Crotchet app starting code

// CrochetPatternApp.js - React Native Mobile App

import React, { useState, useEffect } from 'react';

import {

View,

Text,

ScrollView,

TouchableOpacity,

StyleSheet,

Picker,

Alert,

Modal

} from 'react-native';

// Import the pattern generator logic

import CrochetPatternGenerator from './CrochetPatternGenerator';

const CrochetPatternApp = () => {

const [selectedItem, setSelectedItem] = useState('hat');

const [selectedSize, setSelectedSize] = useState('adult\_medium');

const [selectedYarn, setSelectedYarn] = useState('worsted');

const [selectedStitch, setSelectedStitch] = useState('sc');

const [generatedPattern, setGeneratedPattern] = useState(null);

const [showPattern, setShowPattern] = useState(false);

const [showStitchGuide, setShowStitchGuide] = useState(false);

const [selectedStitchInfo, setSelectedStitchInfo] = useState(null);

const generator = new CrochetPatternGenerator();

const itemTypes = [

{ key: 'hat', label: 'Hat' },

{ key: 'scarf', label: 'Scarf' },

{ key: 'blanket', label: 'Blanket' }

];

const sizes = {

hat: [

{ key: 'baby', label: 'Baby (14")' },

{ key: 'child', label: 'Child (18")' },

{ key: 'adult\_small', label: 'Adult Small (20")' },

{ key: 'adult\_medium', label: 'Adult Medium (22")' },

{ key: 'adult\_large', label: 'Adult Large (24")' }

],

scarf: [

{ key: 'child', label: 'Child (6" x 40")' },

{ key: 'adult', label: 'Adult (8" x 60")' }

],

blanket: [

{ key: 'baby', label: 'Baby (30" x 36")' },

{ key: 'throw', label: 'Throw (50" x 60")' },

{ key: 'twin', label: 'Twin (66" x 90")' }

]

};

const yarnWeights = [

{ key: 'lace', label: 'Lace Weight (0)' },

{ key: 'sport', label: 'Sport Weight (2)' },

{ key: 'dk', label: 'DK Weight (3)' },

{ key: 'worsted', label: 'Worsted Weight (4)' },

{ key: 'chunky', label: 'Chunky Weight (5)' },

{ key: 'super\_chunky', label: 'Super Chunky Weight (6)' }

];

const stitches = [

{ key: 'sc', label: 'Single Crochet (SC)' },

{ key: 'hdc', label: 'Half Double Crochet (HDC)' },

{ key: 'dc', label: 'Double Crochet (DC)' }

];

const generatePattern = () => {

try {

let pattern;

if (selectedItem === 'hat') {

pattern = generator.generateHatPattern(selectedSize, selectedYarn, selectedStitch);

} else if (selectedItem === 'scarf') {

pattern = generator.generateScarfPattern(selectedSize, selectedYarn, selectedStitch);

}

setGeneratedPattern(pattern);

setShowPattern(true);

} catch (error) {

Alert.alert('Error', 'Failed to generate pattern. Please try again.');

}

};

const showStitchInstructions = (stitchKey) => {

const stitchData = {

'sc': {

name: 'Single Crochet',

instructions: 'Insert hook in stitch, yarn over, pull through (2 loops on hook), yarn over, pull through both loops.',

tips: 'Keep consistent tension. This creates the shortest, tightest stitch.'

},

'hdc': {

name: 'Half Double Crochet',

instructions: 'Yarn over, insert hook in stitch, yarn over, pull through (3 loops on hook), yarn over, pull through all 3 loops.',

tips: 'Medium height stitch, good for blankets and scarves.'

},

'dc': {

name: 'Double Crochet',

instructions: 'Yarn over, insert hook in stitch, yarn over, pull through (3 loops on hook), yarn over, pull through 2 loops (2 loops remain), yarn over, pull through last 2 loops.',

tips: 'Tallest basic stitch, works up quickly. Great for hats and blankets.'

}

};

setSelectedStitchInfo(stitchData[stitchKey]);

setShowStitchGuide(true);

