Vietnam National University of HCMC
International University
School of Computer Science and Engineering

UI/UX Design & Evaluation
★ Evaluation: Introduction and Heuristics ★

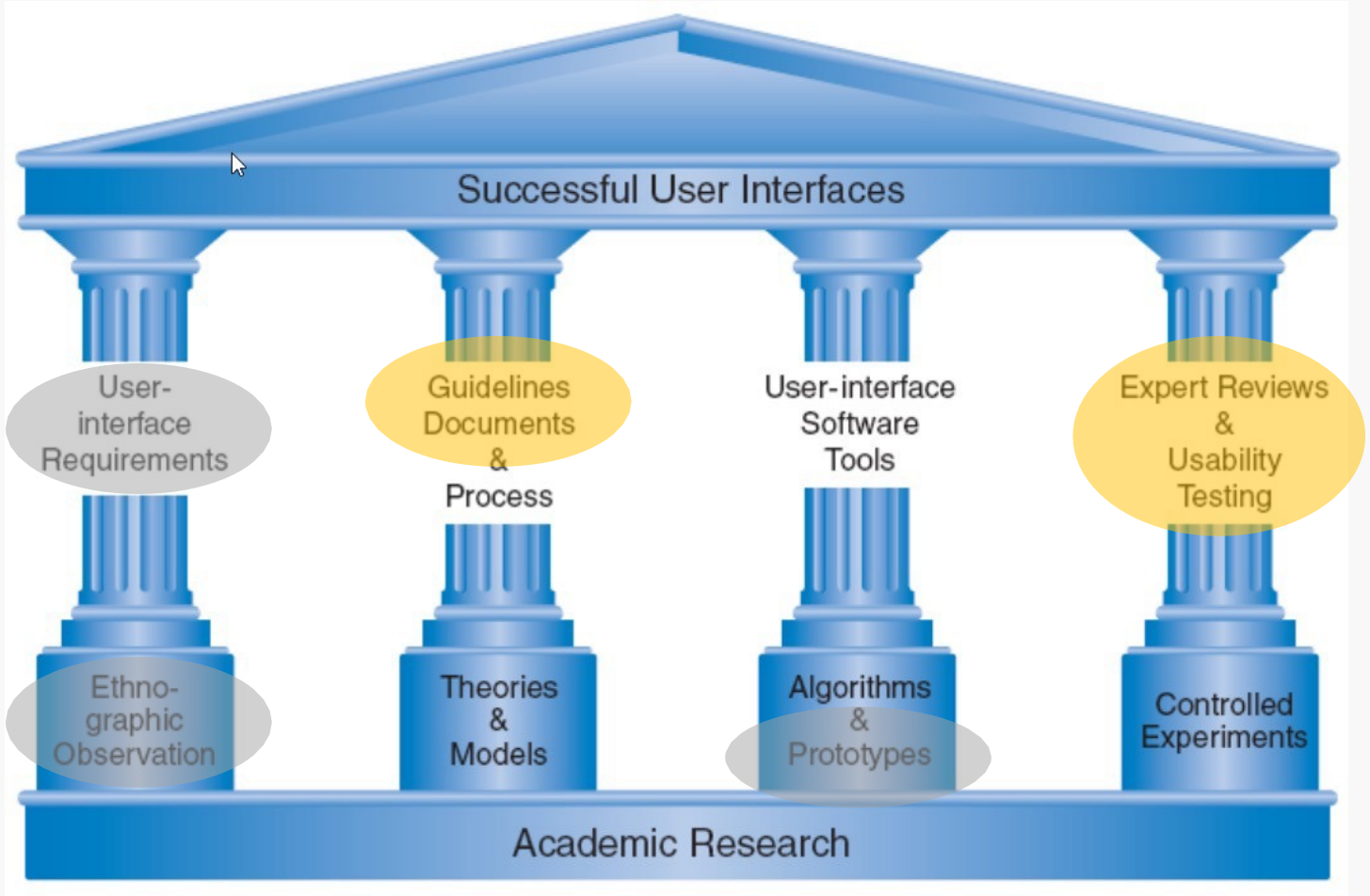
Dr Vi Chi Thanh - vcthanh@hcmiu.edu.vn
https://vichithanh.github.io

SCAN ME

Course Overview

1. Introduction to HCI
2. Needfinding
3. Analyzing and Synthetizing
4. Prototyping
5. Design Theory, Principles, and Guidelines
6. Introduction to Prototyping
7. Visual Design
8. Human abilities and theoretical models
9. Human abilities and theoretical models (cont.)
10. Evaluation: overview. Heuristic evaluation
11. Design Patterns
12. Medium-fidelity Prototypes
13. High-fidelity Prototypes
14. Designing for Diversity
15. Usability Testing & Evaluation Methods
16. Data Collection Techniques & Report Usability Test Results

The Four Pillars of Design



Ben Shneiderman & Catherine Plaisant, Designing the User Interface: Strategies for Effective Human-Computer Interaction

Goals

- **Generating design solutions**
  - Guidelines
  - Principles
  - Theories
  - Design Patterns
- **Evaluating generated designs**
  - Expert reviews and heuristics
  - Usability testing
  - Controlled experiments

Evaluation

Testing the usability, functionality and acceptability of an interactive system

Goal

- Evaluation: «Evaluation tests the usability, functionality and acceptability of an interactive system»
  - According to the design stage (sketch, prototype, final)
  - According to the initial goals
  - Alongside the different usability dimensions
  - Using a range of different techniques
- Identify and correct issues as soon as possible

Usability

- **Usability**: how well users can use the system's functionality.
- Dimensions of usability:
  - **Usefulness**: does it do something people want?
  - **Learnability**: is it easy to learn?
  - **Memorability**: one learned, is it easy to remember?
  - **Effectiveness**: does it allow reaching the goal?
  - **Efficiency**: once learned, is it fast to use?
  - **Visibility**: is the state of the system visible?
  - **Errors**: are errors few and recoverable?
  - **Satisfaction**: is it enjoyable to use?

Functionality

- **Functionality**: the system's functionality must accord with the user's requirements and should enable users to perform their intended tasks.
- Functionality can be tested in different ways:
  - Are the appropriate functionality available within the system?
  - Are they clearly reachable by the user?
  - Do they match the the user's expectations?
- Functionality evaluation may also include measuring the user's performance with the system, to assess the effectiveness of the system in supporting the task.

Acceptability

- Technology **acceptability** is one's perception of a system before use, while technology acceptance is one's perception of the system after use.
- Good User Interface Design can make a product easy to understand and use, which results in greater user acceptance.
- Testing **acceptability** means evaluating the enjoyment and emotional response to a system, particularly in the case of systems that are aimed at leisure or entertainment.
- This may involve:
  - measuring satisfaction and comfort
  - identifying areas of the design that overload the user

Many Evaluation Approaches

- Evaluation may take place:
  - In the laboratory
  - In the field

Many Evaluation Approaches

- In lab studies, users are taken out of their normal work environment to take part in **controlled** tests. They are typically adopted in the early stages of design (e.g., to compare alternatives, you don't need a working implementation).
  - 👍 simulation of dangerous environments
  - 👍 suitable for specific tasks within a system
  - 👎 lack of context
  - 👎 unnatural situations leading to biases
  - 👎 not suitable for all the tasks

Many Evaluation Approaches

- Field studies takes the designer or evaluator out into the user's work environment in order to observe the system in action.
  - 👍open nature: the "real" context
  - 👍users are in their natural environment
  - 👎low degree of control
  - 👎higher costs (you need a working implementation)
  - 👎longer duration

Many Evaluation Approaches

- Evaluation may be based on expert evaluation:
  - Analytic methods
  - Review methods
  - Model-based methods
  - Heuristics
- It is useful to identify any areas that are likely to cause difficulties because they violate known cognitive principles, or ignore accepted empirical results

