```
import React, {useCallback, useState} from 'react';
import {
SafeAreaView,
Text,
Image,
ScrollView,
StyleSheet,
TouchableOpacity,
View,
ActivityIndicator,
} from 'react-native';
import {createResizedImage} from
'rtn-imageresizer/js/ImageResizer';

type ResizedImage = {
path: string;
uri: string;
size: number;
name: string;
width: number;
height: number;
} | null;

const App: () => JSX.Element = () => {
const [image, setImage] = useState<ResizedImage>(null);
```

```
const [isProcessing, setIsProcessing] =
useState<boolean>(false);


const originalHeight = 1024;

const originalWidth = 1024;

const resizeHeight = 720;

const resizeWidth = 720;


const resizeImage = useCallback(async () => {

try {

setIsProcessing(true);

const response = await createResizedImage({

uri:
`https://images.unsplash.com/photo-1707327956851-30a531b7
0cda?q=80&w=${originalWidth}&h=${originalHeight}&auto=for
mat&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDF8MHxwaG90by
1wYWdlfHx8fGVufDB8fHx8fA%3D%3D`,

width: resizeWidth,

height: resizeHeight,

format: 'JPEG',

quality: 100,

rotation: 0,

outputPath: undefined,

keepMeta: false,

});

console.log(
```

```
`Original image dimention height : ${originalHeight} and width :
${originalWidth}`,

);

console.log(

`Resize image dimention height : ${resizeHeight} and width :
${resizeWidth}`,

);

console.log(

'Resized image response =>',

JSON.stringify(response, null, 2),

);

setImage(response);

} catch (error) {

console.log('error', error);

} finally {

setIsProcessing(false);

}

}, [originalHeight, originalWidth, resizeHeight, resizeWidth]);


const renderResizedImage = useCallback(() => {

if (!image) {

return null;

}


return (
```

```
<View style={styles.imageInfo}>
<Image source={{uri: image.uri}} style={styles.image} />
<Text
style={
styles.imageInfoText
}>{`Resized image response :\n${JSON.stringify(image, '', 2)}`}</Text>
</View>
);
}, [image]);

const renderButtons = useCallback(() => {
if (image) {
return (
<TouchableOpacity style={styles.button} onPress={() => setImage(null)}>
<Text style={styles.buttonTitle}> Clear </Text>
</TouchableOpacity>
);
}

return (
<TouchableOpacity
disabled={isProcessing}
style={styles.button}
```

```
onPress={resizeImage}>
{isProcessing ? (
<ActivityIndicator
size={'small'}
color={'white'}
style={styles.loader}
/>
) : null}
<Text style={styles.buttonTitle}>
{isProcessing ? 'Please wait...' : 'Resize Image'}
</Text>
</TouchableOpacity>
);
}, [image, isProcessing, resizeImage]);

return (
<SafeAreaView style={styles.safeArea}>
<View style={styles.container}>
<ScrollView contentContainerStyle={styles.contentContainer}>
{renderButtons()}
{renderResizedImage()}
</ScrollView>
</View>
</SafeAreaView>
```

```
);
};

const styles = StyleSheet.create({
safeArea: {
flex: 1,
},
container: {
flex: 1,
backgroundColor: 'white',
},
contentContainer: {
flexGrow: 1,
backgroundColor: 'white',
padding: 15,
},
imageInfo: {
flex: 1,
marginTop: 15,
},
imageInfoText: {
marginTop: 15,
color: 'black',
},
```

```
image: {
aspectRatio: 16 / 9,
height: undefined,
width: undefined,
borderRadius: 5,
},
loader: {
marginRight: 5,
},
button: {
padding: 10,
borderRadius: 5,
backgroundColor: 'black',
alignItems: 'center',
justifyContent: 'center',
flexDirection: 'row',
},
buttonTitle: {
color: 'white',
textAlign: 'center',
},
});
export default App;
```