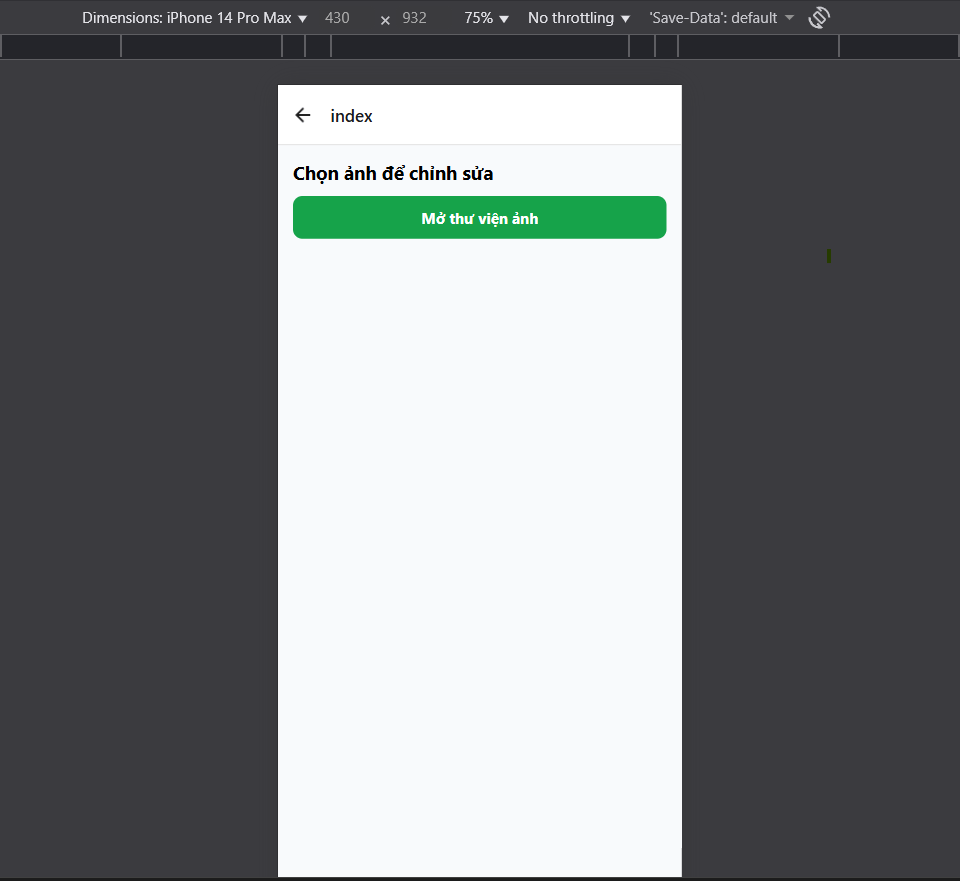
