	19/12/2024 8:04 PM	File folder	
T0	19/12/2024 8:04 PM	File folder	
t1	19/12/2024 8:04 PM	File folder	
U	19/12/2024 8:04 PM	File folder	
u.1	19/12/2024 8:04 PM	File folder	
U0	19/12/2024 8:04 PM	File folder	
u0.1	19/12/2024 8:04 PM	File folder	
u1	19/12/2024 8:04 PM	File folder	
V	19/12/2024 8:04 PM	File folder	
v.1	19/12/2024 8:04 PM	File folder	
V0	19/12/2024 8:04 PM	File folder	
v0.1	19/12/2024 8:04 PM	File folder	