

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❦ ★ ❦ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH BẢO MẬT WEB VÀ ỨNG DỤNG

Lab 05: LẬP TRÌNH AN TOÀN ỨNG DỤNG ANDROID CƠ BẢN

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Phan Thanh Hải | 18520705 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 06/2021

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

----- ❦ ★ ❦ -----



BÁO CÁO KẾT QUẢ THỰC HÀNH BẢO MẬT WEB VÀ ỨNG DỤNG

Lab 05: LẬP TRÌNH AN TOÀN ỨNG DỤNG ANDROID CƠ BẢN

Giảng viên giảng dạy: Nghi Hoàng Khoa

Nhóm sinh viên thực hiện:

- | | |
|-----------------------|----------|
| 1. Phạm Khôi Nguyên | 18520114 |
| 2. Phan Thanh Hải | 18520705 |
| 3. Nguyễn Lý Đình Nhì | 18521205 |

TP. HỒ CHÍ MINH, 06/2021

MỤC LỤC

LỜI NÓI ĐẦU	1
YÊU CẦU 1.....	2
1. Thông tin chung của ứng dụng	2
2. Màn hình đăng nhập	3
3. Màn hình đăng kí	7
4. Màn hình hiển thị đăng nhập thành công	11
YÊU CẦU 2.....	13
1. Thêm thông tin gồm email, username và password vào cơ sở dữ liệu khi người dùng dùng chức năng Đăng ký	14
2. Truy vấn thông tin username và password cho chức năng Đăng nhập	17
YÊU CẦU 3.....	19
YÊU CẦU 4.....	22
YÊU CẦU 5.....	31

LỜI NÓI ĐẦU

Đây là phần bài làm của **nhóm 12** cho bài tập thực hành buổi 05 về lập trình an toàn ứng dụng Android cơ bản. Toàn bộ nội dung thực hành được triển khai trên môi trường Windows của máy thật.

Cảm ơn anh Nghi Hoàng Khoa trong thời gian tuần vừa qua chịu khó trả lời những câu hỏi, những thắc mắc của nhóm về bài tập thực hành buổi 05 trên lớp và ngoài buổi học ạ.

Đây là bài thực hành cuối cùng rồi (*không tính phần thi thực hành cuối cùng*). Những buổi thực hành vừa qua giờ ngẫm lại cảm xúc thấy thật khó tả. Có buồn vì không làm được, có vui vì cuối cùng cũng cài đặt được môi trường thực hành, có giận vì độ khó của bài *trời ơi đất hỡi*, sợ vì câu nói rớt môn của anh Khoa, v.v... Dẫu vậy, cũng rất biết ơn và quý trọng vì kiến thức và lòng nhiệt huyết mà anh Khoa qua môn học vừa rồi đã đem lại cho chúng em. Rất mong sau này sẽ _ được học anh Khoa một lần nữa.

YÊU CẦU 1

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).

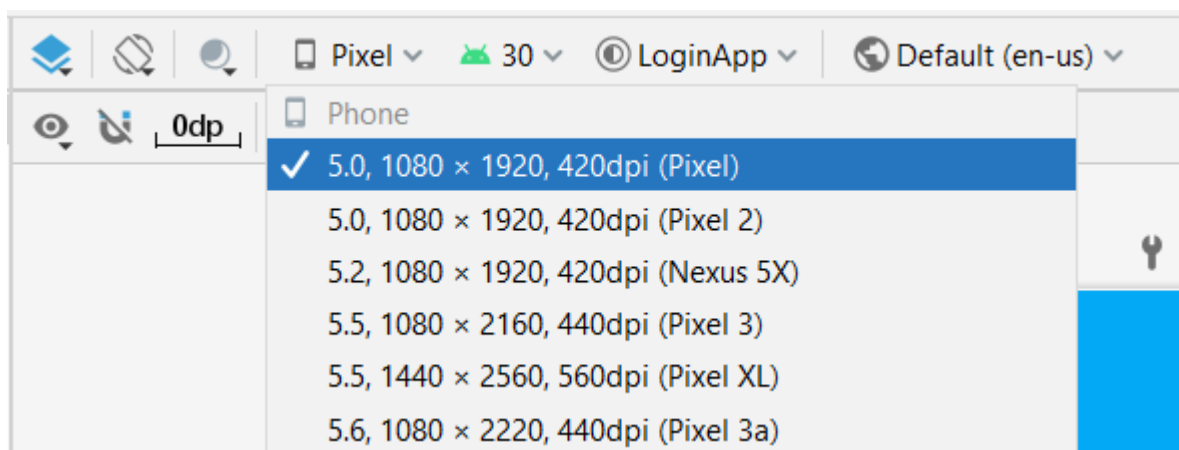
Yêu cầu của ứng dụng:

- Login Activity là main activity (activity khởi chạy ứng dụng), cho phép người dùng nhập thông tin (username và password) đăng nhập hoặc có thể nhấp chọn nút Đăng ký nếu chưa có tài khoản.
- Registry Activity cho phép nhập 3 thông tin như trên để lưu lại trên ứng dụng.
- Display Activity hiển thị lời chào cho người dùng khi đã đăng nhập thành công.

1. Thông tin chung của ứng dụng

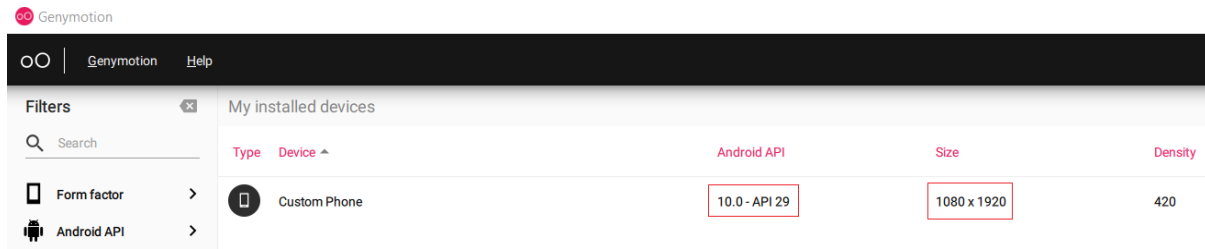
- Tên của ứng dụng là **Login Ứng dụng**.
- Ngôn ngữ sử dụng là **Java**.
- Phiên bản API của hệ thống là **API 29**.

Khi thiết kế màn hình của ứng dụng thì ta chọn kích thước của màn hình như sau:



Để có thể chạy tương thích với màn hình thiết kế trên **Android Studio** thì khi chạy điện thoại ảo trên **Genymotion** ta cũng chọn kích thước màn hình giống với trong **Android**

Studio. Ngoài ra, API của điện thoại ảo trong **Genymotion** phải có giá trị tối thiểu bằng với API của ứng dụng trong **Android Studio**.

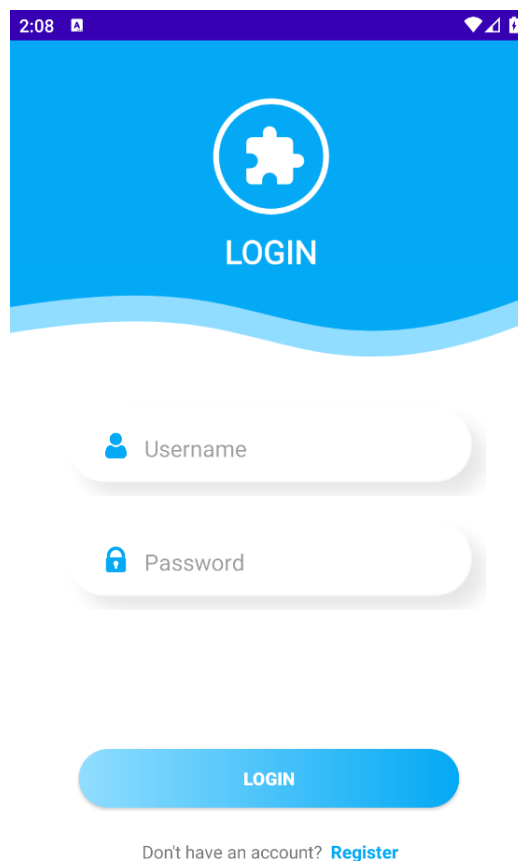


The screenshot shows the Genymotion application window. On the left is a sidebar with 'Filters' and a search bar. The main area is titled 'My installed devices' and contains a table with columns: Type, Device, Android API, Size, and Density. One device is listed: 'Custom Phone' with API level 10.0 - API 29, size 1080 x 1920, and density 420. The 'Android API' and 'Size' values are highlighted with red boxes.

