Chapter 23 Web Applications Principles

Contents

1 Native applications
2 Web applications
3 Native Applications vs Web Applications
4 Usability Principles in the Design of Web Applications
  4.1 Visibility of system status
  4.2 Match between system and the real world
  4.3 User control and freedom
  4.4 Consistency and standards
  4.5 Error prevention
  4.6 Recognition rather than recall
  4.7 Flexibility and efficiency of use
  4.8 Aesthetic and minimalist design
  4.9 Help users recognize, diagnose, and recover from errors
  4.10 Help and documentation
5 ASCII
6 Unicode

Annex 1 – “Newer” Types of application

Annex 2 – Web application vs Website

Syllabus Learning Outcomes

4.2 Web Applications

Understand the concepts and techniques for developing web applications.
4.2.1 Describe the differences between web applications and native applications.
4.2.2 State and apply usability principles in the design of web applications.

3.2 Character Encoding

Understand the use of ASCII code and Unicode to represent characters.
3.2.1 Give examples of where or how Unicode is used.
3.2.2 Use ASCII code in programs.

1 Native Applications

Native applications are installed directly on the device that can work with no internet connection depending on the nature of the application. They are developed specifically for a particular platform or operating system such as Apple iOS, Android or Windows. They can also be designed to take advantage of the device features such as the power of the processor, specific hardware such as GPS and other components of the system such as the notification system and gestures. These apps have more safety and security than web apps, as native apps must be approved by the App Store

Examples: Calculator, Mobile games

2 Web Applications

Web application is a software application accessed and used through a web browser. A Browser is an application that is used to browse the internet. Users don't need to download and install web applications on their devices. Web applications can be accessed from any device with a web browser and an internet connection, regardless of the operating system. They are typically developed as web pages in HTML and CSS while the interactive components of the applications are developed in programming languages such as JQuery and JavaScript.

Examples: LuminPDF, Google Docs

3 Native Applications vs Web Applications

Deployment and maintenance (updates) for a web-based application require deployment on a single set of server machines. However, for native applications, the deployment and any maintenance/patch has to be done on individual client machines separately.

Web applications can also be accessed from anywhere (most locations), so there is no location constraint. However, as native applications are confined to a standalone machine, they can only be accessed from the machines they are deployed in.

Web applications thus, rely heavily on internet connectivity, for their operations. And thus, developers need to account for network reliability before deciding to embark on such projects. Native applications, on the other hand, don’t require the internet for their operations. Some applications just require internet connectivity at the time of update.

Web applications are also often platform-independent, i.e. they can work on different types of platforms with the only requirement being the provision of a web browser.
Native applications, on the other hand, need to be developed separately for different platform machines. (Windows, Linux, Unix, Mac etc)

Web applications lend themselves to higher security risks as they are inherently designed to increase accessibility. This is a trade off that developers must take note and circumvent accordingly with the latest technologies and cyber security intelligence. Native applications, on

the other hand, have better authorisation and administrators have better control, hence they are more secure.

Web apps lack consistency in user experience due to their heavy dependency on browsers. Certain features or images may look different on different browsers. Buttons and menu bar features may be challenging to access from mobile browsers. Browser window resizing may impact the look, feel, and functionality of the web application. Users tend to have a better experience on native mobile apps. For instance, the native app fills the screen and takes control of the entire device. Users get more out of the native app because they are comfortable with the interactions. The native app can also send push notifications to users and get them to re-engage.

The table below shows a summary of the above points of difference.

Native Applications	**Web Applications**
Developed for a specific operating system or platform. As native apps are confined to a standalone machine, they can only be accessed from the machines they are deployed in.	Developed to be accessed via the device's internet browser. Web applications can be accessed from anywhere (most locations), so there is no location constraint.
Need to be installed in the device to function. Native applications need to be developed separately for different platform machines. (Windows, Linux, Unix, Mac etc)	No need to be downloaded or installed. Web applications are platform-independent, they can work on different types of platforms with the only requirement of a web browser.
Have access to system resources such as GPS, camera etc.	No access to system resources such as GPS, camera etc.
May be able to work offline. Native applications don't require the internet for their operations. Some applications just require internet connectivity at the time of update.	Need an active internet connection to work. Web applications rely heavily on internet connectivity, for their operation.
Comparatively faster.	Comparatively slower, depending on the internet connection.
Relatively safer and more secure.	Relatively less secure.

