CELESTIAL SYSTEMS

FLUTTER V/S NATIVE IOS

A comparative analysis based on Shiftboard use case

This whitepaper provides a 1:1 comparison of the same exact application built using Flutter & Native iOS for our client, Shiftboard (www.shiftboard.com).

Topics Included

- What are native & cross-platform apps?
- Why should entrepreneurs consider developing cross-platform mobile apps?
- What is Flutter?
- Overview of Flutter v/s iOS
- Flutter & iOS Architecture
- Use Case Analysis – Shiftboard Mobile Application
- Flutter Business Advantages

What are native & cross-platform applications?

Native vs. Cross-platform is an age-old debate. But then, what does each term mean?

A native mobile app is an application that is developed exclusively for a particular platform, i.e., to meet the requirements of a particular operating system by using its SDK and primary technology stack, as well as hardware memory, camera, sensors, and other programs installed on a device.

Cross-platform apps are those developed to function for multiple mobile platforms and are compatible with multiple operating systems, while maintaining a single code base.

Native Apps	Cross-Platform Apps
High performance	70–90% reusable code
Robust functionality	Easy maintenance & updates
Seamless user experience	Broader reach
	Shorter time to market

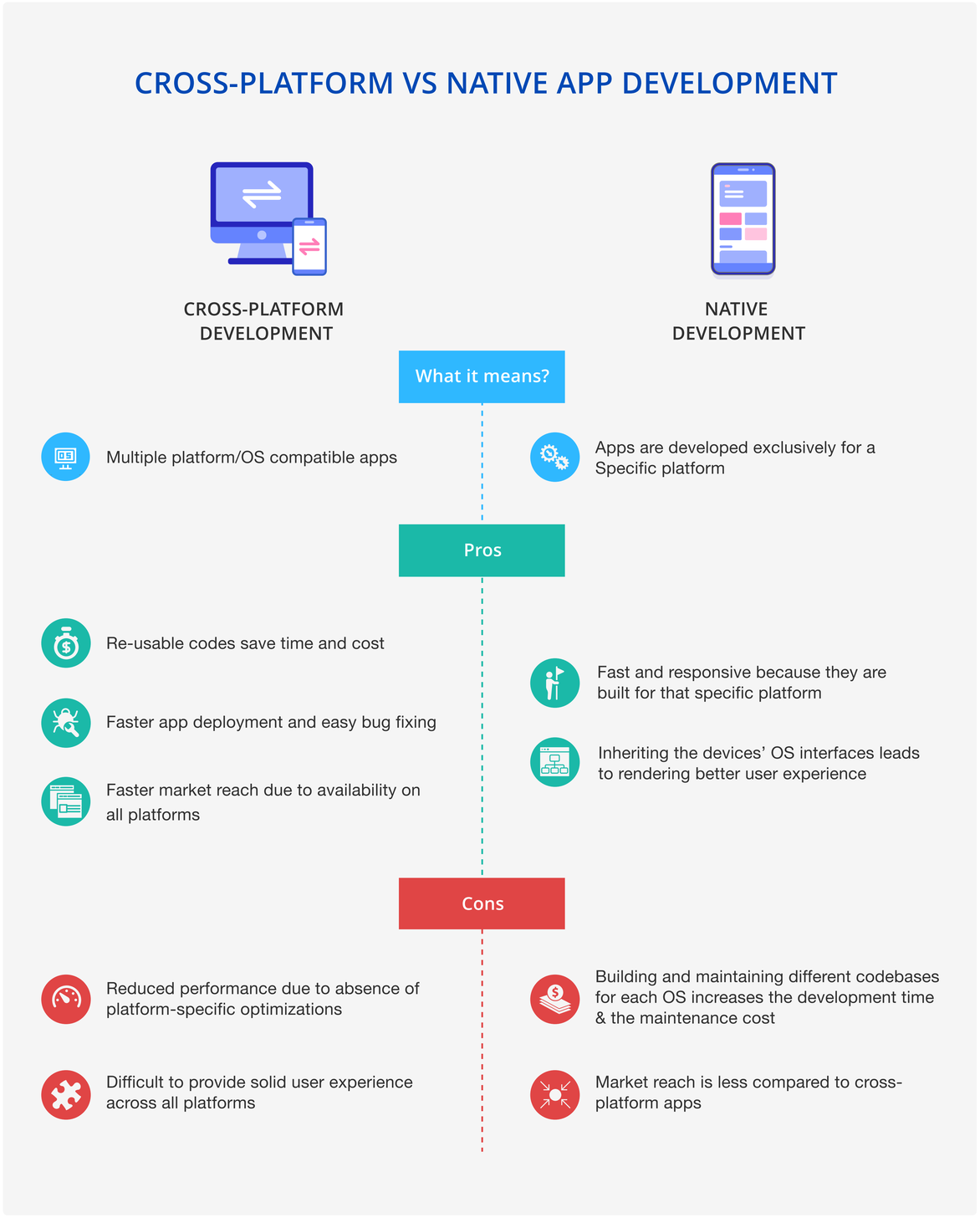
Why should entrepreneurs consider developing cross-platform mobile apps?

