droidcon academy

Biometric Authentication in Android

Musab Nasreldin

@musabnasreldin

in/musabagab

Overview

- Understanding Biometric Authentication
- Explore authentication types
- Checking Biometric Availability
- Implementing Biometric Authentication with BiometricPrompt
- Handling Authentication Callbacks
- Adding Biometric Fallback

Understanding Biometric Authentication

- Uses unique biological characteristics for user verification
- Examples: fingerprints, facial features, and iris patterns
- Advantages over traditional authentication methods

Explore Authentication Types

- BIOMETRIC_STRONG: Requires strong biometric data and user authentication
- BIOMETRIC_WEAK: Allows weak biometric data and user authentication
- DEVICE_CREDENTIAL: Accepts device PIN or password

Checking Biometric Availability

- Use BiometricManager to check if biometric authentication is available
- Handle different cases (available, not available, not enrolled)

Checking Biometric Availability

```kotlin
val biometricManager = BiometricManager.from(this)

when (biometricManager.canAuthenticate(BiometricManager.Authenticators.BIOMETRIC_STRONG)) {

    BiometricManager.BIOMETRIC_SUCCESS -> {
        // Biometric authentication is available
    }
    BiometricManager.BIOMETRIC_ERROR_NO_HARDWARE -> {
        // Biometric hardware is not available on this device
    }
    BiometricManager.BIOMETRIC_ERROR_HW_UNAVAILABLE -> {
        // Biometric hardware is temporarily unavailable
    }
    BiometricManager.BIOMETRIC_ERROR_NONE_ENROLLED -> {
        // Biometric data not enrolled
    }
}
```

Biometric Authentication with BiometricPrompt

- Create a BiometricPrompt instance
- Customize the prompt using BiometricPrompt.PromptInfo.Builder
- Handle authentication callbacks

Biometric Authentication with BiometricPrompt

```kotlin
val biometricPrompt = BiometricPrompt(this, ContextCompat.getMainExecutor(this),
    object : BiometricPrompt.AuthenticationCallback() {
        override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {
            super.onAuthenticationSucceeded(result)
            // Handle authentication success
        }
        override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {
            super.onAuthenticationError(errorCode, errString)
            // Handle authentication error
        }
        override fun onAuthenticationFailed() {
           super.onAuthenticationFailed()
            // Handle authentication failure
        }
    })

val promptInfo = BiometricPrompt.PromptInfo.Builder()
    .setTitle("Biometric Authentication")
    .setSubtitle("Log in using your biometric credential")
    .setNegativeButtonText("Cancel")
    .build()

biometricPrompt.authenticate(promptInfo)
```

Adding Biometric Fallback

```kotlin
val biometricPrompt = BiometricPrompt(this, ContextCompat.getMainExecutor(this),
    object : BiometricPrompt.AuthenticationCallback() {

   override fun onAuthenticationError(errorCode: Int, errString: CharSequence) {

       if (errorCode == BiometricPrompt.ERROR_USER_CANCELED
                    || errorCode == BiometricPrompt.ERROR_CANCELED) {
                // User canceled authentication
                // Handle accordingly (e.g., show PIN/password login option)
            } else {
                // Handle other authentication errors
            }
        }
    })
...
```

Challenge : Determine how the user authenticated

```kotlin
override fun onAuthenticationSucceeded(result: BiometricPrompt.AuthenticationResult) {
    super.onAuthenticationSucceeded(result)

    when (result.authenticationType) {

        AUTHENTICATION_RESULT_TYPE_UNKNOWN -> {

        }

        AUTHENTICATION_RESULT_TYPE_DEVICE_CREDENTIAL -> {

        }

        AUTHENTICATION_RESULT_TYPE_BIOMETRIC -> {

        }
    }
}
```

Summary

- Understanding Biometric Authentication
- Explore authentication types
- Checking Biometric Availability
- Implementing Biometric Authentication with BiometricPrompt
- Handling Authentication Callbacks
- Adding Biometric Fallback