Native applications, on the other hand, have better authorization and administrators have better control, hence more secure.	Web applications are at higher security risks as they are inherently designed to increase accessibility.
May incur maintenance and update costs. Deployment and any maintenance/patch are done on individual client machines separately.	Updated automatically. Deployment and maintenance (updates) for a web-based application require deployment on a single set of server machines.
Tend to give better user experience. Native app fills the screen and takes control of the entire device	Web apps lack consistency in user experience due to their heavy dependency on browsers. Certain features or images may look different on different browsers

4 Usability Principles in the Design of Web Applications

Adapted from: [Jakob Nielsen's 10 general principles for interaction design]

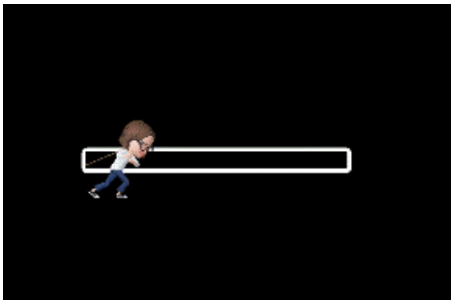
User interface UI refers to the screens, buttons, toggles, icons, and other visual elements that you interact with when using a website, app, or other electronic device. User Experience UX refers to the entire interaction you have with a product, including how you feel about the interaction. While UI can certainly have an impact on UX, the two are distinct, as are the roles that designers play.

Jakob Nielsen has defined 10 usability heuristics — broad principles for usability — that create better user experiences (UX). Think of these as rules of thumb for designing user interfaces (UI) that are intuitive to users to interact with.

Nielsen's usability heuristics are general principles, not specific guidelines. They should guide you in creating a user experience that feels intuitive and clear to people visiting your website or using your products. The goal is to create an interface that's delightful for users while clearly orienting them on their journey.

4.1 Visibility of system status

The application design should always update the users of the system's current status clearly. Prompt and meaningful feedback should be provided so that the users can determine the next steps. Such predictable interactions will create trust in the system. For example, show loading indicators during data processing.



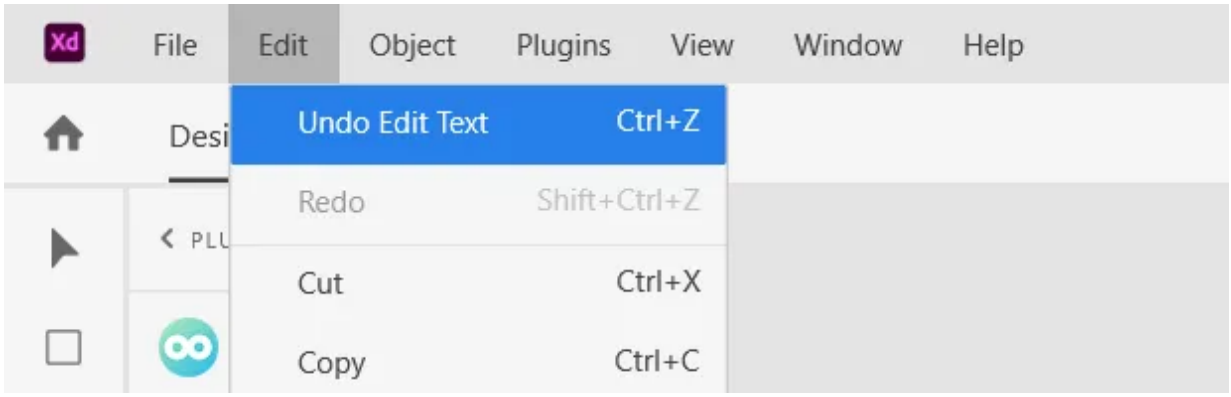
4.2 Match between system and the real world

The terms, images, icons, concepts and words used in the application should also be familiar to the user rather than system-oriented terms. This can be achieved by following real-world conventions so that the information can appear in a natural and logical order which leads to desired outcomes. This is what we call natural mapping. With such intuitive design, users can easily learn how to use the interface and therefore, enhances user experience.