Type	Device	Android API	Size	Density
Custom Phone	Custom Phone	10.0 - API 29	1080 x 1920	420

2. Màn hình đăng nhập

Giao diện màn hình đăng nhập (thiết kế trong tập tin `login.xml`):



Màn hình này sẽ được hiển thị ngay khi khởi chạy ứng dụng. Mặc định, khi ta tạo mới project với `MainActivity` thì `MainActivity` được thiết lập là activity khởi chạy ứng dụng (có thể xem trong tập tin `AndroidManifest.xml`).

```

<activity android:name=".DisplayActivity"></activity>
<activity android:name=".RegisterActivity" />
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

Các thành phần cần quan tâm trong màn hình đăng nhập:

- **EditText** nhập thông tin username.
- **EditText** nhập thông tin password.
- **Button Login.**
- **TextView Register.**

Khi nhấn vào nút **Login** thì hệ thống sẽ kiểm tra tính hợp lệ của username và password thông qua hàm `validateUsername()` và `validatePassword()`. Nếu ít nhất một trong 2 hàm trả về kết quả `false` (tức là nhập username hoặc password không hợp lệ) thì sẽ `return`, không làm gì thêm nữa.

```

// Set login button onclick event
Button button_login = (Button) findViewById(R.id.button_register);
button_login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        TextView text_view_username = (TextView) findViewById(R.id.text_view_username);
        TextView text_view_password = (TextView) findViewById(R.id.text_view_password);
        boolean check = validateUsername(text_view_username);
        check = validatePassword(text_view_password) && check;
        if (!check) {
            return;
        }
    }
}

```

Nếu username hoặc password nhập vào không hợp lệ thì không cho đăng nhập.

Một username được coi là hợp lệ khi:

- Phải có giá trị.
- Có độ dài hơn lớn hơn hoặc bằng 8 ký tự.
- Chỉ chứa các ký tự chữ cái, chữ số, dấu chấm, dấu gạch dưới và dấu cách.

Nếu username không hợp lệ thì hàm `validateUsername()` sẽ trả về kết quả `false` và ngay chỗ **TextView** của username sẽ thông báo lỗi, tùy vào loại lỗi trong code sau.

```

private static final Pattern USERNAME_PATTERN = Pattern.compile("[a-zA-Z0-9._]+$");

public boolean validateUsername(Textview text_view_username) {
    String username_input = text_view_username.getText().toString();
    if (username_input.isEmpty()) {
        text_view_username.setError("Please fill out this field");
        return false;
    }
    else if (username_input.length() < 8)
    {
        text_view_username.setError("The username is too short (< 8)");
        return false;
    }
    else if (!USERNAME_PATTERN.matcher(username_input).matches())
    {
        text_view_username.setError("The username contains special characters");
        return false;
    }
    return true;
}

```

Một password được coi là hợp lệ khi:

- Phải có giá trị.
- Có độ dài hơn lớn hơn hoặc bằng 8 ký tự.

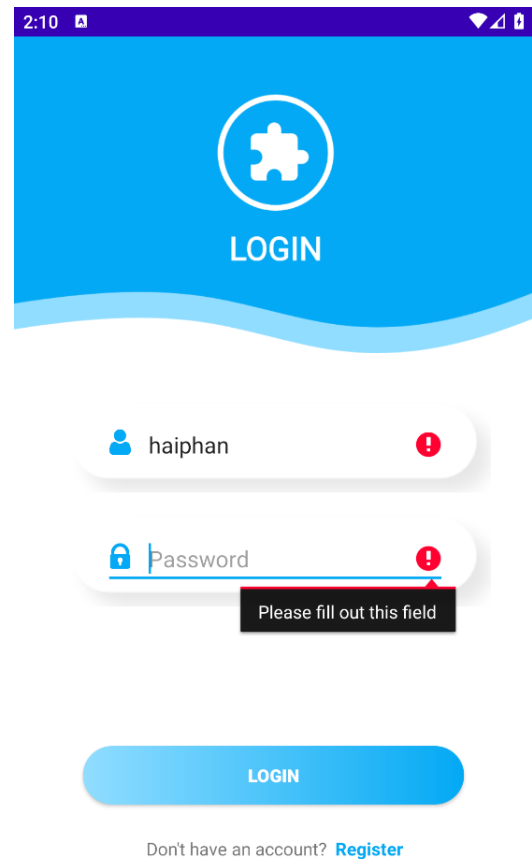
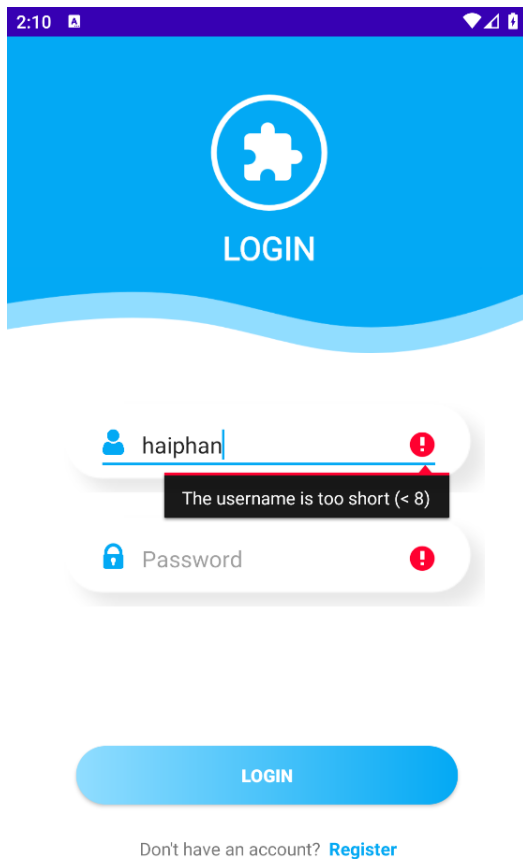
Nếu password không hợp lệ thì hàm `validatePassword()` sẽ trả về kết quả `false` và ngay chỗ **TextView** của password sẽ thông báo lỗi, tùy vào loại lỗi trong code sau.

```

public boolean validatePassword(Textview text_view_password) {
    String password_input = text_view_password.getText().toString();
    if (password_input.isEmpty()) {
        text_view_password.setError("Please fill out this field");
        return false;
    }
    else if (password_input.length() < 8)
    {
        text_view_password.setError("The password is too short (< 8)");
        return false;
    }
    return true;
}

```

Ví dụ khi nhập username và password không hợp lệ sau khi ta nhấn nút **Login**:



Trong ví dụ trên, vì ta nhập giá trị cho trường username quá ngắn nên chương trình thông báo lỗi `The username is too short (< 8)`. Ngoài ra, ta chưa nhập giá trị cho trường password nên chương trình thông báo lỗi `Please fill out this field`. Nếu người dùng chưa có tài khoản thì người dùng có thể đăng kí tài khoản trước bằng cách nhấn vào chữ **Register** để chuyển qua màn hình đăng kí.

```
// Set register text view onclick event to move to Register Activity
TextView text_view_register = (TextView) findViewById(R.id.text_view_login);
text_view_register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { moveToRegisterActivity(); }
});