};

return (

<View style={styles.container}>

<ScrollView style={styles.scrollView}>

<Text style={styles.title}>Crochet Pattern Generator</Text>

{/\* Item Selection \*/}

<View style={styles.section}>

<Text style={styles.sectionTitle}>What would you like to make?</Text>

<View style={styles.buttonRow}>

{itemTypes.map(item => (

<TouchableOpacity

key={item.key}

style={[

styles.selectionButton,

selectedItem === item.key && styles.selectedButton

]}

onPress={() => setSelectedItem(item.key)}

>

<Text style={[

styles.buttonText,

selectedItem === item.key && styles.selectedButtonText

]}>

{item.label}

</Text>

</TouchableOpacity>

))}

</View>

</View>

{/\* Size Selection \*/}

<View style={styles.section}>

<Text style={styles.sectionTitle}>Size</Text>

<View style={styles.pickerContainer}>

<Picker

selectedValue={selectedSize}

onValueChange={setSelectedSize}

style={styles.picker}

>

{sizes[selectedItem]?.map(size => (

<Picker.Item

key={size.key}

label={size.label}

value={size.key}

/>

))}

</Picker>

</View>

</View>

{/\* Yarn Weight Selection \*/}

<View style={styles.section}>

<Text style={styles.sectionTitle}>Yarn Weight</Text>

<View style={styles.pickerContainer}>

<Picker

selectedValue={selectedYarn}

onValueChange={setSelectedYarn}

style={styles.picker}

>

{yarnWeights.map(yarn => (

<Picker.Item

key={yarn.key}

label={yarn.label}

value={yarn.key}

/>

))}

</Picker>

</View>

</View>

{/\* Stitch Selection \*/}

<View style={styles.section}>

<Text style={styles.sectionTitle}>Primary Stitch</Text>

<View style={styles.buttonRow}>

{stitches.map(stitch => (

<TouchableOpacity

key={stitch.key}

style={[

styles.stitchButton,

selectedStitch === stitch.key && styles.selectedButton

]}

onPress={() => setSelectedStitch(stitch.key)}

>

<Text style={[

styles.buttonText,

selectedStitch === stitch.key && styles.selectedButtonText

]}>

{stitch.label}

</Text>

<TouchableOpacity

style={styles.infoButton}

onPress={() => showStitchInstructions(stitch.key)}

>

<Text style={styles.infoButtonText}>?</Text>

</TouchableOpacity>

</TouchableOpacity>

))}

</View>

</View>

{/\* Generate Button \*/}

<TouchableOpacity style={styles.generateButton} onPress={generatePattern}>

<Text style={styles.generateButtonText}>Generate Pattern</Text>

</TouchableOpacity>

</ScrollView>

{/\* Pattern Display Modal \*/}

<Modal

visible={showPattern}

animationType="slide"

onRequestClose={() => setShowPattern(false)}

>

<View style={styles.modalContainer}>

<View style={styles.modalHeader}>

<Text style={styles.modalTitle}>Your Pattern</Text>

<TouchableOpacity

style={styles.closeButton}

onPress={() => setShowPattern(false)}

>

<Text style={styles.closeButtonText}>✕</Text>

</TouchableOpacity>

</View>

<ScrollView style={styles.patternContent}>

{generatedPattern && generatedPattern.map((row, index) => (

<View key={index} style={styles.patternRow}>

{row.type === 'header' ? (

<View style={styles.headerSection}>

<Text style={styles.patternTitle}>{row.content.title}</Text>

<Text style={styles.subTitle}>Materials:</Text>

{row.content.materials.map((material, i) => (

<Text key={i} style={styles.materialText}>• {material}</Text>

))}

<Text style={styles.subTitle}>Abbreviations:</Text>

{row.content.abbreviations.map((abbrev, i) => (

<Text key={i} style={styles.materialText}>• {abbrev}</Text>

))}

</View>

) : (

<View style={styles.instructionRow}>

<Text style={styles.rowNumber}>

{row.row === 'Foundation' ? 'Foundation:' : `Row ${row.row}:`}

</Text>

<Text style={styles.instruction}>{row.instruction}</Text>

</View>

)}

</View>

))}

</ScrollView>

</View>

</Modal>

{/\* Stitch Guide Modal \*/}

<Modal

visible={showStitchGuide}

animationType="fade"

transparent={true}

onRequestClose={() => setShowStitchGuide(false)}

>

<View style={styles.modalOverlay}>

<View style={styles.stitchGuideModal}>

{selectedStitchInfo && (

<>

<Text style={styles.stitchGuideName}>{selectedStitchInfo.name}</Text>

<Text style={styles.stitchGuideInstructions}>

{selectedStitchInfo.instructions}

</Text>

<Text style={styles.stitchGuideTips}>

💡 {selectedStitchInfo.tips}

</Text>

<TouchableOpacity

style={styles.stitchGuideClose}

onPress={() => setShowStitchGuide(false)}

>

<Text style={styles.closeButtonText}>Got it!</Text>

</TouchableOpacity>

</>

)}

</View>

</View>

</Modal>

</View>

);

};

