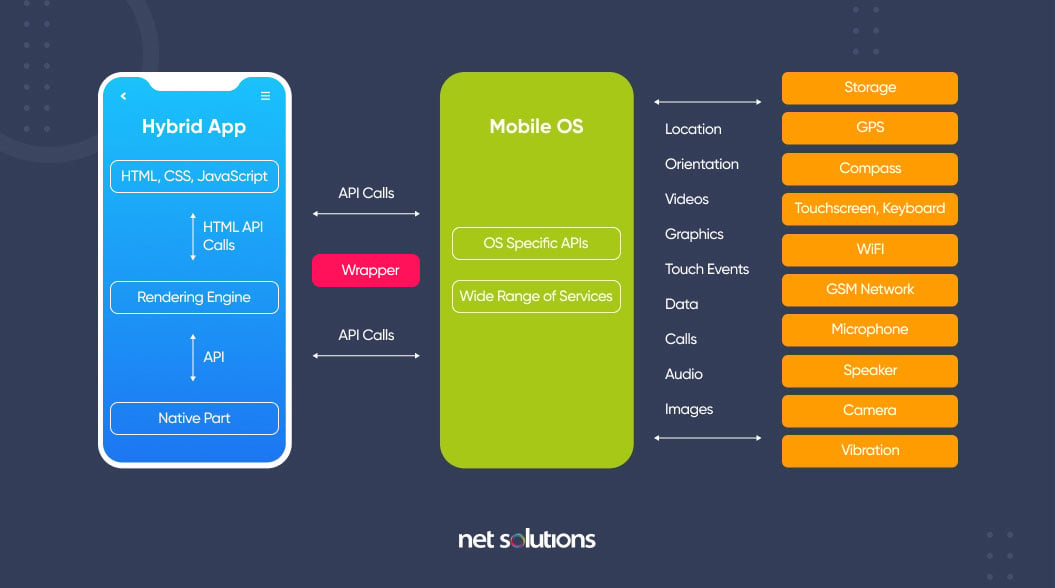
**Hybrid Mobile Application Architecture**

Hybrid mobile apps leverage both native and web solutions. Hybrid apps use native apps as “shells” for the back-end, but platform-neutral [JavaScript](https://www.netsolutions.com/javascript-development-company), HTML and CSS for the front-end. Hybrid apps use plugins such as Apache Cordova or Ionic Capacitor to access native platform features.

Hybrid mobile apps are among the fastest apps to create across a variety of platforms and easy to update, but are not appropriate for complex, interactive, or feature-rich applications.



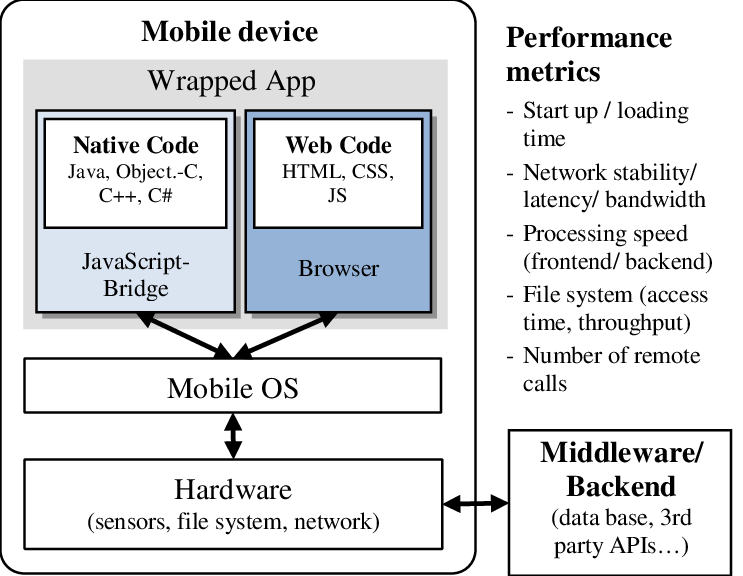
This solution is a blend, hence the name hybrid, of both native and web solutions. Where the core of the application is written using web technologies (HTML, CSS, and JavaScript), which are then encapsulated within a native application. Through the use of plugins, these applications can have full access to the mobile device’s features. To better understand this approach, let’s break down how it all fits together.