👍it can be used at any stage in the development process

👍it is relatively cheap, since it does not require user involvement

👎it does not assess actual use of the system

Many Evaluation Approaches

- Evaluation may involve users:
  - Experimental methods
  - Observational methods
  - Query methods
  - Formal or semi-formal or informal
- l and observational methods, the evaluator chooses a hypothesis to test, which can be determined by measuring some attribute of participant behavior.
  - 👍they provide empirical evidence
  - 👎they require more time to be designed and analyzed
- Query techniques (e.g., interviews) relies on asking the user about the interface directly
  - 👍they are simple and cheap
  - 👎you get subjective results

Many Evaluation Approaches

- We can also adopt automated evaluation:
  - Simulation and software measures
  - Formal evaluation with models and formulas
  - Especially for low-level issues

Cognitive Walkthrough

A simple technique to analyze all individual step in an interaction path

Cognitive Walkthrough

- Step-by-step revision of a sequence of actions (interaction steps) to perform a given task
- Evaluators examine each step, looking for possible problems
- Particularly suited for systems designed for learning-by-exploration:
  - the main focus is to establish how easy a system is to learn
  - the evaluators go through each step in the task and provide a ‘story’ about why that step is or is not good for a new user

Walkthrough Organization

- To do a walkthrough you need:
  - A specification or prototype of the system
    - It doesn't have to be complete, but it should be fairly detailed.
  - A description of the task the user need to perform on the system
    - A representative task that most users will want to do.
  - A complete, written list of the actions needed to complete the task
  - An indication of who the users are (experience, knowledge)

Walkthrough Organization

For each step, you must check

- Is the effect of the action the same as the user's goal at that point?
  - Each user action will have a specific effect within the system. Is this effect the same as what the user is trying to achieve at this point? Will users try to achieve the right result?
- Will users see that the action is available?
  - In other words, is the interactive element that achieves the step visible or easily findable?
- Once users have found the correct action, will they know it is the one they need?
  - Perhaps the right button is visible, but will users understand the label and will they know to engage with it?
- After the action is taken, will users understand the feedback they get?

Walkthrough Organization

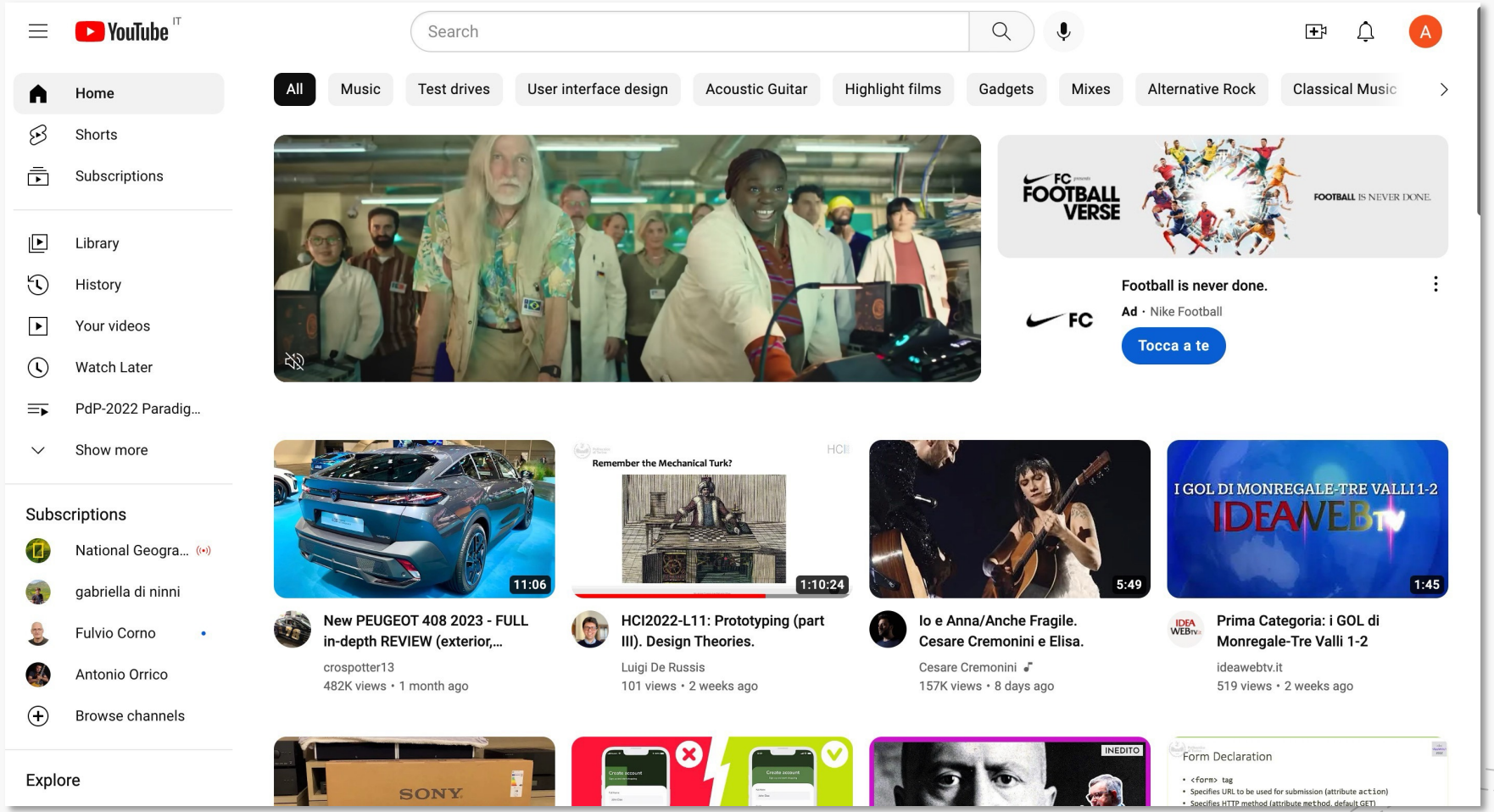
- It is vital to document the cognitive walkthrough to keep a record of what is good and what needs improvement in the design:
  - date, time of the walkthrough, and the names of the evaluators
  - answers of the four questions for each action
  - any negative answer should be documented on a separate usability problem report sheet
  - Each problem should include a degree of severity degree:
    - designers can decide priorities for correcting the design of the identified problem

