MODULE 5 Advanced Topics

ADVENTIST UNIVERSITY OF CENTRAL AFRICA
FACULTY OF INFORMATION TECHNOLOGY
Course: **Mobile Programming**
INSY 429
GROUP IV

July 7, 2023

Group Members

Group IV:

1. RUKUNDO Justin 23456
2. NZEYIMANA Yvon Miracle 23707
3. MUKARWAKA UWASE Mireille 23621
4. Iradukunda Divine 23663
5. ABAYO Florence 23413
6. NIYIRERA Jacqueline 24103
7. kengne klinsman 24042
8. Rafiki Didier 22709
9. Rocky Kayitare 24353
10. Kalisa Sangwa Kelly 23652
11. Ntore Ntwari Rumenera 24438
12. Warda Umutoni 20500
13. Igiraneza Michel 23674
14. Ali Mohamadene Haroun 24315
15. Mucyo Herve 23745
16. Emmanuel MANZI 24031
17. Rwego Muhoza Alain Patrick 23738
18. Niyokwizera Jean de Dieu 21681
19. SHYAKA Dieu-Merci R. 24232
20. Iradukunda Lea 23279
21. IZERE Didier 23423
22. MUGABO André 25337

Table of Contents

INTRODUCTION

Introduction

This presentation will focus on module 5 entitled **Advanced Topics** of our course **Mobile Programming INSY 429** that has as objective of empowering a students with the following topics :

Upon completion of the module, the student should acquire the capacity to:

- Implementing authentication and user management
- Integrating with device features (e.g., Camera, Geolocation, Sensors)
- Push notifications and background tasks
- Performance optimization techniques
- Testing and debugging React Native applications

Now let's see one by one of the above topics

Implementation of authentication and user management

Authentication

Authentication plays a vital role in modern applications and services by enabling them to identify the user and regulate their authorized actions.

- React Native authentication refers to the process of verifying the identity of a user in a React Native app.
- This is typically done by asking the user to provide their login credentials, such as a username and password.
- Then checking those credentials against a database of authorized users.

Implementation of authentication and user management

OAuth 2.0 authorization framework

React Native authentication is commonly implemented with OAuth 2.0

- This allows users to sign in via popular third-party services such as Google, Facebook, or Twitter.
- There are several libraries you can use to implement authentication in React Native-we'll cover three of them in this presentation.
  1. React Native App Auth
  2. React Navigation
  3. React Native OAuth

Why is React Native authentication important?

- React Native authentication is critical because it provides a secure way for users to access protected resources in a React Native app.
- Without proper user authentication, anyone could access sensitive information or perform actions that they should not be able to.

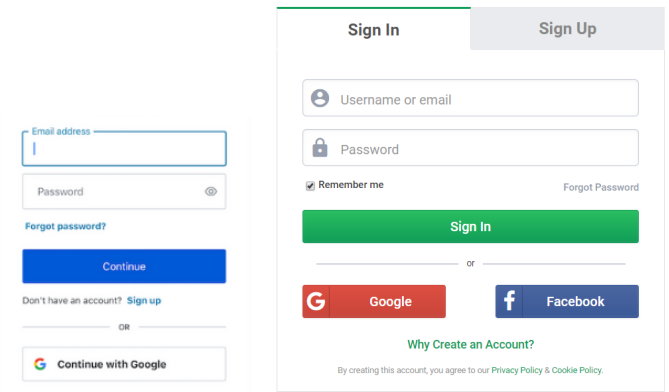
React Native authentication is important for:

- Protecting sensitive data.
- Enhancing security.
- Improving user experience.
- Supporting compliance.

Implementing React Native authentication with OAuth2

OAuth2

- OAuth2 is an open standard for authorization that's widely used to allow users to grant access to resources without sharing their credentials.
- In the context of React Native authentication,
- OAuth2 is commonly used to allow a user to sign in to an application using their existing credentials from a third-party service, such as Google, Facebook, or Twitter.



Implementing React Native authentication with OAuth2

OAuth2

- A common flow for OAuth2 in a React Native application is for the user to click a "Sign in with X" button, which then redirects them to the third-party service's login page.
- Once the user has successfully logged in to the third-party service, they are redirected back to the React Native application, where they grant permission for the application to access their resources.

