Contents

[MY PROJECT 2](#_Toc171695217)

[1.Demo firebase 2](#_Toc171695218)

[- link : https://github.com/giahyng1502/demofirebase 2](#_Toc171695219)

[2.dự án mẫu 2](#_Toc171695220)

[- link : https://github.com/giahyng1502/duanmauAND 2](#_Toc171695221)

[I.Cách tạo SLIDER 2](#_Toc171695222)

[***1. Hiệu ứng chuyển slider*** 4](#_Toc171695223)

[II. CÁCH BO GÓC CỦA 1 IMAGE 6](#_Toc171695224)

[III. Cách đổi màu của 1 Button 6](#_Toc171695225)

[IV. FIRE BASE 7](#_Toc171695226)

[1.Các phương thức cơ bản của REALTIME DATABASE 7](#_Toc171695227)

[a. Tạo dữ liệu 7](#_Toc171695228)

[b. Đọc dữ liệu 8](#_Toc171695229)

[c. Sửa , xóa dữ liệu 8](#_Toc171695230)

[d. Sắp xếp dữ liệu 9](#_Toc171695231)

[2. Firebase Storage 10](#_Toc171695232)

[a. tải ảnh lên firebase 10](#_Toc171695233)

[V. Những thư viện thường sử dụng 12](#_Toc171695234)

[1.AndroidManifest.xml 12](#_Toc171695235)

[a.Quyền đọc và ghi dữ liệu từ bộ nhớ ngoài của thiết bị 12](#_Toc171695236)

[b.Quyền truy cập internet 13](#_Toc171695237)

[c.Quyền truy cập mạng wifi 13](#_Toc171695238)

[d. Quyền sử dụng máy ảnh 13](#_Toc171695239)

[e. Quyền truy cập vị trí (GPS): 13](#_Toc171695240)

[g. Quyền đọc danh bạ: 13](#_Toc171695241)

[h.Quyền gửi tin nhắn SMS: 14](#_Toc171695242)

[i. Quyền gọi điện thoại: 14](#_Toc171695243)

[j.Quyền quản lý cuộc gọi: 14](#_Toc171695244)

[k.Quyền truy cập vào lịch trình (Calendar): 14](#_Toc171695245)

[2.build.gradle.kts 14](#_Toc171695246)

[a.Thư viện load ảnh 14](#_Toc171695247)

[VI.CÁCH NHỚ TÀI KHOẢN MẬT KHẨU 16](#_Toc171695248)

[VII. Tạo NavigationView 17](#_Toc171695249)

[1.Cách Tạo 17](#_Toc171695250)

[2. Cách ẩn 1 item khi trong navigation view 20](#_Toc171695251)

[3.Cách hiển thị tên người dùng lên thanh tiêu đề của NavigationView 21](#_Toc171695252)

[VII. NÂNG CAO VỀ RECYCLERVIEW 22](#_Toc171695253)

[1. Cách scroll (kéo lên sẽ ẩn button add) 22](#_Toc171695254)

# MY PROJECT

## 1.Demo firebase

- thêm sửa xóa , lấy dự liệu lên view

### - link : <https://github.com/giahyng1502/demofirebase>

## 2.dự án mẫu

- app quản lý thư viện với SQLITE

### - link : <https://github.com/giahyng1502/duanmauAND>

### 3. Document Về GIT VÀ GITHUB

- Hướng dẫn sử dụng git và github

- link : <https://fullstack.edu.vn/blog/bo-tui-21-lenh-git-co-ban-cach-nho-giup-newdev-lam-chu-git-quan-ly-tot-ma-ngu>