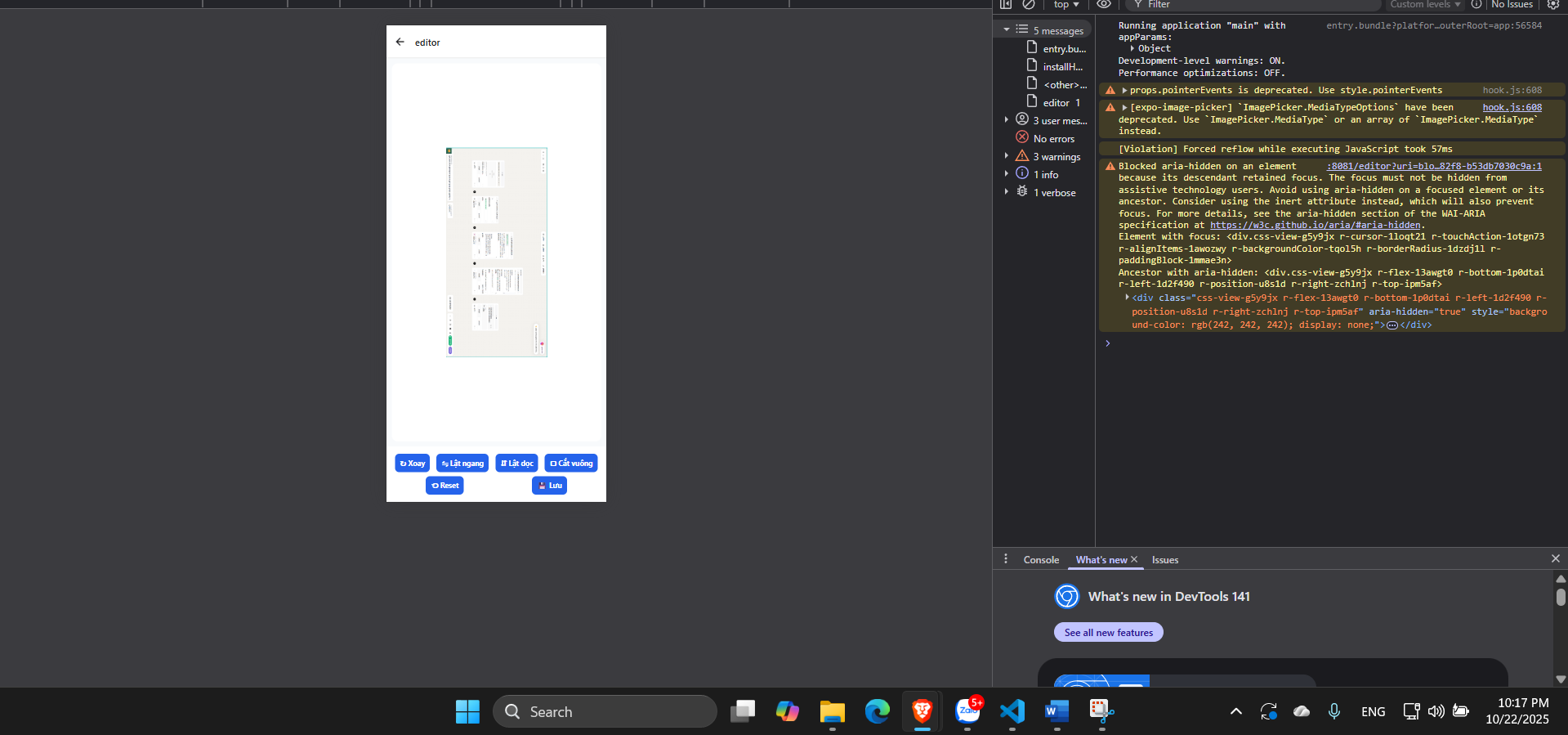
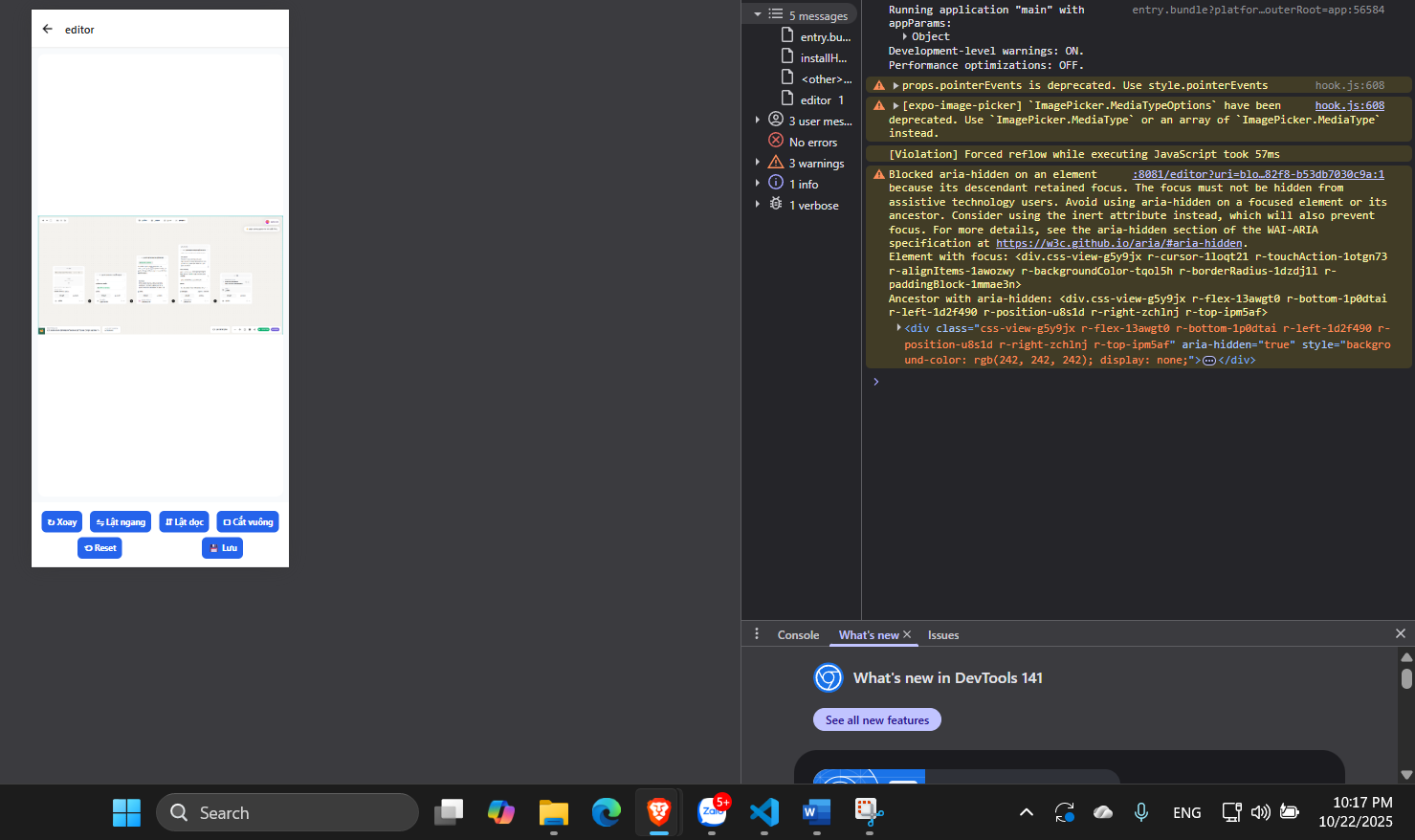