Many entrepreneurs and business owners have already embraced the concept of cross-platform app development. Building cross-platform apps help developers finish the project rapidly by a single-codebase, and enables entrepreneurs to address a huge audience through a user-friendly app.

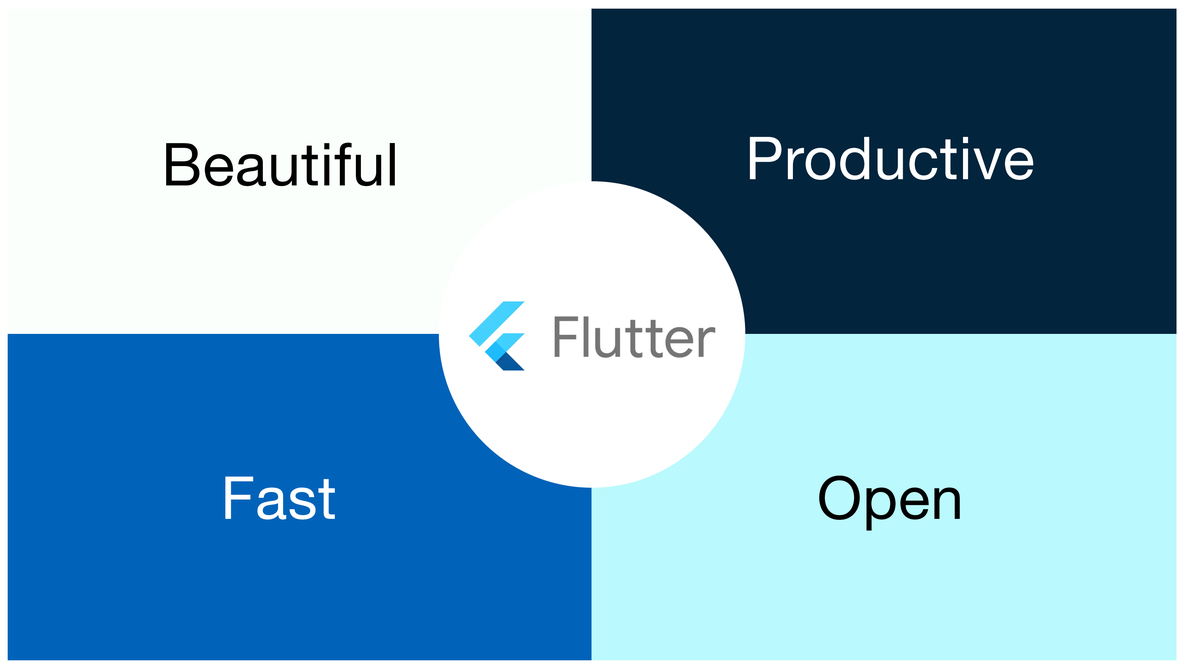
Now, the question is,

1. **How to launch an app under limited budget with around 3.1 billion smartphones and tables in use all across the world?**
2. **How to set a strong foothold in the market and stand out in the crowd where around 305 million start-ups rollout apps annually?**

To address these two crucial questions, entrepreneurs need to focus on building cross platform applications.



