**1. Implement a simple to-do list where users can add items to the list and delete them.**

import React, { useState } from 'react';

import { View, Text, TextInput, Button, FlatList, TouchableOpacity, StyleSheet } from 'react-native';

const TodoList = () => {

  const [todos, setTodos] = useState([]);

  const [text, setText] = useState('');

  const addTodo = () => {

    if (text.trim()) {

      setTodos([...todos, { id: Date.now().toString(), text }]);

      setText('');

    }

  };

  const deleteTodo = (id) => {

    setTodos(todos.filter(todo => todo.id !== id));

  };

  return (

    <View style={styles.container}>

      <TextInput

        style={styles.input}

        placeholder="Add a new task"

        value={text}

        onChangeText={setText}

      />

      <Button title="Add" onPress={addTodo} />

      <FlatList

        data={todos}

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

        renderItem={({ item }) => (

          <View style={styles.item}>

            <Text style={styles.itemText}>{item.text}</Text>

            <TouchableOpacity onPress={() => deleteTodo(item.id)} style={styles.deleteButton}>

              <Text style={styles.deleteButtonText}>Delete</Text>

            </TouchableOpacity>

          </View>

        )}

      />

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 16,

    backgroundColor: '#fff',

  },

  input: {

    height: 40,

    borderColor: 'gray',

    borderWidth: 1,

    marginBottom: 16,

    paddingLeft: 8,

  },

  item: {

    flexDirection: 'row',

    justifyContent: 'space-between',

    padding: 16,

    marginVertical: 8,

    borderWidth: 1,

    borderColor: '#ddd',

    borderRadius: 4,

    backgroundColor: '#f9f9f9',

  },

  itemText: {

    fontSize: 18,

  },

  deleteButton: {

    backgroundColor: '#ff6347',

    padding: 8,

    borderRadius: 4,

  },

  deleteButtonText: {

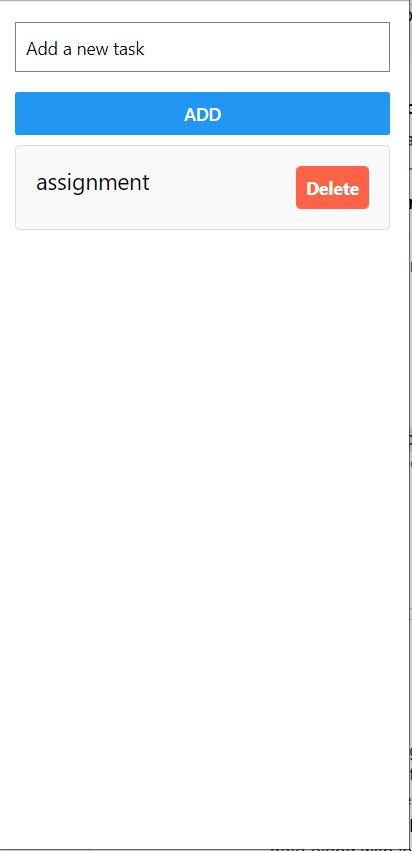
    color: '#fff',

    fontWeight: 'bold',

  },

});

export default TodoList;



**2.Create a component that fetches data from an API and displays it in a list.**

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

import { View, Text, FlatList, ActivityIndicator, StyleSheet } from 'react-native';

const DataList = () => {

  const [data, setData] = useState([]);

  const [loading, setLoading] = useState(true);

  useEffect(() => {

    const fetchData = async () => {

      try {

        const response = await fetch('https://jsonplaceholder.typicode.com/posts');

        const result = await response.json();

        setData(result);

        setLoading(false);

      } catch (error) {

        console.error(error);

        setLoading(false);

      }

    };

    fetchData();

  }, []);

  if (loading) {

    return (

      <View style={styles.container}>

        <ActivityIndicator size="large" color="#0000ff" />

      </View>

    );