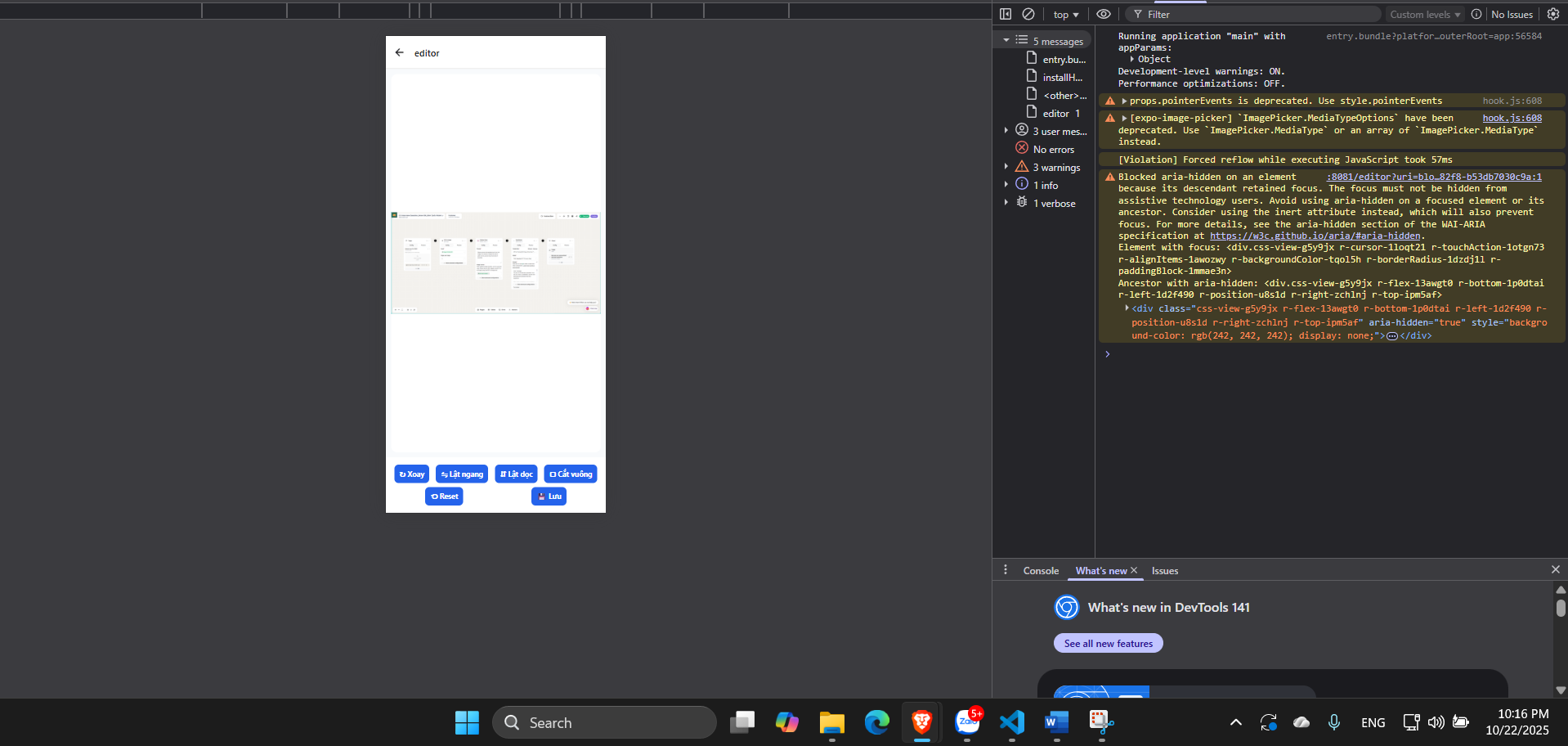
