Temasek Junior College

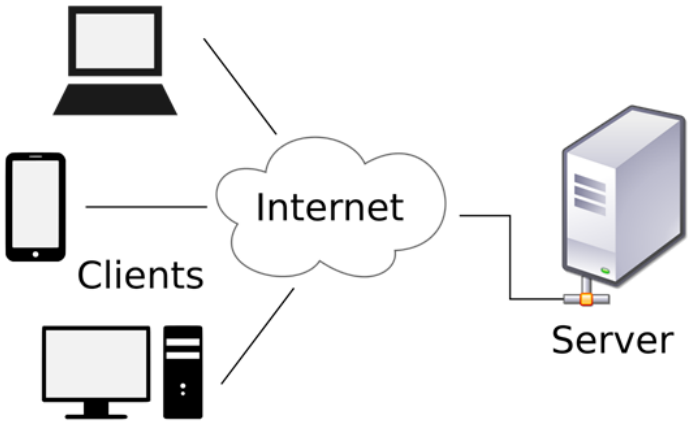
2023 JC2 H2 Computing

Web Applications and Native Applications

1 Web Applications

Web applications, also known as web apps, are client-server programs that run on remote servers. They can be accessed by users (clients) through web browsers using the relevant uniform resource locators (URLs).

Web applications can be seen as a specific variant of client-server software, where the client software is downloaded to the client machine when visiting the relevant web page using standard protocols such as hypertext transfer protocol (HTTP).



Web applications use web documents written in a standard format with web technologies that are supported by a variety of web browsers, such as hypertext markup language (HTML), cascading style sheets (CSS), and JavaScript. Some common web applications include web-based email, online office applications, online retail sales and auctions, browser games, social media, instant messaging services, etc.

1.1 Benefits of Web Applications

Cross-Platform Compatibility
Web apps are designed to be platform-agnostic i.e. they can run on any device with a web browser. This includes desktops, laptops, tablets, and smartphones. During the session, the web browser interprets and displays the pages, and acts as the universal client for any web application. Users can thus access the same web app across different platforms, ensuring a consistent experience and broad reach.

No Requirements for Installation
Web apps eliminate the need for installation or downloads. Users can access them directly by entering the app's URL in a web browser. This ease of access reduces barriers and friction, leading to a faster onboarding process.

Updating and Application Maintenance
Web apps allow for centralized updates and maintenance. Developers can deploy changes on the server, making them available to all users. Hence client web software updates may happen each time the web page is visited. Updates can thus be applied seamlessly without requiring users to manually update the app, ensuring everyone has access to the latest version.

Cost-Effective Development
Developing web apps can be cost-effective as web technologies are widely accessible, and developers can use a single codebase to target multiple platforms. This reduces development time and resources needed for platform-specific development.

Greater Discoverability
Web apps are inherently discoverable through search engines. By optimizing the app's website and content for search engines, web apps can benefit from organic discoverability, potentially attracting a broader user base.

1.2 Shortcomings of Web Applications

Internet Connectivity Dependency
Web apps require an internet connection to function. Limited or no connectivity can hinder the app's usability. However, the development of progressive web apps (PWAs) aims to mitigate this limitation by enabling certain features and content to work offline.

Limited Device Feature Access
Web apps have limited access to device features. While modern web application programming interfaces (APIs) provide access to certain hardware capabilities (such as camera and geolocation), the range and depth of access may be limited. This limitation may impact certain use cases that rely heavily on specific device features.

Performance Constraints
Web apps may face performance constraints. They rely on web browsers, which can introduce variations in rendering speeds and performance optimizations. However, advancements in browser technologies and optimizations, such as hardware acceleration and improved JavaScript engines, have significantly improved web app performance.

Security considerations
Web apps are susceptible to web-based vulnerabilities like cross-site scripting (XSS) and cross-site request forgery (CSRF). Proper security measures, such as input validation, output encoding, and secure communication protocols, must be implemented to protect against these threats.

2 Native Applications

Native applications are software applications developed specifically for a particular platform or operating system. They are built using platform-specific programming languages, frameworks, and development tools. Native apps are installed directly on the user's device.