Resemblance of real life objects in icons

4.3 User control and freedom

Applications that follow good usability principles can accommodate users' mistakes by allowing them to undo and redo easily. They also provide a clear way to exit any unwanted state or interaction. Such design gives users a sense of freedom and control and avoids frustration of getting stuck at a certain state.

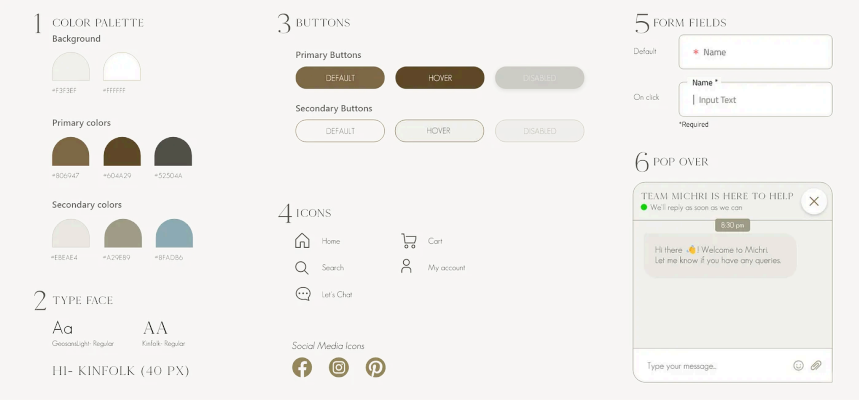


Undo and Redo gives flexibility to users

4.4 Consistency and standards

It is also important to improve users' learnability by ensuring internal and external consistency in using words, symbols and actions. Internal consistency refers to ensuring that your application design follows consistent conventions and design patterns throughout the interface, like using the same button styles and colors for similar actions within a single digital product or a family of products. External consistency refers to following the established industry conventions.

Jakob's Law of Internet UX states that people spend most of their time using their digital products other than yours. As such, users already have certain expectations in using their digital devices. Failure to maintain consistency may increase the users' cognitive load as they are forced to learn something new or contrary to what they are used to.



Following a style guide is a good practice to maintain consistency

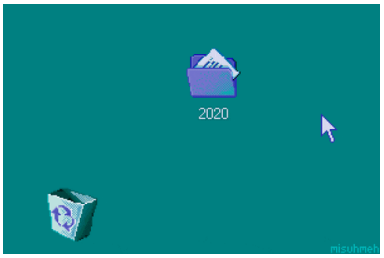
4.5 Error prevention

Web applications should also be developed in such a way that potential problems can be prevented. This can be done by either eliminating such error-prone conditions or by confirming with the users before they commit to a particular action.

In general, there are two types of errors: slips and mistakes.

Slips are unconscious errors caused by inattention. Slips can be avoided by providing helpful constraints and good defaults.

Mistakes are conscious errors based on a mismatch between the user's mental model and the design. Mistakes can be prevented or minimised by allowing users to undo certain actions as well as by notifying or warning the users before they execute the actions. It is also paramount that the design minimises the users' memory load to lower the probability of users making mistakes.

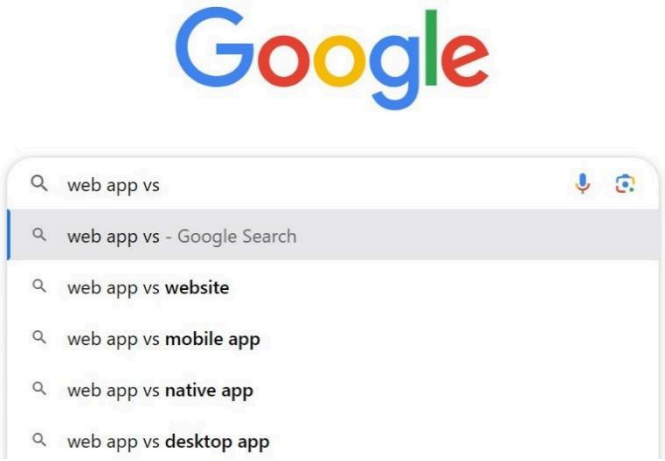


Confirmation messages can prevent unwanted mistakes by user

4.6 Recognition rather than recall

Humans have limited short-term memories. As such, interfaces that promote recognition reduce the amount of cognitive effort required from the users.