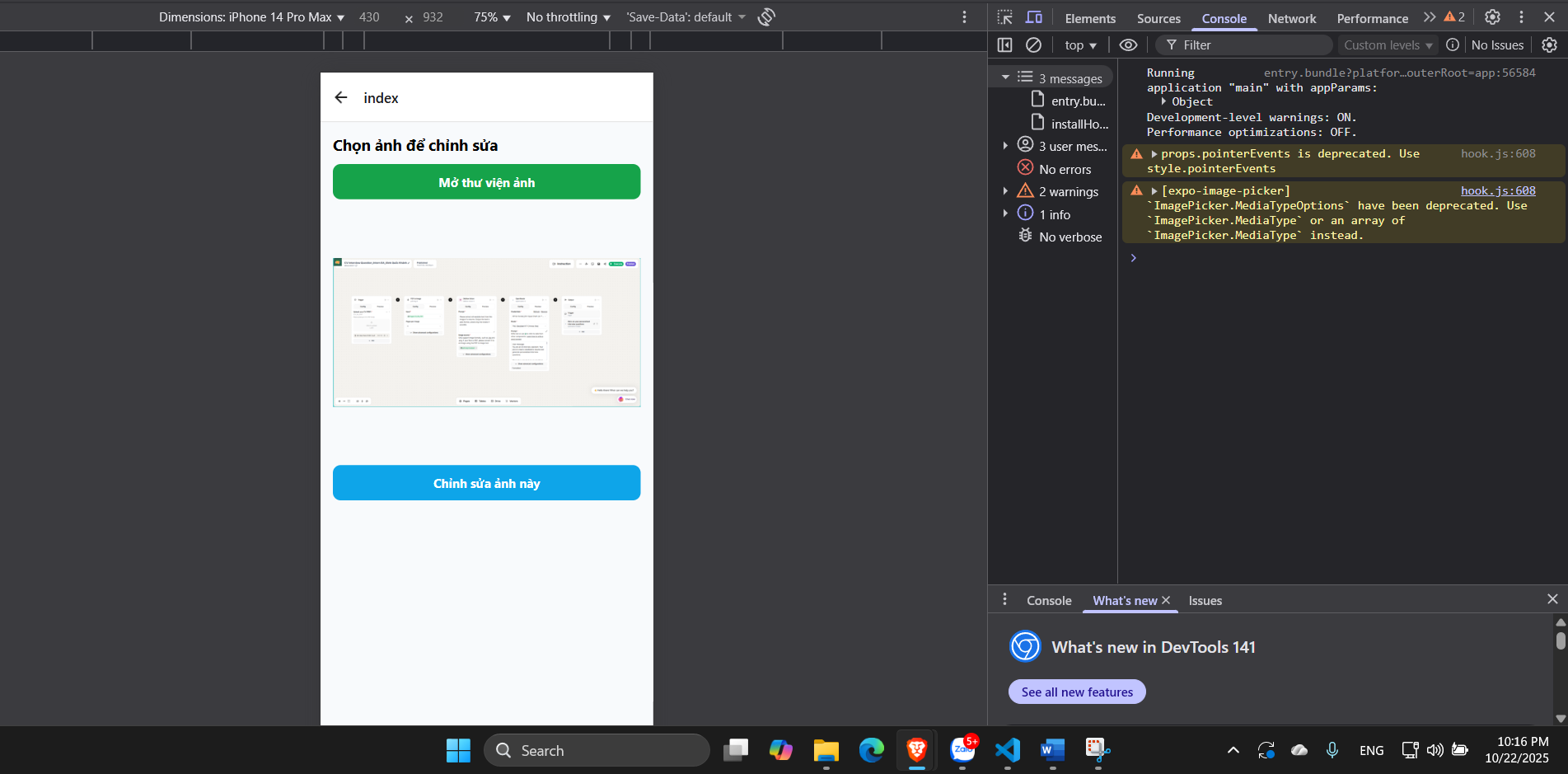
Đinh Quốc Khánh\_21096081