# I.Cách tạo SLIDER

public class MainActivity extends AppCompatActivity {  
 ViewPager2 viewPager2;  
 int index = 0;  
 List<Integer> arrayImage;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 viewPager2 = findViewById(R.id.*viewPage2*);  
 arrayImage = new ArrayList<>();  
 arrayImage.add(R.drawable.*banner1*);  
 arrayImage.add(R.drawable.*banner2*);  
 arrayImage.add(R.drawable.*banner3*);adapterViewPage adapterViewPage = new adapterViewPage(this,arrayImage);  
 viewPager2.setAdapter(adapterViewPage);  
 viewPager2.setClipToPadding(false); *//Mặc định là true. Khi đặt là false,  
 // ViewPager2 sẽ không cắt bỏ nội dung của trang nếu trang đó nằm ngoài padding của ViewPager2.  
 // Điều này cho phép bạn thấy một phần của các trang bên cạnh trang hiện tại.* viewPager2.setClipChildren(false);  
 *//Mặc định là true. Khi đặt là false,  
 // ViewPager2 sẽ không cắt bỏ nội dung của trang nếu trang đó nằm ngoài biên giới của ViewPager2.  
 // Điều này, kết hợp với setClipToPadding(false),  
 // giúp bạn nhìn thấy một phần của các trang kế tiếp hoặc trước đó.* viewPager2.setOffscreenPageLimit(3);  
 *//Thiết lập số lượng trang sẽ được tải trước và giữ trong bộ nhớ.  
 // Mặc định giá trị này là 1, nghĩa là chỉ có một trang trước và một trang sau trang hiện tại sẽ được tải trước.  
 // Khi đặt là 3, ViewPager2 sẽ giữ lại ba trang trước và ba trang sau trang hiện tại trong bộ nhớ, giúp cải thiện hiệu suất khi người dùng lướt qua các trang.* viewPager2.getChildAt(0).setOverScrollMode(RecyclerView.*OVER\_SCROLL\_NEVER*);  
 *// Điều này tắt chế độ over-scroll của RecyclerView bên trong ViewPager2.  
 // Chế độ over-scroll làm cho trang bị co giãn hoặc tạo hiệu ứng "bật lại" khi kéo vượt quá biên giới của nội dung.  
 // Khi đặt là OVER\_SCROLL\_NEVER, ViewPager2 sẽ không có hiệu ứng này, mang lại trải nghiệm mượt mà hơn.*

## ***1. Hiệu ứng chuyển slider***

CompositePageTransformer compositePageTransformer = new

CompositePageTransformer();  
 compositePageTransformer.addTransformer(new MarginPageTransformer(30));

*// Tạo khoảng cách giữa các lần trượt*   
 compositePageTransformer.addTransformer(new ViewPager2.PageTransformer() {  
 @Override  
 public void transformPage(@NonNull View page, float position) {  
*// float r = 1 - Math.abs(position);  
// page.setScaleY(0.85f + r\*0.15f); // thu nhỏ theo trục y tạo hiệu ứng trượt  
//   
 // Ví dụ về việc làm mờ và thu nhỏ trang khi trượt  
// float absPosition = Math.abs(position);  
// page.setAlpha(1.0f - absPosition);  
// page.setScaleY(1.0f - 0.25f \* absPosition);  
  
// page.setTranslationX(-position \* page.getWidth()); // Trượt trang theo trục X  
// page.setAlpha(1 - Math.abs(position)); // Làm mờ trang dựa trên vị trí  
  
// page.setRotationY(position \* -15); // Xoay trang theo trục Y  
  
// float scaleFactor = Math.max(0.85f, 1 - Math.abs(position));  
// page.setScaleX(scaleFactor);  
// page.setScaleY(scaleFactor);  
// page.setAlpha(0.5f + (scaleFactor - 0.85f) / 0.15f \* 0.5f); // Làm mờ trang khi phóng to thu nhỏ  
  
// page.setCameraDistance(20000); // Đặt khoảng cách camera để tạo hiệu ứng 3D  
// page.setRotationY(position \* -40); // Xoay trang theo trục Y để tạo hiệu ứng 3D  
// page.setAlpha(1 - Math.abs(position)); // Làm mờ trang dựa trên vị trí* }  
 });  
 viewPager2.setPageTransformer(compositePageTransformer);  
  