  }

  return (

    <View style={styles.container}>

      <FlatList

        data={data}

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

        renderItem={({ item }) => (

          <View style={styles.item}>

            <Text style={styles.title}>{item.title}</Text>

            <Text style={styles.body}>{item.body}</Text>

          </View>

        )}

      />

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 16,

    backgroundColor: '#fff',

  },

  item: {

    padding: 16,

    marginVertical: 8,

    borderWidth: 1,

    borderColor: '#ddd',

    borderRadius: 4,

    backgroundColor: '#f9f9f9',

  },

  title: {

    fontSize: 18,

    fontWeight: 'bold',

  },

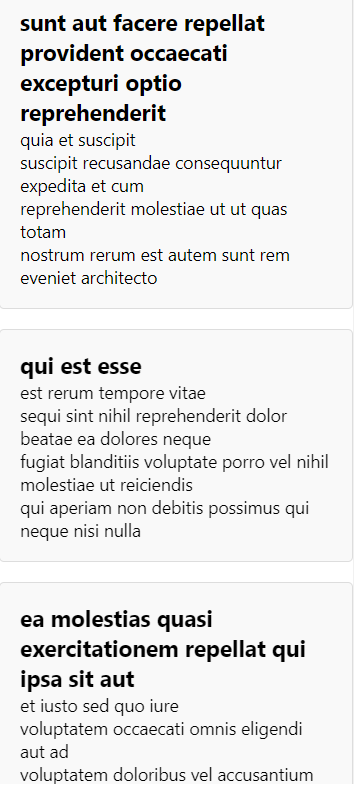
  body: {

    fontSize: 14,

  },

});

export default DataList;



3. Implement a navigation system with two screens. The first screen should have a button that navigates to the second screen, passing a parameter. The second screen should display the passed parameter.

App.js

import React from 'react';

import { StyleSheet, View } from 'react-native';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import FirstScreen from './FirstScreen';

import SecondScreen from './SecondScreen';

const Stack = createStackNavigator();

export default function App() {

  return (

    <NavigationContainer>

      <Stack.Navigator initialRouteName="FirstScreen">

        <Stack.Screen name="FirstScreen" component={FirstScreen} />

        <Stack.Screen name="SecondScreen" component={SecondScreen} />

      </Stack.Navigator>

    </NavigationContainer>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    alignItems: 'center',

  },

});

FirstScreen.js

import React from 'react';

import { View, Button, StyleSheet } from 'react-native';

const FirstScreen = ({ navigation }) => {

  return (

    <View style={styles.container}>

      <Button

        title="Go to Second Screen"

        onPress={() => navigation.navigate('SecondScreen', { message: 'Hello from First Screen!' })}

      />

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    alignItems: 'center',

  },

});

export default FirstScreen;

SecondScreen.js

import React from 'react';

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

const SecondScreen = ({ route }) => {

  const { message } = route.params;

  return (

    <View style={styles.container}>

      <Text style={styles.messageText}>{message}</Text>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    alignItems: 'center',