Link Github : https://github.com/KhanhDinh723/ReactNative/tree/main/React%20Native/DinhQuocKhanh\_img\_edit/image-editor



-



----------------code

import { useEffect, useState } from "react";

import { View, FlatList, Image, Text, StyleSheet } from "react-native";

import AsyncStorage from "@react-native-async-storage/async-storage";

type EditedImage = {

  id: string;

  originalUri: string;

  editedUri: string;

  savedAt: number;

};

const STORAGE\_KEY = "edited\_images\_v1";

export default function GalleryScreen() {

  const [images, setImages] = useState<EditedImage[]>([]);

  useEffect(() => {

    (async () => {

      const data = await AsyncStorage.getItem(STORAGE\_KEY);

      if (data) setImages(JSON.parse(data));

    })();

  }, []);

  return (

    <View style={styles.container}>

      <FlatList

        data={images}

        keyExtractor={(item) => item.id}

        renderItem={({ item }) => (

          <View style={styles.card}>

            <Image source={{ uri: item.editedUri }} style={styles.thumb} />

            <View style={{ flex: 1 }}>

              <Text style={styles.time}>{new Date(item.savedAt).toLocaleString()}</Text>

              <Text numberOfLines={1} style={styles.uri}>

                {item.editedUri}

              </Text>

            </View>

          </View>

        )}

        ListEmptyComponent={<Text style={styles.empty}>Chưa có ảnh đã chỉnh sửa</Text>}

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: { flex: 1, padding: 12, backgroundColor: "#f1f5f9" },

  card: {

    flexDirection: "row",

    backgroundColor: "#fff",

    padding: 8,

    borderRadius: 10,

    marginBottom: 10,

    alignItems: "center",

  },

  thumb: { width: 80, height: 80, borderRadius: 8, marginRight: 10 },

  time: { fontWeight: "bold" },

  uri: { fontSize: 12, color: "#555" },

  empty: { textAlign: "center", marginTop: 20 },

});

import { useLocalSearchParams, useRouter } from "expo-router";

import { useEffect, useState, useMemo } from "react";

import { View, Image, StyleSheet, ActivityIndicator, Alert } from "react-native";

import \* as ImageManipulator from "expo-image-manipulator";

import \* as MediaLibrary from "expo-media-library";

import AsyncStorage from "@react-native-async-storage/async-storage";

import { v4 as uuidv4 } from "uuid";

import EditorToolbar from "../components/EditorToolbar";

type EditedImage = {

  id: string;

  originalUri: string;

  editedUri: string;

  savedAt: number;

