📘 Technical Documentation - Developer Guide

Project: *Parcial*

Name developer: Santiago Gallego Garcia
Platform: React Native (Expo)
Language: JavaScript
Backend: Firebase Authentication
State Management & Storage: AsyncStorage
Navigation: React Navigation

🛠 Requirements

Software

- Node.js (v18 or higher)
- Expo CLI (npm install -g expo-cli)
- Git
- Visual Studio Code (recommended)

Dependencies

Install all project dependencies with:

npm install

🚀 Getting Started

1. **Clone the Repository**

git clone https://github.com/SantiagoG/Parcial.git

cd parcial

2. **Install Dependencies**

npm install

3. **Run the App**

npx expo start

🔐 Firebase Configuration

Firebase is used for user authentication.

Create a firebase.js file in the root or config/ folder:

```
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';

const firebaseConfig = {
  apiKey: "AIzaSyAX8gO1W5lcb1iz1dfJvDESWkiUXmVZKWY",
  authDomain: "parcial-f8bd3.firebaseapp.com",
  projectId: "parcial-f8bd3",
  storageBucket: "parcial-f8bd3.firebasestorage.app",
  messagingSenderId: "268127084597",
  appId: "1:268127084597:web:a940ff699c6e1ec46cd17c",
  measurementId: "G-REXT12KFQZ"
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
```

📱 App Structure

```
📁 screens/
  └── LoginScreen.js
  └── RegisterScreen.js
  └── HomeScreen.js
📁 components/
  └── TaskItem.js
📁 config/
  └── firebase.js
App.js
```

🧭 Navigation

Implemented with @react-navigation/native and @react-navigation/native-stack.

```
<NavigationContainer>
  <Stack.Navigator>
    <Stack.Screen name="Login" component={LoginScreen} />
    <Stack.Screen name="Register" component={RegisterScreen} />
    <Stack.Screen name="Home" component={HomeScreen} />
<Stack.Screen name="AddTask" component={AddTaskScreen} />
  </Stack.Navigator>
</NavigationContainer>
```

💾 AsyncStorage

Used for local persistence of tasks or user state when needed.

```
await AsyncStorage.setItem('tasks', JSON.stringify(taskList));
```

✨ Flash Messages

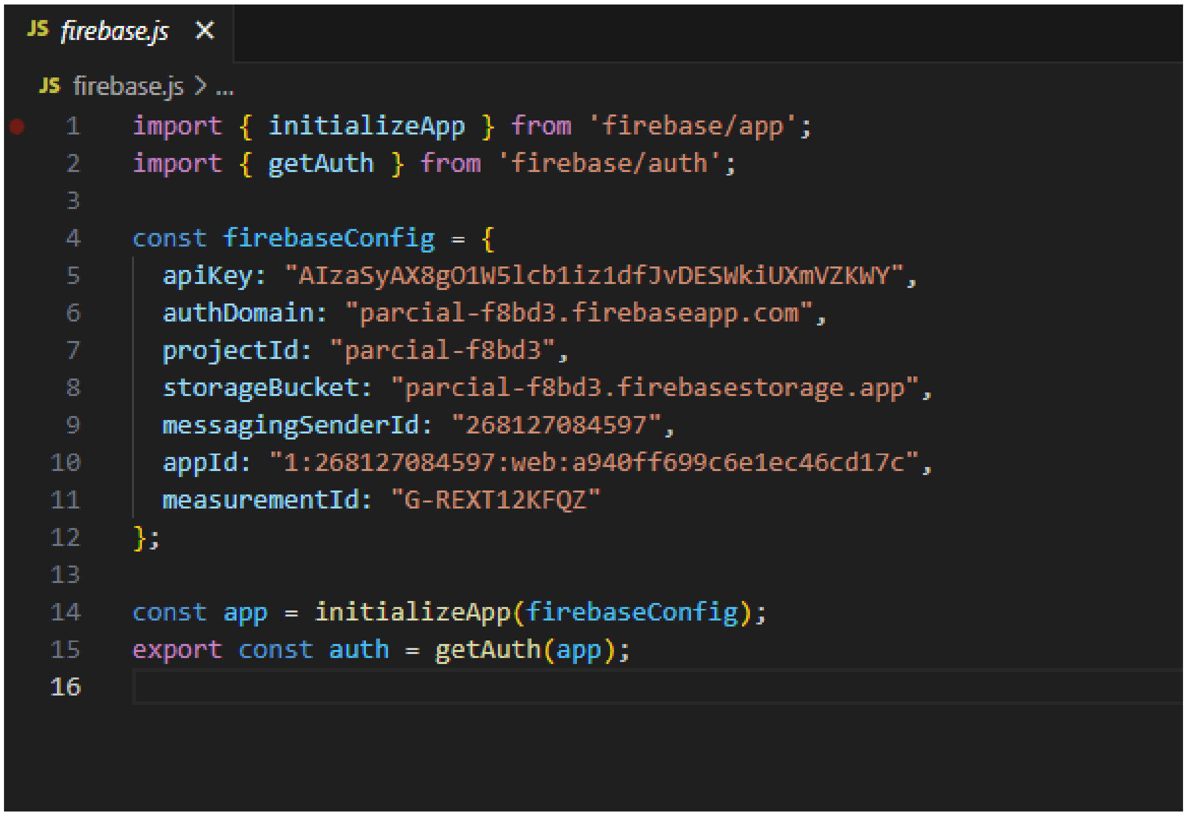
We use react-native-flash-message for better user feedback:

```
import FlashMessage, { showMessage } from "react-native-flash-message";
showMessage({
  message: "Login successful",
  type: "success",
});
```

🐛 Debugging & Troubleshooting

- **Port conflict:** If Expo warns about port 8081, press Y to switch to a new port.
- **Firebase import errors:** Make sure the firebase.js file is in the correct location and properly exported.