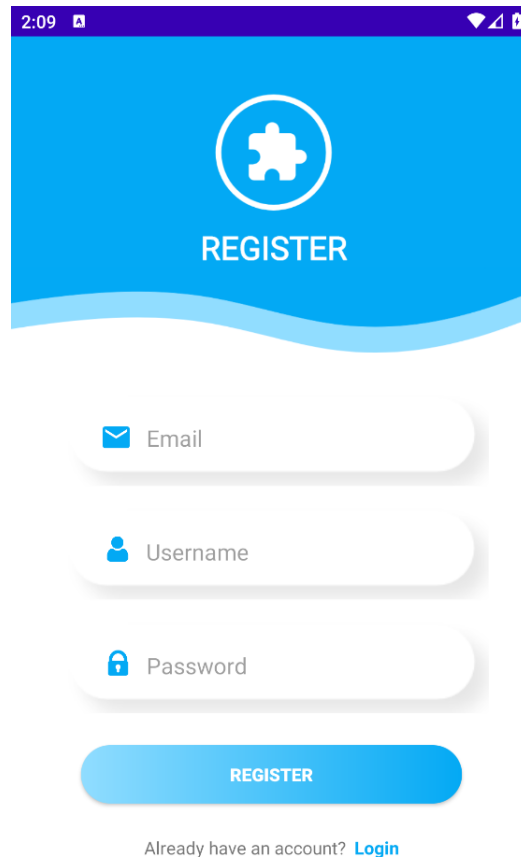
public void moveToRegisterActivity() {
    Intent intent = new Intent( packageContext: this, RegisterActivity.class);
    startActivity(intent);
}
```

Ở đây ta thiết lập sự kiện `onClick()` khi nhấn vào **TextView Register** ở trên màn hình. Khi đó, chương trình sẽ gọi hàm `moveToRegisterActivity()`. Hàm này tạo một explicit intent để gọi `RegisterActivity` từ `MainActivity` này và ta bắt đầu

thực thi activity này. Điều này có nghĩa là ta chuyển qua màn hình đăng kí sau khi bấm vào chữ **Register** trên màn hình.

3. Màn hình đăng kí

Giao diện màn hình đăng kí (thiết kế trong tập tin `register.xml`):



Màn hình này sẽ được hiển thị nếu người dùng nhấn vào chữ **Register** trong màn hình đăng nhập nếu người dùng cần tạo tài khoản để đăng nhập.

Các thành phần cần quan tâm trong màn hình đăng nhập:

- **EditText** nhập thông tin email.
- **EditText** nhập thông tin username.
- **EditText** nhập thông tin password.
- **Button Register**.
- **TextView Login**.

Khi nhấn vào nút **Register** thì hệ thống sẽ kiểm tra tính hợp lệ của email, username và password thông qua hàm `validateEmail()`, `validateUsername()` và `validatePassword()`. Nếu ít nhất một trong 3 hàm trả về kết quả `false` (tức là nhập

email hoặc username hoặc password không hợp lệ) thì sẽ return, không làm gì thêm nữa.

```
// Set register button onclick event
Button button_register = (Button) findViewById(R.id.button_register);
button_register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        TextView text_view_username = (TextView) findViewById(R.id.text_view_username);
        TextView text_view_password = (TextView) findViewById(R.id.text_view_password);
        TextView text_view_email = (TextView) findViewById(R.id.text_view_email);
        MainActivity mActivity = new MainActivity();
        boolean check = mActivity.validateUsername(text_view_username);
        check = mActivity.validatePassword(text_view_password) && check;
        check = validateEmail(text_view_email) && check;
        if (!check) {
            return;
        }
    }
}
```

Nếu email hoặc username hoặc password nhập vào không hợp lệ thì không cho đăng kí tài khoản.

Ở đây, thay vì định nghĩa lại hàm nhập kiểm tra tính hợp lệ của username và password thì ta tận dụng sẵn hàm validateUsername() và validatePassword() ở trong tập tin MainActivity.java bằng cách tạo một đối tượng thuộc về lớp MainActivity và thông qua đối tượng này ta gọi thực hiện phương thức validateUsername() và validatePassword().

Một email được coi là hợp lệ khi:

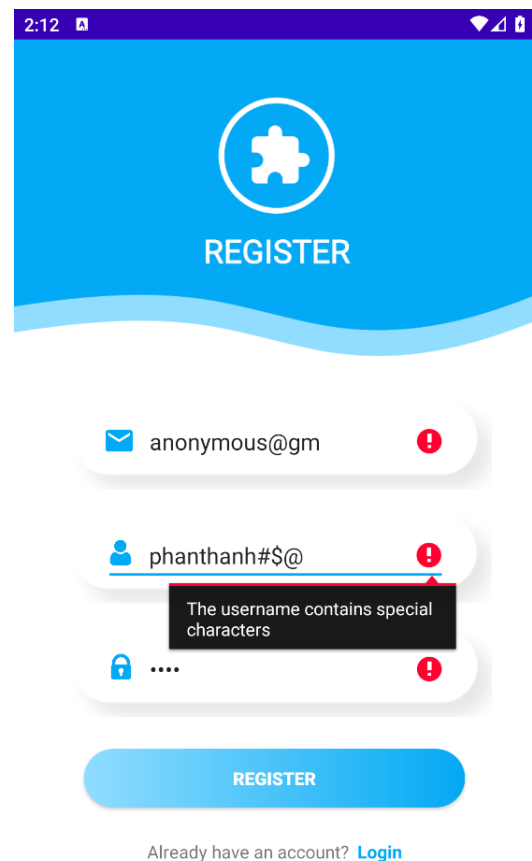
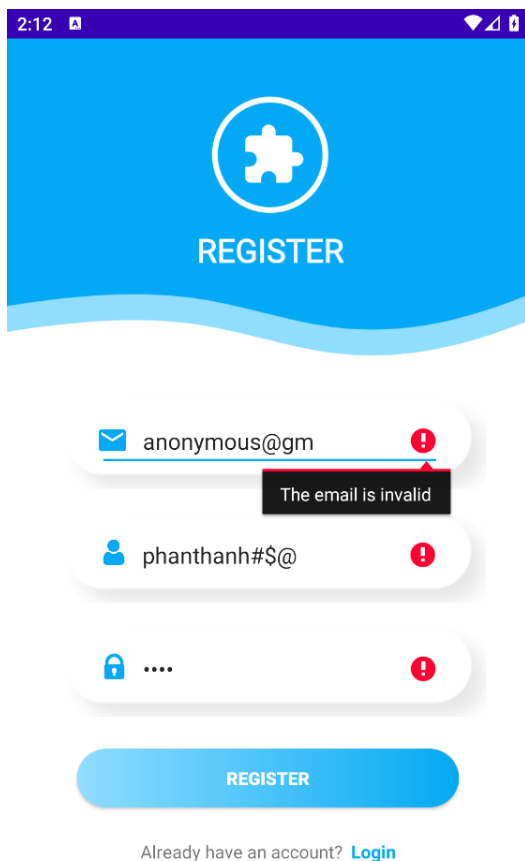
- Phải có giá trị.
- Có độ dài hơn lớn hơn hoặc bằng 8 kí tự.
- Khớp với biểu thức chính quy sau: `[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}` (được tham khảo ở <https://www.regular-expressions.info/email.html>).

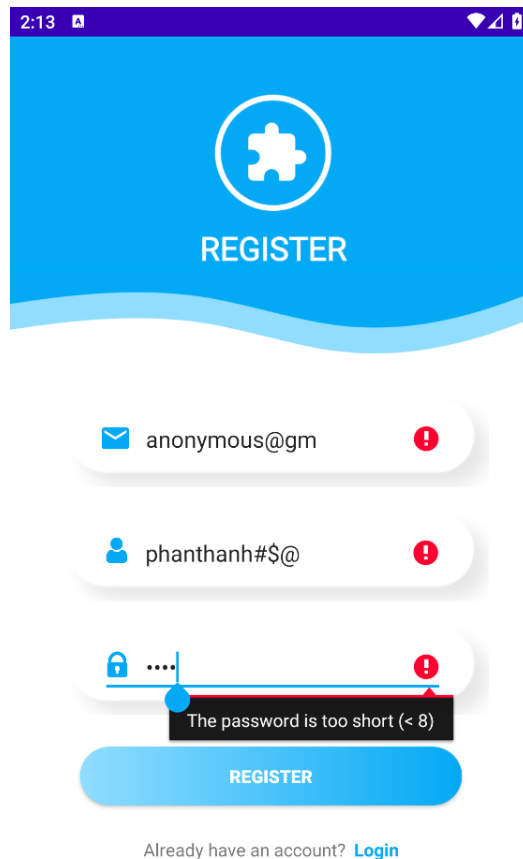
Nếu email không hợp lệ thì hàm validateEmail() sẽ trả về kết quả false và ngay chỗ **TextView** của email sẽ thông báo lỗi, tùy vào loại lỗi trong code sau.