The heart of a hybrid-mobile application is still just an application that is written with HTML, CSS, and JavaScript. However, instead of the app being shown within the user’s browser, it is run from within a native application and its own embedded browser, which is essentially invisible to the user. For example, an iOS application would use the WKWebView to display our application, while on Android it would use the WebView element to do the same function.

This code is then embedded into a native application wrapper using a solution like Apache Cordova (also known as PhoneGap) or [Ionic’s Capacitor](https://capacitor.ionicframework.com/?__hstc=13779304.ff773afa2806bd7f052bf65a96c2684f.1675071220943.1675071220943.1675071220943.1&__hssc=13779304.2.1675071220944&__hsfp=3756997744" \t "_blank). These solutions create a native shell application that is just the platform’s webview component in which it will load your web application. This gives you the ability to create and publish true native applications that can be submitted to each of the platform’s app stores for sale.

Additionally, both Cordova and Capacitor have a plugin system that allows you to extend beyond the limitations of the ‘browser’ and access the full suite of capabilities of a user’s mobile device. So, if you wanted to use TouchID on an iOS device as a login option, or wanted to connect to a Bluetooth device, this can be easily done by installing a plugin. These plugins are created by a wide range of developers and many are actively supported. Ionic even offers a complete ecosystem of supported plugins as part of its [Enterprise](https://ionicframework.com/enterprise?__hstc=13779304.ff773afa2806bd7f052bf65a96c2684f.1675071220943.1675071220943.1675071220943.1&__hssc=13779304.2.1675071220944&__hsfp=3756997744) solution. So, the limitations of a web-only application are easily overcome, allowing your application to have parity with native applications in their features.

However, there are some drawbacks with this option. Similarly to the web-only application solution, the UI library has to be recreated. Here is where solutions like Ionic, NativeScript, Xamarin, React Native, and others step in. These options all provide robust UI components that look and feel like their native counterparts, giving you a full suite of building blocks for your hybrid mobile app.



**Features of hybrid applications**

Hybrid applications features include the following:

* the ability to function whether the device is connected or not;
* integration with the mobile device's file system;
* integration with web-based services; and
* an embedded browser to improve access to dynamic online content.

**How hybrid applications work**

Hybrid apps work in the same manner as web apps, but are downloaded to the device like native apps. Similar to web apps, developers typically write hybrid apps in [HTML5](https://www.techtarget.com/whatis/definition/HTML5), CSS and JavaScript. Hybrid apps run code inside a container. The device's browser engine renders HTML, JavaScript and native [APIs](https://www.techtarget.com/searchapparchitecture/definition/application-program-interface-API) to access device-specific hardware.

Although a hybrid app will typically share similar navigation elements as a web app, whether the application can work offline depends on its functionalities. If an application does not need support from a database, developers can make it function offline.

**Hybrid application pros and cons**

Pros of hybrid apps include the following:

* will operate on different platforms;
* faster build time compared to native apps;
* cheaper to develop compared to building two versions of a native app for two different platforms;
* easier to launch patches and updates; and
* can work online and offline.

**Some cons, however, include the following:**

* Variations due to leaning development on one platform may occur -- for example, if a development team leans their work on one platform, another supported platform may lack in quality or suffer from bugs.
* The appearance of an application may vary from platform to platform.
* Developers need to test the application on a range of devices to ensure proper operation.
* User experience ([UX](https://www.techtarget.com/searchcio/definition/UX-user-experience)) may fail if the user interface ([UI](https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI)) isn't similar to and well enough designed to what browsers the user is used to.