 Handler handler = new Handler();  
 Runnable runnable = new Runnable() {  
 @Override  
 public void run() {  
 next();  
 handler.postDelayed(this,2000);  
 }  
 };  
 handler.post(runnable);  
 }  
 public void next () {  
 if (index < 2) {  
 index ++;  
 viewPager2.setCurrentItem(index);  
*// Log.d("giahyng",index+"");* } else {  
 index = -1;  
 }  
 }  
}

# II. CÁCH BO GÓC CỦA 1 IMAGE

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <com.google.android.material.imageview.ShapeableImageView

*// Thư viện này cho phép bo góc hình ảnh* android:id="@+id/image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:padding="9dp"  
 android:src="@drawable/banner1"  
 android:scaleType="centerCrop"  
app:shapeAppearanceOverlay="@style/CustomShapeableImageView"/> // *: Áp dụng style tùy chỉnh để thay đổi hình dạng của ShapeableImageView ở @colors*

*(<resources>*

*<style name="CustomShapeableImageView" parent="">*

*<item name="cornerFamily">rounded</item>*

*<item name="cornerSize">16dp</item>*

*</style>*

*</resources>)*  
</LinearLayout>

# III. Cách đổi màu của 1 Button

<Button  
 android:layout\_width="wrap\_content"  
 android:background="@drawable/backgroud"  
 app:backgroundTint="#000000"

*// sử dụng app: thay vì andoroid*  
 android:text="START"  
 android:layout\_height="wrap\_content"/>

# IV. FIRE BASE

1. Realtime database

// khai báo toàn cục

public static DatabaseReference *myData*;

{

"Users": {

"user1": {

"name": "John",

"age": 25

},

"user2": {

"name": "Jane",

"age": 30

}

}

}

## 1.Các phương thức cơ bản của REALTIME DATABASE

### a. Tạo dữ liệu

//

- setvalue()

Lưu trữ dữ liệu tại một nút cụ thể. Nếu dữ liệu đã tồn **tại, nó** sẽ bị ghi đè.

* Child()

cho phép bạn truy cập các nút con của một nút cha cụ thể.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users"); ref.child("user1").setValue(new User("John", 25)); |

- Push()

Tạo một ID duy nhất và lưu trữ dữ liệu tại đó.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users”); ref.push().setValue(new User("John", 25)); |

### b. Đọc dữ liệu

- addListenerForSingleValueEvent()

Đọc dữ liệu từ Firebase một lần.

* addValueEventListener()

Lắng nghe các thay đổi dữ liệu liên tục.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users"); ref.addListenerForSingleValueEvent(new ValueEventListener() {  @Override  public void onDataChange(DataSnapshot dataSnapshot) {  for (DataSnapshot snapshot : dataSnapshot.getChildren()) {  User user = snapshot.getValue(User.class);  *// Xử lý dữ liệu ở đây* }  }   @Override  public void onCancelled(DatabaseError databaseError) {  *// Xử lý lỗi ở đây* } }); |

### c. Sửa , xóa dữ liệu

// sử dụng setvalue để sửa dữ liệu

- Xóa dữ liệu

removeValue()

Xóa dữ liệu tại nút cụ thể.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users”); ref.child(“user1”).removeValue()); |

### d. Sắp xếp dữ liệu

- orderByChild(), orderByKey(), orderByValue()

Sắp xếp dữ liệu theo một giá trị, khóa hoặc giá trị cụ thể.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users"); ref.orderByChild("name").addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(DataSnapshot dataSnapshot) {  for (DataSnapshot snapshot : dataSnapshot.getChildren()) {  User user = snapshot.getValue(User.class);  *// Xử lý dữ liệu ở đây* }  } |

- startAt(), endAt(), equalTo()