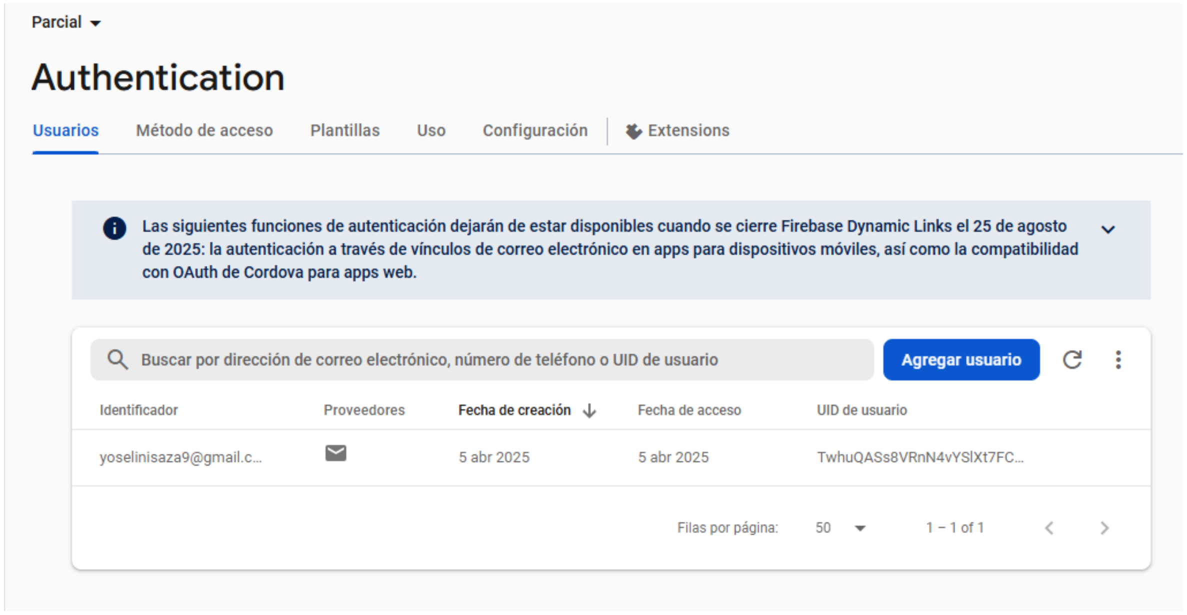
📷 Screenshots



firebase.js

firebase.js > ...

```javascript
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';

const firebaseConfig = {
  apiKey: "AIzaSyAX8gO1W5lcb1iz1dfJvDESWkiUXmVZKWY",
  authDomain: "parcial-f8bd3.firebaseapp.com",
  projectId: "parcial-f8bd3",
  storageBucket: "parcial-f8bd3.firebasestorage.app",
  messagingSenderId: "268127084597",
  appId: "1:268127084597:web:a940ff699c6e1ec46cd17c",
  measurementId: "G-REXT12KFQZ"
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);

```

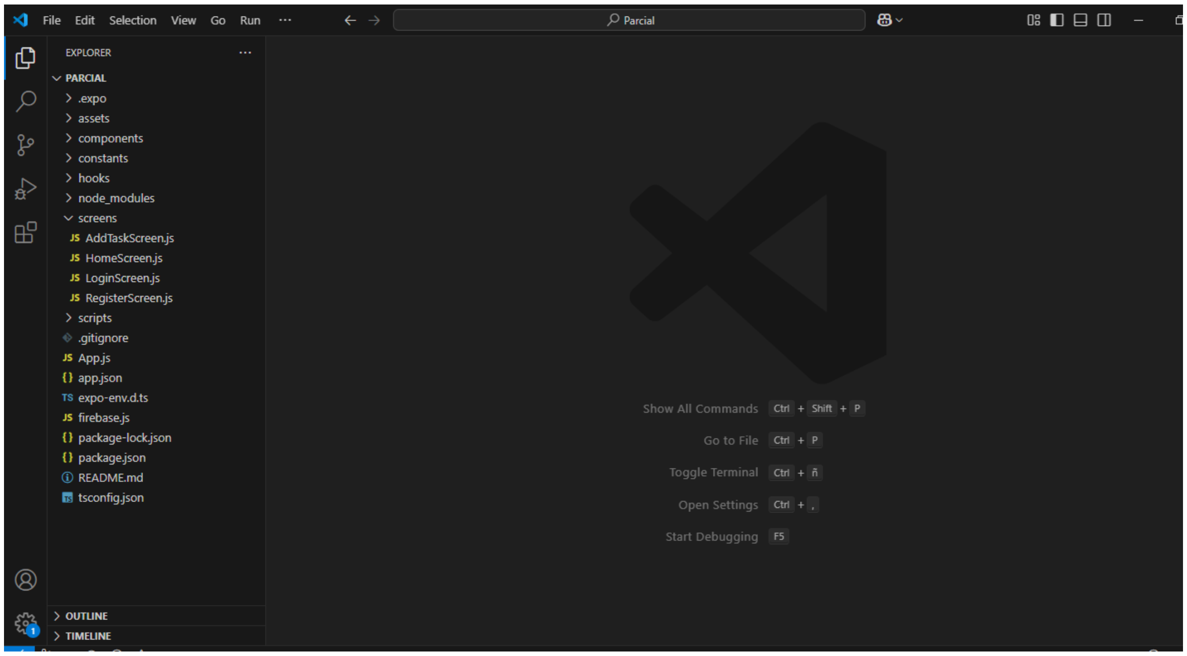


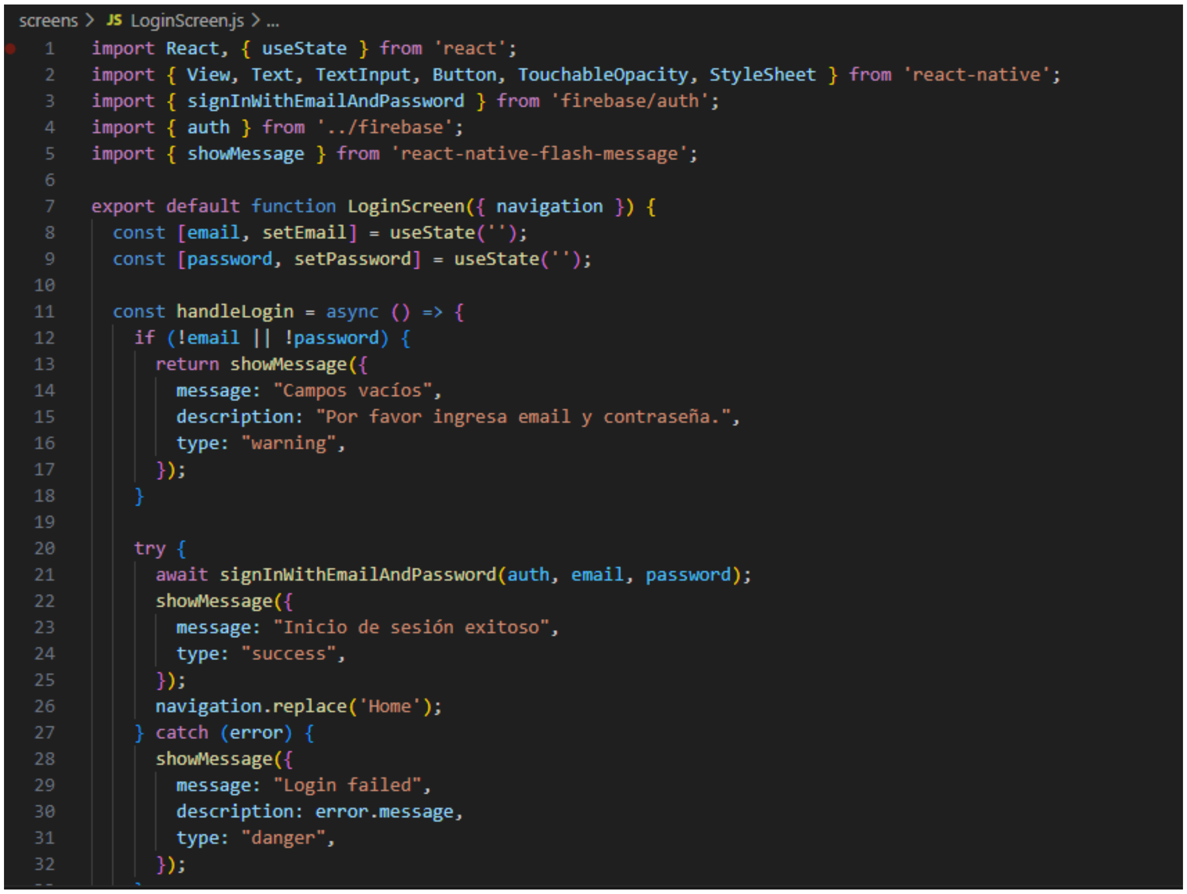
Las siguientes funciones de autenticación dejarán de estar disponibles cuando se cierre Firebase Dynamic Links el 25 de agosto de 2025: la autenticación a través de vínculos de correo electrónico en apps para dispositivos móviles, así como la compatibilidad con OAuth de Cordova para apps web.