  },

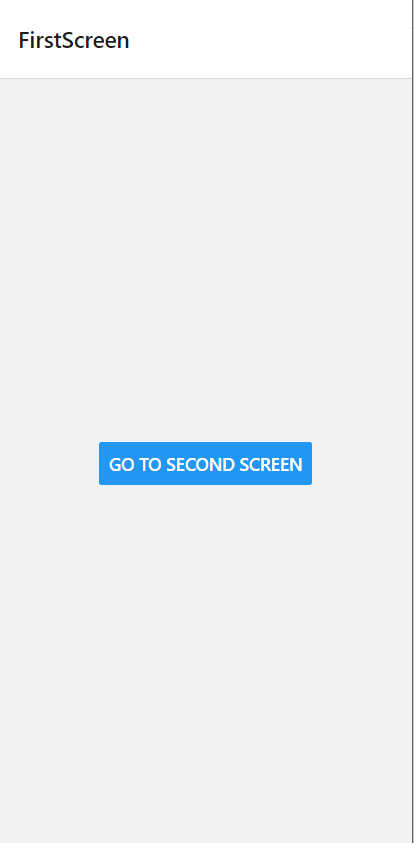
  messageText: {

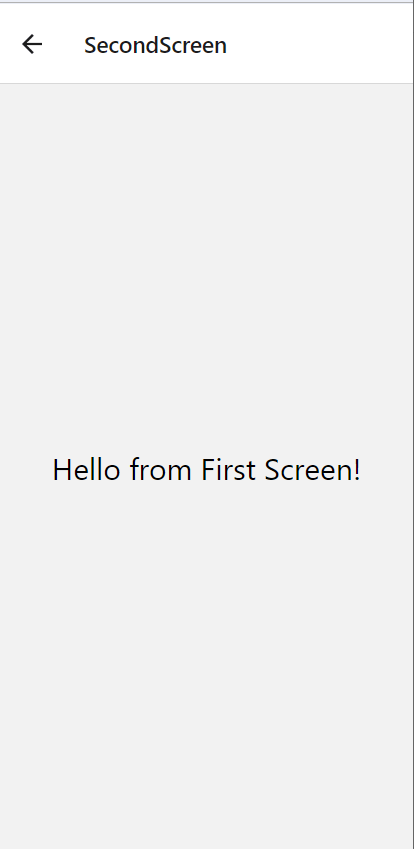
    fontSize: 24,

  },

});

export default SecondScreen;





**4. Create a custom hook useFetch that fetches data from a given URL and returns the data along with loading and error states. Use this hook in a component to display data.**

App.js

import React from 'react';

import { SafeAreaView, StyleSheet } from 'react-native';

import DataDisplay from './DataDisplay'; *// Adjust the path as necessary*

const App = () => {

  return (

    <SafeAreaView style={styles.container}>

      <DataDisplay />

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

  },

});

export default App;

DataDisplay.js

import React from 'react';

import { View, Text, StyleSheet, ActivityIndicator } from 'react-native';

import useFetch from './useFetch'; *// Adjust the path as necessary*

const DataDisplay = () => {

  const { data, loading, error } = useFetch('https://jsonplaceholder.typicode.com/posts'); *// Test with public API*

  if (loading) {

    return (

      <View style={styles.container}>

        <ActivityIndicator size="large" color="#0000ff" />

        <Text>Loading...</Text>

      </View>

    );

  }

  if (error) {

    return (

      <View style={styles.container}>

        <Text>Error: {error}</Text>

      </View>

    );

  }

  return (

    <View style={styles.container}>

      <Text>Data:</Text>

      {data.map((item) => (

        <Text key={item.id}>{item.title}</Text>

      ))}

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    alignItems: 'center',

    padding: 16,

  },

});

export default DataDisplay;

useFetch.js

import { useState, useEffect } from 'react';

const useFetch = (url) => {

  const [data, setData] = useState(null);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  useEffect(() => {

    const fetchData = async () => {

      setLoading(true);

      setError(null);

      try {

        const response = await fetch(url);

        if (!response.ok) {

          throw new Error(`HTTP error! status: ${response.status}`);

        }

        const result = await response.json();

        setData(result);

      } catch (err) {

        console.error("Fetch error: ", err); *// Log error details*

        setError(err.message);

      } finally {

        setLoading(false);

      }

    };

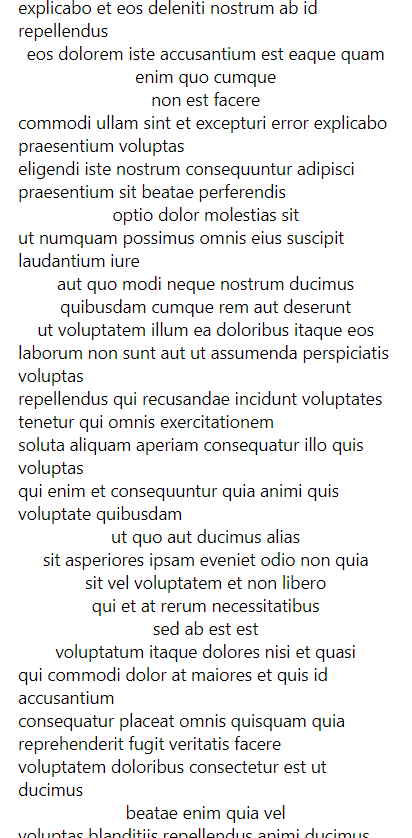
    fetchData();

  }, [url]);

  return { data, loading, error };

};

export default useFetch;



**5.  Implement a component that connects to a WebSocket server, sends a message, and displays messages received from the server.**

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

import { View, Text, Button, StyleSheet, FlatList, TextInput, Alert } from 'react-native';

