```
import React, {useCallback} from 'react';
import {StyleSheet, TouchableOpacity, useWindowDimensions} from 'react-
native';
import Animated, {
  interpolate,
  interpolateColor,
  useAnimatedStyle,
  useDerivedValue,
  useSharedValue,
  withTiming,
} from 'react-native-reanimated';
import DraggableView from './DraggableView';
import IonicIcon from 'react-native-vector-icons/Ionicons';
import {Colors} from '../../Assets/AssetsRoot';

interface IPropsAnimatedFAB {}

const FAB_SIZE = 60;
const GAP_BTN_FAB = 10;
const DURATION = 400;

const AnimatedFAB: React.FC<IPropsAnimatedFAB> = () => {
  const {height: dHeight} = useWindowDimensions();
  const aIsActionBtnsVisible = useSharedValue(0);
  const aInputRangeStart = useSharedValue(FAB_SIZE);
  const aOutputRangeStart = useSharedValue(FAB_SIZE + GAP_BTN_FAB);

  const dDistance = useDerivedValue(() => {
    let space = -FAB_SIZE;
    let distance = {
      settings: 0,
    };

    if (aIsActionBtnsVisible.value == 1) {
      distance.settings = space;
    }

    return distance;
  });

  const fabIconBgColorStyle = useAnimatedStyle(() => {
    return {
      backgroundColor: withTiming(
        interpolateColor(
          aIsActionBtnsVisible.value,
          [0, 1],
          [Colors.CuriousBlue, Colors.Red],
        ),
        {duration: DURATION},
      ),
    };
  });

  const fabIconRotateZStyle = useAnimatedStyle(() => {
    const rotateZValue = interpolate(
      aIsActionBtnsVisible.value,
      [1, 0],
      [45, 0],
    );
    return {
      transform: [
        {rotateZ: withTiming(`${rotateZValue}deg`, {duration:
DURATION})},
      ],
    };
  });

  const settingsAnimateStyles = useAnimatedStyle(() => {
    const input_range = [aInputRangeStart.value, 0];
    const output_range = [aOutputRangeStart.value, 0];
    return {
      transform: [
        {
          translateY: withTiming(
            interpolate(dDistance.value.settings, input_range,
output_range),
            {duration: DURATION},
          ),
        },
      ],
    };
  });

  const onClickFAB = useCallback(() => {
    aIsActionBtnsVisible.value = aIsActionBtnsVisible.value == 0 ? 1 : 0;
  }, []);

  const onDragEnd = useCallback(
    (positionX: number, positionY: number) => {
      'worklet';
      if (positionY <= dHeight / 2) {
        aInputRangeStart.value = -FAB_SIZE;
      } else {
        aInputRangeStart.value = FAB_SIZE;
      }
    },
    [dHeight],
  );

  const renderFAB = useCallback(() => {
    return (
      <Animated.View style={[styles.fab, fabIconBgColorStyle]}>
        <TouchableOpacity
          style={styles.fabBtn}
          onPress={onClickFAB}
          activeOpacity={0.85}>
          <Animated.View style={fabIconRotateZStyle}>
            <IonicIcon name="add-outline" size={32} color="white" />
          </Animated.View>
        </TouchableOpacity>
      </Animated.View>
    );
  }, []);

  const renderSettings = useCallback(() => {
    return (
      <Animated.View style={[styles.actionView, settingsAnimateStyles]}>
        <TouchableOpacity style={styles.actionBtn}>
          <IonicIcon name="settings-outline" size={25} color="white" />
        </TouchableOpacity>
      </Animated.View>
    );
  }, []);

  return (
    <DraggableView size={FAB_SIZE} onDragEnd={onDragEnd}>
      {renderFAB()}
      {renderSettings()}
    </DraggableView>
  );
};

const styles = StyleSheet.create({
  fab: {
    backgroundColor: Colors.CuriousBlue,
    height: FAB_SIZE,
    width: FAB_SIZE,
    borderRadius: FAB_SIZE / 2,
    overflow: 'hidden',
    alignItems: 'center',
    justifyContent: 'center',
  },
  fabBtn: {
    alignItems: 'center',
    justifyContent: 'center',
    flex: 1,
  },
  actionView: {
    width: FAB_SIZE,
    height: FAB_SIZE,
    borderRadius: FAB_SIZE / 2,
    backgroundColor: Colors.CuriousBlue,
    justifyContent: 'center',
    alignItems: 'center',
    position: 'absolute',
    zIndex: -1,
  },
  actionBtn: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
});

export default AnimatedFAB;
```