Lọc dữ liệu theo một giá trị cụ thể.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users"); ref.orderByChild("age").startAt(25).endAt(30).addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(DataSnapshot dataSnapshot) {  for (DataSnapshot snapshot : dataSnapshot.getChildren()) {  User user = snapshot.getValue(User.class);  *// Xử lý dữ liệu ở đây* }  } |

- limitToFirst(), limitToLast()

Giới hạn số lượng kết quả trả về.

|  |
| --- |
| DatabaseReference ref = FirebaseDatabase.*getInstance*().getReference("Users"); ref.orderByKey().limitToFirst.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(DataSnapshot dataSnapshot) {  for (DataSnapshot snapshot : dataSnapshot.getChildren()) {  User user = snapshot.getValue(User.class);  *// Xử lý dữ liệu ở đây* }  } |

2. Firebase Storage

a. tải ảnh lên firebase

chuẩn bị : Import Thư Viện

* AndroidManifest.xml : [cấp quyền truy cập ảnh](#_a.Quyền_đọc_và)
* build.gradle.kts : [thư viện loại ảnh lên view](#_a.Thư_viện_load)

Bước 1. Mở Anbulm ảnh từ máy ảo

|  |
| --- |
| StorageReference storageRef // khai báo cục bộ  storageRef = FirebaseStorage.*getInstance*().getReference().child("Image User"); |

|  |
| --- |
| private void openAnbulm() {  Intent intent = new Intent();  intent.setType("image/\*");  *// (“\*/\*”) : truy cập đến toàn bộ file*  *// (“video/\*”) : truy cập đến file video*  intent.setAction(Intent.*ACTION\_GET\_CONTENT*);  activityResultLauncher.launch(intent); } |

Bước 2 : Nhận ảnh đã chọn từ máy

// mục đích : Lấy được link url của ảnh

|  |
| --- |
| *// Khởi tạo ActivityResultLauncher để nhận kết quả từ intent khác (chọn ảnh)* private final ActivityResultLauncher<Intent> activityResultLauncher  = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(),  new ActivityResultCallback<ActivityResult>() {  @Override  public void onActivityResult(ActivityResult result) {  if (result.getResultCode() == *RESULT\_OK* && result.getData() != null) {  imageUri = result.getData().getData(); *// Lấy URI của ảnh đã chọn* if (imageUri != null) {  ivAvatar.setImageURI(imageUri); *// Hiển thị ảnh đã chọn trên ImageView* }  }  }  }); |

//Uri imageUrl : được khai báo cục bộ

Bước 3 : Tải ảnh đã lấy lên firebase

|  |
| --- |
| private void themData(User user, final Dialog dialog) {  final StorageReference fileReference = storageRef.child(user.getID() + ".jpg"); // đặt tên ảnh  fileReference.putFile(imageUri) // tải ảnh lên firebase  .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {  @Override  public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {  fileReference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {  @Override  public void onSuccess(Uri uri) {  String imageUrl = uri.toString();  // lấy link uri lưu vào user rôi thêm data lên firebase  user.setAvatar(imageUrl); *// Đặt URL ảnh cho user* SaveData(user, dialog); *// Lưu user* }  });  } })  .addOnFailureListener(new OnFailureListener() {  @Override  public void onFailure(@NonNull Exception e) {  showLoading(1); *// Đóng dialog loading* Toast.*makeText*(MainActivity.this, "Thêm ảnh thất bại: " + e.getMessage(), Toast.*LENGTH\_SHORT*).show();  }  }); } |

# V. Những thư viện thường sử dụng

## 1.AndroidManifest.xml

### a.Quyền đọc và ghi dữ liệu từ bộ nhớ ngoài của thiết bị

|  |
| --- |
| <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  Quyền này cho phép ứng dụng đọc dữ liệu từ bộ nhớ ngoài của thiết bị, chẳng hạn như đọc các tập tin ảnh, video, âm nhạc từ thẻ nhớ hoặc bộ nhớ trong của thiết bị. |
| <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  Quyền này cho phép ứng dụng ghi dữ liệu vào bộ nhớ ngoài của thiết bị, chẳng hạn như lưu trữ các tập tin ảnh, video, âm nhạc từ ứng dụng của bạn vào thẻ nhớ hoặc bộ nhớ trong của thiết bị. |