*// Replace with your WebSocket server URL*

const WS\_SERVER\_URL = 'ws://localhost:8080';

export default function App() {

  const [messages, setMessages] = useState([]);

  const [inputMessage, setInputMessage] = useState('');

  const ws = useRef(null);

  useEffect(() => {

    const connectWebSocket = () => {

      ws.current = new WebSocket(WS\_SERVER\_URL);

      ws.current.onopen = () => {

        console.log('WebSocket connection opened');

      };

      ws.current.onmessage = (event) => {

        const newMessage = event.data;

        setMessages((prevMessages) => [...prevMessages, newMessage]);

      };

      ws.current.onerror = (error) => {

        console.error('WebSocket error:', error);

        Alert.alert('Error', 'WebSocket error occurred');

      };

      ws.current.onclose = () => {

        console.log('WebSocket connection closed');

*// Attempt to reconnect after a delay*

        setTimeout(connectWebSocket, 3000);

      };

    };

    connectWebSocket();

    return () => {

      if (ws.current) {

        ws.current.close();

      }

    };

  }, []);

  const sendMessage = () => {

    if (ws.current && ws.current.readyState === WebSocket.OPEN) {

      ws.current.send(inputMessage);

      setInputMessage('');

    } else {

      Alert.alert('Error', 'WebSocket is not connected.');

    }

  };

  return (

    <View style={styles.container}>

      <FlatList

        data={messages}

        renderItem={({ item }) => <Text style={styles.message}>{item}</Text>}

        keyExtractor={(item, index) => index.toString()}

        style={styles.list}

      />

      <View style={styles.inputContainer}>

        <TextInput

          style={styles.input}

          value={inputMessage}

          onChangeText={setInputMessage}

          placeholder="Type a message"

        />

        <Button title="Send" onPress={sendMessage} />

      </View>

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 20,

    justifyContent: 'center',

  },

  list: {

    flex: 1,

    marginBottom: 10,

  },

  message: {

    padding: 10,

    backgroundColor: '#f0f0f0',

    marginBottom: 5,

  },

  inputContainer: {

    flexDirection: 'row',

    alignItems: 'center',

  },

  input: {

    flex: 1,

    borderWidth: 1,

    borderColor: '#ccc',

    padding: 10,

    marginRight: 10,

  },

});



**6. Create a component that animates a box moving from the top to the bottom of the screen using Animated API.**

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

import { View, Animated, StyleSheet, Dimensions } from 'react-native';

export default function App() {

  const screenHeight = Dimensions.get('window').height;

  const translateY = useRef(new Animated.Value(0)).current;

  useEffect(() => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(translateY, {

          toValue: screenHeight - 100, *// Adjust the value to fit the box height*

          duration: 2000, *// Duration of the animation*

          useNativeDriver: true,

        }),

        Animated.timing(translateY, {

          toValue: 0,

          duration: 2000,

          useNativeDriver: true,

        }),

      ])

    ).start();

  }, [screenHeight, translateY]);

  return (

    <View style={styles.container}>

      <Animated.View style={[styles.box, { transform: [{ translateY }] }]} />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    alignItems: 'center',

    backgroundColor: '#fff',

  },

  box: {

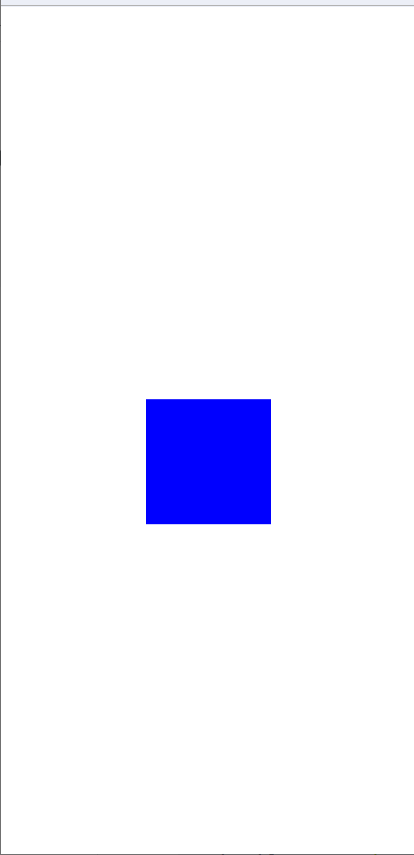
    width: 100,

    height: 100,

    backgroundColor: 'blue',

  },

});