Example

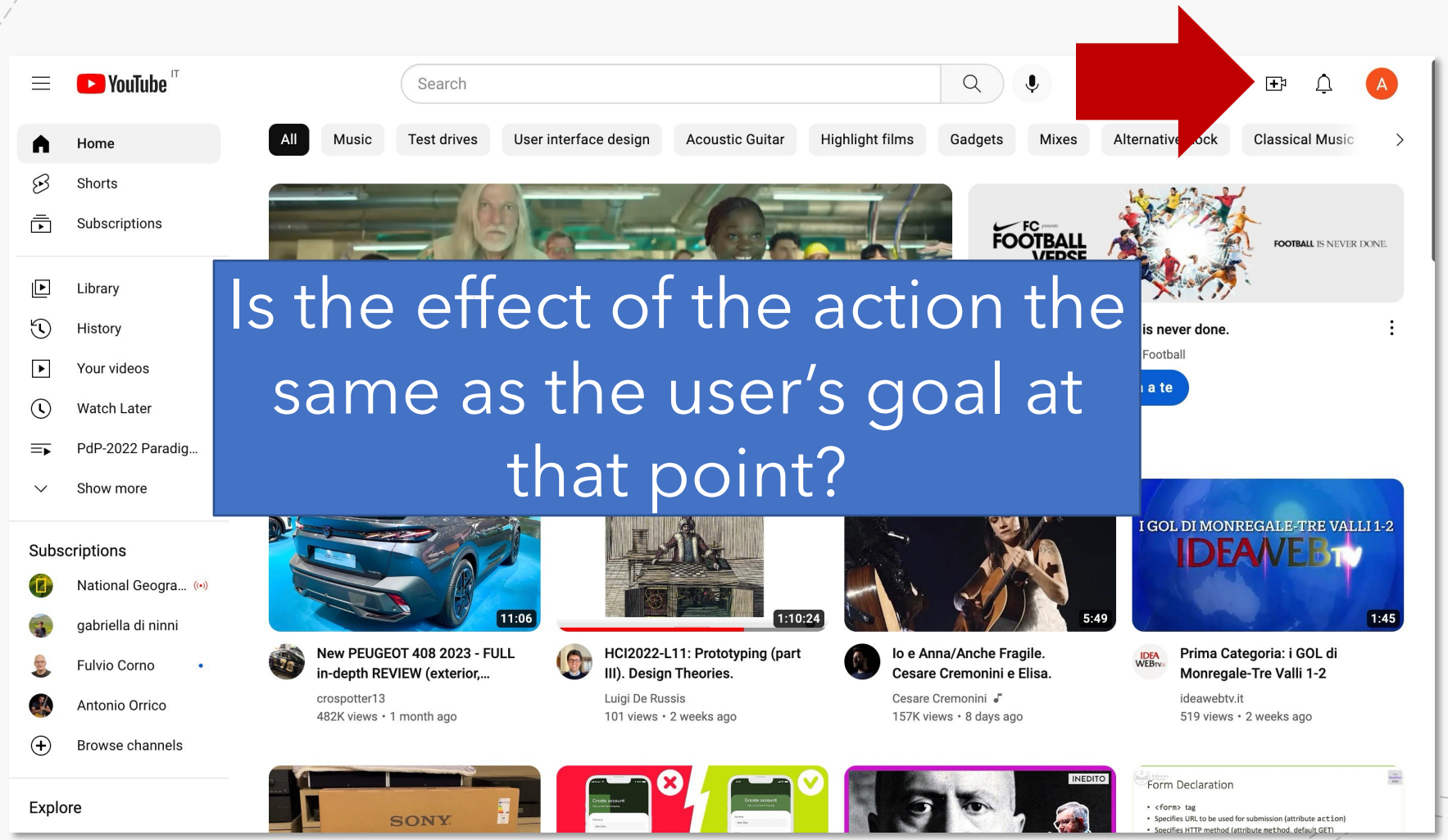
TASK: Upload a video on YouTube

- In the homepage, click on the create video icon
- Click on “Upload video” in the dropdown menu
- Drag and drop an “.mp4” file on the upload modal or click on the “SELECT FILE” button to select the file from your PC
- Insert the title and the description of the video in the two related text fields
- Click on the “NEXT” button
- […]