Buscar por dirección de correo electrónico, número de teléfono o UID de usuario
Agregar usuario

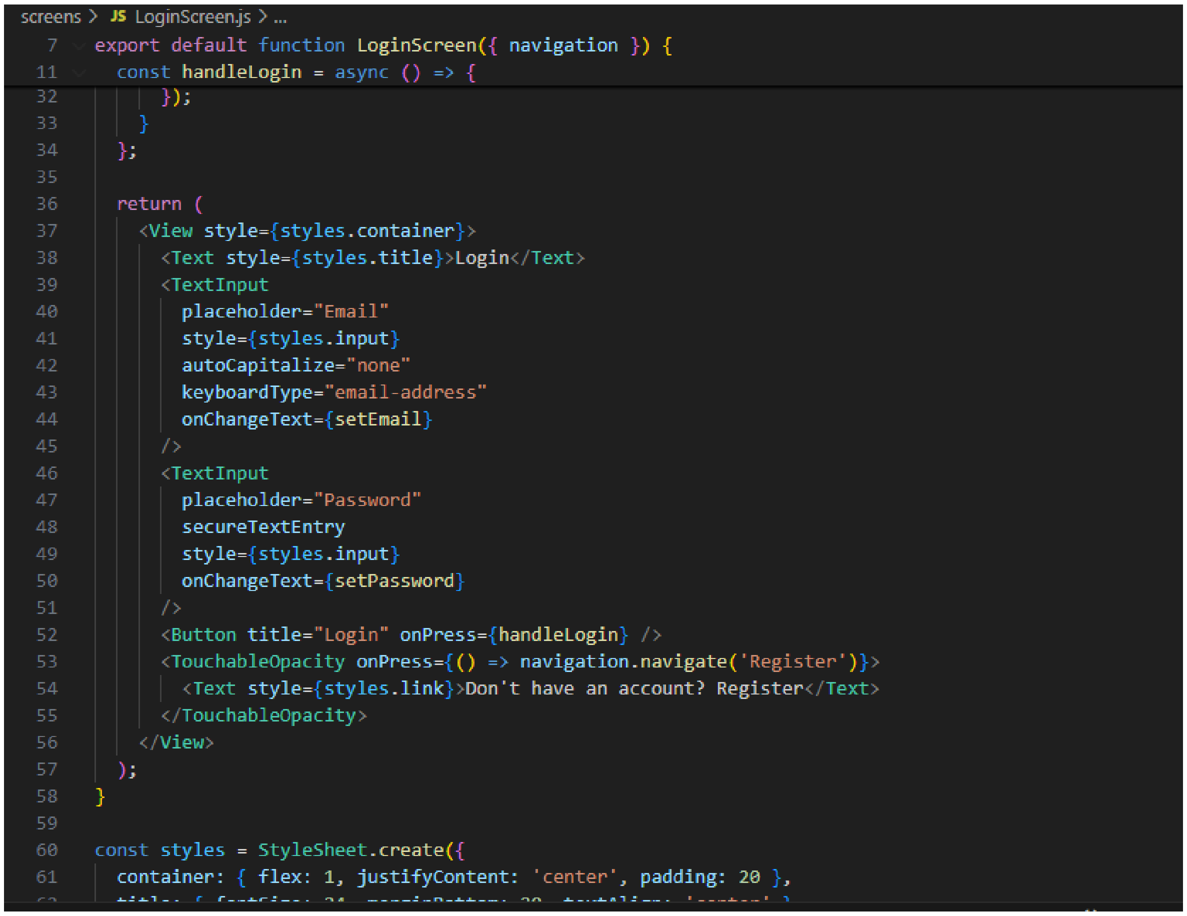
Identificador	Proveedores	Fecha de creación	Fecha de acceso	UID de usuario
yoselinisaza9@gmail.c...	✉	5 abr 2025	5 abr 2025	TwhuQASs8VRnN4vYSlXt7FC...

Filas por página: 50 1 – 1 of 1



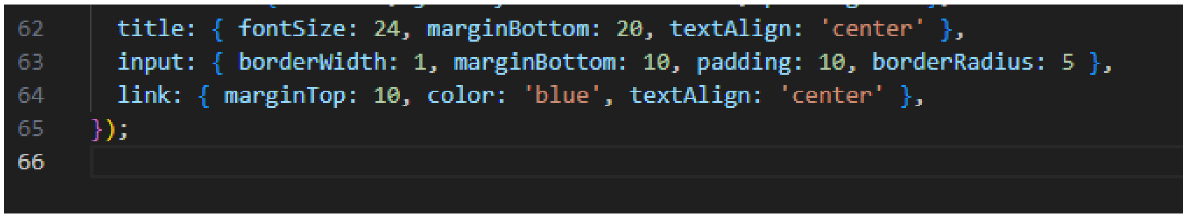


```
import React, { useState } from 'react';
import { View, Text, TextInput, Button, TouchableOpacity, StyleSheet } from 'react-native';
import { signInWithEmailAndPassword } from 'firebase/auth';
import { auth } from '../firebase';
import { showMessage } from 'react-native-flash-message';

export default function LoginScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    if (!email || !password) {
      return showMessage({
        message: "Campos vacíos",
        description: "Por favor ingresa email y contraseña.",
        type: "warning",
      });
    }

    try {
      await signInWithEmailAndPassword(auth, email, password);
      showMessage({
        message: "Inicio de sesión exitoso",
        type: "success",
      });
      navigation.replace('Home');
    } catch (error) {
      showMessage({
        message: "Login failed",
        description: error.message,
        type: "danger",
      });
```

