The UXAccessibilityService.java

```
package com.example.uxinteractionlogger;
import android.accessibilityservice.AccessibilityService;
import android.graphics.Rect;
import android.util.Log;
import android.view.accessibility.AccessibilityEvent;
import android.view.accessibility.AccessibilityNodeInfo;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.*;

public class UXAccessibilityService extends AccessibilityService {

    private static final String TAG = "UXLogger";
    private static final SimpleDateFormat timeFormat =
            new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS", Locale.US);

    private File logFile;
    private String sessionId;
    private int swipeCounter = 0;

    private static class NodeSnapshot {
        String label;
        Rect bounds;
        String itemId;

        NodeSnapshot(String label, Rect bounds, String itemId) {
            this.label = label;
            this.bounds = bounds;
            this.itemId = itemId;
        }
    String getKey() {
        return label + "|" + bounds.centerX() + "," + bounds.centerY();
    }
}

private static class RemovedItem {
    String swipeId;
    String itemId;
    String label;
    String visibleUi;
    String packageName;

    RemovedItem(String swipeId, String itemId, String label, String visibleUi, String packageName) {
        this.swipeId = swipeId;
        this.itemId = itemId;
        this.label = label;
        this.visibleUi = visibleUi;
        this.packageName = packageName;
    }
}

private final Map<String, String> itemKeyToStableId = new HashMap<>();
private final Set<String> seenItemIdsInMainUI = new HashSet<>();
private final List<RemovedItem> recentlyRemovedItems = new ArrayList<>();
private List<NodeSnapshot> previousSnapshot = new ArrayList<>();

@Override
public void onCreate() {
    super.onCreate();
    sessionId = UUID.randomUUID().toString();
    File dir = getExternalFilesDir(null);
    if (dir != null) {
        logFile = new File(dir, "UX_Log.txt");
        try (FileWriter writer = new FileWriter(logFile, false)) {
            writer.write("=== New Session Started ===\n");
        } catch (IOException e) {
            Log.e(TAG, "Failed to initialize log file", e);
        }
    }
}

@Override
public void onAccessibilityEvent(AccessibilityEvent event) {
    AccessibilityNodeInfo rootNode = getRootInActiveWindow();
    if (rootNode == null) return;

    String timestamp = timeFormat.format(new Date());
    String visibleUi = extractVisibleUILabel(rootNode);

    if (event.getEventType() == AccessibilityEvent.TYPE_VIEW_SCROLLED) {
        List<NodeSnapshot> currentSnapshot = buildNodeSnapshot(rootNode);

        // 1. Undo Detection
        Iterator<RemovedItem> iter = recentlyRemovedItems.iterator();
        while (iter.hasNext()) {
            RemovedItem removed = iter.next();
            if (!removed.visibleUi.equals(visibleUi)) continue;

            for (NodeSnapshot current : currentSnapshot) {
                if (current.itemId.equals(removed.itemId)) {
                    log(timestamp + " | UNDO_ACTION"
                            + " | swipe_id: " + removed.swipeId
                            + " | package: " + removed.packageName
                            + " | undone_ui_item: \"" + removed.label + "\""
                            + " | item_id: " + current.itemId
                            + " | visible_ui: " + visibleUi);
                    iter.remove();
                    break;
                }
            }
        }

        // 2. Swipe Detection
        Set<String> currentItemIds = new HashSet<>();
        for (NodeSnapshot snap : currentSnapshot) currentItemIds.add(snap.itemId);

        for (NodeSnapshot prev : previousSnapshot) {
            if (!currentItemIds.contains(prev.itemId)) {
                // Make sure it's not a full refresh
                int remaining = 0;
                for (NodeSnapshot other : previousSnapshot) {
                    if (!other.itemId.equals(prev.itemId) && currentItemIds.contains(other.itemId)) {
                        remaining++;
                    }
                }

                if (remaining >= 2) {
                    swipeCounter++;
                    String swipeId = "swipe_" + swipeCounter;

                    log(timestamp + " | SWIPE_EVENT"
                            + " | swipe_id: " + swipeId
                            + " | package: " + event.getPackageName()
                            + " | direction: left"
                            + " | swiped_ui_item: \"" + prev.label + "\""
                            + " | item_id: " + prev.itemId
                            + " | triggered_action: removed"