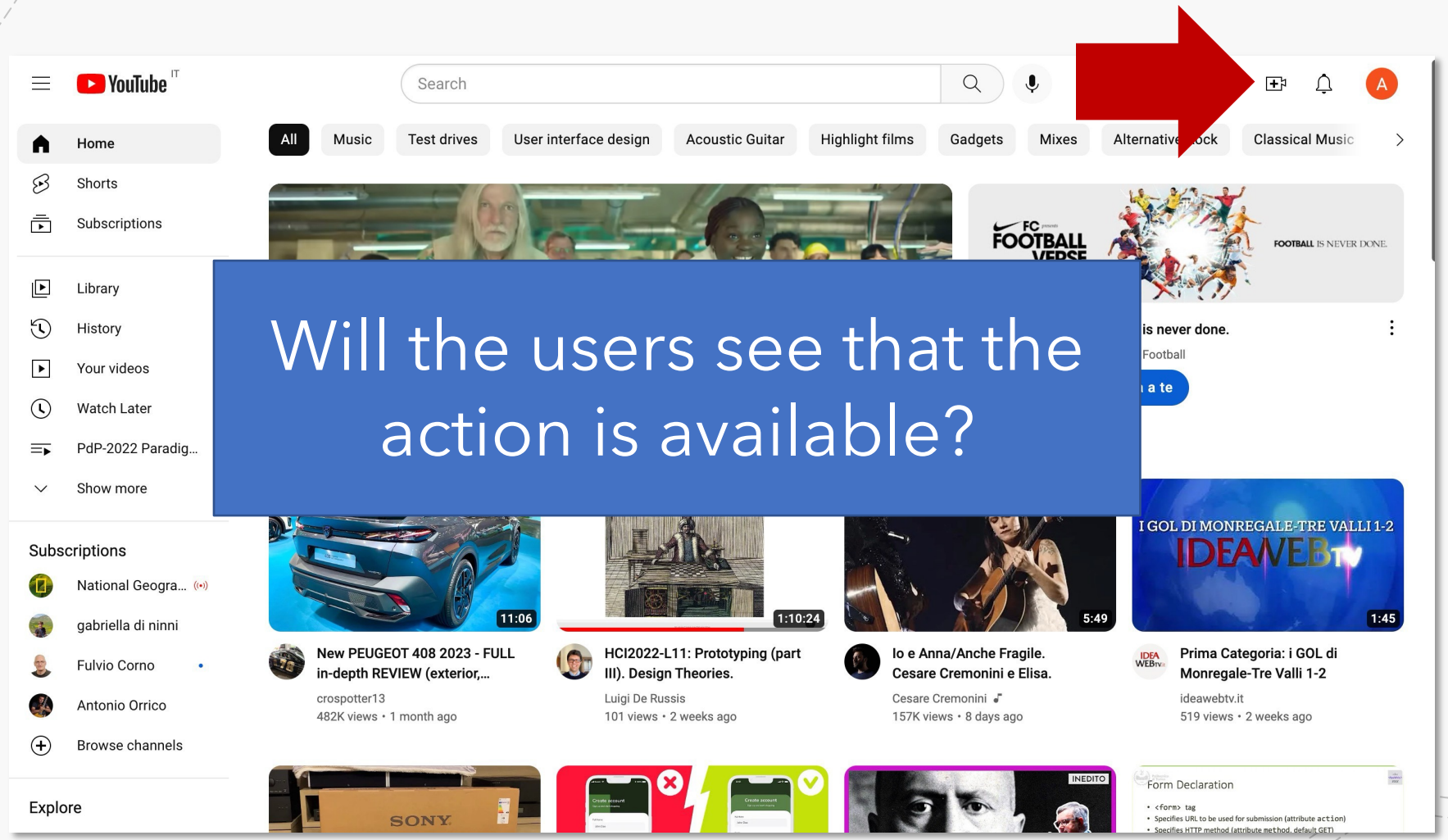
Example: click on the create video icon



Example: click on the create video icon

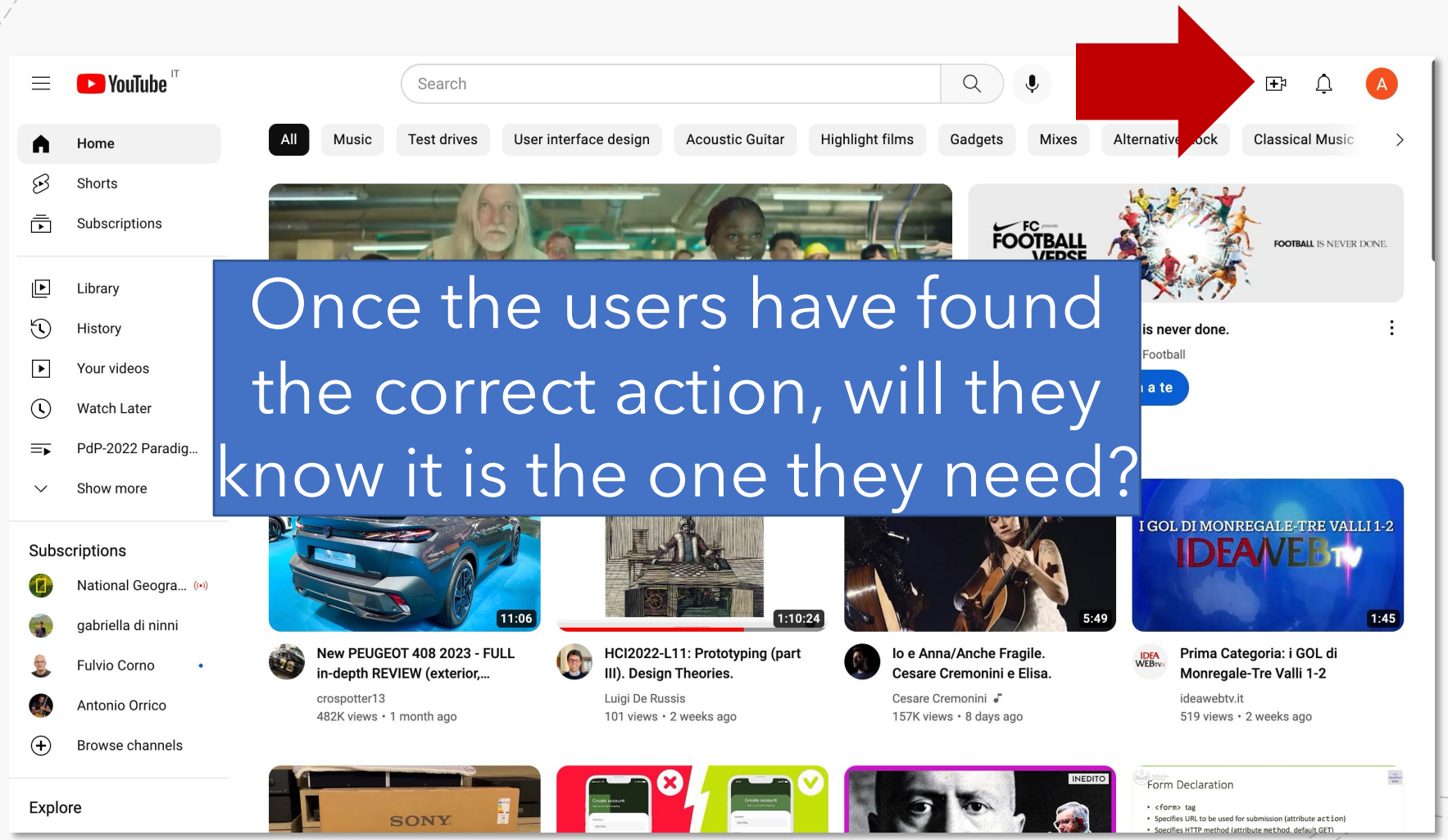


Example: click on the create video icon



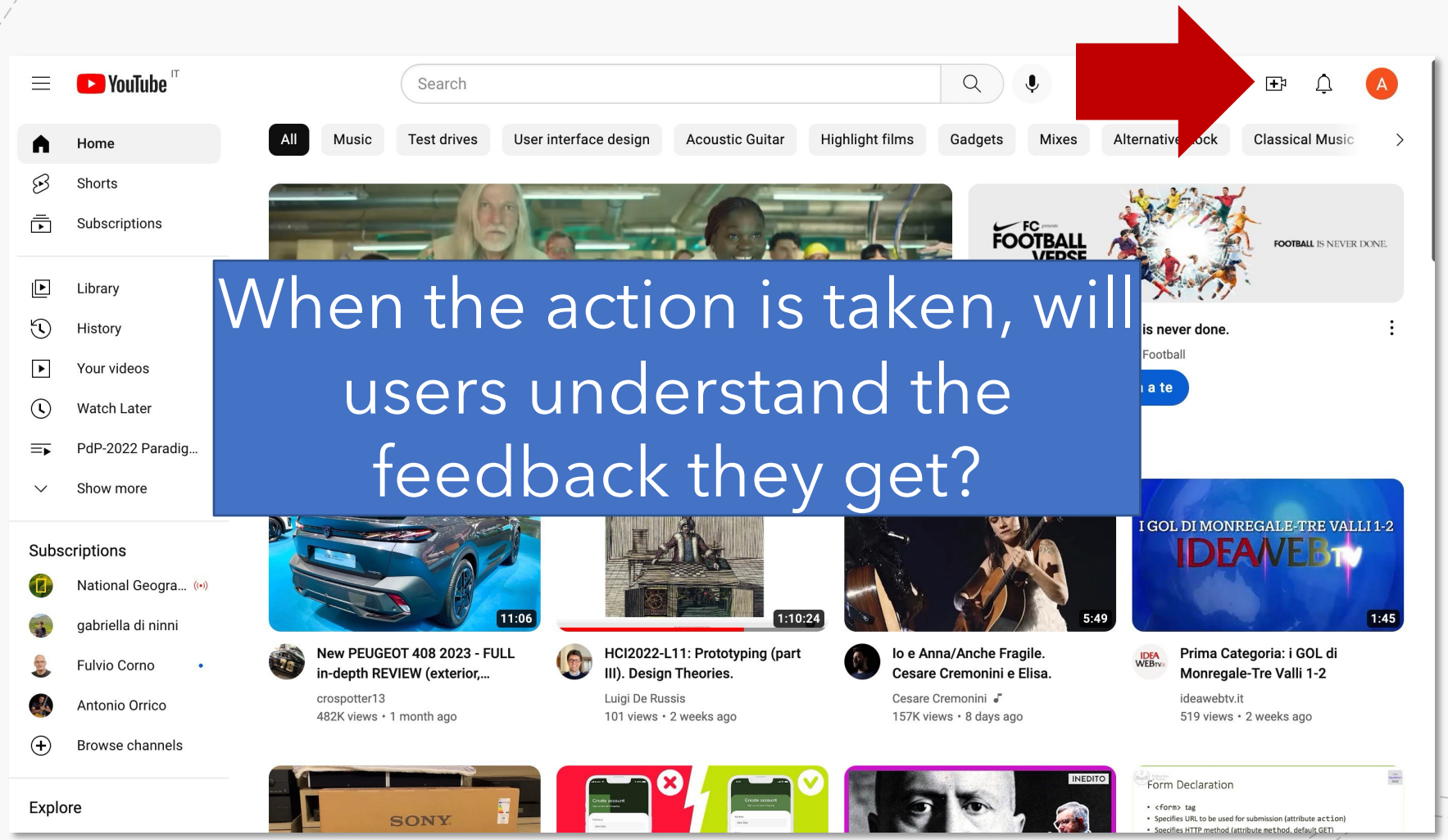
Will the users see that the action is available?

Example: click on the create video icon



Once the users have found the correct action, will they know it is the one they need?

Example: click on the create video icon



Example

Home | Study at Bristol | About | Schools & faculties | Research | Business & partnerships | News | People

University of BRISTOL

New students

Current students | Current staff | Alumni

search

- New students
- Before you arrive
- When you arrive
- Get ready to study
- Accommodation
- Health and wellbeing
- Sport
- New international students
  - Before you arrive
  - Travel to Bristol
  - After you arrive
  - Welcome events
  - Your life in Bristol
- Help and contacts
- Study Abroad students

New international students

Starting university in a different country is a big step. Find out the extra things you need to know as an international student before you start your adventure.

If you're already living in Bristol, check our information for current international students.

University of Bristol app

Make the most of your time at Bristol. Download from Google Play or the App Store.

Information for all new students

Before you arrive

Heuristic Evaluation

Experts check potential issues on your design, by referring to a set of heuristic criteria

When Is Design Critique Useful?