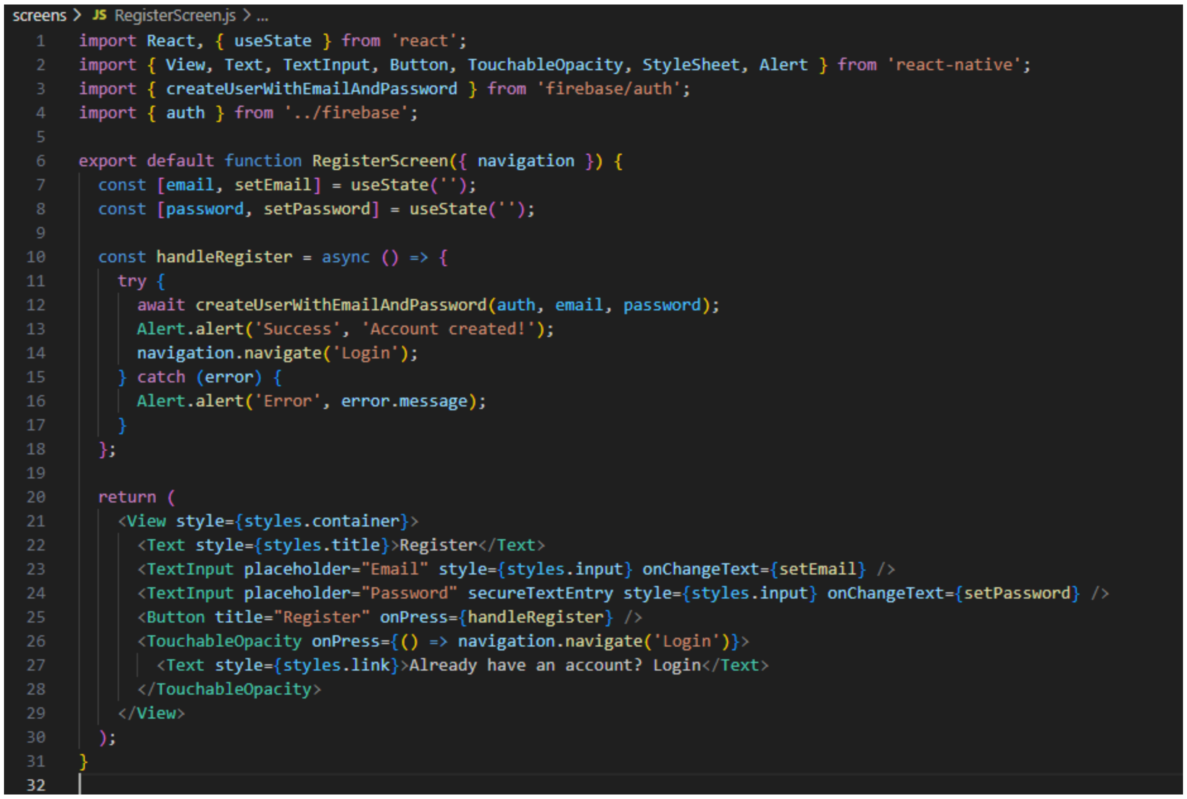


screens > JS LoginScreen.js > ...

```
export default function LoginScreen({ navigation }) {
  const handleLogin = async () => {
      });
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Login</Text>
      <TextInput
        placeholder="Email"
        style={styles.input}
        autoCapitalize="none"
        keyboardType="email-address"
        onChangeText={setEmail}
      />
      <TextInput
        placeholder="Password"
        secureTextEntry
        style={styles.input}
        onChangeText={setPassword}
      />
      <Button title="Login" onPress={handleLogin} />
      <TouchableOpacity onPress={() => navigation.navigate('Register')}>
        <Text style={styles.link}>Don't have an account? Register</Text>
      </TouchableOpacity>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20 },
```

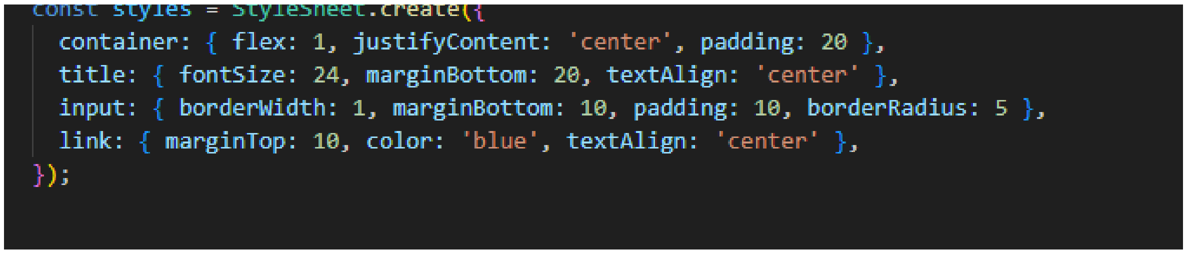


```
  title: { fontSize: 24, marginBottom: 20, textAlign: 'center' },
  input: { borderWidth: 1, marginBottom: 10, padding: 10, borderRadius: 5 },
  link: { marginTop: 10, color: 'blue', textAlign: 'center' },
});
```