What is Flutter?



Flutter is an open-source Software Development Kit (SDK) launched by Google to develop robust, scalable, and high-performing mobile apps for multiple operating systems by using a single codebase. It uses the Dart programming language for developing both iOS and Android apps. By saving time and effort and by lessening the amount of coding, Flutter is perfect for start-ups building a minimum viable product (MVP). Rather than being a framework, it is a complete SDK (software development kit) that contains everything you require for cross-platform mobile app development.

Built on the four pillars – beautiful, productive, fast, and open, Flutter helps build 'beautiful native apps in record time' and is picking up steam over other mobile frameworks like React Native, Xamarin, Angular JS. Flutter takes a unique method, unlike any other cross-platform frameworks. This enables it to achieve near-native performance. Also, the Dart language makes it easier for developers in making the apps attractive and intuitive to the users.

Flutter is here to stay, sustain and evolve as a platform with growing market demands.

Key things to know about Flutter

Programming Language	-	Dart
For whom is it easier to start	-	For C# or Java engineers
Ready-made widgets & components	-	Uses its own widget library to display Flutter UI Huge library of plugins and widgets
Development tools & Documentation	-	Great documentation & starter toolkit
Choose if	-	UI is a key priority of your app
Do not choose if	-	Your app should be smaller than 4mb
Time to Market	-	Faster
Code Reusability	-	80% - 90%
Apps	-	Alibaba, Google Ads etc

Keynote

According to the GitHub's OCTOVERSE report of 2019, Flutter is one of the fastest growing open-source projects and it has climbed to the #2 spot.

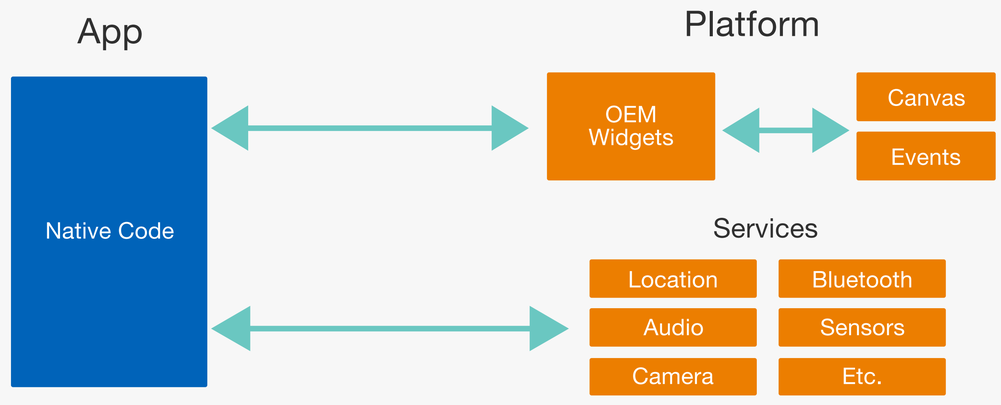
Overview of Flutter v/s iOS

Initially, the programming language used by Apple was **Objective-C** for iOS development, though it was not widely accepted by all developers. But fast-forward to 2014, **Swift** was introduced which was type-safe and used concise but expressive syntax, and developers just loved it. It was also fully compatible with Objective-C and the existing iOS APIs.

Flutter, unlike the other cross-platform solutions, helps build and deploy visually attractive, fast mobile apps on both Android and iOS platforms. The first "stable" version of the Framework, Flutter 1.0 was released on December 4, 2018, and is the complete UI kit that allows developers to develop high-scale applications with the best typography, icons and scrolling behaviours.