- Before user testing
  - To save effort
  - Solving easy-to-solve problems
  - Leaving user testing for bigger issues
- Before redesigning
  - Identify the good parts (to be kept) and the bad ones (to be redesigned)
- To generate evidence for problems that are known (or suspected)
  - From ‘murmurs’ or ‘impressions’ to hard evidence
- Before release
  - Smoothing and polishing

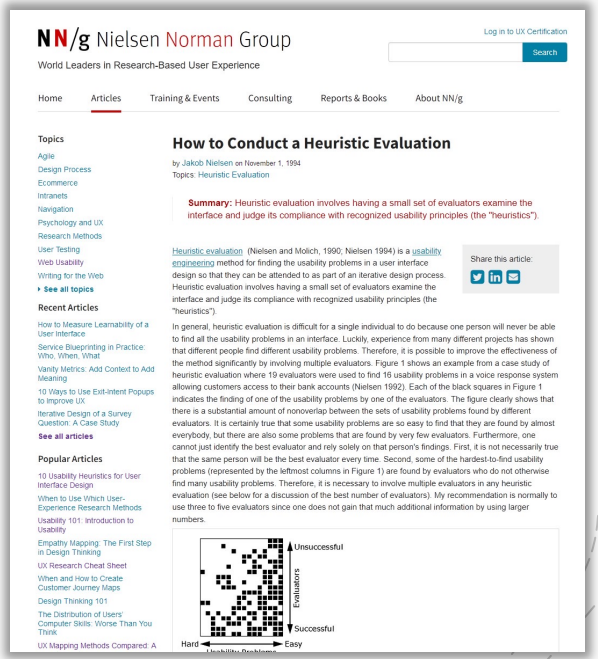
Heuristic Evaluation

- A method developed by Jacob Nielsen (1994)
  - Structured design critique
  - Using a set of simple and general heuristics
  - Executed by a small group of experts (3-5)
  - Suitable for any stage of the design (sketches, UI, …)
  - Goal: find usability problems in a design
- Also popularized as “Discount Usability”
- A heuristic is a guideline or general principle or rule of thumb that can guide a design decision or be used to critique a decision that has already been made.

Basic Idea

https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

- Define a set of heuristics (or principles)
- Give those heuristics to a group of experts
  - Each expert will use heuristics to look for problems in the design
- Experts work independently
  - Each expert will find different problems
- At the end, experts communicate and share their findings
  - Findings are analyzed, aggregated, ranked
- The discovered *violations* of the heuristics are used to fix problems or to re-design



Heuristics

- Nielsen proposed 10 heuristic rules
  - Good at finding most design problems
  - Inspired and connected to the Design Principles (→Guidelines)
- In a specific context, application domain, or for specific design goals…
  - … new heuristics can be defined
  - … some heuristic can be ignored

Phases of Heuristic Evaluation

1. Pre-evaluation training
   - Give evaluator information about the domain and the scenario to be evaluated
2. Evaluation
   - Individual
3. Severity Rating
   - First, individually
   - Then, aggregate and find consensus
4. Debriefing
   - Review with the design team

Evaluation (I)

- Define a set of tasks, that the evaluators should analyze
- For each task, the evaluator should step through the design several times, and inspect the UI elements
  - On the real design, or on a preliminary prototype
- At each step, check the design according to each of the heuristics
  - 1st step, get a general feeling for the interaction flow and general scope
  - 2nd step (and following), focus on specific UI elements, knowing where they fit in the general picture
- Heuristics are used as a “reminder” of things to look for
  - Other types of problems can also be reported

Evaluation (II)

- Comments from each evaluator should be recorded or written
  - There may be an observer, taking notes
  - The observer may provide clarifications, especially it the evaluator is not a domain expert
- Session duration is normally 1h – 2h
- Each evaluator should provide a list of usability problems
  - Which heuristic (or other usability rule) has been violated, and why
    - Not a subjective comment, but a reference to a known principle
  - Each problem reported separately, in detail

Evaluation (III)

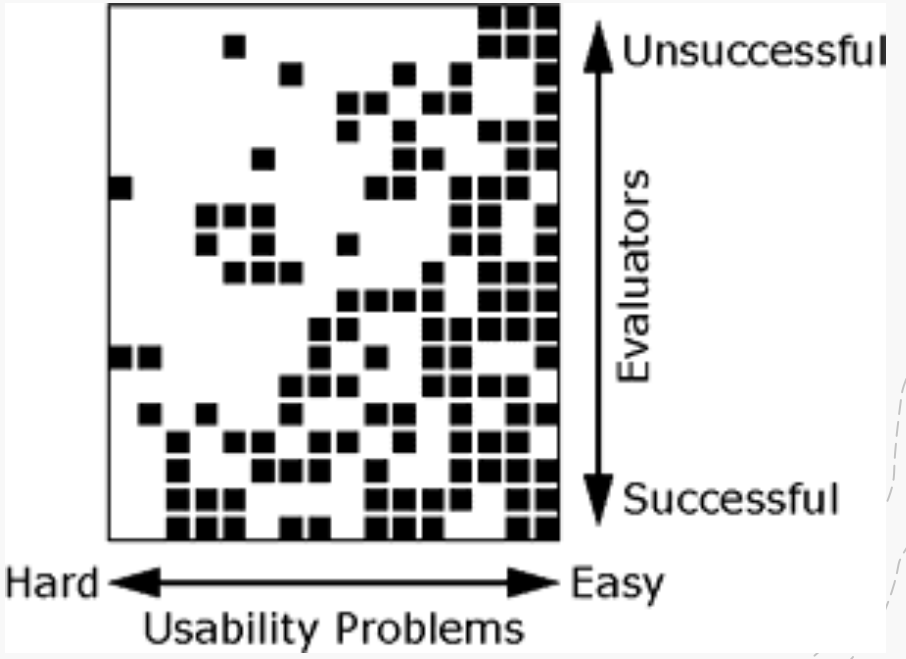
https://www.nngroup.com/articles/usability-problems-found-by-heuristic-evaluation/

Where problems may be found

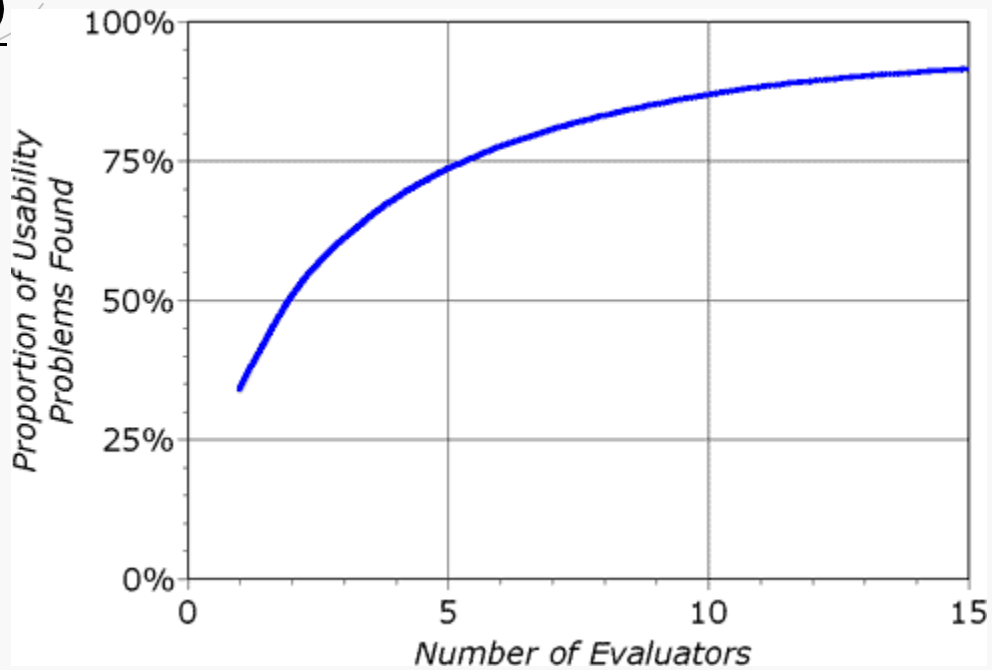
- A single location in the UI
- Two or more locations that need to be compared
- Problem with the overall UI structure
- Something is missing
  - May be due to prototype approximation
  - May still be unimplemented

Multiple Evaluators

- No evaluator finds all problems
  - Even the best one finds only ~1/3
- Different evaluators find different problems
  - Substantial amount of nonoverlap
- Some evaluators find more problems than others



How Many Evaluators?