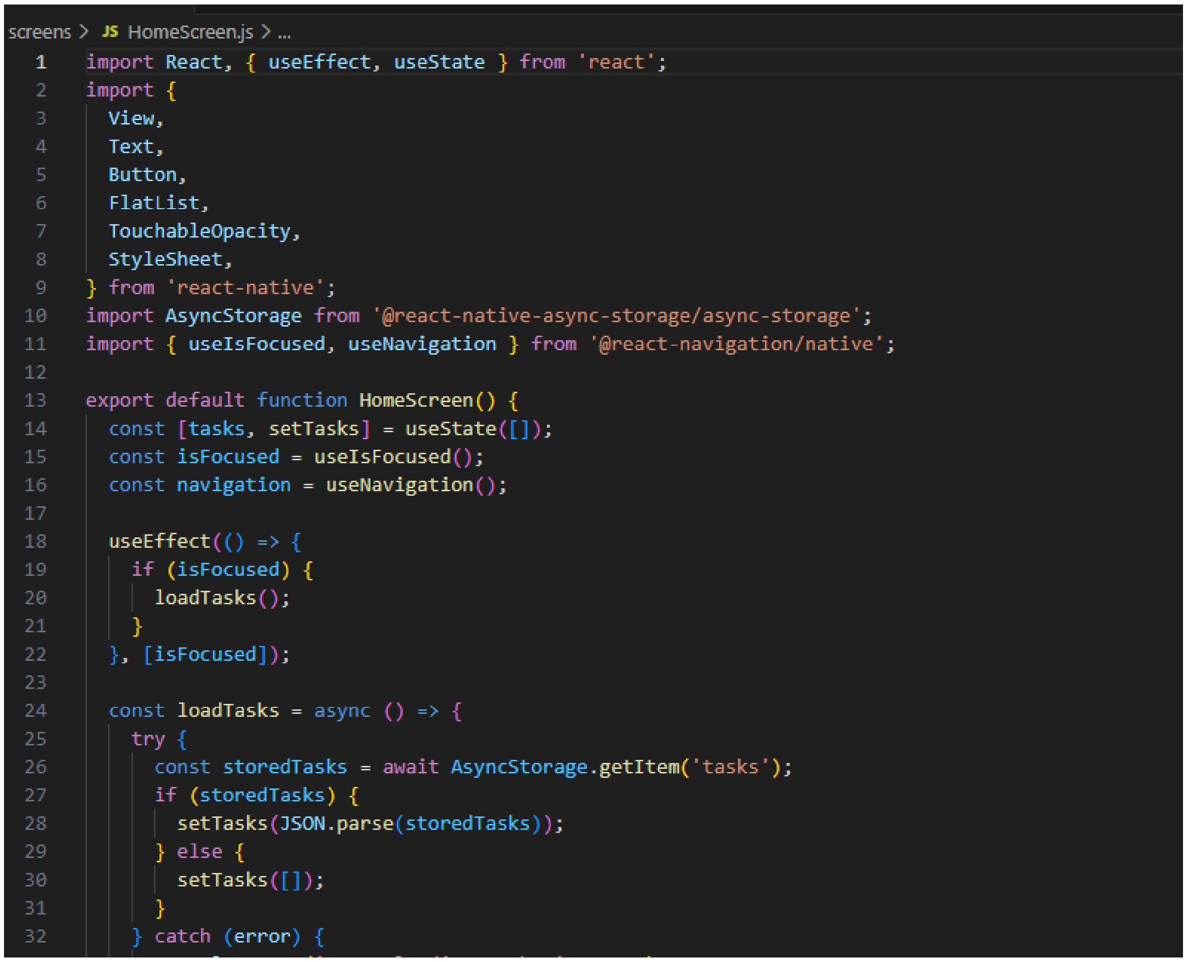


screens > JS RegisterScreen.js > ...

```javascript
import React, { useState } from 'react';
import { View, Text, TextInput, Button, TouchableOpacity, StyleSheet, Alert } from 'react-native';
import { createUserWithEmailAndPassword } from 'firebase/auth';
import { auth } from '../firebase';

export default function RegisterScreen({ navigation }) {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleRegister = async () => {
    try {
      await createUserWithEmailAndPassword(auth, email, password);
      Alert.alert('Success', 'Account created!');
      navigation.navigate('Login');
    } catch (error) {
      Alert.alert('Error', error.message);
    }
  };

  return (
    <View style={styles.container}>
      <Text style={styles.title}>Register</Text>
      <TextInput placeholder="Email" style={styles.input} onChangeText={setEmail} />
      <TextInput placeholder="Password" secureTextEntry style={styles.input} onChangeText={setPassword} />
      <Button title="Register" onPress={handleRegister} />
      <TouchableOpacity onPress={() => navigation.navigate('Login')}>
        <Text style={styles.link}>Already have an account? Login</Text>
      </TouchableOpacity>
    </View>
  );
}

```

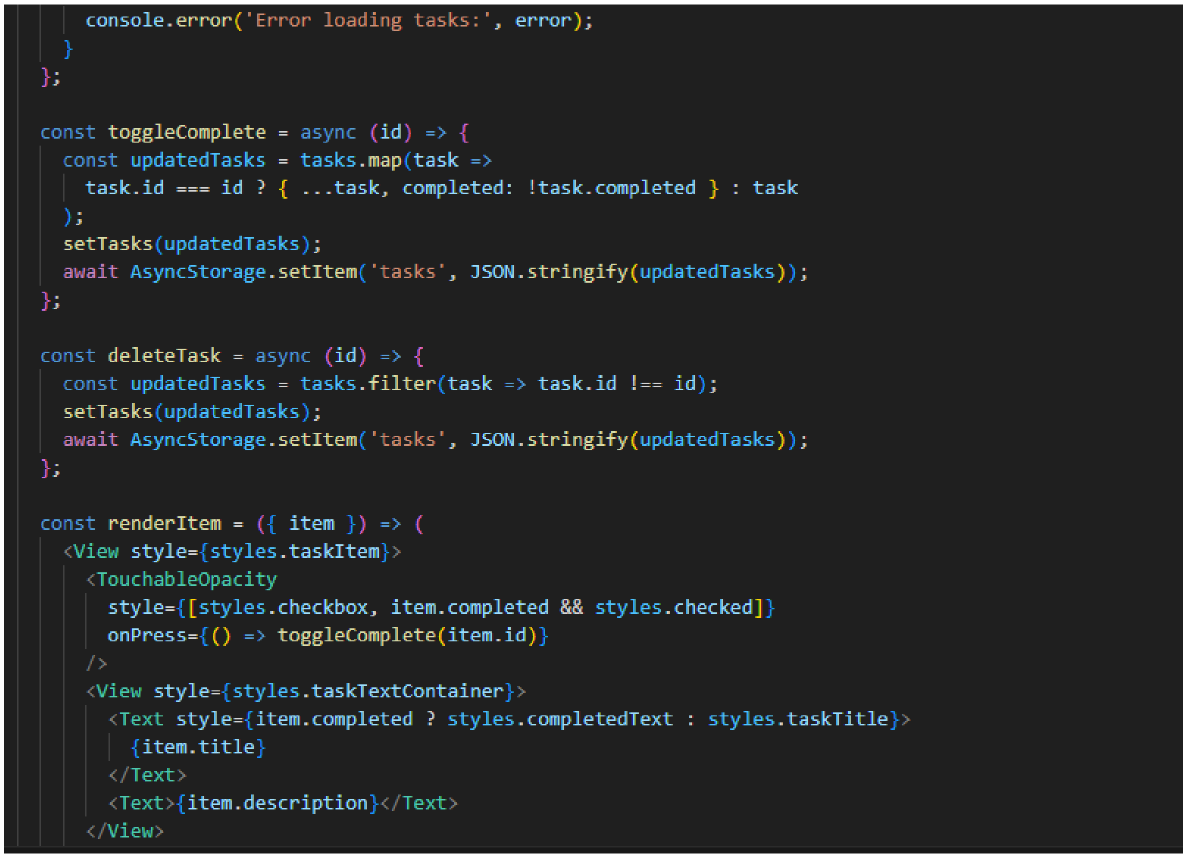


```javascript
const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', padding: 20 },
  title: { fontSize: 24, marginBottom: 20, textAlign: 'center' },
  input: { borderWidth: 1, marginBottom: 10, padding: 10, borderRadius: 5 },
  link: { marginTop: 10, color: 'blue', textAlign: 'center' },
});
```

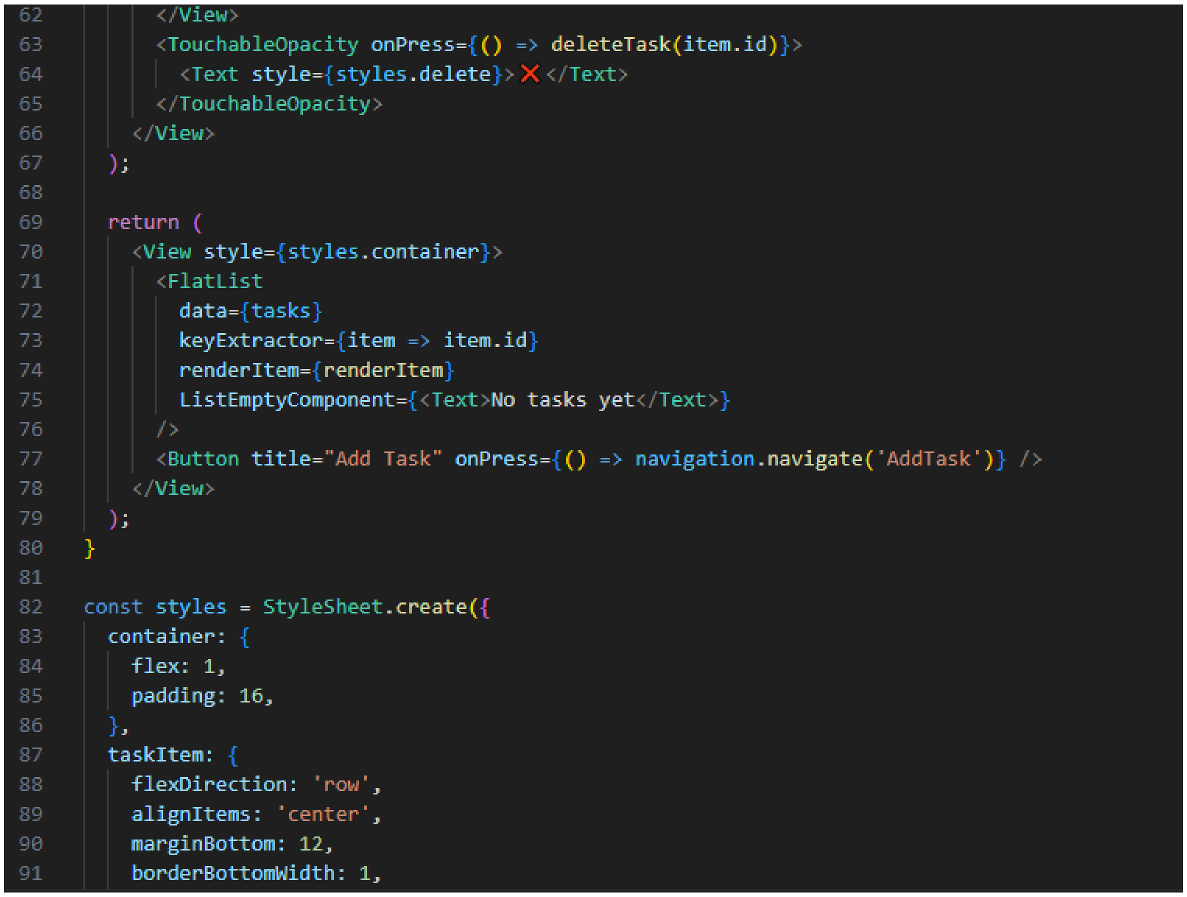


screens > JS HomeScreen.js > ...