};

const STORAGE\_KEY = "edited\_images\_v1";

export default function EditorScreen() {

  const { uri } = useLocalSearchParams<{ uri: string }>();

  const router = useRouter();

  const [imageUri, setImageUri] = useState(uri || "");

  const [saving, setSaving] = useState(false);

  const [rotate, setRotate] = useState(0);

  const [flipH, setFlipH] = useState(false);

  const [flipV, setFlipV] = useState(false);

  const [cropSquare, setCropSquare] = useState(false);

  if (!imageUri) return null;

  const transforms = useMemo(() => {

    const t: any[] = [];

    if (rotate) t.push({ rotate: `${rotate}deg` });

    if (flipH) t.push({ scaleX: -1 });

    if (flipV) t.push({ scaleY: -1 });

    return t;

  }, [rotate, flipH, flipV]);

  const handleSave = async () => {

    try {

      setSaving(true);

      const perm = await MediaLibrary.requestPermissionsAsync();

      if (!perm.granted) throw new Error("Thiếu quyền truy cập thư viện ảnh.");

      const actions: ImageManipulator.Action[] = [];

      if (rotate) actions.push({ rotate });

      if (flipH || flipV) actions.push({ flip: { horizontal: flipH, vertical: flipV } });

      if (cropSquare) actions.push({ crop: { originX: 0, originY: 0, width: 600, height: 600 } });

      const result = await ImageManipulator.manipulateAsync(imageUri, actions, {

        compress: 1,

        format: ImageManipulator.SaveFormat.JPEG,

      });

      const asset = await MediaLibrary.createAssetAsync(result.uri);

      const album = await MediaLibrary.getAlbumAsync("ImageEditor");

      if (album) await MediaLibrary.addAssetsToAlbumAsync([asset], album, false);

      else await MediaLibrary.createAlbumAsync("ImageEditor", asset, false);

      const record: EditedImage = {

        id: uuidv4(),

        originalUri: imageUri,

        editedUri: result.uri,

        savedAt: Date.now(),

      };

      const existing = await AsyncStorage.getItem(STORAGE\_KEY);

      const parsed = existing ? JSON.parse(existing) : [];

      parsed.unshift(record);

      await AsyncStorage.setItem(STORAGE\_KEY, JSON.stringify(parsed));

      Alert.alert("✅ Thành công", "Ảnh đã được lưu!", [

        { text: "Xem danh sách", onPress: () => router.push("/gallery") },

        { text: "OK" },

      ]);

    } catch (e: any) {

      Alert.alert("Lỗi", e.message || "Không thể lưu ảnh.");

    } finally {

      setSaving(false);

    }

  };

  return (

    <View style={styles.container}>

      <View style={styles.previewWrap}>

        <Image source={{ uri: imageUri }} style={[styles.image, { transform: transforms }]} resizeMode="contain" />

        {saving && (

          <View style={styles.overlay}>

            <ActivityIndicator size="large" />

          </View>

        )}

      </View>

      <EditorToolbar

        onRotate={() => setRotate((r) => (r + 90) % 360)}

        onFlipH={() => setFlipH((v) => !v)}

        onFlipV={() => setFlipV((v) => !v)}

        onCropSquare={() => setCropSquare((v) => !v)}

        onReset={() => {

          setRotate(0);

          setFlipH(false);

          setFlipV(false);

          setCropSquare(false);

        }}

        onSave={handleSave}

      />

    </View>

  );