2.1 Benefits of Native Applications

Optimum Access to Device Hardware and Features
Native apps can be designed to have settings that allow an optimal level of access to device hardware and features. Developers can utilize the full range of device capabilities, including camera, microphone, GPS, accelerometer, gyroscope, NFC, and more. This enables the creation of highly specialized and feature-rich applications.

Potential for High Performance and High Responsiveness
Native apps offer enhanced performance and responsiveness. They are optimized for the specific platform, utilizing low-level APIs and hardware acceleration. Native apps can deliver smoother animations, faster load times, and a more fluid user experience.

Enhanced User Experience
Native apps can be designed to provide a rich and immersive user experience. They can leverage platform-specific UI components, design patterns, and navigation gestures, resulting in a seamless and familiar interface for users. This level of customization enhances user engagement and satisfaction.

Offline Capabilities
Native apps can function offline, allowing users to access certain features and content without an internet connection. By storing data locally on the device, native apps can continue to operate and provide functionality in offline or low-connectivity scenarios.

2.2 Shortcomings of Native Applications

Longer Development Time and Higher Development Costs
Developing native apps requires separate codebases for each platform. Platform-specific programming languages (e.g., Swift for iOS, Java or Kotlin for Android) and development environments must be used. Additionally, ongoing maintenance efforts, updates, and bug fixes must be performed separately for each platform. Developing and maintaining separate codebases for multiple platforms increases the development timeline, cost, complexity, resource demands.

App Store Approval
Native apps need to go through the app store approval process, adhering to guidelines and policies specific to each platform. The approval process can introduce additional time and potential challenges, especially if the app requires compliance with strict guidelines or regulations.

Limited Distribution and Discoverability Outside App Stores
App store distribution may limit app discovery and require adherence to specific distribution rules. This can make it challenging for users to find them outside of app store ecosystems, limiting the reach and visibility of the app to users who do not actively search within app stores.

Fragmentation and Compatibility Challenges

The mobile computing landscape is fragmented, with multiple OS versions, device models, and screen sizes. Native apps must be optimized and tested across a wide range of devices, OS versions, and resolutions, which can increase development and testing efforts.

3 Web Applications vs. Native Applications

The table below provides a comparison of the two types of applications:

Criteria	Web Applications	Native Applications
Performance	• Rely on web browsers, which introduce additional layers and potential performance constraints.	• Generally offer superior performance compared to web apps due to their direct access to device resources, hardware acceleration, and platform optimization. • Compiled into machine code, allowing for efficient execution and utilization of the device's capabilities.
User Experience	• Can achieve responsiveness and adaptability. • May lack the native feel and consistency provided by native apps.	• Provide a tailored and consistent user experience, adhering to the design principles and guidelines of the platform. • Can leverage platform-specific UI components, navigation patterns, and gestures, resulting in a native and intuitive user interface.
Development and Maintenance	• Generally have shorter development timelines and lower costs as developers can use a single codebase, to target multiple platforms, reducing the need for platform-specific development efforts.	• Offer greater control over the user experience and access to device capabilities. • However, requires platform-specific expertise, multiple codebases, and ongoing maintenance efforts for each platform.
Accessiblity	• A single codebase facilitates accessibility from any device with a compatible web browser, eliminating the need to switch between platforms or download different versions of the app.	• Requires users to switch between different platforms (e.g., iOS, Android) to gain access to the same application, resulting in fragmentation and potentially limited user base.
Offline Functionality	• Typically require an active internet connection to function	• Have inherent offline capabilities, allowing users to access certain features and

	• Progressive web apps (PWAs) have introduced offline capabilities using technologies like Service Workers and local storage.	content even without an internet connection. • Can store data locally and synchronize changes once connectivity is restored.

4 Web Applications or Native Applications?

Web applications and native applications offer different strengths and limitations.

The choice between web and native depends on factors such as project requirements, target audience, development resources, and desired user experience.

Web apps provide cross-platform compatibility, easy updates, and cost-effective development but may have limitations in device feature access and performance.

On the other hand, native apps offer superior performance, access to device capabilities, offline functionality, and a tailored user experience but require separate development efforts, longer timelines, and platform-specific expertise.

Developers should consider these factors and prioritize them according to the specific project's needs to make an informed decision.