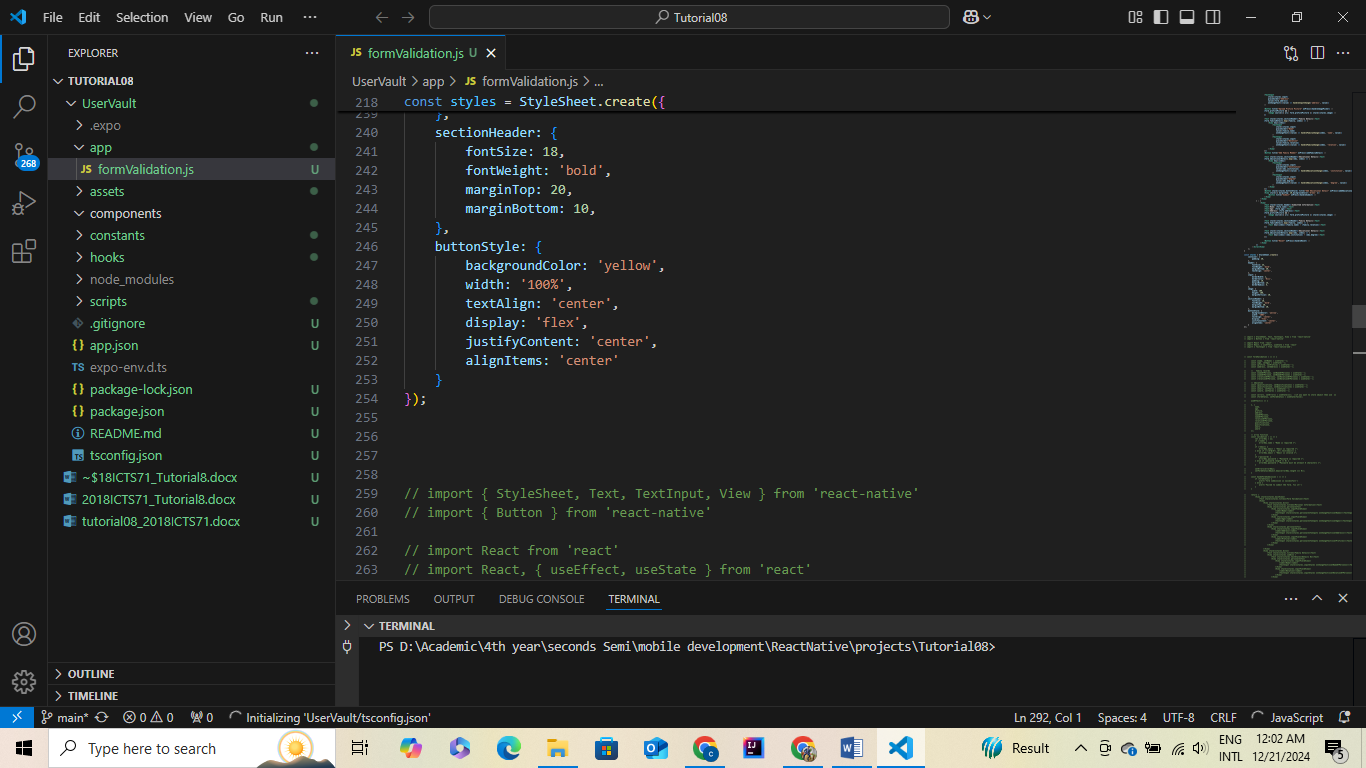
**Code Explanation**

The provided code is a simple **React Native** application that form validation of input values. It consists of a functional component called FormValidation, which manages user inputs and validate and handles input validation to ensure that the fields are not empty and satisfy the requirement.

**1. Functional Components**

In React, a functional component is a JavaScript function that returns a React element. It can accept props as an argument and use hooks for state management. In this project, FormValidation is a functional component.

Step1



Coding

 // UserVault App Implementation

import React, { useState } from 'react';

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

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

export default function UserVault() {

    const [form, setForm] = useState({

        name: '',

        age: '',

        address: '',

        profilePicture: null,

        familyDetails: [{ name: '', relation: '' }],

        educationalDetails: [{ institution: '', degree: '' }],

    });

    const [submitted, setSubmitted] = useState(false);

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

    // Input Handlers

    const handleInputChange = (field, value) => {

        setForm({ ...form, [field]: value });

    };

    const handleFamilyChange = (index, field, value) => {

        const updatedFamily = [...form.familyDetails];

        updatedFamily[index][field] = value;

        setForm({ ...form, familyDetails: updatedFamily });

    };

    const handleEducationChange = (index, field, value) => {

        const updatedEducation = [...form.educationalDetails];

        updatedEducation[index][field] = value;

        setForm({ ...form, educationalDetails: updatedEducation });

    };

    // Add or Remove Family/Education Details

    const addFamilyDetail = () => {

        setForm({ ...form, familyDetails: [...form.familyDetails, { name: '', relation: '' }] });

    };

    const addEducationalDetail = () => {

        setForm({

            ...form,

            educationalDetails: [...form.educationalDetails, { institution: '', degree: '' }],

        });

    };

    const handleImagePicker = async () => {

        const result = await ImagePicker.launchImageLibraryAsync({

            mediaTypes: ImagePicker.MediaTypeOptions.Images,

            allowsEditing: true,

            quality: 1,

        });

        if (!result.canceled) {

            setForm({ ...form, profilePicture: result.uri });

        }

    };

