```
import React from 'react';
import { useState, useRef, useEffect } from 'react';
import {
  View,
  Text,
  TouchableOpacity,
  StyleSheet,
  SafeAreaView,
  ScrollView,
  Switch,
  TextInput,
  Alert,
  Vibration,
  StatusBar,
} from 'react-native';
import { CameraView, useCameraPermissions } from 'expo-camera';
import * as Speech from 'expo-speech';

var BG      = '#0a0a0f';
var SURFACE = '#12121a';
var CARD    = '#1a1a26';
var BORDER  = '#2a2a3a';
var ACCENT  = '#00e5ff';
var WARN    = '#ff6d00';
var DANGER  = '#ff1744';
var SAFE    = '#00e676';
var TXTCOL  = '#e8e8f0';
var MUTED   = '#6b6b88';

var INTERVAL = 2500;

var PROMPT = 'You are an AI assistant for visually impaired people. '
  + 'Analyze this image and respond ONLY with this JSON format, no other text: '
  + '{"objects":[{"name":"object name","direction":"left or right or front or bac
  + '"estimatedDistance":2.5,"riskLevel":"danger or warn or safe",'
  + '"speechText":"short alert message"}],'
  + '"overallSafe":true,"sceneSummary":"brief summary"}';

function callClaude(b64, key) {
  return fetch('https://api.anthropic.com/v1/messages', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'x-api-key': key,
      'anthropic-version': '2023-06-01',
    },
    body: JSON.stringify({
      model: 'claude-opus-4-6',
      max_tokens: 800,
      messages: [{
        role: 'user',
        content: [
          { type: 'image', source: { type: 'base64', media_type: 'image/jpeg', da
          { type: 'text', text: PROMPT },
        ],
      }],
    }),
  })
  .then(function(r) { return r.json(); })
  .then(function(d) {
    var txt = d.content && d.content[0] ? d.content[0].text : '{}';
    var m = txt.match(/\{[\s\S]*\}/);
    return m ? JSON.parse(m[0]) : { objects: [], overallSafe: true };
  });
}

function sayIt(text) {
  Speech.stop();
  Speech.speak(text, { language: 'en-US', rate: 0.85 });
}

function buzz(level) {
  if (level === 'danger') Vibration.vibrate([100, 60, 100, 60, 300]);
  else if (level === 'warn') Vibration.vibrate([100, 60, 150]);
}

function Banner(props) {
  var obj = props.obj;
  if (!obj) return null;
  var c = obj.riskLevel === 'danger' ? DANGER : WARN;
  return (
    <View style={[sty.banner, { backgroundColor: c }]}>
      <Text style={sty.bannerTxt}>{obj.speechText}</Text>
    </View>
  );
}

function Overlay(props) {
  var analyzing = props.analyzing;
  var objects = props.objects || [];
  return (
    <View style={StyleSheet.absoluteFill} pointerEvents='none'>
      <View style={[sty.corner, { top: 14, left: 14, borderTopWidth: 2, borderLef
      <View style={[sty.corner, { top: 14, right: 14, borderTopWidth: 2, borderRi
      <View style={[sty.corner, { bottom: 14, left: 14, borderBottomWidth: 2, bor
      <View style={[sty.corner, { bottom: 14, right: 14, borderBottomWidth: 2, bo
      <View style={sty.chip}>
        <View style={{ width: 7, height: 7, borderRadius: 4,
          backgroundColor: analyzing ? WARN : SAFE }} />
        <Text style={{ color: TXTCOL, fontSize: 11 }}>
          {analyzing ? 'Analyzing...' : 'AI Ready'}
        </Text>
      </View>
      {objects.filter(function(o) { return o.riskLevel !== 'safe'; })
        .map(function(o, i) {
          var c = o.riskLevel === 'danger' ? DANGER : WARN;
          var pos = o.direction === 'left'  ? { left: '4%', top: '35%' }
                  : o.direction === 'right' ? { right: '4%', top: '35%' }
                  : { alignSelf: 'center', top: '10%' };
          return (
            <View key={i} style={[sty.badge, pos, { borderColor: c }]}>
              <Text style={{ color: c, fontWeight: '700', fontSize: 12 }}>{o.name
              <Text style={{ color: c, fontSize: 10 }}>{o.estimatedDistance}m</Te
            </View>
          );
        })
      }
    </View>
  );
}

function Log(props) {
  var logs = props.logs || [];
  var onClear = props.onClear;
  var cmap = { danger: DANGER, warn: WARN, safe: SAFE, info: MUTED };
  return (
    <View style={{ flex: 1, paddingHorizontal: 14, paddingTop: 8 }}>
      <View style={{ flexDirection: 'row', justifyContent: 'space-between', margi
        <Text style={{ color: MUTED, fontSize: 11, textTransform: 'uppercase' }}>
        {logs.length > 0 &&
          <TouchableOpacity onPress={onClear}>
            <Text style={{ color: ACCENT, fontSize: 12 }}>Clear</Text>
          </TouchableOpacity>
        }
      </View>
      <ScrollView showsVerticalScrollIndicator={false}>
        {logs.length === 0 &&
          <Text style={{ color: MUTED, textAlign: 'center', marginTop: 10, fontSi
```

```
            No alerts yet
          </Text>
        }
        {logs.map(function(item) {
          return (
            <View key={item.id} style={[sty.logRow, { borderLeftColor: cmap[item.