### 

### b.Quyền truy cập internet

Cho phép ứng dụng truy cập vào internet để tải xuống dữ liệu từ mạng và tương tác với các dịch vụ trực tuyến.

|  |
| --- |
| <uses-permission android:name="android.permission.INTERNET" /> |

### c.Quyền truy cập mạng wifi

Cho phép ứng dụng truy cập mạng của wifi để kết nối và tương tác với wifi của thiết bị.

|  |
| --- |
| <uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" /> |

### d. Quyền sử dụng máy ảnh

Cho phép ứng dụng sử dụng máy ảnh của thiết bị để chụp ảnh.

|  |
| --- |
| <uses-permission android:name="android.permission.CAMERA" /> |

### e. Quyền truy cập vị trí (GPS):

Cho phép ứng dụng truy cập và sử dụng thông tin vị trí từ GPS hoặc mạng để cung cấp dịch vụ dựa trên vị trí.

|  |
| --- |
| <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" /> <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" /> |

### g. Quyền đọc danh bạ:

Cho phép ứng dụng đọc danh bạ của thiết bị để có thể hiển thị thông tin liên lạc.

|  |
| --- |
| <uses-permission android:name="android.permission.READ\_CONTACTS" /> |

h.Quyền gửi tin nhắn SMS:

Cho phép ứng dụng gửi và nhận tin nhắn SMS.

|  |
| --- |
| <uses-permission android:name="android.permission.SEND\_SMS" /> |

i. Quyền gọi điện thoại:

Cho phép ứng dụng thực hiện cuộc gọi điện thoại.

|  |
| --- |
| <uses-permission android:name="android.permission.CALL\_PHONE" /> |

j.Quyền quản lý cuộc gọi:

Cho phép ứng dụng quản lý cuộc gọi đến, cuộc gọi đi và các thông tin liên quan.

|  |
| --- |
| <uses-permission android:name="android.permission.READ\_CALL\_LOG" /> <uses-permission android:name="android.permission.WRITE\_CALL\_LOG" /> |

k.Quyền truy cập vào lịch trình (Calendar):

Cho phép ứng dụng đọc và ghi lịch trình của thiết bị.

|  |
| --- |
| <uses-permission android:name="android.permission.READ\_CALENDAR" /> <uses-permission android:name="android.permission.WRITE\_CALENDAR" /> |

## 2.build.gradle.kts

### a.Thư viện load ảnh

|  |
| --- |
| *implementation* ("com.github.bumptech.glide:glide:4.12.0") *annotationProcessor* ("com.github.bumptech.glide:compiler:4.12.0") |

- Cách sử dụng

|  |
| --- |
| User user = list.get(position); Glide.*with*(context).load(user.getAvatar()).into(holder.ivAvatar);  // bổ xung  *// Load ảnh từ URL vào ImageView sử dụng Glide* Glide.*with*(context)  .load("https://example.com/image.jpg")  .placeholder(R.drawable.placeholder) *// Ảnh placeholder hiển thị trong khi đang tải* .error(R.drawable.error\_image) *// Ảnh sẽ hiển thị khi tải lỗi* .override(300, 200) *// Kích thước ảnh được tải xuống (300x200 px)* .centerCrop() *// Căn giữa và cắt phần thừa nếu kích thước không khớp* .diskCacheStrategy(DiskCacheStrategy.ALL) *// Chiến lược lưu cache của ảnh* .skipMemoryCache(true) *// Bỏ qua lưu cache trong bộ nhớ* .listener(new RequestListener<Drawable>() {  @Override  public boolean onLoadFailed(@Nullable GlideException e, Object model, Target<Drawable> target, boolean isFirstResource) {  *// Xử lý khi tải ảnh thất bại* return false;  }   @Override  public boolean onResourceReady(Drawable resource, Object model, Target<Drawable> target, DataSource dataSource, boolean isFirstResource) {  *// Xử lý khi tải ảnh thành công* return false;  }  })  .into(imageView); *// Thiết lập ImageView để hiển thị ảnh* |

