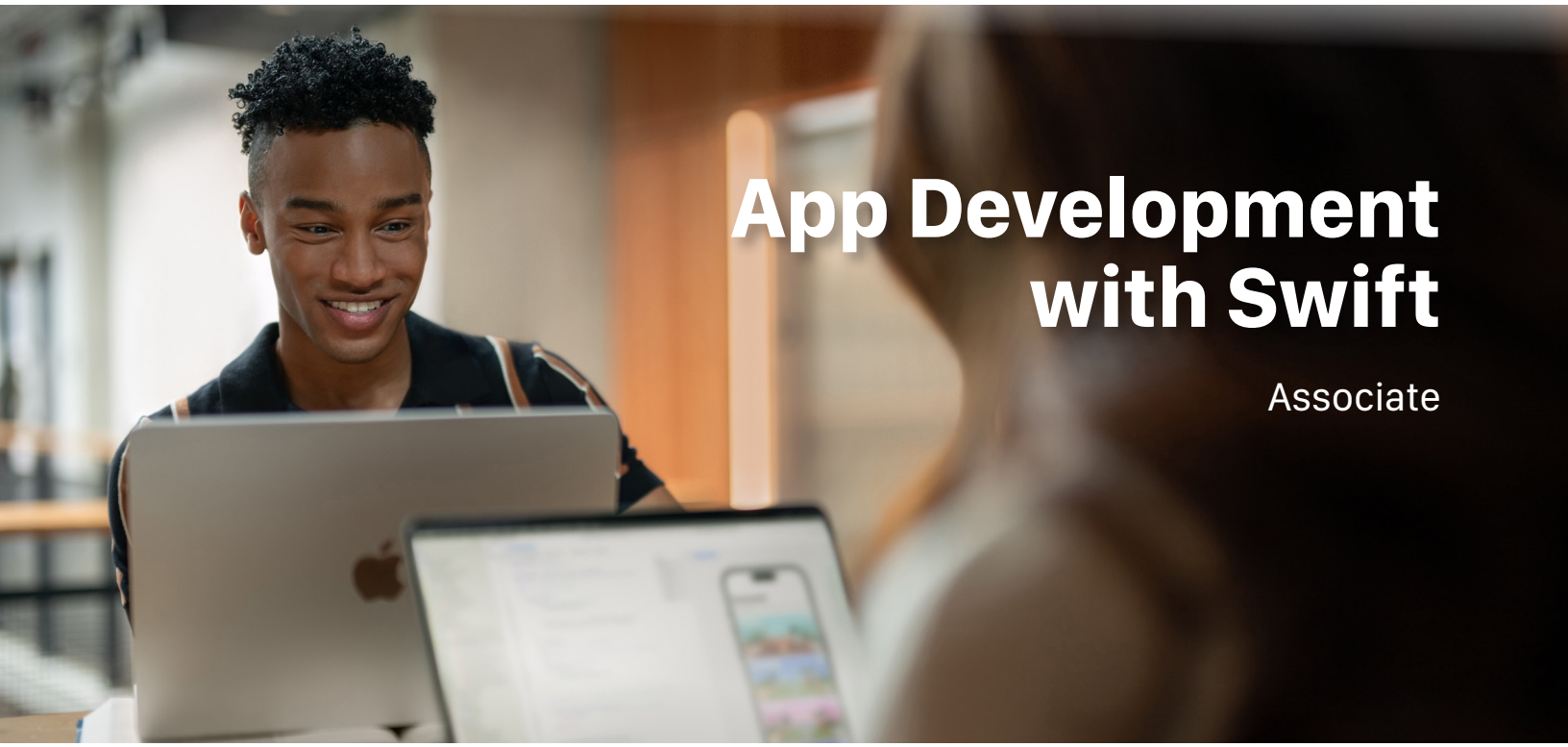
CERTIPORT
A PEARSON VUE BUSINESS



App Development with Swift

Associate

Objective Domains

Earning the App Development with Swift Associate certification demonstrates knowledge of key computing concepts and a solid foundation in programming with Swift and SwiftUI. They'll demonstrate knowledge of the impact of computing and apps on society, economies, and cultures while exploring app development.

Planning and Design

1.1. Summarize the design cycle

- 1.1.1. Brainstorm, plan, prototype, evaluate

1.2. Summarize how sensitive data can be protected and compromised

- 1.2.1. Sharing personal and application information
- 1.2.2. Security challenges
- 1.2.3. Legal, ethical and socioeconomic impacts

1.3. Assess a visual design with accessibility in mind

XCode Project Navigation

2.1. Differentiate between basic file types

2.2. After an asset has been imported, recognize available assets and how they are used in a project

2.3. Import and/or use an asset

2.4. Select the appropriate actions to configure different areas of the user interface

Swift Language Usage

3.1. Write, call and/or evaluate the execution of functions

- 3.1.1. Evaluate the use of argument labels, parameters and returns

3.2. Calculate the results when using various operators

3.3. Create and evaluate structures

- 3.3.1. Declare the properties of a structure
- 3.3.2. Initialize the properties of a structure
- 3.3.3. Define methods
- 3.3.4. Create an instance of a structure
- 3.3.5. Use an instance of a structure

APP DEVELOPMENT
WITH SWIFT
Associate

CERTIPORT
A PEARSON VUE BUSINESS

App Development with Swift

Associate

Swift Language Usage (Continued)

3.4. Create and manipulate arrays

- 3.4.1. Declare and/or initialize an array with values
- 3.4.2. Identify and/or modify an array element using its index
- 3.4.3. Use and/or evaluate array properties and/or methods

3.5. Demonstrate how to control the flow of execution

- 3.5.1. Create, analyze and predict loop structures and their results
- 3.5.2. Create and interpret the outcome of conditional statements

3.6. Declare and/or evaluate constants and variables of different data types

- 3.6.1. Differentiate between constants and variables
- 3.6.2. Apply type inference
- 3.6.3. Use explicit typing

3.7. Use the appropriate naming syntax

- 3.7.1. Use appropriate camel casing
- 3.7.2. Apply Swift identifier rules

View Building with SwiftUI

4.1. Differentiate between imperative and declarative programming

4.2. Create Content Views using Text, Image, Shape, and/or Color

4.3. Implement Modifiers including, but not limited to, .padding, .background, .frame, .foregroundColor, .font, and .resizable

4.4. Create Container Views (HStack, VStack, ZStack, Spacer) and arrange Views inside of Stack Views

4.5. Explain the View hierarchy produced by a program

4.6. Create and/or apply Interactive Views including, but not limited to, Button, TextField, Slider, and Toggle

4.7. Use @State Property Wrapper to control the appearance of a View

Debugging

5.1. Differentiate between syntax and run-time errors when building and running an app

5.2. Interpret error messages



© 2025 Certiport, Inc. Certiport and the Certiport logo are registered trademarks of Certiport Inc. Apple, Xcode, Swift and SwiftUI are trademarks of Apple Inc., registered in the U.S. and other countries and regions.