              <Text style={{ flex: 1, color: TXTCOL, fontSize: 12 }} numberOfLine
                {item.text}
              </Text>
              <Text style={{ color: MUTED, fontSize: 10 }}>{item.time}</Text>
            </View>
          );
        })}
      </ScrollView>
    </View>
  );
}

function Settings(props) {
  var apiKey = props.apiKey;
  var setApiKey = props.setApiKey;
  var cfg = props.cfg;
  var setCfg = props.setCfg;
  var onBack = props.onBack;
  var inp = useState(apiKey);
  var inputKey = inp[0];
  var setInputKey = inp[1];

  function save() {
    if (inputKey.indexOf('sk-ant-') !== 0) {
      Alert.alert('Error', 'API Key must start with sk-ant-');
      return;
    }
    setApiKey(inputKey.trim());
    Alert.alert('Saved', 'API Key saved!');
  }

  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: BG }}>
      <ScrollView contentContainerStyle={{ padding: 20 }}>
        <View style={{ flexDirection: 'row', alignItems: 'center', marginBottom:
          <TouchableOpacity onPress={onBack} style={{ marginRight: 16 }}>
            <Text style={{ color: ACCENT, fontSize: 16 }}>Back</Text>
          </TouchableOpacity>
          <Text style={{ color: TXTCOL, fontSize: 20, fontWeight: '700' }}>Settin
        </View>
```

```
<View style={[sty.card, { marginBottom: 16 }]}>
  <Text style={sty.cardTitle}>Claude API Key</Text>
  <TextInput
    style={sty.input}
    value={inputKey}
    onChangeText={setInputKey}
    placeholder='sk-ant-api03-...'
    placeholderTextColor={MUTED}
    secureTextEntry={true}
    autoCapitalize='none'
    autoCorrect={false}
  />
  <Text style={{ color: MUTED, fontSize: 11, marginBottom: 8 }}>
    Get key at console.anthropic.com
  </Text>
  <TouchableOpacity style={sty.saveBtn} onPress={save}>
    <Text style={{ color: '#000', fontWeight: '700' }}>Save Key</Text>
  </TouchableOpacity>
</View>

<View style={[sty.card, { marginBottom: 16 }]}>
  <Text style={sty.cardTitle}>Voice and Vibration</Text>
  <View style={{ flexDirection: 'row', justifyContent: 'space-between', a
    <Text style={{ color: TXTCOL }}>Voice alerts</Text>
    <Switch value={cfg.speech}
      onValueChange={function(v) { setCfg(function(s) { return Object.ass
      trackColor={{ false: BORDER, true: ACCENT }} />
  </View>
  <View style={{ flexDirection: 'row', justifyContent: 'space-between', a
    <Text style={{ color: TXTCOL }}>Vibration</Text>
    <Switch value={cfg.vibrate}
      onValueChange={function(v) { setCfg(function(s) { return Object.ass
      trackColor={{ false: BORDER, true: ACCENT }} />
  </View>
  <TouchableOpacity
    style={[sty.saveBtn, { backgroundColor: SURFACE, borderWidth: 1, bord
    onPress={function() { sayIt('Test. EyeAI is ready to help you.'); }}>
    <Text style={{ color: ACCENT, fontWeight: '600' }}>Test Voice</Text>
  </TouchableOpacity>
</View>

<View style={sty.card}>
  <Text style={sty.cardTitle}>Analysis Speed</Text>
  <View style={{ flexDirection: 'row', gap: 8 }}>
    {[1500, 2000, 2500, 3000].map(function(ms) {
      var active = cfg.interval === ms;
      return (
                <TouchableOpacity key={ms}
                  style={[sty.chip2, active && { backgroundColor: ACCENT, borderC
                  onPress={function() { setCfg(function(s) { return Object.assign
                  <Text style={{ color: active ? '#000' : MUTED, fontWeight: '600
                    {ms / 1000}s
                  </Text>
                </TouchableOpacity>
              );
            })}
          </View>
        </View>
      </ScrollView>
    </SafeAreaView>
  );
}

function Camera(props) {
  var apiKey = props.apiKey;
  var cfg = props.cfg;
  var onSettings = props.onSettings;

  var permState = useCameraPermissions();
  var permission = permState[0];
  var requestPermission = permState[1];

  var camRef    = useRef(null);