```
private static final Pattern EMAIL_PATTERN = Pattern.compile("[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}");

public boolean validateEmail(Textview text_view_email) {
    String email_input = text_view_email.getText().toString();
    if (email_input.isEmpty()) {
        text_view_email.setError("Please fill out this field");
        return false;
    }
    else if (email_input.length() < 8) {
        text_view_email.setError("The email is too short (< 8)");
        return false;
    }
    else if (!EMAIL_PATTERN.matcher(email_input).matches()) {
        text_view_email.setError("The email is invalid");
        return false;
    }
    return true;
}
```

Ví dụ khi nhập email, username và password không hợp lệ sau khi ta nhấn nút **Register**:





Trong ví dụ trên, vì ta nhập giá trị cho trường không khớp với biểu thức chính quy trong đoạn code nên chương trình thông báo lỗi `The email is invalid`. Ngoài ra, ta nhập kí tự không được cho phép trong trường username nên chương trình thông báo lỗi `The username contains special characters`. Ta cũng nhập giá trị cho trường password quá ngắn nên chương trình thông báo lỗi `The password is too short (< 8)`.

Nếu người dùng đã có tài khoản thì người dùng có thể đăng nhập tài khoản bằng cách nhấn vào chữ **Login** để chuyển qua màn hình đăng nhập.

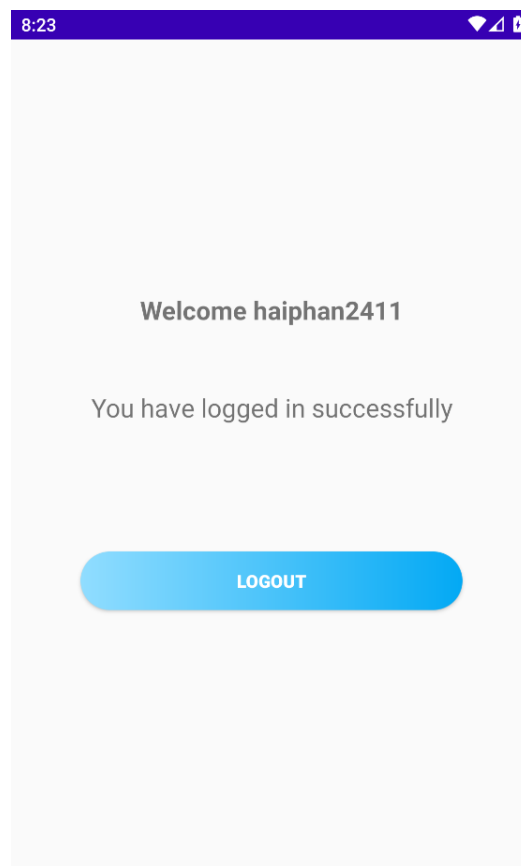
```
// Set login text view onclick event to move to Login Activity
TextView text_view_login = (TextView) findViewById(R.id.text_view_login);
text_view_login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { moveToLoginActivity(); }
});

public void moveToLoginActivity() {
    Intent intent = new Intent( packageContext: this, MainActivity.class);
    startActivity(intent);
}
```

Ở đây ta thiết lập sự kiện `onClickListener` khi nhấn vào **TextView Login** ở trên màn hình. Khi đó, chương trình sẽ gọi hàm `moveToLoginActivity()`. Hàm này tạo một explicit intent để gọi `MainActivity` từ `RegisterActivity` này và ta bắt đầu thực thi activity này. Điều này có nghĩa là ta chuyển qua màn hình đăng nhập sau khi bấm vào chữ **Login** trên màn hình.

4. Màn hình hiển thị đăng nhập thành công

Giao diện hiển thị khi đăng nhập thành công:



Màn hình này sẽ được hiển thị sau khi người dùng đăng nhập thành công.

Các thành phần cần quan tâm trong màn hình đăng nhập:

- **TextView** chào mừng người dùng.
- **Button Logout**.

TextView chào mừng người dùng sẽ có cấu trúc là `Welcome <tên người dùng>` với `<tên người dùng>` là username đã đăng nhập thành công ở màn hình đăng nhập. Để có thể lấy thông tin username của bên `MainActivity` thì ta sẽ dùng hàm `getStringExtra()` lấy dữ liệu được truyền từ `MainActivity`.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.display);

    // Get username data from Login Activity
    Intent intent = getIntent();
    String username = intent.getStringExtra( name: "username");

    // Set welcome string
    TextView text_view_welcome = (TextView) findViewById(R.id.text_view_welcome);
    text_view_welcome.setText("Welcome " + username);

```

Để nhận dữ liệu được truyền thì trước đó, trong MainActivity, ta dùng hàm `putExtra()` gửi dữ liệu đi.

```

String username_input = text_view_username.getText().toString();
Intent intent = new Intent( packageContext: this, DisplayActivity.class);
intent.putExtra( name: "username", username_input);
startActivity(intent);

```

Nếu người dùng muốn thoát khỏi phiên đăng nhập thì nhấn vào nút **Logout** để đăng xuất, quay về màn hình đăng nhập.

```

// Set logout button onclick event
Button button_logout = (Button) findViewById(R.id.button_logout);
button_logout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { moveToLoginActivity(); }
});

public void moveToLoginActivity() {
    Intent intent = new Intent( packageContext: this, MainActivity.class);
    startActivity(intent);
}

```

Ở đây ta thiết lập sự kiện `onClick()` khi nhấn vào nút **Logout** ở trên màn hình. Khi đó, chương trình sẽ gọi hàm `moveToLoginActivity()`. Hàm này tạo một explicit intent để gọi MainActivity từ DisplayActivity này và ta bắt đầu thực thi activity này. Điều này có nghĩa là ta chuyển qua màn hình đăng nhập sau khi nhấn vào nút **Logout** trên màn hình.

YÊU CẦU 2

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.

- a. Thêm thông tin gồm email, username và password vào cơ sở dữ liệu khi người dùng dùng chức năng Đăng ký.
- b. Truy vấn thông tin username và password cho chức năng Đăng nhập.

Ta thấy trong tập tin `SQLiteConnector` có sử dụng một đối tượng thuộc về lớp `User`. Tuy nhiên, ta không thấy phần định nghĩa của lớp `User` đó. Do đó, ta sẽ tạo một tập tin java định nghĩa lớp `User` đó.

Thuộc tính cơ bản của một lớp `User` bao gồm:

```
public class User {  
    private int id;  
    private String name;  
    private String email;  
    private String password;
```

Trong tập tin `SQLiteConnector` chỉ sử dụng những phương thức getter và setter của lớp `User` để tương tác với thế giới bên ngoài. Do đó, trong lớp `User` ta định nghĩa thêm các phương thức getter và setter tương ứng với mỗi thuộc tính của lớp `User`.

Thay vì nhập bàn phím thủ công thì ta có thể tạo tự động phương thức getter và setter. Trong **Android Studio** có hỗ trợ làm việc này bằng cách nhấn chuột phải trên màn hình code, chọn **Generate... → Getter and Setter**. Sau đó, ta chọn những thuộc tính cần định nghĩa phương thức getter và setter tương ứng rồi nhấn nút **OK**. Kết quả sẽ như sau:


```

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

public String getPassword() { return password; }

public void setPassword(String password) { this.password = password; }

```

1. Thêm thông tin gồm email, username và password vào cơ sở dữ liệu khi người dùng dùng chức năng Đăng ký

Trong RegisterActivity, ta thêm thuộc tính user thuộc về lớp User và thuộc tính conn thuộc về lớp SQLiteConnector phục vụ cho việc kết nối cơ sở dữ liệu SQLite. Sau đó, ta khởi tạo thông tin cho 2 thuộc tính này trong hàm initObjects() và gọi hàm này khi tiến hành khởi tạo activity của RequestActivity.

```

private User user;
private SQLiteConnector conn;

private void initObjects() {
    conn = new SQLiteConnector( context: this);
    user = new User();
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.register);

    initObjects();
}

```

Trong phương thức `onClick()` của nút **Login** thì ta bổ sung thêm phương thức `checkRegister()` nhận đầu vào là 3 **TextView** của email, username và password.

```
// Set register button onclick event
Button button_register = (Button) findViewById(R.id.button_register);
button_register.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        TextView text_view_username = (TextView) findViewById(R.id.text_view_username);
        TextView text_view_password = (TextView) findViewById(R.id.text_view_password);
        TextView text_view_email = (TextView) findViewById(R.id.text_view_email);
        MainActivity mActivity = new MainActivity();
        boolean check = mActivity.validateUsername(text_view_username);
        check = mActivity.validatePassword(text_view_password) && check;
        check = validateEmail(text_view_email) && check;
        if (!check) {
            return;
        }
        checkRegister(text_view_username, text_view_password, text_view_email);
    }
});
```

Trong phương thức `checkRegister()`, ta kiểm tra email đăng kí đã có trong cơ sở dữ liệu trước đó hay chưa thông qua phương thức `checkUser()` của lớp `SQLiteConnector` với 1 tham số đầu vào. Nếu đúng, thì thông báo `There already exists an account registered with this email address. Please try again` để cho người dùng đăng kí sử dụng mail khác. Nếu không thì tiến hành lưu toàn bộ thông tin vào trong cơ sở dữ liệu.

```
public void checkRegister(TextView text_view_username, TextView text_view_password, TextView text_view_email) {
    String username_input = text_view_username.getText().toString();
    String password_input = text_view_password.getText().toString();
    String email_input = text_view_email.getText().toString();
    if (conn.checkUser(email_input)) {
        Toast.makeText(context, this, text: "There already exists an account registered with this email address. " +
            "Please try again", Toast.LENGTH_LONG).show();
    }
    else {
        user.setName(username_input);
        user.setPassword(password_input);
        user.setEmail(email_input);
        conn.addUser(user);
    }
}
```

