Sure! Here are three full examples demonstrating the usage of `useContext` in React:

Example 1: Theme Switcher

In this example, we'll create a simple theme switcher using React context and `useContext`.

```jsx
// ThemeContext.js
import React from 'react';

const ThemeContext = React.createContext();

export default ThemeContext;

// ThemeProvider.js
import React, { useState } from 'react';
import ThemeContext from './ThemeContext';

const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light');

  const toggleTheme = () => {
    setTheme((prevTheme) => (prevTheme === 'light' ? 'dark' : 'light'));
  };

  return (
    <ThemeContext.Provider value={{ theme, toggleTheme }}>
      {children}
    </ThemeContext.Provider>
  );
};

export default ThemeProvider;

// ThemeToggler.js
import React, { useContext } from 'react';
import ThemeContext from './ThemeContext';

const ThemeToggler = () => {
  const { theme, toggleTheme } = useContext(ThemeContext);

  return (
    <button onClick={toggleTheme}>
      Toggle Theme ({theme})
    </button>
  );
};

export default ThemeToggler;

// App.js
import React from 'react';
import ThemeProvider from './ThemeProvider';
import ThemeToggler from './ThemeToggler';

const App = () => {
  return (
    <ThemeProvider>
      <ThemeToggler />
    </ThemeProvider>
  );
};

export default App;
```

Example 2: Language Switcher

In this example, we'll create a language switcher using React context and `useContext`.

```jsx
// LanguageContext.js
import React from 'react';

const LanguageContext = React.createContext();

export default LanguageContext;

// LanguageProvider.js
import React, { useState } from 'react';
import LanguageContext from './LanguageContext';

const LanguageProvider = ({ children }) => {
  const [language, setLanguage] = useState('en');

  const toggleLanguage = () => {
    setLanguage((prevLanguage) => (prevLanguage === 'en' ? 'fr' : 'en'));
  };
  return (
    <LanguageContext.Provider value={{ language, toggleLanguage }}>
      {children}
    </LanguageContext.Provider>
  );
};

export default LanguageProvider;

// LanguageToggler.js
import React, { useContext } from 'react';
import LanguageContext from './LanguageContext';

const LanguageToggler = () => {
  const { language, toggleLanguage } = useContext(LanguageContext);

  return (
    <button onClick={toggleLanguage}>
      Toggle Language ({language})
    </button>
  );
};

export default LanguageToggler;

// App.js
import React from 'react';
import LanguageProvider from './LanguageProvider';
import LanguageToggler from './LanguageToggler';

const App = () => {
  return (
    <LanguageProvider>
      <LanguageToggler />
    </LanguageProvider>
  );
};

export default App;
```

Example 3: Authentication

In this example, we'll create an authentication system using React context and `useContext`.

```jsx
// AuthContext.js
import React, { createContext, useState } from 'react';

const AuthContext = createContext();

const AuthProvider = ({ children }) => {
  const [isLoggedIn, setIsLoggedIn] = useState(false);

  const login = () => {
    setIsLoggedIn(true);
  };

  const logout = () => {
    setIsLoggedIn(false);
  };

  return (
    <AuthContext.Provider value={{ isLoggedIn, login, logout }}>
      {children}
    </AuthContext.Provider>
  );
};

export { AuthContext, AuthProvider };

// Login.js
import React, { useContext } from 'react';
import { AuthContext } from './AuthContext';

const Login = () => {
  const { login } = useContext(AuthContext);

  return (
    <button onClick={login}>
      Login
    </button>
  );
};

export default Login;
```

```
// Logout.js
import React, { useContext } from 'react';
import { AuthContext } from './AuthContext';

const Logout = () => {
  const { logout } = useContext(AuthContext);

  return (
    <button onClick={logout}>
      Logout
    </button>
  );
};

export default Logout;

// App.js
import React from 'react';
import { AuthProvider } from './AuthContext';
import Login from './Login';
import Logout from './Logout';

const App = () => {
  return (
    <AuthProvider>
      <div>
        <h1>Authentication Example</h1>
        <Login />
        <Logout />
      </div>
    </AuthProvider>
  );
};

export default App;
```

These examples showcase different scenarios where `useContext` can be used to share state across components without prop drilling.