**7. Create a component that allows users to save a piece of text locally using AsyncStorage, and retrieve it when the component is mounted**

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

import { View, Text, Button, StyleSheet, TextInput, Alert } from 'react-native';

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

const STORAGE\_KEY = 'savedText'; *// Key for AsyncStorage*

export default function App() {

  const [inputText, setInputText] = useState('');

  const [storedText, setStoredText] = useState('');

  useEffect(() => {

*// Fetch the stored text when the component mounts*

    const fetchStoredText = async () => {

      try {

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

        if (text !== null) {

          setStoredText(text);

        }

      } catch (error) {

        Alert.alert('Error', 'Failed to load saved text');

      }

    };

    fetchStoredText();

  }, []);

  const saveText = async () => {

    try {

      await AsyncStorage.setItem(STORAGE\_KEY, inputText);

      setStoredText(inputText);

      setInputText('');

      Alert.alert('Success', 'Text saved successfully');

    } catch (error) {

      Alert.alert('Error', 'Failed to save text');

    }

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>AsyncStorage Example</Text>

      <TextInput

        style={styles.input}

        value={inputText}

        onChangeText={setInputText}

        placeholder="Type something..."

      />

      <Button title="Save Text" onPress={saveText} />

      <Text style={styles.storedText}>Stored Text: {storedText}</Text>

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 20,

    justifyContent: 'center',

  },

  title: {

    fontSize: 24,

    marginBottom: 20,

  },

  input: {

    borderWidth: 1,

    borderColor: '#ccc',

    padding: 10,

    marginBottom: 20,

  },

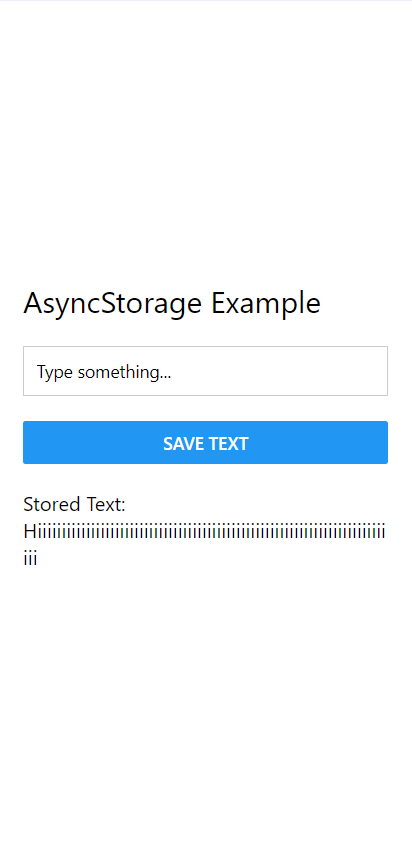
  storedText: {

    marginTop: 20,

    fontSize: 16,

  },

});



**8. Use the Context API to create a global state that keeps track of the user's theme preference (light or dark) and allows toggling between them.**

import React, { createContext, useState, useContext } from 'react';

import { View, Text, Button, StyleSheet, SafeAreaView } from 'react-native';

// Create the Context

const ThemeContext = createContext();

// Create a custom hook to use the ThemeContext

const useTheme = () => useContext(ThemeContext);

// Create a Provider component

const ThemeProvider = ({ children }) => {

const [theme, setTheme] = useState('light'); // Default to light theme

// Toggle theme between light and dark

const toggleTheme = () => {

setTheme((prevTheme) => (prevTheme === 'light' ? 'dark' : 'light'));

};

return (

<ThemeContext.Provider value={{ theme, toggleTheme }}>

{children}

</ThemeContext.Provider>

);

};

// Component to toggle theme

const ThemeToggleButton = () => {

const { theme, toggleTheme } = useTheme();

return (

<Button

title={`Switch to ${theme === 'light' ? 'dark' : 'light'} mode`}

onPress={toggleTheme}

/>

);