  var timerRef  = useRef(null);
  var busyRef   = useRef(false);
  var lastRef   = useRef(0);

  var activeState   = useState(false);
  var isActive      = activeState[0];
  var setActive     = activeState[1];

  var analyzingState = useState(false);
  var isAnalyzing    = analyzingState[0];
  var setAnalyzing   = analyzingState[1];

  var objState   = useState([]);
  var objects    = objState[0];
  var setObjects = objState[1];

  var logState = useState([]);
  var logs     = logState[0];
  var setLogs  = logState[1];

  var topObj = objects.find(function(o) { return o.riskLevel === 'danger'; })
  || objects.find(function(o) { return o.riskLevel === 'warn'; })
  || null;

function addLog(text, level) {
  var d = new Date();
  var t = [d.getHours(), d.getMinutes(), d.getSeconds()]
    .map(function(n) { return String(n).padStart(2, '0'); }).join(':');
  setLogs(function(prev) {
    return [{ id: String(Date.now()), text: text, riskLevel: level, time: t }]
      .concat(prev.slice(0, 49));
  });
}

function doCapture() {
  if (!camRef.current || busyRef.current || !apiKey) return;
  busyRef.current = true;
  setAnalyzing(true);

  camRef.current.takePictureAsync({ base64: true, quality: 0.35, skipProcessing
    .then(function(photo) {
      if (!photo || !photo.base64) throw new Error('No photo');
      return callClaude(photo.base64, apiKey);
    })
    .then(function(result) {
      var objs = result.objects || [];
      setObjects(objs);
      var alerts = objs.filter(function(o) { return o.riskLevel !== 'safe'; });
      var now = Date.now();
      if (alerts.length > 0 && now - lastRef.current > cfg.interval * 1.5) {
        lastRef.current = now;
        var top = alerts.find(function(o) { return o.riskLevel === 'danger'; })
        if (cfg.speech)  sayIt(top.speechText);
        if (cfg.vibrate) buzz(top.riskLevel);
        alerts.forEach(function(o) { addLog(o.speechText, o.riskLevel); });
      } else if (result.overallSafe && now - lastRef.current > 15000) {
        lastRef.current = now;
        if (cfg.speech) sayIt('Path ahead is clear.');
        addLog('Path is clear', 'safe');
      }
    })
    .catch(function(e) {
      addLog('Error: check internet / API key', 'info');
    })
    .finally(function() {
      busyRef.current = false;
      setAnalyzing(false);
    });
```

```
}

useEffect(function() {
  if (isActive) {
    doCapture();
    timerRef.current = setInterval(doCapture, cfg.interval);
  } else {
    clearInterval(timerRef.current);
    setObjects([]);
  }
  return function() { clearInterval(timerRef.current); };
}, [isActive, cfg.interval, apiKey]);

function toggle() {
  if (!apiKey) {
    Alert.alert('No API Key', 'Go to Settings and enter your Claude API Key fir
      { text: 'Settings', onPress: onSettings }
    ]);
    return;
  }
  if (!isActive) sayIt('EyeAI activated.');
  else Speech.stop();
  setActive(function(v) { return !v; });
}

if (!permission) return <View style={{ flex: 1, backgroundColor: BG }} />;

if (!permission.granted) {
  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: BG, alignItems: 'center',
      <Text style={{ color: TXTCOL, fontSize: 17, textAlign: 'center', marginBo
        Camera permission required
      </Text>
      <TouchableOpacity style={sty.startBtn} onPress={requestPermission}>
        <Text style={{ color: '#000', fontWeight: '700', fontSize: 16 }}>Allow
      </TouchableOpacity>
    </SafeAreaView>
  );
}

return (
  <SafeAreaView style={{ flex: 1, backgroundColor: BG }}>
    <View style={{ flex: 1.5, backgroundColor: '#000' }}>
      <CameraView ref={camRef} style={StyleSheet.absoluteFill} facing='back' />
      <Overlay analyzing={isAnalyzing} objects={objects} />
      {isActive && <Banner obj={topObj} />}
      {!isActive &&
          <View style={{ ...StyleSheet.absoluteFillObject, backgroundColor: 'rgba