Ví dụ khi đăng kí thất bại:

1:12

REGISTER

✉ haiphan2411@gmail.com

👤 haiphan123

🔒

REGISTER

There already exists an account registered with this email address. Please try again

Already have an account? [Login](#)

Ví dụ khi đăng kí thành công:

9:24

REGISTER

✉ haiphan2411@gmail.com

👤 haiphan123

🔒

REGISTER

You have successfully registered

Already have an account? [Login](#)

```

PS D:\Virtual Machine\platform-tools> .\adb shell
vbox86p:/ # cd /data/data/com.example.loginapp/databases/
vbox86p:/data/data/com.example.loginapp/databases # ls
UserManager.db UserManager.db-journal
vbox86p:/data/data/com.example.loginapp/databases # sqlite3 UserManager.db
SQLite version 3.22.0 2018-12-19 01:30:22
Enter ".help" for usage hints.
sqlite> SELECT * FROM user;
1|haiphphan123|haiphphan2411@gmail.com|haiphphan2411
sqlite>

```

2. Truy vấn thông tin username và password cho chức năng Đăng nhập

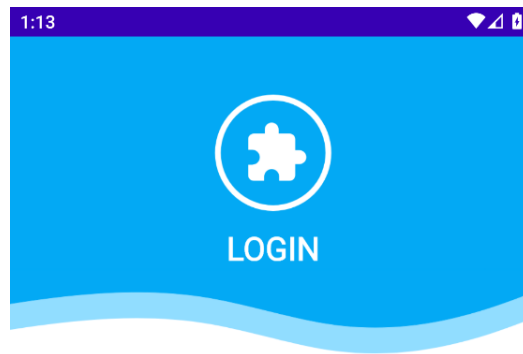
Qua bên MainActivity, ta cũng làm tương tự như RegisterActivity. Tuy nhiên, trong phương thức checkLogin() sẽ kiểm tra nếu cả thông tin username và password có trong cơ sở dữ liệu và password đúng ứng với username thông qua phương thức checkUser() của lớp SQLiteConnector với 2 tham số đầu vào. Nếu đúng thì ta chuyển qua màn hình hiển thị đăng nhập thành công. Nếu không, thì ta hiển thị thông báo Username or Password is incorrect. Please try again. khi người dùng nhập sai username hoặc password.

```

public void checkLogin(Textview text_view_username, Textview text_view_password) {
    String username_input = text_view_username.getText().toString();
    String password_input = text_view_password.getText().toString();
    if (conn.checkUser(username_input, password_input)) {
        Intent intent = new Intent( packageContext: this, DisplayActivity.class);
        intent.putExtra( name: "username", username_input);
        startActivity(intent);
    }
    else
    {
        Toast.makeText( context: this, text: "Username or Password is incorrect. Please try again", Toast.LENGTH_LONG).show();
    }
}

```

Ví dụ khi đăng nhập thất bại:



 haiphan2411



LOGIN
Username or Password is incorrect. Please
try again

Don't have an account? [Register](#)

YÊU CẦU 3

Điều chỉnh mã nguồn để password được lưu và kiểm tra dưới dạng mã hash thay vì plaintext.

Ở 2 yêu cầu, ta lưu thẳng chuỗi mật khẩu dưới dạng plaintext vào trong cơ sở dữ liệu. Như thế sẽ không đảm bảo tính bảo mật của dữ liệu. Do đó, ta cần mã hóa mật khẩu sang dạng khác.

Ở đây, nhóm sử dụng kỹ thuật mã hóa đối xứng thay vì sử dụng kỹ thuật hash (*bởi vì kỹ thuật mã hóa đối xứng nó an toàn hơn so với kỹ thuật hash*). Thuật toán mã hóa được nhóm sử dụng là thuật toán AES. Chú ý là với thuật toán AES thì kích thước của khóa phải là 128, 192 hoặc 256 bits.

Nhóm tạo một lớp Crypto dùng để thực hiện việc mã hóa và giải mã chuỗi bất kì. Khóa ở đây nhóm tạo mặc định là `example key text`.

```
package com.example.loginapp;

import android.util.Base64;
import javax.crypto.*;
import java.security.*;
import javax.crypto.spec.*;

public class Crypto {
    private static byte[] key;
    private static byte[] iv;
    private static String CIPHER_ALGORITHM;

    public Crypto() {
        this.key = "example key text".getBytes();
        this.iv = new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        this.CIPHER_ALGORITHM = "AES/CBC/PKCS5PADDING";
    }
}
```

Lớp Crypto có 2 phương thức chính là `encryptString` dùng để mã hóa chuỗi `plain_text` ban đầu và `decryptString` dùng để giải mã chuỗi đã được mã hóa.

```

public String encryptString(String plain_text) throws NoSuchAlgorithmException, NoSuchPaddingException {
    SecretKeySpec key = new SecretKeySpec(this.key, algorithm: "AES");
    IvParameterSpec iv_spec = new IvParameterSpec(this.iv);
    Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
    cipher.init(Cipher.ENCRYPT_MODE, key, iv_spec);
    byte[] ciphertext = cipher.doFinal(plain_text.getBytes());
    return Base64.encodeToString(ciphertext, flags: 0);
}

public String decryptString(String encrypt_string) throws NoSuchAlgorithmException, NoSuchPaddingException {
    byte[] decrypted_bytes = Base64.decode(encrypt_string, Base64.DEFAULT);
    SecretKeySpec key = new SecretKeySpec(this.key, algorithm: "AES");
    IvParameterSpec iv_spec = new IvParameterSpec(this.iv);
    Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
    cipher.init(Cipher.DECRYPT_MODE, key, iv_spec);
    byte[] ciphertext = cipher.doFinal(decrypted_bytes);

    return new String(ciphertext);
}

```

Ở trong phương thức checkRegister() của RegisterActivity, ta gọi thực hiện phương thức encryptString() của lớp Crypto để mã hóa chuỗi mật khẩu lưu xuống cơ sở dữ liệu như sau:

```

public void checkRegister(TextView text_view_username, TextView text_view_password, TextView text_view_email) {
    String username_input = text_view_username.getText().toString();
    String password_input = text_view_password.getText().toString();
    password_input = new Crypto().encryptString(password_input);
    String email_input = text_view_email.getText().toString();

    if (conn.checkUser(email_input)) {
        Toast.makeText(context: this, text: "There already exists an account registered with this email address. Please try again", Toast.LENGTH_LONG).show();
    }
    else {

```

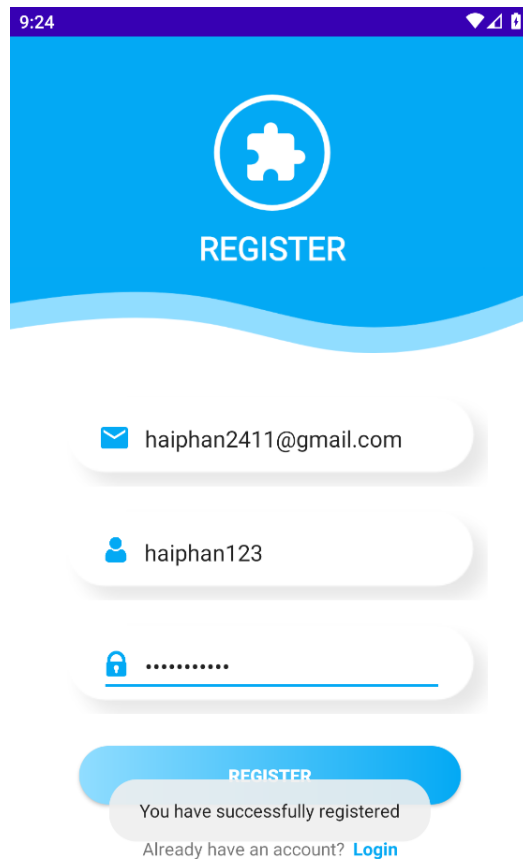
Tương tự, ở trong phương thức checkLogin() của MainActivity, ta gọi thực hiện phương thức encryptString() của lớp Crypto để mã hóa chuỗi mật khẩu khi kiểm tra với mật khẩu mã hóa trong cơ sở dữ liệu như sau:

```

public void checkLogin(TextView text_view_username, TextView text_view_password) {
    String username_input = text_view_username.getText().toString();
    String password_input = text_view_password.getText().toString();
    password_input = new Crypto().encryptString(password_input);
    if (conn.checkUser(username_input, password_input)) {
        Intent intent = new Intent(packageContext: this, DisplayActivity.class);
        intent.putExtra(name: "username", username_input);
        startActivity(intent);
    }
    else

```

Sau đó, ta thử đăng kí tài khoản trong ứng dụng với mật khẩu là haiphan2411.