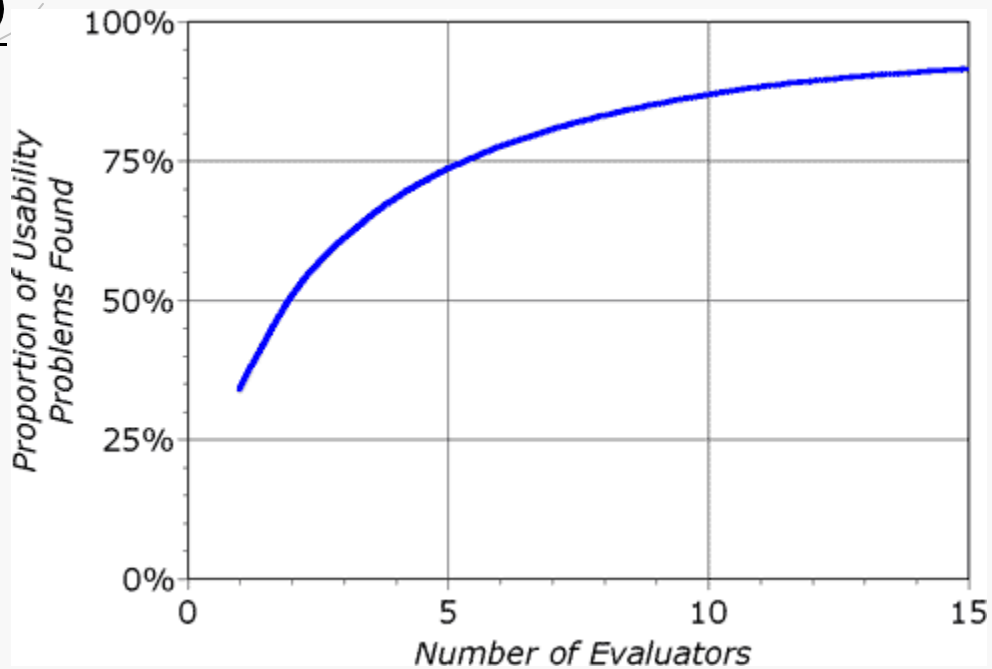
- $PF(i) = N(1 - (1 - l)^{!}$
- $PF(i)$: problems found
- i: number of independent evaluators
- N: number of existing (but unknown) usability problems
- l: ratio of usability problems found by a single evaluator

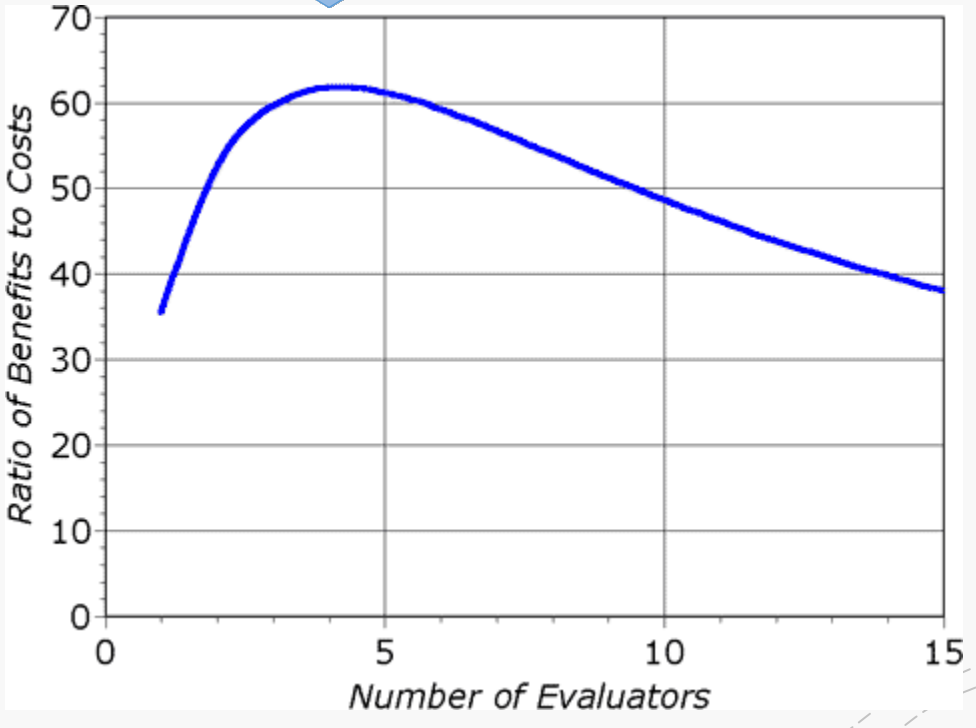
How Many Evaluators?

$Cost\ (i\ )$ = Fixed + Fee×i

$\frac{PF(i)/N}{Cost(i)}$

3-5

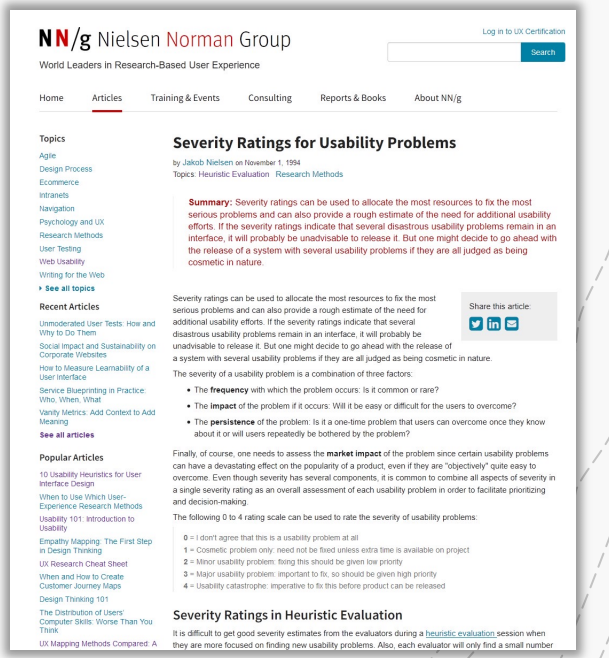




Severity Rating

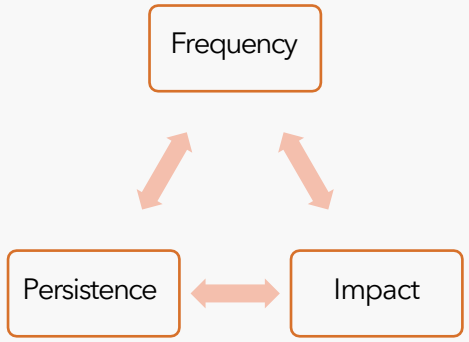
https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/

- We need to allocate the most resources to fix the most serious problems
- We need to understand if additional usability efforts are required
- **Severity** is a combination of:
  - **Frequency** with which the problem occurs: common or rare?
  - **Impact** of the problem if it occurs: easy to overcome or difficult?
  - **Persistence**, is it one-time or will it occur many times to users?
- Define a *combined severity rating*
  - Individually, for each evaluator