            alignItems: 'center', justifyContent: 'center' }}>
            <Text style={{ color: MUTED, fontSize: 15 }}>Tap START to activate AI
          </View>
        }
      </View>

      <View style={{ flex: 1, backgroundColor: SURFACE, borderTopWidth: 1, border
        <Log logs={logs} onClear={function() { setLogs([]); }} />
        <View style={sty.controls}>
          <TouchableOpacity style={[sty.mainBtn, isActive ? sty.stopBtn : sty.sta
            <Text style={{ color: '#000', fontWeight: '700', fontSize: 16 }}>
              {isActive ? 'STOP' : 'START AI'}
            </Text>
          </TouchableOpacity>
          <TouchableOpacity style={sty.iconBtn} onPress={onSettings}>
            <Text style={{ color: ACCENT, fontSize: 13, fontWeight: '600' }}>SET<
          </TouchableOpacity>
        </View>
      </View>
    </SafeAreaView>
  );
}

export default function App() {
  var scr    = useState('cam');
  var screen = scr[0];
  var setScreen = scr[1];

  var kst    = useState('');
  var apiKey = kst[0];
  var setApiKey = kst[1];

  var cfgst = useState({ speech: true, vibrate: true, interval: INTERVAL });
  var cfg   = cfgst[0];
  var setCfg = cfgst[1];

  return (
    <View style={{ flex: 1, backgroundColor: BG }}>
      <StatusBar barStyle='light-content' backgroundColor={BG} />
      {screen === 'cam'
        ? <Camera apiKey={apiKey} cfg={cfg} onSettings={function() { setScreen('s
        : <Settings apiKey={apiKey} setApiKey={setApiKey} cfg={cfg} setCfg={setCf
            onBack={function() { setScreen('cam'); }} />
      }
    </View>
  );
```

```
}

var sty = StyleSheet.create({
  corner: {
    position: 'absolute', width: 26, height: 26, borderColor: ACCENT,
  },
  chip: {
    position: 'absolute', bottom: 12, alignSelf: 'center',
    flexDirection: 'row', alignItems: 'center', gap: 6,
    backgroundColor: 'rgba(0,0,0,0.65)', borderRadius: 20,
    paddingHorizontal: 14, paddingVertical: 6,
    borderWidth: 1, borderColor: BORDER,
  },
  badge: {
    position: 'absolute', borderWidth: 1.5, borderRadius: 8,
    paddingHorizontal: 8, paddingVertical: 5,
    backgroundColor: 'rgba(0,0,0,0.75)', alignItems: 'center', minWidth: 68,
  },
  banner: {
    position: 'absolute', top: 18, alignSelf: 'center',
    paddingHorizontal: 20, paddingVertical: 10, borderRadius: 28, zIndex: 99,
    maxWidth: '88%',
  },
  bannerTxt: { color: '#fff', fontWeight: '700', fontSize: 14, textAlign: 'center
  logRow: {
    flexDirection: 'row', alignItems: 'center', gap: 8,
    backgroundColor: CARD, borderRadius: 10, marginBottom: 5,
    paddingHorizontal: 12, paddingVertical: 8, borderLeftWidth: 3,
  },
  controls: {
    flexDirection: 'row', padding: 12, gap: 10,
    borderTopWidth: 1, borderTopColor: BORDER,
  },
  mainBtn: {
    flex: 1, paddingVertical: 15, borderRadius: 18,
    alignItems: 'center', justifyContent: 'center',
  },
  startBtn: { backgroundColor: ACCENT },
  stopBtn:  { backgroundColor: DANGER },
  iconBtn: {
    width: 58, height: 52, borderRadius: 14,
    backgroundColor: CARD, borderWidth: 1, borderColor: BORDER,
    alignItems: 'center', justifyContent: 'center',
  },
  card: {
    backgroundColor: CARD, borderRadius: 16, padding: 16,
    borderWidth: 1, borderColor: BORDER,
  },
  cardTitle: { color: TXTCOL, fontSize: 15, fontWeight: '700', marginBottom: 12 }
  input: {
    backgroundColor: SURFACE, color: TXTCOL, borderRadius: 10,
    paddingHorizontal: 14, paddingVertical: 12, fontSize: 13,
    borderWidth: 1, borderColor: BORDER, marginBottom: 8,
  },
  saveBtn: {
    backgroundColor: ACCENT, borderRadius: 12, padding: 13, alignItems: 'center',
  },
  chip2: {
    flex: 1, padding: 10, borderRadius: 10,
    borderWidth: 1, borderColor: BORDER, alignItems: 'center', backgroundColor: S
  },
});
```