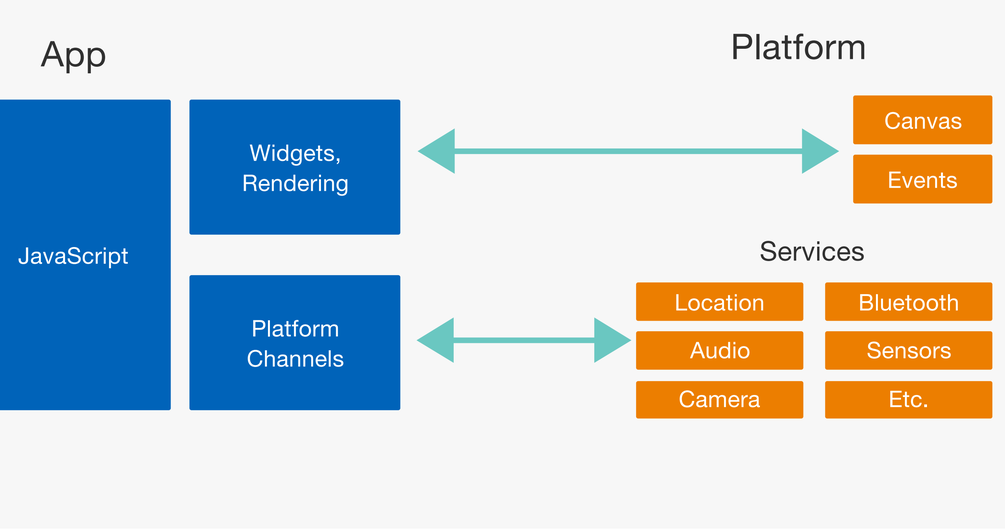
Understanding the Flutter & iOS Architecture

Native Application



There are numerous languages that can be used for building native mobile applications and a few of examples of these are: Kotlin, Java and Swift. In native apps, whether it is Android or iOS, the native app code talks to the Platform to create OEM Widgets or for accessing various Services like audio, sensors, camera, etc. The widgets are rendered to a screen canvas, and events are sent back to the widgets. This architecture restricts the widgets to be reused across all platforms as they are OEM specific. This is the reason that we need to write the whole application separately for each platform.

Flutter Apps



Flutter solves the most challenging part of the other cross-platform frameworks, i.e. getting rid of the BRIDGE. Flutter does not use the OEM widgets; it provides its own widgets. Flutter moves the widgets and the renderer from the Platform into the app, which allows them to be customizable and extensible. This also helps Flutter to maintain a consistent 60 FPS.

The one reason that Flutter has been widely adapted by developers across the globe is the similarity of Dart's language with existing mobile languages like Java/Kotlin/Swift.

Keynote

Flutter's framework is backed by advanced API that enables you to leverage faster, cleaner, and smoother app animations.

Use Case Analysis – Shiftboard Mobile Application

Let us take Shiftboard project as an example and look into some of our findings while we built the same mobile app in Flutter & Native iOS.

Who is Shiftboard?

Shiftboard is an award-winning frontline workforce scheduling software company backed by innovative technology. By optimizing the workforce scheduling to align with strategic business goals, it has a proven track record of delivering tangible results.

Consumer behaviour keeps on changing; adopting strategies that help move where the customers are gives your brand a competitive edge. This is one reason companies are moving rapidly towards making mobile apps as business tools and assets.

Here, the project requirement was to build a B2B Mobile SaaS app, for an already built web application, designed for estimating and managing employee time schedules and thus improving the internal employee engagement of the organization.

Faster development time, good performance, faster time-to-market, and user-friendly UI are a few major factors which let choose the best solution for a mobile app development.

Here is a comparative study on the two different solutions – native iOS and Flutter - to build a business mobile application for Shiftboard.

Business Requirements

- Design and develop a SaaS mobile application with employee time & schedule management features
- Customize a calendar with intuitive UI
- Leave application tracking feature

Challenges

- To build the app in a short period of time
- Complex logic & Customized UI for iOS

Features Included

- Schedule Management Feature
- Notifications Feature
- Leave Management Feature

The main comparison points are as follows:

1. Development Speed
2. Programming Language
3. Customized UI
4. Testing
5. Development cost

1. Development Speed

Creating an app is a time-intensive process. There are a lot of factors that affects development speed like the functionality of the app, app complexity, idea validation, testing and even more.

