Mobile Application Development

Introduction

Dr. Hammoudeh Alamri

Introduction to Mobile Application Development

- Mobile application development focuses on creating software for mobile devices.
- The main platforms are iOS (Apple) and Android (Google).
- Mobile apps are classified into three types: native, hybrid, and cross-platform.

Why Mobile Apps are Important

•Mobile devices enable high user engagement.

•Mobile apps provide convenience and easy accessibility.

•They support business growth by connecting with users on smartphones and tablets.

Types of Mobile Apps

- Native Apps: Developed specifically for one platform (iOS or Android).
- Hybrid Apps: A combination of web apps and native apps.
- Cross-Platform Apps: Apps that work on multiple platforms.

Native Mobile App Development

- Native apps are written in the programming language specific to the platform.
- iOS apps use Swift or Objective-C.
- Android apps use Java or Kotlin.
- Advantages: High performance and access to device features.

Challenges with Native Development

- Separate codebases for each platform increase development time and cost.
- Maintenance and updates must be done separately for each platform.
- Lack of code sharing between iOS and Android.

Hybrid mobile development

- Hybrid mobile app development mixes the characteristics of a native application and web app.
- Mobile developers create hybrid mobile apps using well-known languages such as JavaScript, HTML, and CSS.
- A hybrid app combines elements from both mobile and web apps to rapidly produce a finished product that may be targeted across platforms.

Introduction to Cross-Platform Development

- Cross-platform development allows a single codebase to run on multiple platforms.
- Tools like Flutter, React Native, and Xamarin enable this approach.
- Faster development cycles with reduced costs.

Cross-Platform Frameworks Overview

- Flutter: Developed by Google, uses Dart.
- React Native: Developed by Facebook, uses JavaScript.
- Xamarin: Developed by Microsoft, uses C#.

Flutter: An Introduction

- Flutter is an open-source UI software development kit by Google.
- Enables developers to build natively compiled apps for mobile, web, and desktop from a single codebase.
- Uses Dart programming language.

Key Features of Flutter

- Fast Development with 'Hot Reload'.
- Expressive and Flexible UI.
- Single codebase for multiple platforms.
- Native performance across platforms.

Flutter's Architecture

- Flutter includes a rendering engine, framework, and widgets.
- The Skia engine powers the rendering and draws the UI.
- Widgets are used to build complex UIs by composing smaller elements.

Flutter's Layered Architecture

- Flutter's architecture is layered, separating concerns.
- Lowest Layer: Engine (written in C++) for rendering, events, and input.
- Middle Layer: Framework (written in Dart) for handling gestures, animations, and painting.
- Top Layer: Widgets, built using the framework for UI composition.

Flutter's Rendering Process

- The framework builds a widget tree representing the UI.
- The rendering engine turns the widget tree into pixels on the screen.
- Flutter performs rendering independently of platform-specific UI components.

Dart's Role in Flutter's Architecture

- Dart is the language used for building Flutter apps.
- Flutter uses Dart's reactive programming model for UI updates.
- Dart supports both Just-In-Time (JIT) and Ahead-Of-Time (AOT) compilation for speed and performance.

Flutter Installation and Setup

- Flutter SDK can be installed on macOS, Windows, or Linux.
- Official documentation guides setup for each platform.
- Install Android Studio or Xcode for emulators and device testing.

Setting Up Flutter on macOS

- Install Flutter SDK from flutter.dev.
- Install Xcode and enable command-line tools.
- Configure iOS simulators for testing Flutter apps.

