Yes, you can implement **Sign in with Google** in React Native without using Firebase by directly using the **Google OAuth 2.0 API**. This approach involves manually handling the OAuth flow and using a library like `expo-auth-session` (for Expo) or `react-native-app-auth` (for React Native CLI) to authenticate users.

Below is a step-by-step guide to implement Google Sign-In without Firebase:

Step 1: Set Up Google API Console

1. Go to the [Google API Console](https://console.developers.google.com/).

2. Create a new project (or use an existing one).

3. Enable the **Google Sign-In API** for your project.

4. Create OAuth 2.0 credentials:

   - Set the **Authorized Redirect URIs** to a custom URI scheme (e.g., `com.yourapp:/oauth2redirect`).

   - Note down the **Client ID** and **Client Secret**.

Step 2: Install Required Libraries

For React Native CLI, you can use the `react-native-app-auth` library to handle the OAuth flow:

```bash

npm install react-native-app-auth

```

or

```bash
yarn add react-native-app-auth
```

For iOS, install the necessary pods:

```bash
cd ios && pod install && cd ..
```

Step 3: Configure Android

1. Add the following to `android/app/src/main/AndroidManifest.xml`:

```xml
<activity
    android:name="net.openid.appauth.RedirectUriReceiverActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="com.yourapp" /> <!-- Replace with your app's scheme -->
    </intent-filter>
</activity>
```

```

---

### Step 4: Configure iOS

1. Open your project in Xcode.

2. Go to **Info** > **URL Types** and add a new URL scheme (e.g., `com.yourapp`).

---

### Step 5: Implement Google Sign-In

Here’s how you can implement the Google Sign-In flow using `react-native-app-auth`:

```javascript
import React from 'react';
import { Button, View, Text } from 'react-native';
import { authorize } from 'react-native-app-auth';

const config = {
  issuer: 'https://accounts.google.com',
  clientId: 'YOUR_CLIENT_ID', // From Google API Console
  redirectUrl: 'com.yourapp:/oauth2redirect', // Must match the redirect URI in Google API Console
  scopes: ['openid', 'profile', 'email'],
};
```
```

```
const App = () => {
  const [userInfo, setUserInfo] = React.useState(null);

  const signIn = async () => {
    try {
      const result = await authorize(config);
      console.log('Auth Result:', result);

      // Fetch user info using the access token
      const userInfoResponse = await fetch('https://www.googleapis.com/oauth2/v2/userinfo', {
        headers: { Authorization: `Bearer ${result.accessToken}` },
      });
      const userInfo = await userInfoResponse.json();
      setUserInfo(userInfo);
    } catch (error) {
      console.log('Error:', error);
    }
  };

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      {userInfo ? (
        <Text>Welcome, {userInfo.name}!</Text>
```

```
      ) : (
        <Button title="Sign in with Google" onPress={signIn} />
      )}
    </View>
  );
};

export default App;
```

Step 6: Test the Implementation

1. Run your app on an Android or iOS device/emulator:

   ```bash
   npx react-native run-android
   ```

   or

   ```bash
   npx react-native run-ios
   ```

2. Test the Google Sign-In button and ensure it works as expected.

Key Points

- Replace `YOUR_CLIENT_ID` with your actual client ID from the Google API Console.

- Ensure the **redirect URL** matches the one configured in the Google API Console.

- This approach does not require Firebase, and you directly interact with the Google OAuth 2.0 API.

Alternative: Use Expo (if using Expo Managed Workflow)

If you're using Expo, you can use the `expo-auth-session` library to handle Google Sign-In without Firebase. Here's a quick example:

```javascript
import React from 'react';
import { Button, View, Text } from 'react-native';
import * as WebBrowser from 'expo-web-browser';
import * as Google from 'expo-auth-session/providers/google';

WebBrowser.maybeCompleteAuthSession();

const App = () => {
  const [request, response, promptAsync] = Google.useAuthRequest({
    expoClientId: 'YOUR_EXPO_CLIENT_ID',
    iosClientId: 'YOUR_IOS_CLIENT_ID',
```

```
    androidClientId: 'YOUR_ANDROID_CLIENT_ID',
  });

  React.useEffect(() => {
    if (response?.type === 'success') {
      const { authentication } = response;
      console.log('Access Token:', authentication.accessToken);
      // Fetch user info using the access token
    }
  }, [response]);

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Button
        title="Sign in with Google"
        disabled={!request}
        onPress={() => promptAsync()}
      />
    </View>
  );
};

export default App;
```

This approach is simpler if you're using Expo, but for React Native CLI, `react-native-app-auth` is the recommended library. Let me know if you need further assistance!