However, reducing the app development time, without trading the app quality, is a requisite in keeping the app development costs from spiralling out of control.

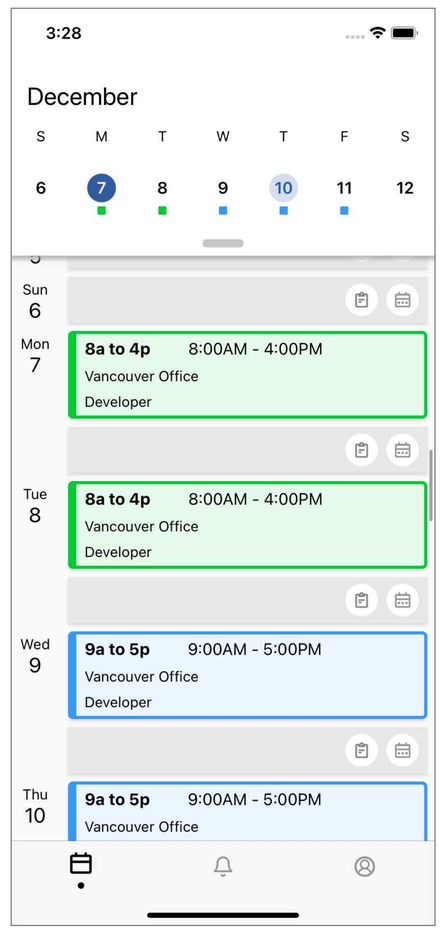
Swift

For native iOS, the initial build phase took more than 25 days to complete. The target was to implement a Schedule Management system that can reduce conflicts in managing shifts, especially with complex shift patterns for a larger workforce. With more discernability to the roadmap and workflow, this feature makes it easier to schedule work. Creating the Schedule List UI took more of developmental time as the TableView caused many issues while implementation, and lots of research was needed. Some issues took more than 3 days to resolve.

Flutter

The initial build time was less compared to native iOS and took around 17 days for its completion. With ready-to-use widgets, customizing the UI was easier with Flutter.

Schedule Management feature was implemented with less effort compared to native due to the highly customizable widgets in Flutter. Creation of Schedule List UI in Flutter took only 2 hours of work, whereas the native iOS took 8 hours of work.



WINNER:

Flutter

The ready-to-use widgets and the hot reload feature of Flutter reduced the app development time considerably.

2. Programming Language

When considering programming languages, frameworks, and SDKs for mobile apps, we have to not just consider the frontend (UI) development environment but also be aware of the backend (server-side) development environment. For Shiftboard we were looking for minimal coding to reduce the app complexity and provide smoother user experience.

Swift

Shiftboard mobile app was developed with iOS Native using Swift language. Swift helps write consistent code and provides safeguards to prevent errors and improve readability.

Flutter

Dart framework has most of the components inbuilt, hence it is bigger in size and often does not require the bridge to communicate with the native modules. We used Dart to create and customize the entire UI layout providing the users with easy navigation and flexibility in the application. The complete development took half the time of native app.

WINNER:

Flutter

Dart and Flutter improve performance, safety, and tools for cross-platform mobile app development.

3. Customized UI

We wanted to build an interactive and engaging GUI for Shiftboard mobile app to improve user experience with preferred choices and personalization.