                            + " | visible_ui: " + visibleUi);

                    recentlyRemovedItems.add(new RemovedItem(
                            swipeId, prev.itemId, prev.label, visibleUi, event.getPackageName().toString()
                    ));
```

```
                break;
            }
        }
    }

    previousSnapshot = currentSnapshot;
  }
}

private List<NodeSnapshot> buildNodeSnapshot(AccessibilityNodeInfo root) {
  List<NodeSnapshot> result = new ArrayList<>();
  Queue<AccessibilityNodeInfo> queue = new LinkedList<>();
  queue.add(root);

  while (!queue.isEmpty()) {
    AccessibilityNodeInfo node = queue.poll();
    if (node == null) continue;

    CharSequence text = node.getText();
    Rect bounds = new Rect();
    node.getBoundsInScreen(bounds);

    if (text != null && !text.toString().trim().isEmpty()
        && bounds.width() > 0 && bounds.height() > 0) {
      String label = text.toString().trim();
      String key = label + "|" + bounds.centerX() + "," + bounds.centerY();
      String itemId;

      if (itemKeyToStableId.containsKey(key)) {
        itemId = itemKeyToStableId.get(key);
      } else {
        itemId = "item_" + UUID.randomUUID().toString().substring(0, 8);
        itemKeyToStableId.put(key, itemId);
      }
```

```
            seenItemIdsInMainUI.add(itemId); // track as seen
            result.add(new NodeSnapshot(label, bounds, itemId));
        }

        for (int i = 0; i < node.getChildCount(); i++) {
            AccessibilityNodeInfo child = node.getChild(i);
            if (child != null) queue.add(child);
        }
    }

    return result;
}

private String extractVisibleUILabel(AccessibilityNodeInfo root) {
    if (root == null) return "unknown";
    Queue<AccessibilityNodeInfo> queue = new LinkedList<>();
    queue.add(root);
    List<String> labels = new ArrayList<>();

    while (!queue.isEmpty()) {
        AccessibilityNodeInfo node = queue.poll();
        if (node == null || node.isClickable() || node.isCheckable()) continue;

        CharSequence text = node.getText();
        CharSequence desc = node.getContentDescription();

        if (text != null && isTitleLike(text.toString())) labels.add(text.toString().trim());
        if (desc != null && isTitleLike(desc.toString())) labels.add(desc.toString().trim());

        for (int i = 0; i < node.getChildCount(); i++) {
            AccessibilityNodeInfo child = node.getChild(i);
            if (child != null) queue.add(child);
        }
        }

        return labels.isEmpty() ? "unknown" : labels.get(0);
    }

    private boolean isTitleLike(String text) {
        if (text == null || text.length() > 30 || text.matches(".*\\d{3,}.*")) return false;
        String lower = text.toLowerCase();
        return !(lower.contains("toolbar") || lower.contains("layout") ||
                lower.contains("button") || lower.contains("search") ||
                lower.contains("background"));
    }

    private void log(String text) {
        if (logFile == null) return;
        try (FileWriter writer = new FileWriter(logFile, true)) {
            writer.write(text + "\n");
        } catch (IOException e) {
            Log.e(TAG, "Failed to log", e);
        }
    }

    @Override
    public void onInterrupt() {
        Log.d(TAG, "Service interrupted");
    }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.UXInteractionLogger"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.UXInteractionLogger">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- Accessibility Service -->
        <service
            android:name=".UXAccessibilityService"
            android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
            android:exported="true">
            <intent-filter>
                <action android:name="android.accessibilityservice.AccessibilityService" />
            </intent-filter>

            <meta-data
                android:name="android.accessibilityservice"
                android:resource="@xml/accessibility_service_config" />
        </service>
    </application>

</manifest>
```

Accessibility_service_config

```
<accessibility-service xmlns:android="http://schemas.android.com/apk/res/android"

android:accessibilityEventTypes="typeViewScrolled|typeWindowStateChanged|typeViewClicked|typeViewText
Changed|typeWindowContentChanged|typeViewFocused"
    android:accessibilityFeedbackType="feedbackGeneric"
    android:notificationTimeout="100"
    android:canRetrieveWindowContent="true"

android:accessibilityFlags="flagReportViewIds|flagRetrieveInteractiveWindows|flagIncludeNotImportantViews
"
    android:description="@string/accessibility_service_description" />
```