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#f8f9fa',

},

scrollView: {

flex: 1,

padding: 20,

},

title: {

fontSize: 28,

fontWeight: 'bold',

color: '#2c3e50',

textAlign: 'center',

marginBottom: 30,

marginTop: 20,

},

section: {

marginBottom: 25,

backgroundColor: 'white',

padding: 20,

borderRadius: 12,

shadowColor: '#000',

shadowOffset: { width: 0, height: 2 },

shadowOpacity: 0.1,

shadowRadius: 4,

elevation: 3,

},

sectionTitle: {

fontSize: 18,

fontWeight: '600',

color: '#34495e',

marginBottom: 15,

},

buttonRow: {

flexDirection: 'row',

flexWrap: 'wrap',

gap: 10,

},

selectionButton: {

backgroundColor: '#ecf0f1',

paddingHorizontal: 20,

paddingVertical: 12,

borderRadius: 25,

minWidth: 80,

alignItems: 'center',

},

stitchButton: {

backgroundColor: '#ecf0f1',

paddingHorizontal: 15,

paddingVertical: 12,

borderRadius: 8,

flexDirection: 'row',

alignItems: 'center',

flex: 1,

marginBottom: 10,

},

selectedButton: {

backgroundColor: '#3498db',

},

buttonText: {

fontSize: 16,

color: '#2c3e50',

fontWeight: '500',

flex: 1,

},

selectedButtonText: {

color: 'white',

},

infoButton: {

backgroundColor: '#95a5a6',

width: 24,

height: 24,

borderRadius: 12,

alignItems: 'center',

justifyContent: 'center',

marginLeft: 10,

},

infoButtonText: {

color: 'white',

fontSize: 12,

fontWeight: 'bold',

},

pickerContainer: {

backgroundColor: '#ecf0f1',

borderRadius: 8,

overflow: 'hidden',

},

picker: {

height: 50,

},

generateButton: {

backgroundColor: '#e74c3c',

paddingVertical: 18,

borderRadius: 30,

alignItems: 'center',

marginTop: 20,

marginBottom: 40,

shadowColor: '#e74c3c',

shadowOffset: { width: 0, height: 4 },

shadowOpacity: 0.3,

shadowRadius: 8,

elevation: 5,

},

generateButtonText: {

color: 'white',

fontSize: 20,

fontWeight: 'bold',

},

modalContainer: {

flex: 1,

backgroundColor: 'white',

},

modalHeader: {

flexDirection: 'row',

justifyContent: 'space-between',

alignItems: 'center',

padding: 20,

borderBottomWidth: 1,

borderBottomColor: '#ecf0f1',

},

modalTitle: {

fontSize: 24,

fontWeight: 'bold',

color: '#2c3e50',

},

closeButton: {

backgroundColor: '#95a5a6',

width: 30,

height: 30,

borderRadius: 15,

alignItems: 'center',

justifyContent: 'center',

},

closeButtonText: {

color: 'white',

fontSize: 16,

fontWeight: 'bold',

},

patternContent: {

flex: 1,

padding: 20,

},

headerSection: {

marginBottom: 20,

padding: 15,

backgroundColor: '#f8f9fa',

borderRadius: 8,

},

patternTitle: {

fontSize: 20,

fontWeight: 'bold',

color: '#2c3e50',

marginBottom: 15,

},

subTitle: {

fontSize: 16,

fontWeight: '600',

color: '#34495e',

marginTop: 10,

marginBottom: 5,

},

materialText: {

fontSize: 14,

color: '#7f8c8d',

marginLeft: 10,

marginBottom: 2,

},

instructionRow: {

marginBottom: 15,

padding: 15,

backgroundColor: 'white',

borderRadius: 8,

borderLeftWidth: 4,

borderLeftColor: '#3498db',

},

rowNumber: {

fontSize: 16,

fontWeight: 'bold',

color: '#2c3e50',

marginBottom: 5,

},

instruction: {

fontSize: 14,

color: '#34495e',

lineHeight: 20,

},

modalOverlay: {

flex: 1,

backgroundColor: 'rgba(0,0,0,0.5)',

justifyContent: 'center',

alignItems: 'center',

padding: 20,

},

stitchGuideModal: {

backgroundColor: 'white',

borderRadius: 12,

padding: 25,

maxWidth: 300,

alignItems: 'center',

},

stitchGuideName: {

fontSize: 20,

fontWeight: 'bold',

color: '#2c3e50',

marginBottom: 15,

textAlign: 'center',

},

stitchGuideInstructions: {

fontSize: 14,

color: '#34495e',

lineHeight: 20,

marginBottom: 15,

textAlign: 'center',

},

stitchGuideTips: {

fontSize: 12,

color: '#7f8c8d',

fontStyle: 'italic',

marginBottom: 20,

textAlign: 'center',

},

stit