    // Form Validation

    const validateForm = () => {

        const { name, age, address, profilePicture, familyDetails, educationalDetails } = form;

        if (!name || /[^a-zA-Z\s]/.test(name)) {

            Alert.alert('Validation Error', 'Name is required and must contain only letters.');

            return false;

        }

        if (!age || !Number.isInteger(Number(age)) || age < 1 || age > 120) {

            Alert.alert('Validation Error', 'Age must be a positive integer between 1 and 120.');

            return false;

        }

        if (!address || address.length < 10) {

            Alert.alert('Validation Error', 'Address is required and must be at least 10 characters long.');

            return false;

        }

        if (!profilePicture) {

            Alert.alert('Validation Error', 'Please upload or select a valid profile picture.');

            return false;

        }

        if (!familyDetails.length || familyDetails.some((f) => !f.name || !f.relation)) {

            Alert.alert('Validation Error', 'Please include at least one family member with a name and relation.');

            return false;

        }

        if (!educationalDetails.length || educationalDetails.some((e) => !e.institution || !e.degree)) {

            Alert.alert('Validation Error', 'Please include at least one educational qualification.');

            return false;

        }

        return true;

    };

    // Handle Form Submission

    const handleSubmit = () => {

        if (validateForm()) {

            setSubmitted(true);

        }

    };

    // Reset Form

    const handleReset = () => {

        setForm({

            name: '',

            age: '',

            address: '',

            profilePicture: null,

            familyDetails: [{ name: '', relation: '' }],

            educationalDetails: [{ institution: '', degree: '' }],

        });

        setSubmitted(false);

    };

    // Render Form or Summary Screen

    return (

        <ScrollView style={styles.container}>

            {!submitted ? (

                <View>

                    <Text style={styles.header}>FormValidation</Text>

                    <TextInput

                        style={styles.input}

                        placeholder="Name"

                        value={form.name}

                        onChangeText={(value) => handleInputChange('name', value)}

                    />

                    <TextInput

                        style={styles.input}

                        placeholder="Age"

                        keyboardType="numeric"

                        value={form.age}

                        onChangeText={(value) => handleInputChange('age', value)}

                    />

                    <TextInput

                        style={styles.input}

                        placeholder="Address"

                        value={form.address}

                        onChangeText={(value) => handleInputChange('address', value)}

                    />

                    <Button title="Upload Profile Picture" onPress={handleImagePicker} />

                    {form.profilePicture && (

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

                    )}

                    <Text style={styles.sectionHeader}>Family Details</Text>

                    {form.familyDetails.map((family, index) => (

                        <View key={index}>

                            <TextInput

                                style={styles.input}

                                placeholder="Name"

                                value={family.name}

                                onChangeText={(value) => handleFamilyChange(index, 'name', value)}

                            />

                            <TextInput

                                style={styles.input}

                                placeholder="Relation"

                                value={family.relation}

                                onChangeText={(value) => handleFamilyChange(index, 'relation', value)}

                            />

                        </View>

                    ))}

                    <Button title="Add Family Member" onPress={addFamilyDetail} />

                    <Text style={styles.sectionHeader}>Educational Details</Text>

                    {form.educationalDetails.map((edu, index) => (

                        <View key={index}>

                            <TextInput

                                style={styles.input}

                                placeholder="Institution"

                                value={edu.institution}

                                onChangeText={(value) => handleEducationChange(index, 'institution', value)}

                            />

                            <TextInput

                                style={styles.input}

                                placeholder="Degree"

                                value={edu.degree}

                                onChangeText={(value) => handleEducationChange(index, 'degree', value)}

                            />

                        </View>

                    ))}

                    <Button style={styles.buttonStyle} title="Add Educational Detail" onPress={addEducationalDetail} />

                    <View style={{ marginTop: 20,backgroundColor:'Green' }}>

                        <Button title="Submit" onPress={handleSubmit} />

                    </View>

                </View>

            ) : (

                <View>

                    <Text style={styles.header}>Submitted Information</Text>

                    <Text>Name: {form.name}</Text>

                    <Text>Age: {form.age}</Text>

                    <Text>Address: {form.address}</Text>

                    {form.profilePicture && (

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

                    )}

                    <Text style={styles.sectionHeader}>Family Details</Text>

                    {form.familyDetails.map((family, index) => (

                        <Text key={index}>{family.name} - {family.relation}</Text>

                    ))}

                    <Text style={styles.sectionHeader}>Educational Details</Text>

                    {form.educationalDetails.map((edu, index) => (

                        <Text key={index}>{edu.institution} - {edu.degree}</Text>

                    ))}

                    <Button title="Reset" onPress={handleReset} />

                </View>

            )}

        </ScrollView>

    );

}

const styles = StyleSheet.create({

    container: {

        padding: 20,

    },

    header: {

        fontSize: 24,

        fontWeight: 'bold',

        marginBottom: 20,

        textAlign: 'center',

    },

    input: {

        borderWidth: 1,

        borderColor: '#ccc',

        padding: 10,

        marginBottom: 15,

        borderRadius: 5,

    },

    image: {

        width: 100,

        height: 100,

        marginVertical: 10,

    },

    sectionHeader: {

        fontSize: 18,

        fontWeight: 'bold',

        marginTop: 20,

        marginBottom: 10,

    },

    buttonStyle: {

        backgroundColor: 'yellow',

        width: '100%',

        textAlign: 'center',

        display: 'flex',

        justifyContent: 'center',

        alignItems: 'center'

    }

});