# VI.CÁCH NHỚ TÀI KHOẢN MẬT KHẨU

Bước 1. Tạo 1 bộ nhớ ngay trên máy điện thoại

Phương thức này khởi tạo ở hàm oncreat để khi vào app sẽ được chạy ngay

|  |
| --- |
| private void setChkRemember() {  SharedPreferences sharedPreferences = getSharedPreferences("USER\_FILE",*MODE\_PRIVATE*);  *//Tạo một đối tượng SharedPreferences có tên là "USER\_FILE".  // Đối tượng này sẽ được sử dụng để lưu trữ các thông tin người dùng.* edtUsername.setText(sharedPreferences.getString("USER",""));  *//Lấy giá trị của chuỗi có key là "USER" từ SharedPreferences và đặt nó vào EditText edtUsername.  // Nếu không tìm thấy giá trị, sử dụng chuỗi rỗng mặc định.* edtPassword.setText(sharedPreferences.getString("PASS",""));  chkRemember.setChecked(sharedPreferences.getBoolean("REMEMBER",false)); } |

Bước 2 : lưu trữ thông tin người dùng vào SharedPreferences.

MODE\_PRIVATE : chế độ chỉ được đọc viết của sharePreferences

|  |
| --- |
| private void rememberUser(String user,String pass,boolean status) {  SharedPreferences preferences = getSharedPreferences("USER\_FILE",*MODE\_PRIVATE*);  SharedPreferences.Editor editor = preferences.edit();  *// Tạo một đối tượng Editor để thực hiện chỉnh sửa SharedPreferences.* if (status == false) {  editor.clear();  *// nếu người dùng không nhấn chkremember thì clear toàn bộ dự liệu* } else {  editor.putString("USER",user);  editor.putString("PASS",pass);  editor.putBoolean("REMEMBER",status);  }  editor.commit();  *// lưu lại dữ liệu* } |

Bước 3 :

Sử dụng thuật toán khi nào tài khoản mật khẩu đúng thì sẽ gọi đến phương thức rememberUser(u,p,status) đề lưu lại tài khoản mật khẩu

# VII. Tạo NavigationView

### 1.Cách Tạo

Bước 1. Tạo View

|  |
| --- |
| *<!-- activity\_main.xml -->* <androidx.drawerlayout.widget.DrawerLayout  xmlns:android="http://schemas.android.com/apk/res/android"  android:id="@+id/drawer\_layout"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   *<!-- Content view -->* <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   *<!-- Toolbar -->* <androidx.appcompat.widget.Toolbar  android:id="@+id/toolbar"  android:layout\_width="match\_parent"  android:layout\_height="?attr/actionBarSize"  android:background="?attr/colorPrimary" />   *<!-- Main content -->* <FrameLayout  android:id="@+id/nav\_host\_fragment"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </LinearLayout>   *<!-- NavigationView -->* <com.google.android.material.navigation.NavigationView  android:id="@+id/nav\_view"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:layout\_gravity="start"  app:headerLayout="@layout/nav\_header"  app:menu="@menu/nav\_menu"  />  </androidx.drawerlayout.widget.DrawerLayout> |