};

// Main App component

const App = () => {

const { theme } = useTheme();

return (

<SafeAreaView style={styles[theme].container}>

<Text style={styles[theme].text}>Hello, World!</Text>

<ThemeToggleButton />

</SafeAreaView>

);

};

// Define styles for light and dark themes

const styles = StyleSheet.create({

light: {

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: 'white',

},

text: {

color: 'black',

fontSize: 20,

},

},

dark: {

container: {

flex: 1,

justifyContent: 'center',

alignItems: 'center',

backgroundColor: 'black',

},

text: {

color: 'white',

fontSize: 20,

},

},

});

// Render the application

export default function Main() {

return (

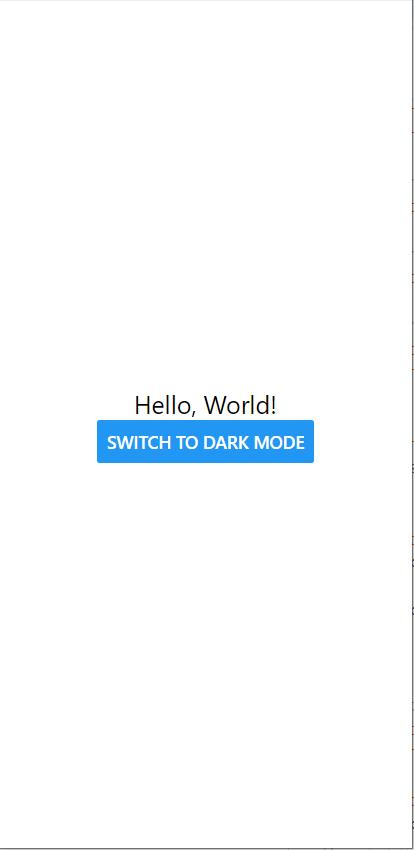
<ThemeProvider>

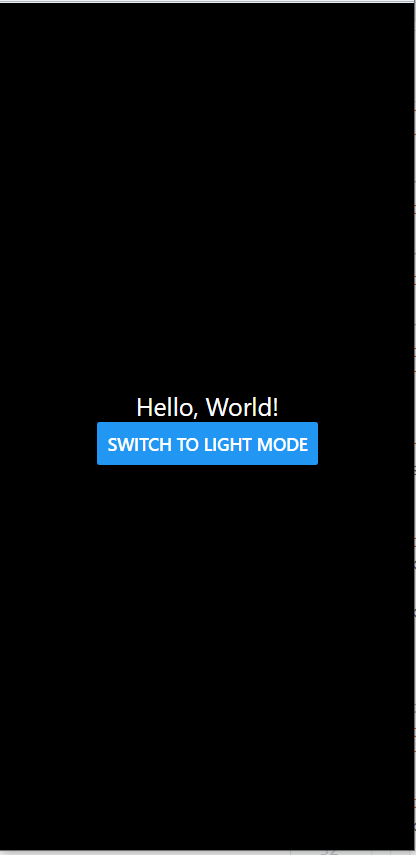
<App />

</ThemeProvider>

);

}





**9. Create a component that allows users to pick an image from their device's gallery and display the selected image.**

import React, { useState } from 'react';

import { View, Text, Image, Button, StyleSheet, Alert } from 'react-native';

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

export default function App() {

  const [imageUri, setImageUri] = useState(null);

  const pickImage = async () => {

*// Request permission to access the gallery*

    const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();

    if (status !== 'granted') {

      Alert.alert('Permission required', 'Sorry, we need camera roll permissions to make this work!');

      return;

    }

*// Open image picker and let the user select an image*

    const result = await ImagePicker.launchImageLibraryAsync({

      mediaTypes: ImagePicker.MediaTypeOptions.Images,

      allowsEditing: false,

      aspect: [4, 3],

      quality: 1,

    });

    if (!result.canceled) {

      setImageUri(result.assets[0].uri); *// Set the selected image URI*

    }

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Image Picker Example</Text>

      {imageUri ? (

        <Image source={{ uri: imageUri }} style={styles.image} />

      ) : (

        <Text style={styles.placeholder}>No image selected</Text>

      )}

      <Button title="Pick an image from gallery" onPress={pickImage} />

    </View>

  );