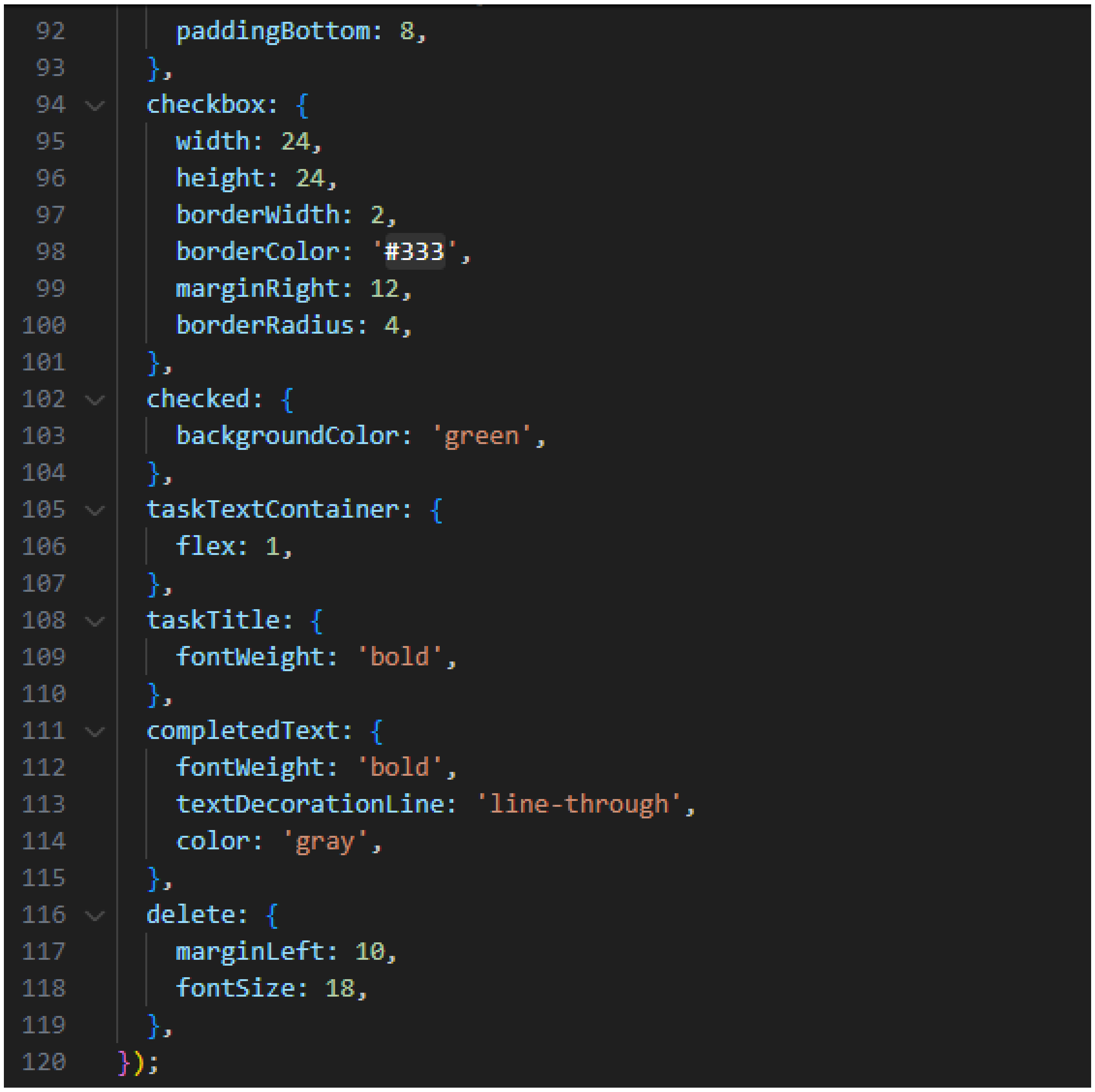
```
import React, { useEffect, useState } from 'react';
import {
  View,
  Text,
  Button,
  FlatList,
  TouchableOpacity,
  StyleSheet,
} from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useIsFocused, useNavigation } from '@react-navigation/native';

export default function HomeScreen() {
  const [tasks, setTasks] = useState([]);
  const isFocused = useIsFocused();
  const navigation = useNavigation();

  useEffect(() => {
    if (isFocused) {
      loadTasks();
    }
  }, [isFocused]);

  const loadTasks = async () => {
    try {
      const storedTasks = await AsyncStorage.getItem('tasks');
      if (storedTasks) {
        setTasks(JSON.parse(storedTasks));
      } else {
        setTasks([]);
      }
    } catch (error) {
```