Severity Ratings scale

0	No problem	I don't agree that this is a usability problem at all
1	**Cosmetic** problem only	need not be fixed unless extra time is available on project
2	**Minor** usability problem	fixing this should be given low priority
3	**Major** usability problem	important to fix, so should be given high priority
4	Usability **catastrophe**	imperative to fix this before product can be released



Combined Severity Ratings

- Severity ratings from *one* evaluator have been found *unreliable*, they should not be used
- After all evaluators completed their rankings
  - Either let them discuss, and agree on a consensus ranking
  - Or just compute the average of the 3-5 ratings

Debriefing

- Meeting of all evaluators, with observers, and members of the development team
- Line-by-line analysis of the problems identified
  - Discussion: how can we fix it?
  - Discussion: how much will it cost to fix it?
- Can also be used to brainstorm general design ideas

Heuristic Evaluation vs. User Testing

Heuristic Evaluation

- Faster (1-2h per evaluator)
- Results are pre-interpreted (thanks to the evaluators)
- Could generate *false positives*
- Might *miss* some problems

User Testing

- Need to develop software, and prepare the set-up
- More accurate (by definition!)
  - Actual users and tasks
- … *more on this later in the course!*

Heuristic Evaluation vs. User Testing

Heuristic Evaluation

- Faster (1-2h per evaluator)
- Results are pre-interpreted (thanks to the evaluators)
- Could generate f
- Might *miss* som

User Testing

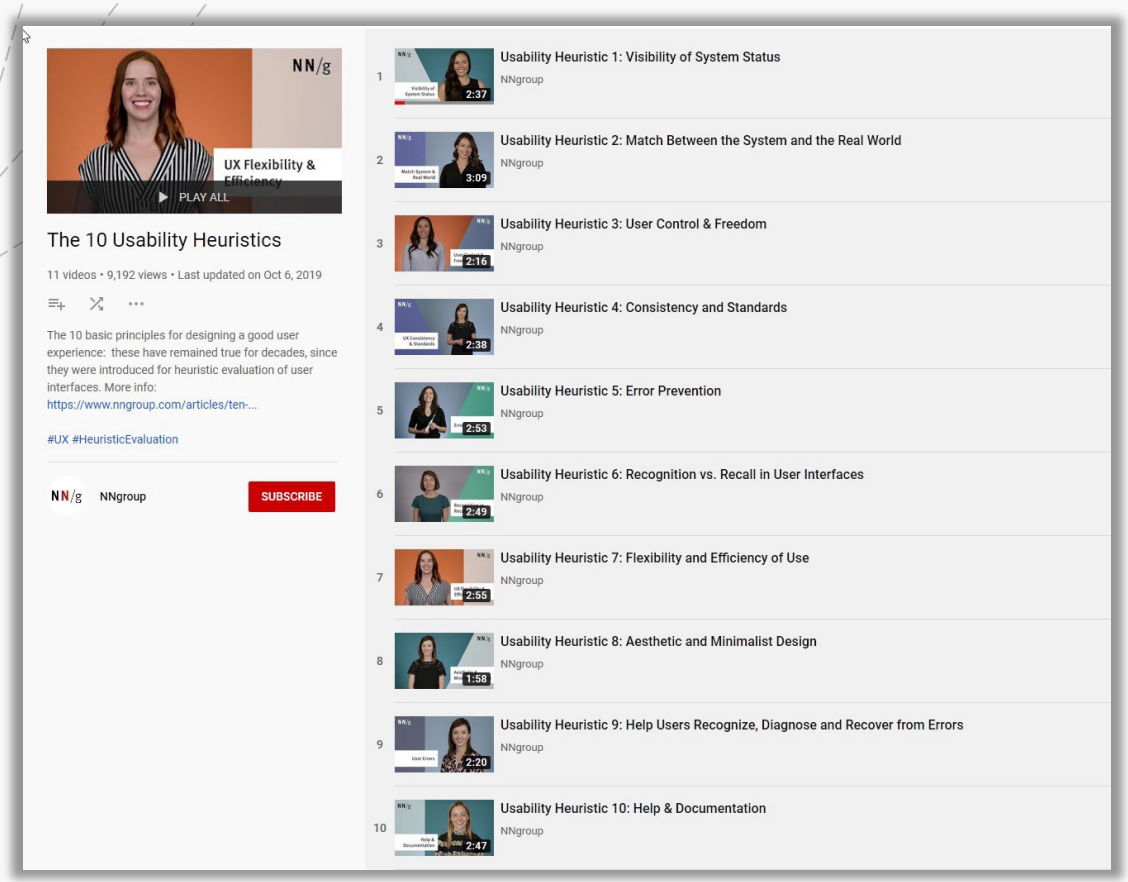
- Need to develop software, and prepare the set-up
- More accurate (by definition!)
  - Actual users and tasks
  - *this later in the*

- Alternate the methods!
  - Find different problems
  - Do not waste participants