}

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 20,

    justifyContent: 'center',

    alignItems: 'center',

  },

  title: {

    fontSize: 24,

    marginBottom: 20,

  },

  image: {

    width: 300,

    height: 300,

    marginBottom: 20,

    borderRadius: 10,

  },

  placeholder: {

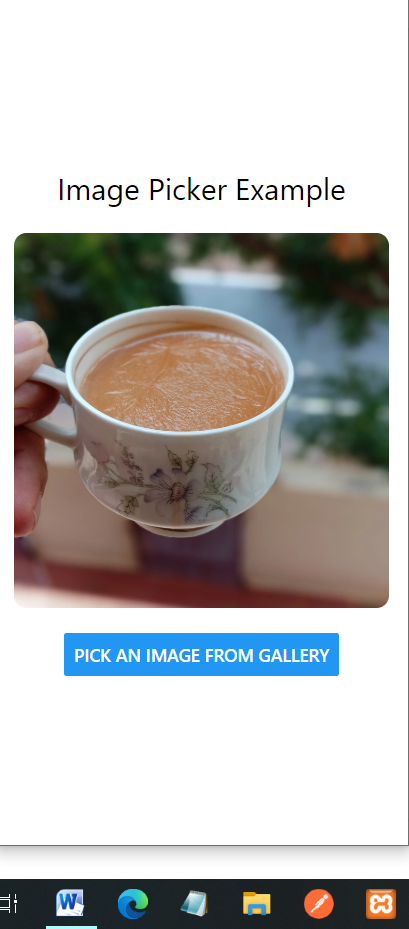
    fontSize: 16,

    color: 'gray',

    marginBottom: 20,

  },

});



**10. Create a form with two fields: name and email. Validate that the email is in a correct format before submitting.**

import React, { useState } from 'react';

import { View, Text, TextInput, TouchableOpacity, StyleSheet, SafeAreaView, Alert } from 'react-native';

*// Validate email format using a regular expression*

const isValidEmail = (email) => {

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

  return emailRegex.test(email);

};

const App = () => {

  const [name, setName] = useState('');

  const [email, setEmail] = useState('');

*// Handle form submission*

  const handleSubmit = () => {

    if (name.trim() === '' || email.trim() === '') {

      Alert.alert('Error', 'All fields are required');

      return;

    }

    if (!isValidEmail(email)) {

      Alert.alert('Error', 'Please enter a valid email address');

      return;

    }

    Alert.alert('Success', 'Form submitted successfully');

*// Reset form fields*

    setName('');

    setEmail('');

  };

  return (

    <SafeAreaView style={styles.container}>

      <Text style={styles.title}>Registration Form</Text>

      <Text style={styles.label}>Name:</Text>

      <TextInput

        style={styles.input}

        value={name}

        onChangeText={setName}

        placeholder="Enter your name"

        placeholderTextColor="#888"

      />

      <Text style={styles.label}>Email:</Text>

      <TextInput

        style={styles.input}

        value={email}

        onChangeText={setEmail}

        placeholder="Enter your email"

        placeholderTextColor="#888"

        keyboardType="email-address"

      />

      <TouchableOpacity style={styles.button} onPress={handleSubmit}>

        <Text style={styles.buttonText}>Submit</Text>

      </TouchableOpacity>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    justifyContent: 'center',

    padding: 20,

    backgroundColor: '#f5f5f5',

  },

  title: {

    fontSize: 24,

    fontWeight: 'bold',

    marginBottom: 20,

    textAlign: 'center',

    color: '#333',

  },

  label: {

    fontSize: 18,

    marginBottom: 8,

    color: '#333',

  },

  input: {

    height: 40,

    borderColor: '#ccc',

    borderWidth: 1,

    marginBottom: 20,

    paddingHorizontal: 10,

    borderRadius: 5,

    backgroundColor: '#fff',

  },

  button: {

    backgroundColor: '#007BFF',

    paddingVertical: 10,

    borderRadius: 5,

    alignItems: 'center',

  },

  buttonText: {

    color: '#fff',

    fontSize: 18,

    fontWeight: 'bold',

  },

});

export default App;