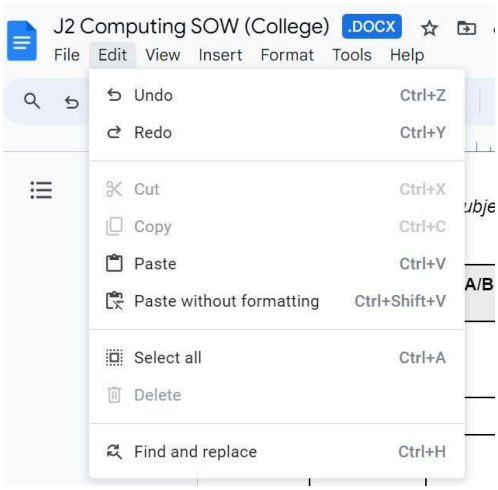
Users' memory load can be minimised by making the elements, actions and options in your application visible and easily retrievable. The users should not have to remember information from one part of the interface to another. For example, if the users need certain information to use the application such as field labels or menu items, these elements should be visible and easily retrievable to the users.



Google Suggest is designed to help users find relevant search queries more quickly

4.7 Flexibility and efficiency of use

A well-designed application allows differentiation for both novice users and expert users. Accelerators like keyboard shortcuts and touch gestures that are hidden from novice users may speed up the interaction for the expert and experienced users. The application should also be developed in such a way that the users can personalise how they prefer to interact with the application by customizing or tailoring their frequent actions, content and other functionality. An application that allows customisations will provide users to make selections about how they want the product to work.

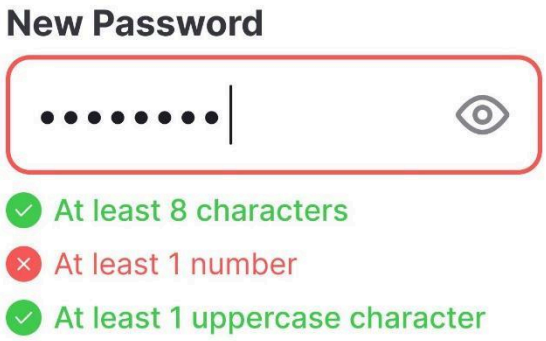


4.8 Aesthetic and minimalist design

The content and visual design of the User Interface (UI) should also focus on the essentials. The interfaces should not contain irrelevant content or rarely-needed information. This is because every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility. As such, it is important not to distract the users with unnecessary elements and to ensure that the visual elements of the interface support the user's primary goals.

4.9 Help users recognize, diagnose, and recover from errors

Errors cannot be eradicated completely. As such, helpful error messages are vital in web applications. These messages should attract users' immediate attention by ensuring that the messages follow the traditional visuals such as bold, red text. The messages should also be expressed in simple language and not in error codes or other technical jargon. The system should then indicate the problem and suggest constructive suggestions to solve the problem.


New Password
At least 8 characters
At least 1 number
At least 1 uppercase character

4.10 Help and documentation

Help and documentation help users to understand how to use the applications and to complete their tasks. These should be easy to search and tailored to the context in which the users are currently in. The help and support should also be concise by listing the concrete steps that need to be taken.

References

https://bootcamp.uxdesign.cc/simplified-jakob-nielsens-10-usability-heuristics-36a860c2ac9c
Jakob Nielsen's 10 general principles for interaction design

UI elements guide:
https://www.usability.gov/how-to-and-tools/methods/user-interface-elements.html

UI best practices:
https://www.usability.gov/what-and-why/user-interface-design.html
https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/

5 ASCII

ASCII stands for 'American Standard Code for Information Interchange'. It was defined in 1963 and was one of the most common character sets used. The ASCII standard defines how numbers are used to represent common characters that can be typed using a keyboard, such as upper-case and lower-case letters.

Using ASCII, only 128 distinct characters can be represented because ASCII codes are defined to be exactly seven bits long. This means that the value of ASCII codes can only vary from (000 0000)2 to (1111111)2 , which is the same as (00)16 to (7F)16 or 0 to 127.