Kết quả hiển thị dữ liệu trong cơ sở dữ liệu:

```
PS D:\Virtual Machine\platform-tools> .\adb shell
vbox86p:/ # cd /data/data/com.example.loginapp/databases/
vbox86p:/data/data/com.example.loginapp/databases # ls
 UserManager.db UserManager.db-journal
vbox86p:/data/data/com.example.loginapp/databases # sqlite3 UserManager.db
SQLite version 3.22.0 2018-12-19 01:30:22
Enter ".help" for usage hints.
sqlite> SELECT * FROM user;
1|haiphan123|haiphan2411@gmail.com|FLAU1+pd3SkCAUbIdrQ1Sg==
```

Như ta thấy, chuỗi mật khẩu haiphan2411 đã được mã hóa thành chuỗi FLAU1+pd3SkCAUbIdrQ1Sg==.

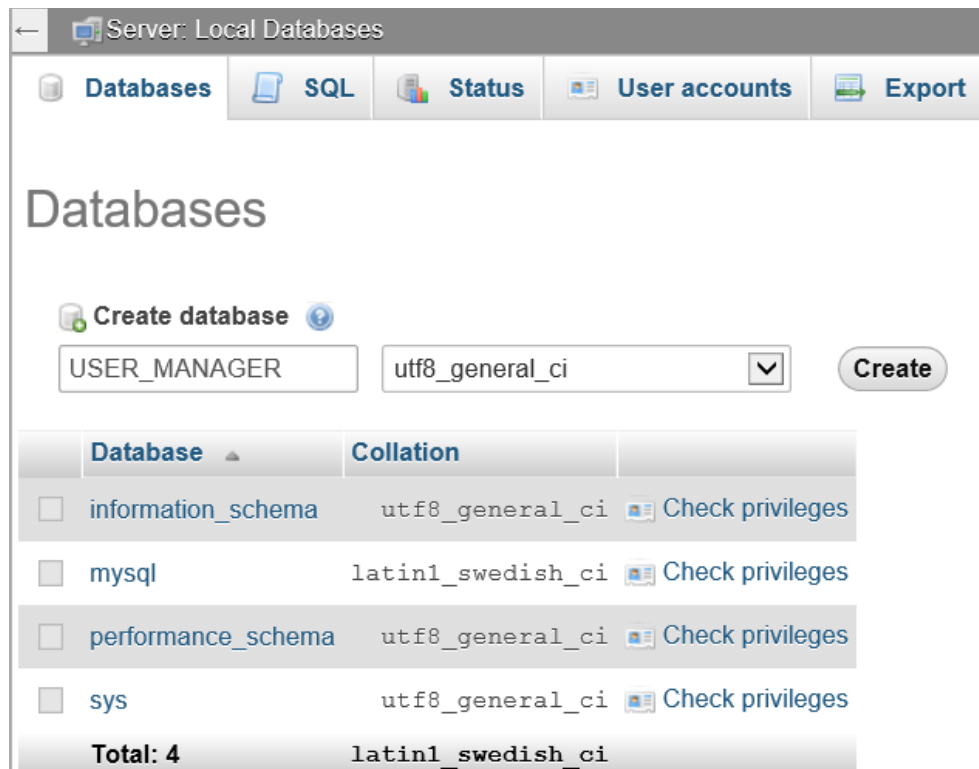
YÊU CẦU 4

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.

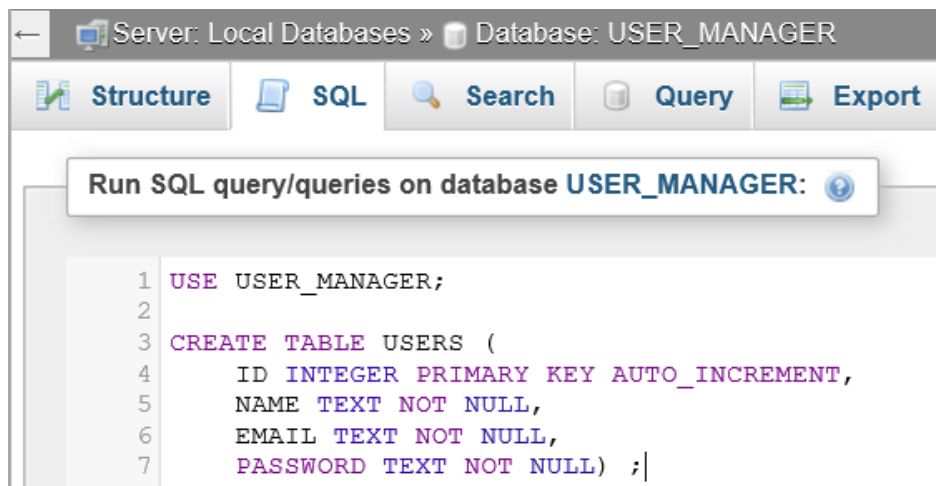
Gợi ý: Sinh viên có thể tận dụng CSDL MySQL và các tập tin xử lý PHP đã thực hiện ở bài thực hành trước và kết nối sử dụng Web REST API. Lưu ý, có cần điều chỉnh quyền hạn gì của ứng dụng hay không?

Đầu tiên, nhóm tạo cơ sở dữ liệu MySQL dùng để lưu trữ thông tin của người dùng. Nhóm sử dụng **phpMyAdmin** trên **WampServer** để tiến hành quản lý cơ sở dữ liệu.

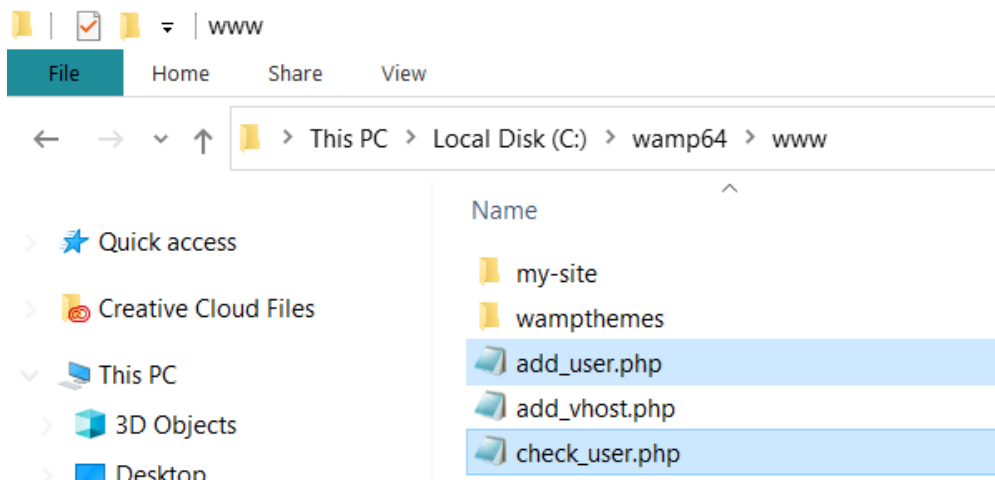
Bước 1: Tạo cơ sở dữ liệu USER_MANAGER như hình bên dưới và nhấn nút **Create**.



Bước 2: Tạo bảng USERS dùng để lưu trữ thông tin của người dùng. Ta tạo bảng USERS có cấu trúc như hình bên dưới:



Tiếp theo, ta tạo 2 tập tin `check_user.php` và `add_user.php` dùng để kiểm tra dữ liệu và thêm dữ liệu người dùng mới thông qua Web Rest API. Hai tập tin này được đặt trong thư mục **www** của **WampServer**.