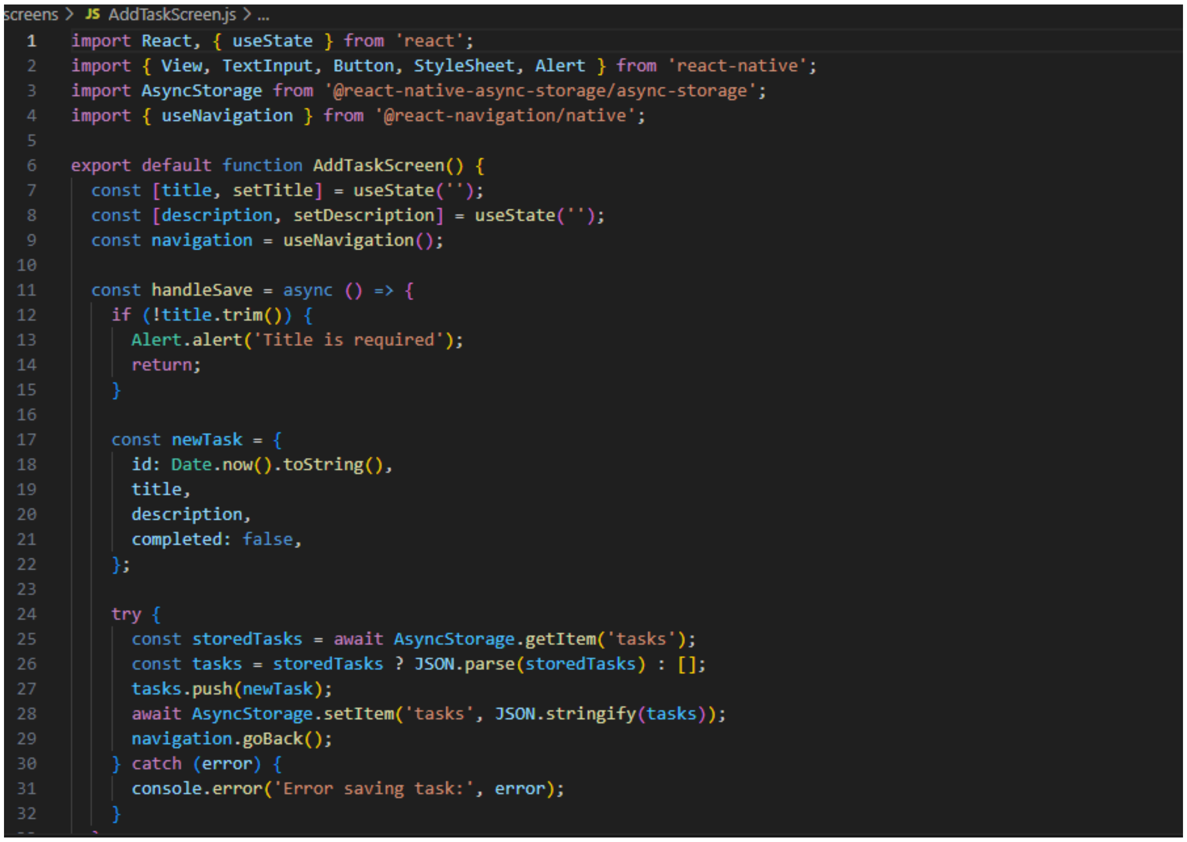
```
      console.error('Error loading tasks:', error);
    }
  };

  const toggleComplete = async (id) => {
    const updatedTasks = tasks.map(task =>
      task.id === id ? { ...task, completed: !task.completed } : task
    );
    setTasks(updatedTasks);
    await AsyncStorage.setItem('tasks', JSON.stringify(updatedTasks));
  };

  const deleteTask = async (id) => {
    const updatedTasks = tasks.filter(task => task.id !== id);
    setTasks(updatedTasks);
    await AsyncStorage.setItem('tasks', JSON.stringify(updatedTasks));
  };

  const renderItem = ({ item }) => (
    <View style={styles.taskItem}>
      <TouchableOpacity
        style={[styles.checkbox, item.completed && styles.checked]}
        onPress={() => toggleComplete(item.id)}
      />
      <View style={styles.taskTextContainer}>
        <Text style={item.completed ? styles.completedText : styles.taskTitle}>
          {item.title}
        </Text>
        <Text>{item.description}</Text>
      </View>
```



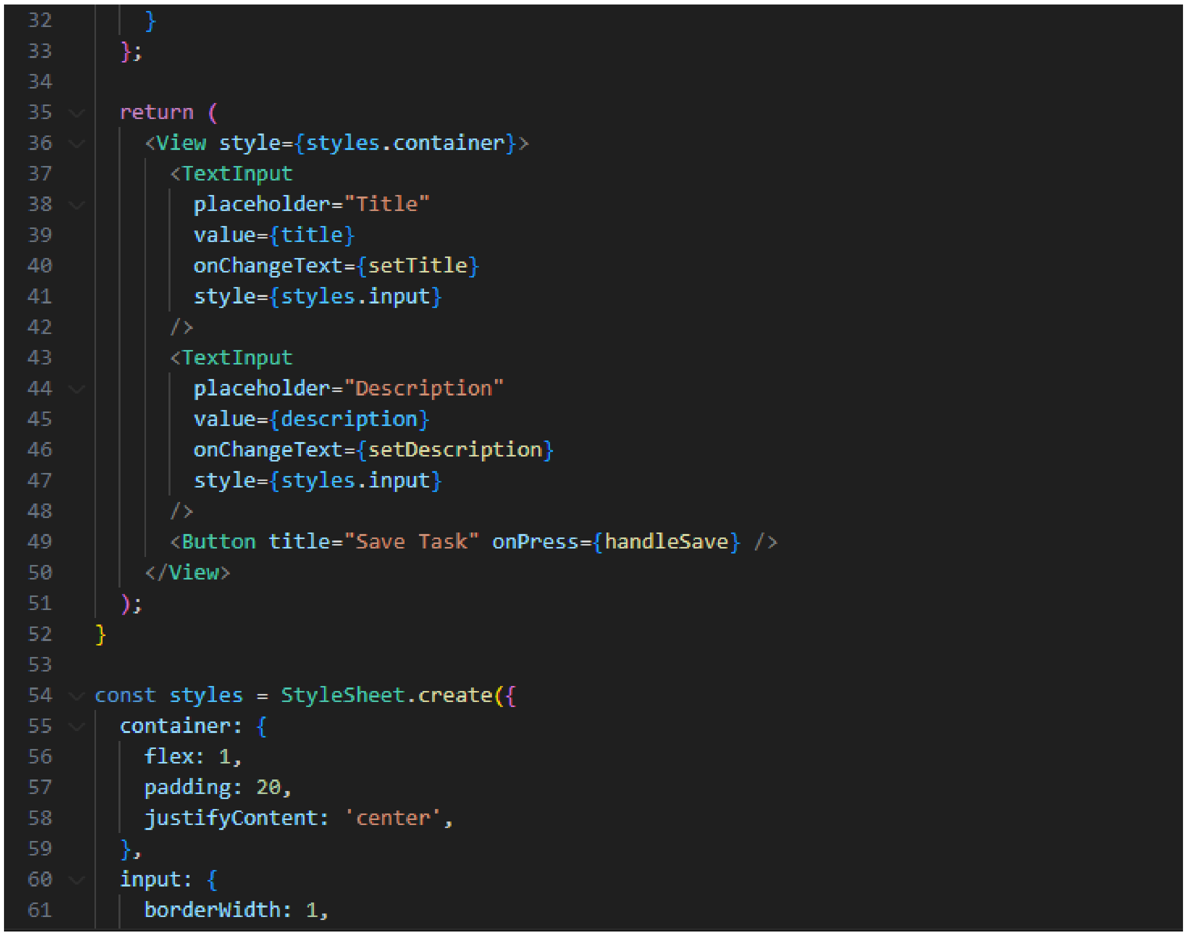
```jsx
      </View>
      <TouchableOpacity onPress={() => deleteTask(item.id)}>
        <Text style={styles.delete}>❌</Text>
      </TouchableOpacity>
    </View>
  );

  return (
    <View style={styles.container}>
      <FlatList
        data={tasks}
        keyExtractor={item => item.id}
        renderItem={renderItem}
        ListEmptyComponent={<Text>No tasks yet</Text>}
      />
      <Button title="Add Task" onPress={() => navigation.navigate('AddTask')} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
  },
  taskItem: {
    flexDirection: 'row',
    alignItems: 'center',
    marginBottom: 12,
    borderBottomWidth: 1,
```



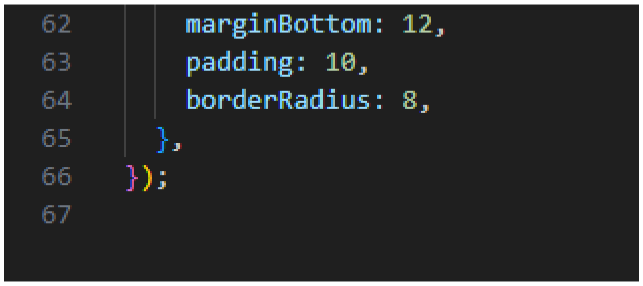
```
        paddingBottom: 8,
    },
    checkbox: {
        width: 24,
        height: 24,
        borderWidth: 2,
        borderColor: '#333',
        marginRight: 12,
        borderRadius: 4,
    },
    checked: {
        backgroundColor: 'green',
    },
    taskTextContainer: {
        flex: 1,
    },
    taskTitle: {
        fontWeight: 'bold',
    },
    completedText: {
        fontWeight: 'bold',
        textDecorationLine: 'line-through',
        color: 'gray',
    },
    delete: {
        marginLeft: 10,
        fontSize: 18,
    },
});
```


screens > JS AddTaskScreen.js > ...