Implementing React Native authentication with OAuth2

OAuth2

- Redirects are an important part of the OAuth2 flow in React Native applications. The redirects are used to redirect the user to the third-party service's login page, and then back to the React Native application after the user has successfully logged in.
- These redirects are typically implemented using the **WebView component** in React Native, which is used to display web pages within the mobile application.

What is Proof of Key Code Exchange or PKCE?

Proof of Key Code Exchange, or PKCE

- Proof of Key Code Exchange, or PKCE, is an extension to the OAuth2 authorization code flow that provides additional security for public client applications.
- In the standard OAuth2 authorization code flow, an authorization code is exchanged for an access token. This can create security vulnerabilities, as a malicious actor could intercept the authorization code and use it to obtain an access token for the user's resources.
- PKCE solves this problem by adding an extra step to the authorization code flow, where a code verifier is generated and passed along with the authorization code. The code verifier is then used to verify the authorization code when it is exchanged for an access token, providing an additional layer of security.

React Native authentication libraries

React Native authentication libraries

There is a variety of libraries you can use to implement authentication in your React Native apps. The three discussed here are only a few of the possible options-there are many more libraries you could use to implement authentication in React.

React Native authentication libraries

React Native App Auth

- **React Native App Auth** is an open-source library for implementing authentication in React Native apps using OAuth 2.0 and OpenID Connect. It provides a set of tools and APIs for interacting with OAuth 2.0 and OpenID Connect-compliant identity providers, such as Google, Facebook, and Microsoft, to authenticate users.
- React Native App Auth supports both Android and iOS platforms and can be easily integrated into a React Native app to handle authentication tasks, such as user login and logout, securely storing authentication tokens, and refreshing access tokens.

React Native authentication libraries

React Native App Auth

- The library abstracts the underlying authentication protocol details, making it easier for developers to implement authentication in their apps without having to deal with the complexity of the underlying protocols.

React Navigation

- **React Navigation** is a popular library for navigation in React Native apps. It provides tools and APIs for navigation between screens, handling navigation history, and configuring the navigation header.
- React Navigation can be used in combination with a separate authentication library to implement authentication in a React Native app by controlling the navigation flow based on the authentication state.

React Native authentication libraries

React Native OAuth

- React Native OAuth is a library for adding OAuth authentication to React Native apps. With React Native OAuth, developers can add OAuth authentication to their apps and interact with OAuth-compliant identity providers, such as Google, Facebook, and Microsoft, to authenticate users. The library provides a set of APIs and tools for handling common OAuth tasks, such as obtaining and refreshing access tokens, securely storing tokens, and handling token expiration.
- React Native OAuth supports both Android and iOS platforms and can be easily integrated into a React Native app to handle authentication tasks.

React Native authentication example

Authentication Example

Here's an example of how to use the React Native App Auth library to implement OAuth 2.0 authentication in a React Native app.
First, install the React Native App Auth library:

```
npm i react-native-app-auth
```

Import the React Native App Auth module into your React Native app:

```
1 import { authorize, signOut } from 'react-native-app-auth';
```

Auth.js

Add a function to handle the authorization flow:

React Native authentication example

Authentication Example

```
const handleAuthorize = async () => {
  try {
    const result = await authorize({
      issuer: "https://YOUR_ISSUER",
      clientId: "YOUR_CLIENT_ID",
      redirectUrl: "YOUR_REDIRECT_URI",
      scopes: ["openid", "profile", "email"],
      serviceConfiguration: {
        authorizationEndpoint: "https://YOUR_AUTHORIZATION_ENDPOINT
            ",
        tokenEndpoint: "https://YOUR_TOKEN_ENDPOINT",
      },
    });
    console.log(result);
  } catch (error) {
    console.error(error);
  }
};
```

handleAuthorize.js

React Native authentication example

Authentication Example

Add a function to handle sign-out:

```
const handleSignOut = async () => {
  try {
    await signOut(options);
    console.log("User signed out");
  } catch (error) {
    console.error(error);
  }
};
```

handleSignOut.js

Use the above functions in your React Native app to authorize and sign out users.