The first 32 ASCII codes (0 to 31) represent non-printing characters that are not written symbols or the space character. These codes are instead used to control devices or to represent special input such as typing the Backspace and Esc keys on the keyboard. The last ASCII code 127 is also used to represent the special input of typing the Delete key on the keyboard.

Hexadecimal digits are often used to express ASCII codes as only two hexadecimal digits are needed to express any ASCII code. This is more compact than using three denary digits or seven binary digits.

6 Unicode

Besides ASCII, there is another expanded standard called Unicode. Unicode provides a unique number for every character, no matter what the platform, program or language is. Therefore, Unicode is used to represent characters in languages other than English such as Tamil and Mandarin.

While ASCII codes are defined to be exactly seven bits long, Unicode is not as restrictive. In Unicode, the number of bits used to represent each character can vary from 8 to 32 bits depending on the encoding scheme used. This flexibility means that Unicode can be used to represent over a million unique characters from many different written languages all over the world, which makes it a preferred choice for use over the ASCII standard.

The largest number that can be used to represent a character in Unicode is 1,114,111, which is (100001111111111111111)2 and requires 21 binary digits. This means that if Unicode values are stored directly as binary numbers, each character would take up 21 bits. Doing so would be wasteful, however, as not all Unicode characters are used equally often. The designers of Unicode realised that it would be more efficient to use fewer bits for frequently used characters at the expense of using more bits for rarely used characters. This explains why the number of bits used to represent a Unicode character can vary between 8 and 32.

References

ASCII vs Unicode
ASCII Code Website
Unicode in Python
Unicode official website
Unicode official website
https://www.bbc.co.uk/bitesize/guides/zp73wmn/revision/5
https://www.bbc.co.uk/bitesize/guides/zsnbr82/revision/5

Annex 1 – "Newer" Types of Applications

Hybrid Applications

A hybrid application combines the elements of a native application (an application developed for a specific platform like Android or iOS) and a web application (an app that can be accessed on the internet via a browser). It's built with popular front-end development technologies and languages like HTML5, JavaScript, and CSS, providing cross-platform functionality. In other words, developers don't have to create separate code for Android and iOS. They can write code for a mobile app once, while still accommodating multiple platforms.

A hybrid application needs to be installed on the device. Once users download a hybrid application from an app store and install it locally, its native shell will connect to their mobile platform's capabilities through a browser embedded in the app.

Examples: Gmail/Meet, Instagram, Discord

Progressive Web Applications (PWA)

These are similar to hybrid applications in that they both combine the elements of native and web applications. However, PWAs can be accessed directly through a web browser and be added as a shortcut on your home screen, which can then be accessed offline. PWAs currently are not able to access internal APIs and device-specific resources.

Examples: Spotify

Annex 2 – Web application vs Website

There are many similarities between websites and web apps. Many people would argue that web apps are the next stage of evolution for websites. Still, there are critical differences between these two technologies that should be considered. Essential differences between web applications and websites include:

- Function
- Complexity
- Access

Function
From a user perspective, the significant difference between websites and web applications is function. Websites serve to inform, and web apps serve to help. The content on a website can be viewed, read, or listened to, but the user cannot manipulate it. Conversely, content on web applications is not only viewable but contains interactive elements. Web applications allow users to manipulate data. A simple example of interactive web content is a form. Anything on the Internet that performs a service is likely a web app.

Complexity
Web applications are far more complex than websites. Websites are just a collection of static web pages. Yes, modern web development standards have made websites more interactive, but they are still relatively simple to build compared to web applications. Not only do web applications have to provide a service and function seamlessly, but they require backend services, varying user levels, and data processing capabilities. In addition, web application security requires more advanced solutions.

While web applications can be built with the same web technologies like websites, such as JavaScript, HTML, and CSS, apps will also need to use more advanced programming languages, like PHP, frameworks, and server-side scripts. Additionally, UX/UI design is more crucial to web applications since users actively interact with them.

Access
Public access is a hallmark of websites, but almost all web applications require registration and authentication. In most cases, unregistered website visitors have the same access and experience as registered users. On the other hand, web apps almost always require user authentication because they provide services customized to the user's requirements. Banking apps provide a great real-world example of this. Yes, all account holders are offered similar services online, but each user has a unique experience based on their personal banking information.