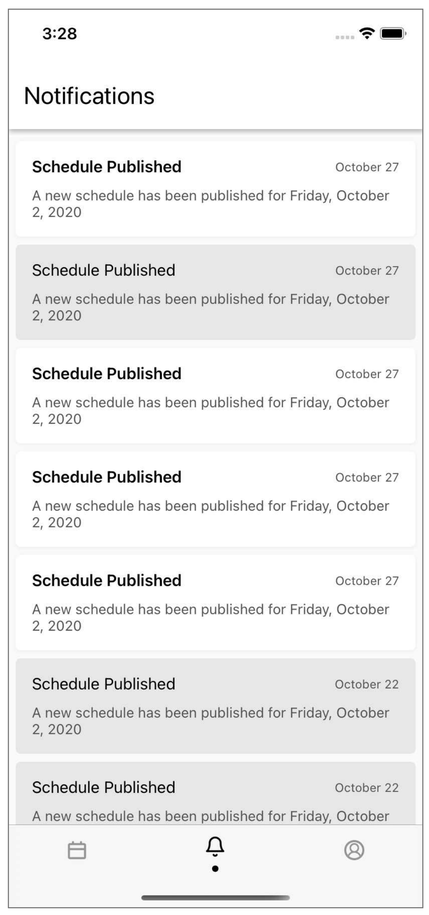
Swift

Designing the customized UI was a tough task, and the absence of a good Library to implement all features without hassle took more development time. We researched FSCalendar vs JTAppleCalendar for Calendar Schedule Summary Pagination and then finalized JTAppleCalendar for final customized integration.

Flutter

One major difference here is that Flutter gives a natural platform feel due to Material Design (Android) and Cupertino (iOS) widgets, allowing it to imitate the platform design itself.

Customizing the UI in Flutter was much easier compared to native. Calendar research took less time than native as we found multiple libraries for calendar, and finalized one to have heavily customized implementation.



WINNER:

Flutter

Flutter took only 4 hours compared to native in creating the calendar view and customizing UI in Flutter is easy compared with Native.

4. Testing

The project was started in the month of September and there was a time constraint on the development. With automated tests and fewer bugs, the target was set to reduce the time-to-market to at least half the pre-set time.

Swift

Due to the complexity in UI, we were getting a lot of bugs and UI bugs fixing took more time as compared to Flutter.

Flutter

Flutter comes with an enriched set of widget testing features at various levels such as unit, widget and integration stage. Due to this reason, fixing UI bugs did not take much time.

WINNER:

Flutter

Testing was completed in 80 hours for the native app and for Flutter it took just 60 hours.

5. Development Cost

The objective was to develop a cost-effective application in a shorter period of time.

Swift

The complexity in UI, the push notifications feature, and the schedule rework took more time for development, hence more developmental cost. Also, for MSAL implementation part, MSAL Keys, URL & Credentials were not provided prior. Figuring out the correct implementation took extra time, adding to the cost.

Flutter

Flutter uses significantly fewer human resources, material, and time, for developing apps across platforms making it cost-effective. For Shiftboard, after all the specified features were implemented and tested for bugs, the application was ready for market earlier as compared to native iOS. This reduction in time estimation has helped reduce the development cost too.

WINNER:

Flutter

Flutter by providing customized widgets and UI is a clear winner in terms of reducing development cost.

Expected benefits of Flutter implementation

- Increased time efficiency & productive hours by 80%
- 70% faster time-to-market
- Automated and faster schedule management and leave requests
- Better user interaction with customized calendars with intuitive UIs

Findings

Flutter Development Total Time: **229:50:00**

iOS Development Total Time: **400:30:00**

Considering the challenges in the application, comparison between iOS and Flutter

Custom UI Development:

iOS: Time consuming and difficult

Flutter: Very easy

Microsoft Authentication Library:

iOS: Normal Implementation Time

Flutter: Extra effort spent on research as there were multiple options available

Key Takeaway

"With Flutter, developers have to write a single code base to build cross-platform applications that closely perform like a Native app in terms of features, UI/UX and functionality".

Conclusion

Based on our findings we can conclude the following:

- Development time was considerably faster for Flutter; this was because custom components were to be created which was not easy in iOS
- Performance was a huge concern, but Flutter renders at 60 fps which is fairly good for an application that isn't graphic intensive
- Development effort would be much less if you were to build a cross platform app for multiple platforms, like iOS, Android etc. as opposed to building native apps for each.

In this subjective evaluation, Flutter emerged to be the clear winner. That being said, our suggestion would be to take up each project as a separate case and evaluate them based on their functionality for a native or cross-platform approach. But if development effort and time are the prime concerns, then cross-platform would be the way to go; Flutter, in this case.