}

const styles = StyleSheet.create({

  container: { flex: 1, backgroundColor: "#f8fafc" },

  previewWrap: {

    flex: 1,

    margin: 10,

    backgroundColor: "#fff",

    borderRadius: 12,

    overflow: "hidden",

    alignItems: "center",

    justifyContent: "center",

  },

  image: { width: "100%", height: "100%" },

  overlay: {

    ...StyleSheet.absoluteFillObject,

    justifyContent: "center",

    alignItems: "center",

    backgroundColor: "rgba(0,0,0,0.3)",

  },

});

import { View, Text, Pressable, Image, StyleSheet, Alert } from "react-native";

import \* as ImagePicker from "expo-image-picker";

import { useState } from "react";

import { useRouter } from "expo-router";

export default function HomeScreen() {

  const [uri, setUri] = useState<string | null>(null);

  const router = useRouter();

  const pickImage = async () => {

    const perm = await ImagePicker.requestMediaLibraryPermissionsAsync();

    if (!perm.granted) {

      Alert.alert("Thiếu quyền", "Cần quyền truy cập thư viện ảnh.");

      return;

    }

    const res = await ImagePicker.launchImageLibraryAsync({

      mediaTypes: ImagePicker.MediaTypeOptions.Images,

      quality: 1,

    });

    if (!res.canceled) {

      setUri(res.assets[0].uri);

    }

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Chọn ảnh để chỉnh sửa</Text>

      <Pressable style={styles.primary} onPress={pickImage}>

        <Text style={styles.primaryText}>Mở thư viện ảnh</Text>

      </Pressable>

      {uri && (

        <>

          <Image source={{ uri }} style={styles.preview} resizeMode="contain" />

          <Pressable

            style={styles.secondary}

            onPress={() => router.push(`/editor?uri=${encodeURIComponent(uri!)}`)}

          >

            <Text style={styles.secondaryText}>Chỉnh sửa ảnh này</Text>

          </Pressable>

        </>

      )}

      <Pressable style={styles.link} onPress={() => router.push("/gallery")}>

        <Text style={styles.linkText}>Xem ảnh đã chỉnh</Text>

      </Pressable>

    </View>

  );

}

const styles = StyleSheet.create({

  container: { flex: 1, padding: 16, gap: 12, backgroundColor: "#f8fafc" },

  title: { fontSize: 20, fontWeight: "bold" },

  primary: {

    backgroundColor: "#16a34a",

    paddingVertical: 12,

    borderRadius: 10,

    alignItems: "center",

  },

  primaryText: { color: "#fff", fontSize: 16, fontWeight: "700" },

  preview: { width: "100%", height: 300, marginVertical: 10, borderRadius: 10 },

  secondary: {

    backgroundColor: "#0ea5e9",

    paddingVertical: 12,

    borderRadius: 10,

    alignItems: "center",

  },

  secondaryText: { color: "#fff", fontSize: 16, fontWeight: "700" },

  link: { alignItems: "center", marginTop: "auto", paddingVertical: 12 },

  linkText: { color: "#2563eb", fontWeight: "700" },

});

import { View, Pressable, Text, StyleSheet } from "react-native";

type Props = {

  onRotate: () => void;

  onFlipH: () => void;

  onFlipV: () => void;

  onCropSquare: () => void;

  onReset: () => void;

  onSave: () => void;

};

export default function EditorToolbar(props: Props) {

  return (

    <View style={styles.bar}>

      <Button label="↻ Xoay" onPress={props.onRotate} />

      <Button label="⇋ Lật ngang" onPress={props.onFlipH} />

      <Button label="⇵ Lật dọc" onPress={props.onFlipV} />

      <Button label="▢ Cắt vuông" onPress={props.onCropSquare} />

      <Button label="⟲ Reset" onPress={props.onReset} />

      <Button label="💾 Lưu" onPress={props.onSave} />

    </View>

  );

}

function Button({ label, onPress }: { label: string; onPress: () => void }) {

  return (

    <Pressable style={styles.btn} onPress={onPress}>

      <Text style={styles.txt}>{label}</Text>

    </Pressable>

  );

}

const styles = StyleSheet.create({

  bar: {

    flexDirection: "row",

    flexWrap: "wrap",

    justifyContent: "space-around",

    padding: 10,

    backgroundColor: "#fff",

  },

  btn: {

    backgroundColor: "#2563eb",

    paddingHorizontal: 10,

    paddingVertical: 8,

    borderRadius: 8,

    marginVertical: 4,

  },

  txt: { color: "#fff", fontWeight: "bold" },

});