Trong tập tin `check_user.php`, ta sẽ kiểm tra 2 trường hợp (tùy vào giá trị của tham số của phương thức POST). Một là kiểm tra tên đăng nhập và mật khẩu có nằm trong cơ sở dữ liệu hay không. Hai là kiểm tra mail đã được đăng kí trong cơ sở dữ liệu hay chưa. Nếu kết quả tìm thấy có, ta sẽ in ra dòng chữ **Success**. Ngược lại, nếu không có hoặc xảy ra lỗi thì ra dòng chữ như trong code ở hình bên dưới.

```

1  <?php
2      session_start();
3
4      $servername = "localhost";
5      $username = "root";
6      $password = "";
7      $database = "USER_MANAGER";
8      $table = "USERS";
9
10     $conn = mysqli_connect($servername, $username, $password, $database);
11     if ($conn->connect_error) {
12         echo $conn->connect_error;
13     }
14     else {
15         $name = mysqli_real_escape_string($conn, $_POST['name']);
16         $email = mysqli_real_escape_string($conn, $_POST['email']);
17         $password = mysqli_real_escape_string($conn, $_POST['password']);
18         $sql_command = "";
19         if ($email === "") {
20             $sql_command = "SELECT * FROM $table WHERE NAME = '$name' AND PASSWORD = '$password'";
21         }
22         else {
23             $sql_command = "SELECT * FROM $table WHERE EMAIL = '$email'";
24         }
25         $result = mysqli_query($conn, $sql_command);
26         if ($result) {
27             $row = mysqli_num_rows($result);
28             if ($row > 0) {
29                 echo "Success";
30             }
31             else {
32                 echo "Fail";
33             }
34         }
35         else {
36             echo mysqli_error($conn);
37         }
38         mysqli_close($conn);
39     }

```

Trong tập tin `add_user.php`, ta sẽ thực hiện việc thêm dữ liệu của người dùng mới vào trong cơ sở dữ liệu. Nếu thành công, ta sẽ in ra dòng chữ `Success`. Ngược lại, nếu xảy ra lỗi thì ra dòng chữ như trong code ở hình bên dưới.

```

1  <?php
2      session_start();
3
4      $servername = "localhost";
5      $username = "root";
6      $password = "";
7      $database = "USER_MANAGER";
8      $table = "USERS";
9
10     $conn = mysqli_connect($servername, $username, $password, $database);
11     if ($conn->connect_error) {
12         echo $conn->connect_error;
13     }
14     else {
15         $id = mysqli_real_escape_string($conn, $_POST['id']);
16         $name = mysqli_real_escape_string($conn, $_POST['name']);
17         $email = mysqli_real_escape_string($conn, $_POST['email']);
18         $password = mysqli_real_escape_string($conn, $_POST['password']);
19         $sql_command = "INSERT INTO $table VALUES($id, '$name', '$email', '$password')";
20         if (mysqli_query($conn, $sql_command)) {
21             echo "Success";
22         }
23         else {
24             echo mysqli_error($conn);
25         }
26         mysqli_close($conn);
27     }
28  ?>

```

Cuối cùng, ta sẽ viết code trong ứng dụng để thực hiện việc kết nối cơ sở dữ liệu và thực thi các tập tin PHP ở trên.

Ta cần điều chỉnh quyền hạn của ứng dụng cho phép ứng dụng kết nối vào mạng bằng cách thêm dòng `<uses-permission android:name="android.permission.INTERNET" />`. Ngoài ra, ta cũng thiết lập ứng dụng cho phép sử dụng cleartext HTTP bằng cách thêm dòng `android:usesCleartextTraffic="true"`. Những dòng code được thêm này sẽ được thêm vào trong tập tin `AndroidManifest.xml` của ứng dụng.

```

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.loginapp">
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Login App"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.LoginApp"
        android:usesCleartextTraffic="true" >
        <activity android:name=".DisplayActivity"></activity>
        <activity android:name=".RegisterActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

Nhóm tạo một lớp MySQLConnector dùng để kết nối với cơ sở dữ MySQL ở bên ngoài như hình bên dưới. Chú ý là url có dạng http://<địa chỉ ip của máy>:<cổng máy chủ Apache>.

```

public class MySQLConnector {
    private Context context;
    private static final String url = "http://192.168.1.8:8080";
    private String response_text;

    public MySQLConnector(Context context){
        this.context = context;
    }
}

```

Tương tự như các hàm sử dụng trong SQLiteConnector, ta cũng tạo phương thức checkUser() và addUser() trong lớp MySQLConnector với ý nghĩa tương tự.

Ở đây, nhóm sử dụng thư viện **Volley** dùng gửi và nhận phản hồi từ server sử dụng giao thức HTTP.

Để sử dụng được thư viện **Volley** thì ta cần phải import nó bằng cách thêm dòng lệnh sau vào phần dependencies trong tập tin build.gradle của ứng dụng.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.android.volley:volley:1.2.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Với phương thức `addUser()` thì ta sẽ gửi đi giá trị của 4 tham số `id`, `name`, `email` và `password` để thêm chúng vào trong cơ sở dữ liệu.

```
public void addUser(User user) {
    RequestQueue queue = Volley.newRequestQueue(this.context);
    StringRequest request = new StringRequest(Request.Method.POST, url: url + "/add_user.php", new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            response_text = response;
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "ERROR", error.toString());
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<String, String>();
            params.put("id", String.valueOf(user.getId()));
            params.put("name", user.getName());
            params.put("email", user.getEmail());
            params.put("password", user.getPassword());
            return params;
        }
    };
    queue.add(request);
}
```

Với phương thức `checkUser()` thì tùy vào tham số đầu vào, ta sẽ gửi giá trị rỗng cho các tham số còn lại hoặc không. Ngoài ra, ta cũng kiểm tra chuỗi phản hồi sau khi gửi phản hồi. Nếu chuỗi phản hồi là `Success` thì ta thông báo trong cơ sở dữ liệu đã có thông tin đó và trả về kết quả là `true`. Ngược lại thì trả về kết quả là `false`.

- `checkUser()` với thông tin kiểm tra là email:

```

public boolean checkUser(String email) {
    RequestQueue queue = Volley.newRequestQueue(this.context);
    StringRequest request = new StringRequest(Request.Method.POST, url + "/check_user.php", new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            response_text = response;
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "ERROR", error.toString());
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<>();
            params.put("name", "");
            params.put("email", email);
            params.put("password", "");
            return params;
        }
    };
    queue.add(request);
    if (response_text.equals("Success")) {
        Log.i( tag: "RESPONSE", msg: " return True");
        return true;
    }
    return false;
}

```

- checkUser() với thông tin kiểm tra là tên đăng nhập và mật khẩu:

```

public boolean checkUser(String name, String password)
{
    RequestQueue queue = Volley.newRequestQueue(this.context);
    StringRequest request = new StringRequest(Request.Method.POST, url + "/check_user.php", new Response.Listener<String>() {
        @Override
        public void onResponse(String response) {
            response_text = response;
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e( tag: "ERROR", error.toString());
        }
    }) {
        @Override
        protected Map<String, String> getParams() {
            Map<String, String> params = new HashMap<>();
            params.put("name", name);
            params.put("email", "");
            params.put("password", password);
            return params;
        }
    };
    queue.add(request);
    if (response_text.equals("Success")) {
        return true;
    }
    return false;
}

```

Trong tập tin RegisterActivity, ta khai báo thuộc tính conn có kiểu dữ liệu là MySQLConnector. Cũng giống như thuộc tính conn của lớp SQLiteConnector, ta cũng khởi tạo giá trị ban đầu cho nó. Phần code còn lại là gọi phương thức checkUser() và addUser() của thuộc tính conn thì ta không cần thay đổi gì, cứ để nguyên như lúc code trong **Yêu cầu 2**.