Bước 2 : Xử lý sự kiện khi click vào

|  |
| --- |
| *// Khi vào app ngay lập tức xuất hiện ở fragament sách .* getSupportFragmentManager().beginTransaction()  .replace(R.id.*fragment*, new FragmentQLSach())  .commit();  *// Ẩn tiêu đề trên ActionBar để tăng không gian hiển thị* getSupportActionBar().setDisplayShowTitleEnabled(false);  *// Thiết lập ActionBarDrawerToggle để có thể click vào dấu 3 gạch để xuất hiện navigation* ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar, 0, 0);  *0, 0: Hai tham số cuối cùng là ID của strings để mô tả mở và đóng. Trong trường hợp này, chúng được đặt là 0, vì không cần thiết để cung cấp chuỗi mô tả.*  *// Thêm một DrawerListener vào DrawerLayout để lắng nghe các sự kiện mở và đóng của Drawer.*  drawerLayout.addDrawerListener(toggle);  *// syncState: Đồng bộ hóa trạng thái của DrawerLayout và* *toggle với trạng thái hiện tại của ActionBar. Nó cần được gọi sau khi gọi addDrawerListener để đảm bảo rằng trạng thái mở hoặc đóng của DrawerLayout được phản ánh chính xác trên ActionBar.*  toggle.syncState();  *// Xử lý sự kiện khi người dùng chọn mục trong NavigationView* navigationView.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {  @Override  public boolean onNavigationItemSelected(MenuItem item) {  Fragment fragment = null;   *// Xử lý từng lựa chọn từ menu NavigationView* if (item.getItemId() == R.id.*nav\_PhieuMuon*) {  fragment = new FragmentPhieuMuon();  } else if (item.getItemId() == R.id.*nav\_ThanhVien*) {  fragment = new FragmentQuanLyThanhVien();  }  *// Nếu fragment được chọn không null, thay thế fragment hiện tại và đóng DrawerLayout* if (fragment != null) {  getSupportFragmentManager().beginTransaction()  .replace(R.id.*fragment*, fragment)  .commit();  drawerLayout.close();  }   return true;  }  }); |

### 2. Cách ẩn 1 item khi trong navigation view

|  |
| --- |
| if (navigationView != null) {  if (!checkAdmin(*thuThu*)) {  *// lấy menu từ navigationView . Tìm item theo ID. Ẩn bằng phương thực setVisible(false);* navigationView.getMenu().findItem(R.id.*qlUser*).  setVisible(false);  }} } |

### 3.Cách hiển thị tên người dùng lên thanh tiêu đề của NavigationView

|  |
| --- |
| *// Phương thức này được gọi để hiển thị tên người dùng lên thanh tiêu đề của NavigationView* private void setNameUser() {  *// Lấy Intent từ Activity hiện tại* Intent intent = getIntent();  *// Lấy chuỗi "user" từ Intent, giả định rằng đây là tên người dùng* String user = intent.getStringExtra("user");   *// Lấy thông tin chi tiết của người dùng từ cơ sở dữ liệu  thuThu* = *daoThuthu*.getID(user);   *// Lấy View của header trong NavigationView để hiển thị thông tin người dùng* View view = navigationView.getHeaderView(0);  *// Đối với phương thức getHeaderView(0), nó trả về header đầu tiên trong NavigationView. Trong hầu hết các trường hợp, NavigationView chỉ có một header duy nhất, do đó bạn thường sẽ sử dụng getHeaderView(0) để lấy ra header này.*  *// Tìm TextView có id là tvUser trong header View* TextView tvTen = view.findViewById(R.id.*tvUser*);  *// Thiết lập văn bản của TextView thành tên của người dùng* tvTen.setText(*thuThu*.getHoTen()); } |

# VII. NÂNG CAO VỀ RECYCLERVIEW

1. Cách scroll (kéo lên sẽ ẩn button add)

|  |
| --- |
| recyclerViewSach.addOnScrollListener(new RecyclerView.OnScrollListener() {  @Override  public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {  super.onScrolled(recyclerView, dx, dy);  if (dy > 0) {  btnAdd.hide();  } else if (dy < 0) {  btnAdd.show();  }  } }); |