Setting Up Flutter on Windows

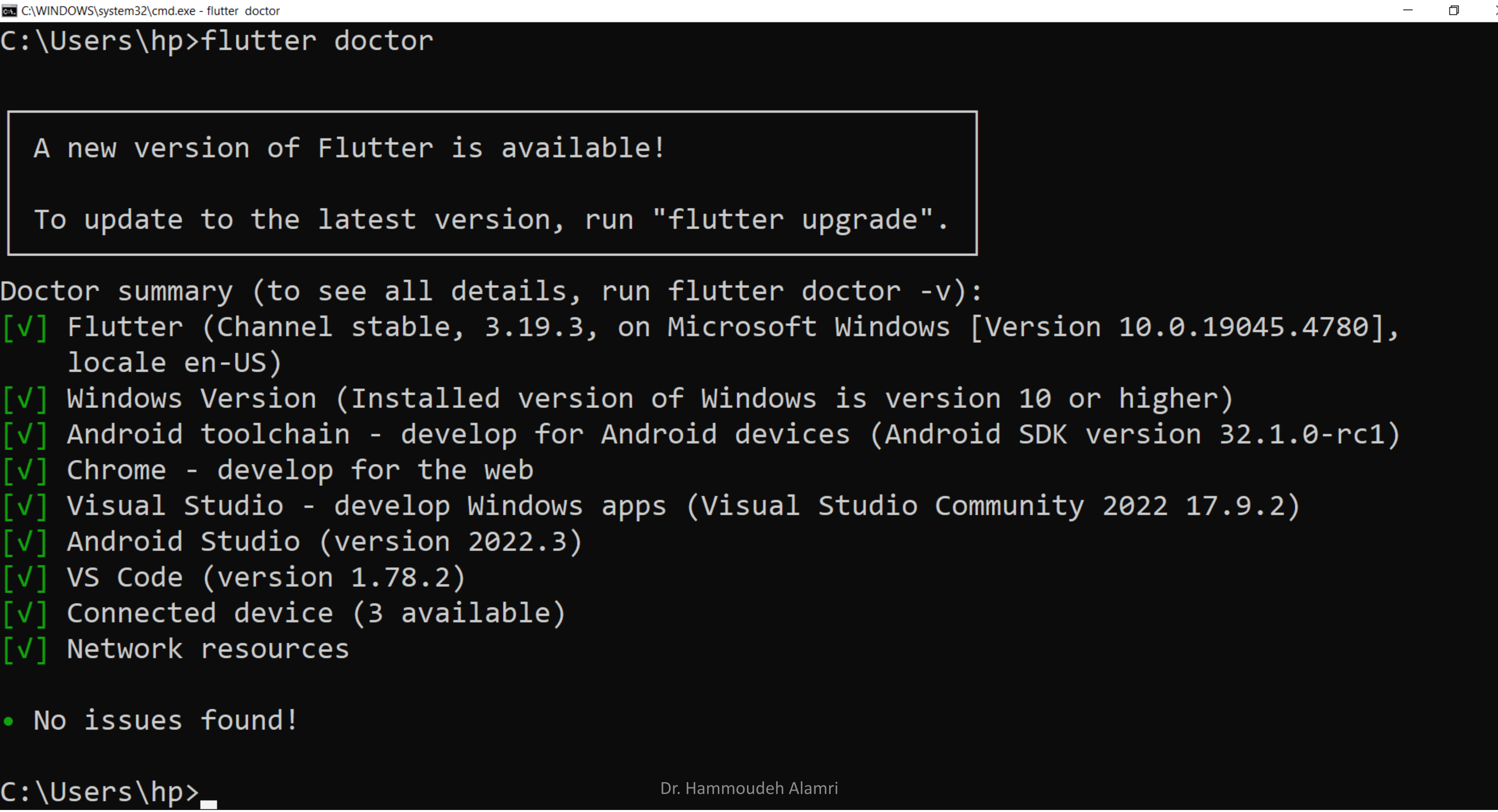
- Download and install Flutter SDK from flutter.dev.
- Install Android Studio for Android development and emulators.
- Use the command-line tools to run Flutter apps.

Tools for Flutter Development

- Flutter SDK: The core toolkit for building Flutter apps.
- Visual Studio Code: A lightweight IDE with Flutter extensions.
- Android Studio: Provides powerful features for Flutter development.
- Dart DevTools: Debugging and performance tools for Flutter apps.

Flutter Command-Line Tools

- `flutter doctor`: Checks for system requirements.
- `flutter create`: Creates a new Flutter project.
- `flutter run`: Runs the app on connected devices or emulators.
- `flutter build`: Compiles the app for release.



```
C:\WINDOWS\system32\cmd.exe - flutter  doctor
C:\Users\hp>flutter doctor

┌─────────────────────────────────────────────────────────┐
│ A new version of Flutter is available!                  │
│                                                         │
│ To update to the latest version, run "flutter upgrade". │
└─────────────────────────────────────────────────────────┘
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.19.3, on Microsoft Windows [Version 10.0.19045.4780],
    locale en-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 32.1.0-rc1)
[√] Chrome - develop for the web
[√] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.9.2)
[√] Android Studio (version 2022.3)
[√] VS Code (version 1.78.2)
[√] Connected device (3 available)
[√] Network resources

• No issues found!

C:\Users\hp>_
```

IDEs for Flutter Development

- Visual Studio Code: Lightweight, with Flutter extensions.
- Android Studio: Offers advanced debugging and emulator features.
- IntelliJ IDEA: Supports Flutter with a rich plugin ecosystem.

Advantages of Flutter

- Single codebase reduces development time and cost.
- High-performance apps with native-like performance.
- Customizable UI with Flutter's wide range of widgets.
- Supports mobile, web, and desktop apps.

Limitations of Flutter

- Larger app size compared to native apps.
- Limited access to platform-specific features, though plugins help mitigate this.
- The ecosystem is still growing, and some features may require native code integration.

Flutter vs React Native

- Flutter uses Dart, while React Native uses JavaScript.
- Flutter offers a fully custom rendering engine, while React Native relies on native components.
- Both have strong community support and are used by large companies.

Dart: The Programming Language for Flutter

- Dart is a client-optimized language for fast apps on any platform.
- Supports both just-in-time (JIT) and ahead-of-time (AOT) compilation.
- Strongly typed language with asynchronous programming support.

How Flutter Works: Hot Reload

- Hot Reload allows developers to instantly see changes in the UI without restarting the app.
- Speeds up development and bug fixing.
- Allows fast iteration for UI and logic changes.

Flutter's Widget-Based Architecture

- Everything in Flutter is a widget, from buttons to layouts.
- Widgets can be composed to build complex UIs.
- Supports both stateful and stateless widgets for dynamic or static UI.

Flutter in the Industry

- Used by companies like Google, Alibaba, and eBay.
- Popular for building high-quality apps on both mobile and web.
- Growing community with extensive support.

Real-World Apps Built with Flutter

- Google Ads: A mobile app for managing Google Ads campaigns.
- Alibaba: E-commerce app in China using Flutter for high performance.
- Reflectly: A mental health journaling app with rich UI.

Conclusion: The Future of Mobile App Development

- Cross-platform frameworks like Flutter are shaping the future of mobile app development.
- Native performance with reduced costs and faster development times.
- Expect more advancements in mobile, web, and desktop app integration.