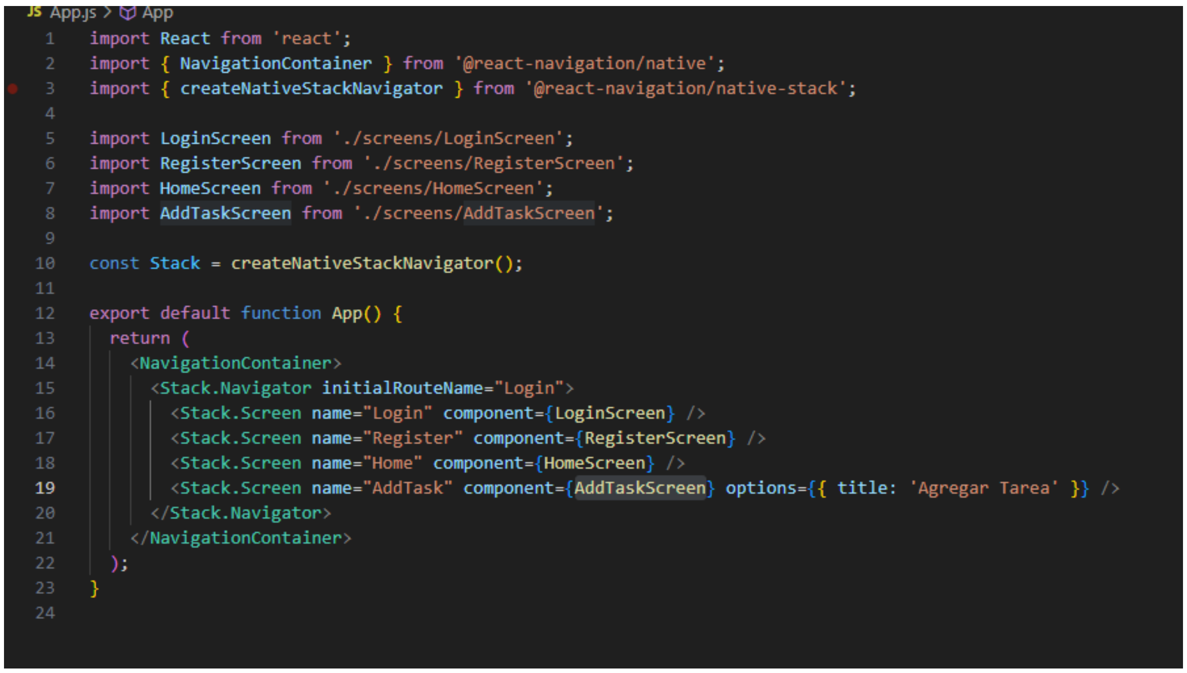
```javascript
import React, { useState } from 'react';
import { View, TextInput, Button, StyleSheet, Alert } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { useNavigation } from '@react-navigation/native';

export default function AddTaskScreen() {
  const [title, setTitle] = useState('');
  const [description, setDescription] = useState('');
  const navigation = useNavigation();

  const handleSave = async () => {
    if (!title.trim()) {
      Alert.alert('Title is required');
      return;
    }

    const newTask = {
      id: Date.now().toString(),
      title,
      description,
      completed: false,
    };

    try {
      const storedTasks = await AsyncStorage.getItem('tasks');
      const tasks = storedTasks ? JSON.parse(storedTasks) : [];
      tasks.push(newTask);
      await AsyncStorage.setItem('tasks', JSON.stringify(tasks));
      navigation.goBack();
    } catch (error) {
      console.error('Error saving task:', error);
    }
```



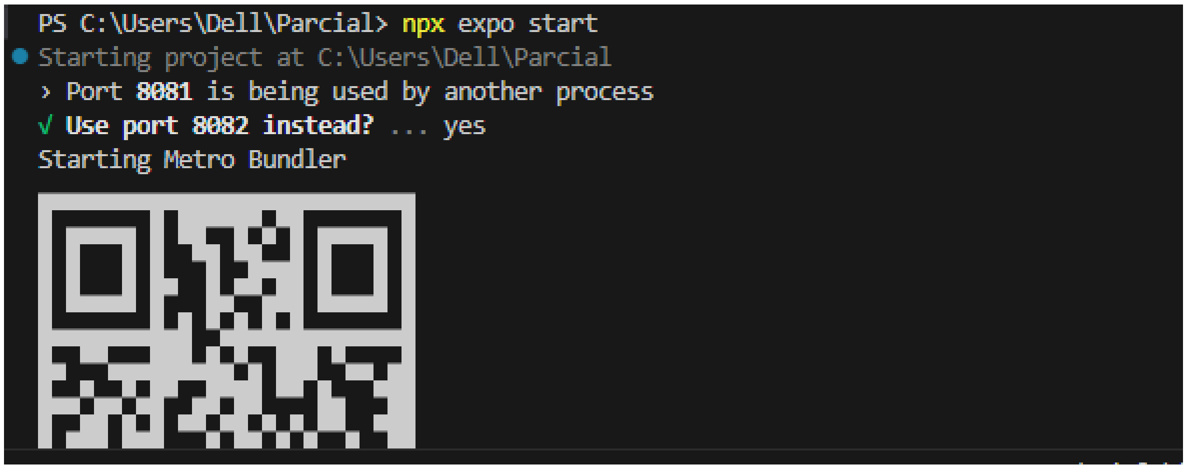
```
    }
  };

  return (
    <View style={styles.container}>
      <TextInput
        placeholder="Title"
        value={title}
        onChangeText={setTitle}
        style={styles.input}
      />
      <TextInput
        placeholder="Description"
        value={description}
        onChangeText={setDescription}
        style={styles.input}
      />
      <Button title="Save Task" onPress={handleSave} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    justifyContent: 'center',
  },
  input: {
    borderWidth: 1,
```



```
    marginBottom: 12,
    padding: 10,
    borderRadius: 8,
  },
});
```



```
App.js > App
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

import LoginScreen from './screens/LoginScreen';
import RegisterScreen from './screens/RegisterScreen';
import HomeScreen from './screens/HomeScreen';
import AddTaskScreen from './screens/AddTaskScreen';

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Login">
        <Stack.Screen name="Login" component={LoginScreen} />
        <Stack.Screen name="Register" component={RegisterScreen} />
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="AddTask" component={AddTaskScreen} options={{ title: 'Agregar Tarea' }} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```



```
PS C:\Users\Dell\Parcial> npx expo start
Starting project at C:\Users\Dell\Parcial
› Port 8081 is being used by another process
√ Use port 8082 instead? ... yes
Starting Metro Bundler
```

and here the application should run without any problem.