```

public class RegisterActivity extends AppCompatActivity {
    private static final Pattern EMAIL_PATTERN = Pattern.compile("[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,}");

    private User user;
    // private SQLiteConnector conn;
    private MySQLConnector conn;

    private void initObjects() {
        // conn = new SQLiteConnector(this);
        conn = new MySQLConnector(context: this);
        user = new User();
    }
}

```

Một cách tương tự, trong tập tin MainActivity, ta cũng khai báo thuộc tính conn có kiểu dữ liệu là MySQLConnector. Cũng giống như thuộc tính conn của lớp SQLiteConnector, ta cũng khởi tạo giá trị ban đầu cho nó. Phần code còn lại là gọi phương thức checkUser() và addUser() của thuộc tính conn thì ta không cần thay đổi gì, cứ để nguyên như lúc code trong **Yêu cầu 2**.

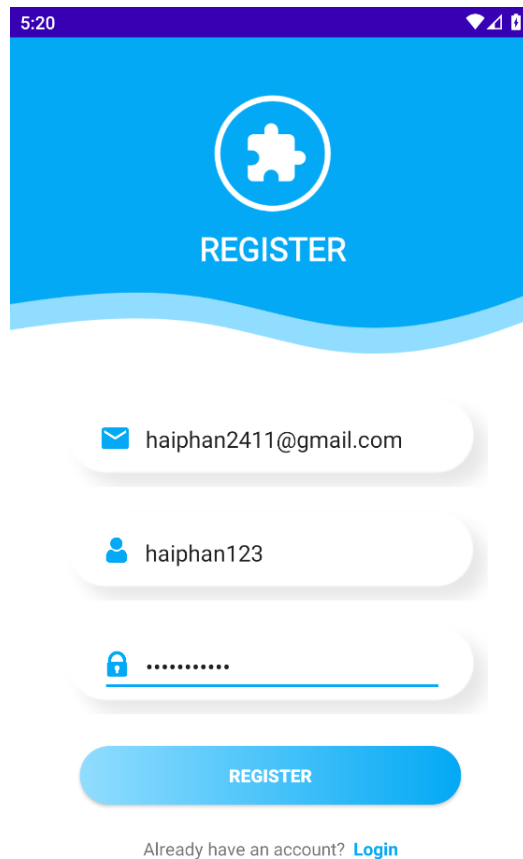
```

public class MainActivity extends AppCompatActivity {
    private static final Pattern USERNAME_PATTERN = Pattern.compile("^ [a-zA-Z0-9._]+$");
    private User user;
    // private SQLiteConnector conn;
    private MySQLConnector conn;

    private void initObjects() {
        // conn = new SQLiteConnector(this);
        conn = new MySQLConnector(context: this);
        user = new User();
    }
}

```

Giờ ta chạy ứng dụng để tiến hành đăng kí tài khoản thử:



5:20

REGISTER

✉ haiphan2411@gmail.com

👤 haiphan123

🔒

REGISTER

Already have an account? [Login](#)

Kết quả hiển thị trong cơ sở dữ liệu MySQL trong **phpMyAdmin**:

(Ở đây là id = 2 là do nhóm có chèn dữ liệu trước đó và xóa đi rồi nên khi thêm mới vô thì id nó lên 2)

Server: Local Databases » Database: user_manager » Table: users

Browse Structure SQL Search Insert Export Import Privileges

✓ Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)

```
SELECT * FROM `users`
```

☐ Show all | Number of rows: 25 | Filter rows: Search this table

+ Options

	ID	NAME	EMAIL	PASSWORD
<input type="checkbox"/> Edit Copy Delete	2	haiphan123	haiphan2411@gmail.com	FLAUI+pd3SkCAUblDrQISg==

↑ ☐ Check all | With selected: Edit Copy Delete Export

☐ Show all | Number of rows: 25 | Filter rows: Search this table

YÊU CẦU 5

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?

Ta mở tập tin `build.gradle` của ứng dụng và thêm vào những dòng lệnh sau (nếu chưa có):

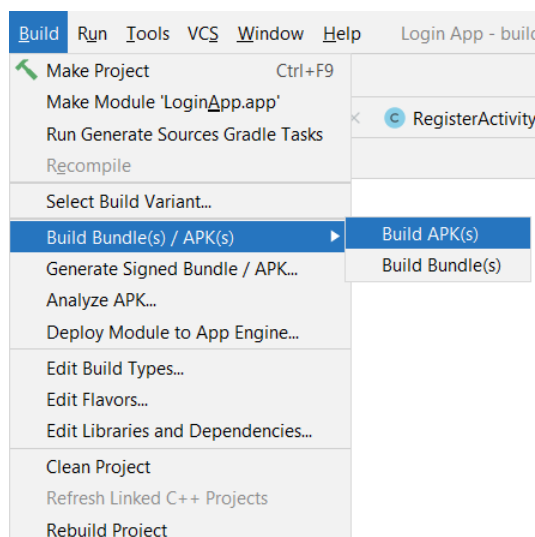
```
buildTypes {  
    release {  
        minifyEnabled true  
        shrinkResources true  
        useProguard true  
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
    }  
}
```

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,...

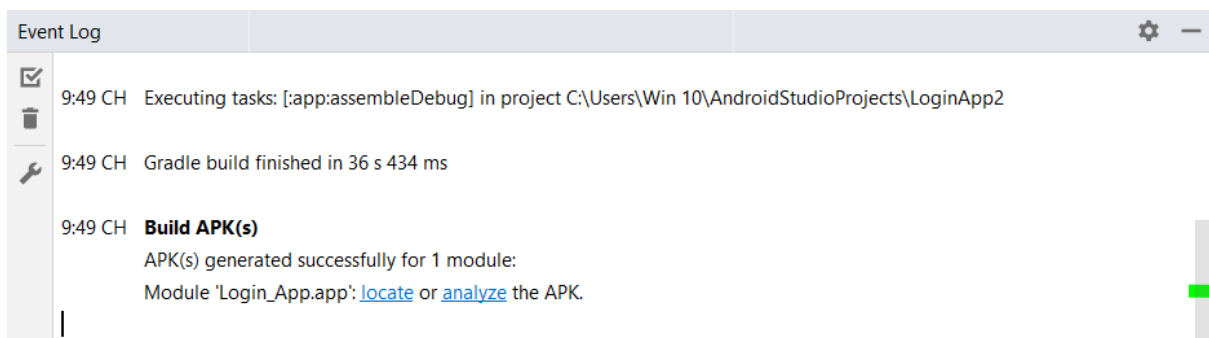
Khi ta thiết lập giá trị `shrinkResources` bằng `true` thì sẽ loại bỏ code dư thừa, thu gọn kích thước của code.

Ngoài ra, ta có thể cấu hình ProGuard thông qua tập tin `proguard-rules.pro`. Ở đây, nhóm sử dụng cấu hình mặc định của ProGuard nên sẽ không có chỉnh sửa gì thêm trong tập tin `proguard-rules.pro`.

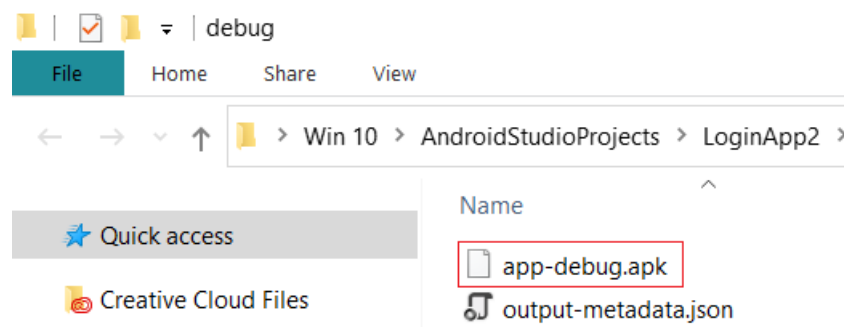
Giờ ta sẽ build ứng dụng của ta thành tập tin APK bằng cách vào **Build → Build Bundle(s) / APK(s) → Build APK(s)**.



Khi build xong thì sẽ hiển thị Event Log như thế này:



Ta nhấn vào **locate** để xem vị trí của tập tin APPK vừa build:



Sau đó, ta dùng các công cụ decompile (nhóm sử dụng **apktool**) để giải nén tập tin APK nhằm so sánh khác biệt trước và sau sử dụng công cụ ProGuard.

```
D:\Virtual Machine\platform-tools>apktool d app-debug.apk
I: Using Apktool 2.5.0 on app-debug.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\Win 10\AppData\Local\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Baksmaling classes3.dex...
I: Baksmaling classes2.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Ta mở thử một tập tin Java bất kỳ trong ứng dụng để so sánh. Ở đây, nhóm sẽ mở thử tập tin `Crypto` theo đường dẫn `app-debug\smali_classes3\com\example\loginapp`:

```

1  .class public Lcom/example/loginapp/Crypto;
2  .super Ljava/lang/Object;
3  .source "Crypto.java"
4
5
6  # static fields
7  .field private static CIPHER_ALGORITHM:Ljava/lang/String;
8
9  .field private static iv:[B
10
11  .field private static key:[B
12
13
14  # direct methods
15  √ .method public constructor <init>()V
16  |   .locals 1

```

Như ta thấy thì đoạn code bây giờ được viết lại một cách khó hiểu, *có chớ mới hiểu*. Nhưng như thế mới tránh việc dịch ngược và đảm bảo an toàn hơn.

----- HẾT -----