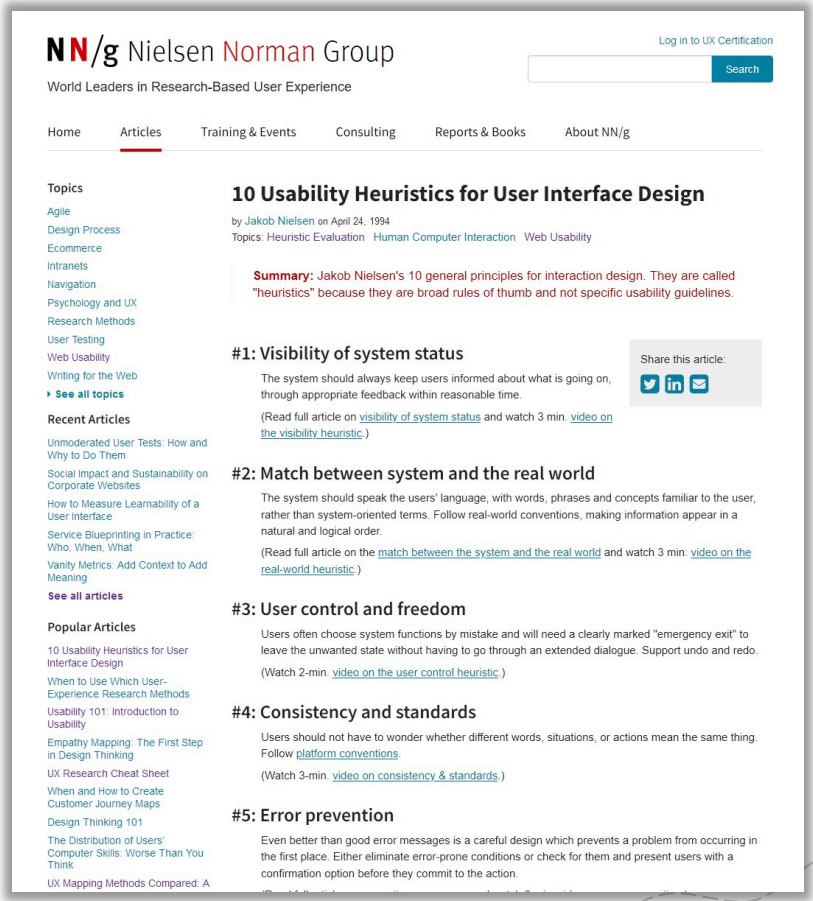
Nielsen’s Usability Heuristics

10 Usability Principles to be used in Heuristic Evaluation

10 Nielsen's Usability Heuristics



https://www.youtube.com/playlist?list=PLJOFJ3Ok_idtb2YeifXlG1-TYoMBLoG6I



https://www.nngroup.com/articles/ten-usability-heuristics/

10 Nielsen's Usability Heuristics

https://www.nngroup.com/articles/ten-usability-heuristics/

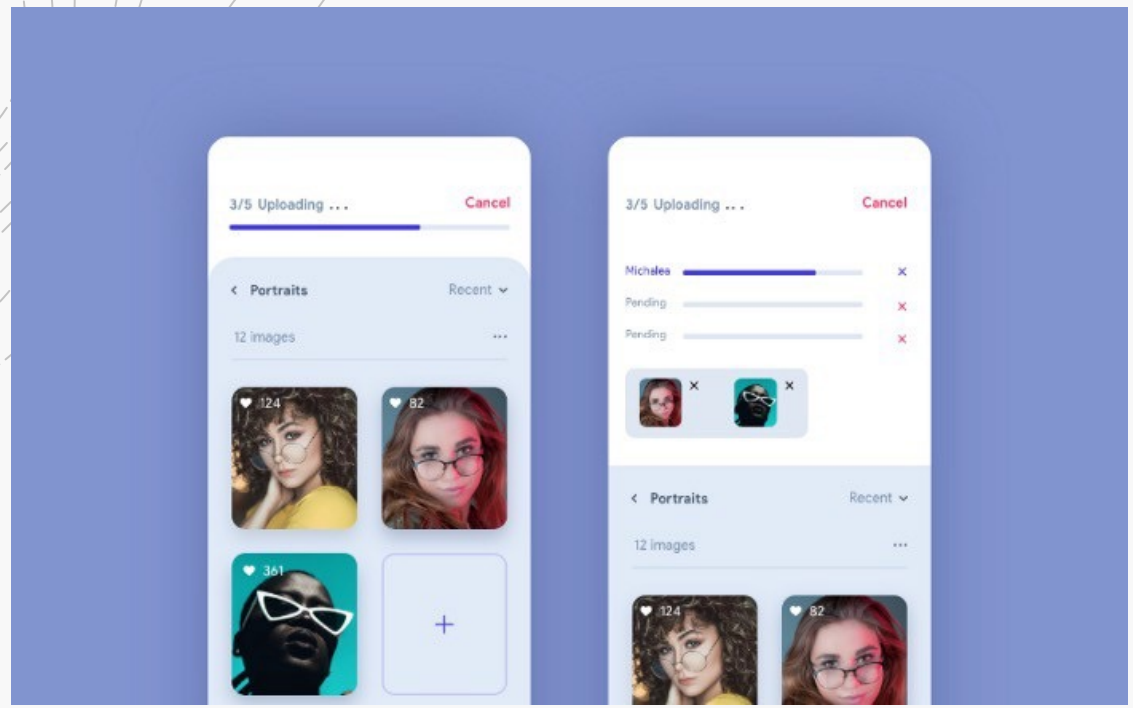
- #1: Visibility of system status
- #2: Match between system and the real world
- #3: User control and freedom
- #4: Consistency and standards
- #5: Error prevention
- #6: Recognition rather than recall
- #7: Flexibility and efficiency of use
- #8: Aesthetic and minimalist design
- #9: Help users recognize, diagnose, and recover from errors
- #10: Help and documentation

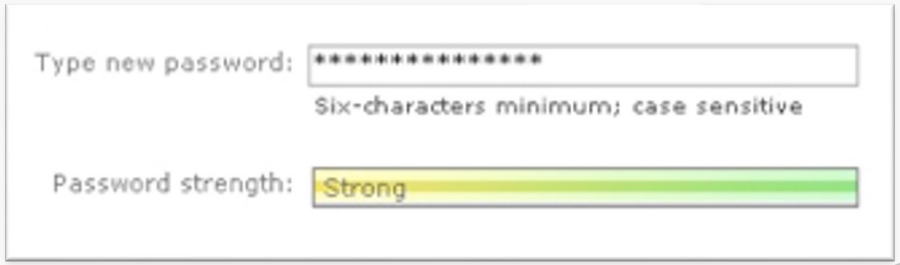
#1: Visibility of system status

- The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
- When users know the current system status, they learn the outcome of their prior interactions and determine next steps.

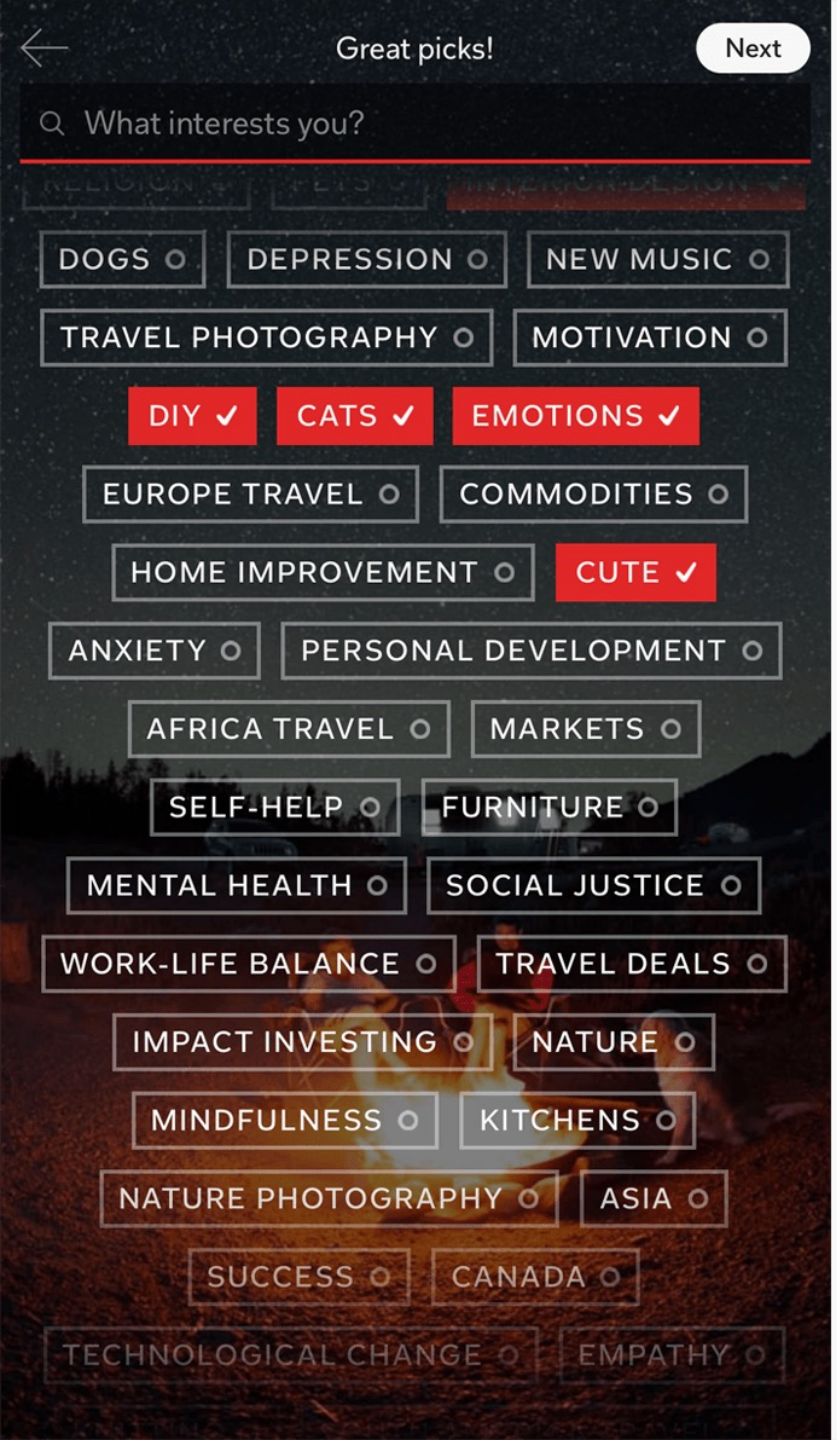
https://www.nngroup.com/articles/visibility-system-status/

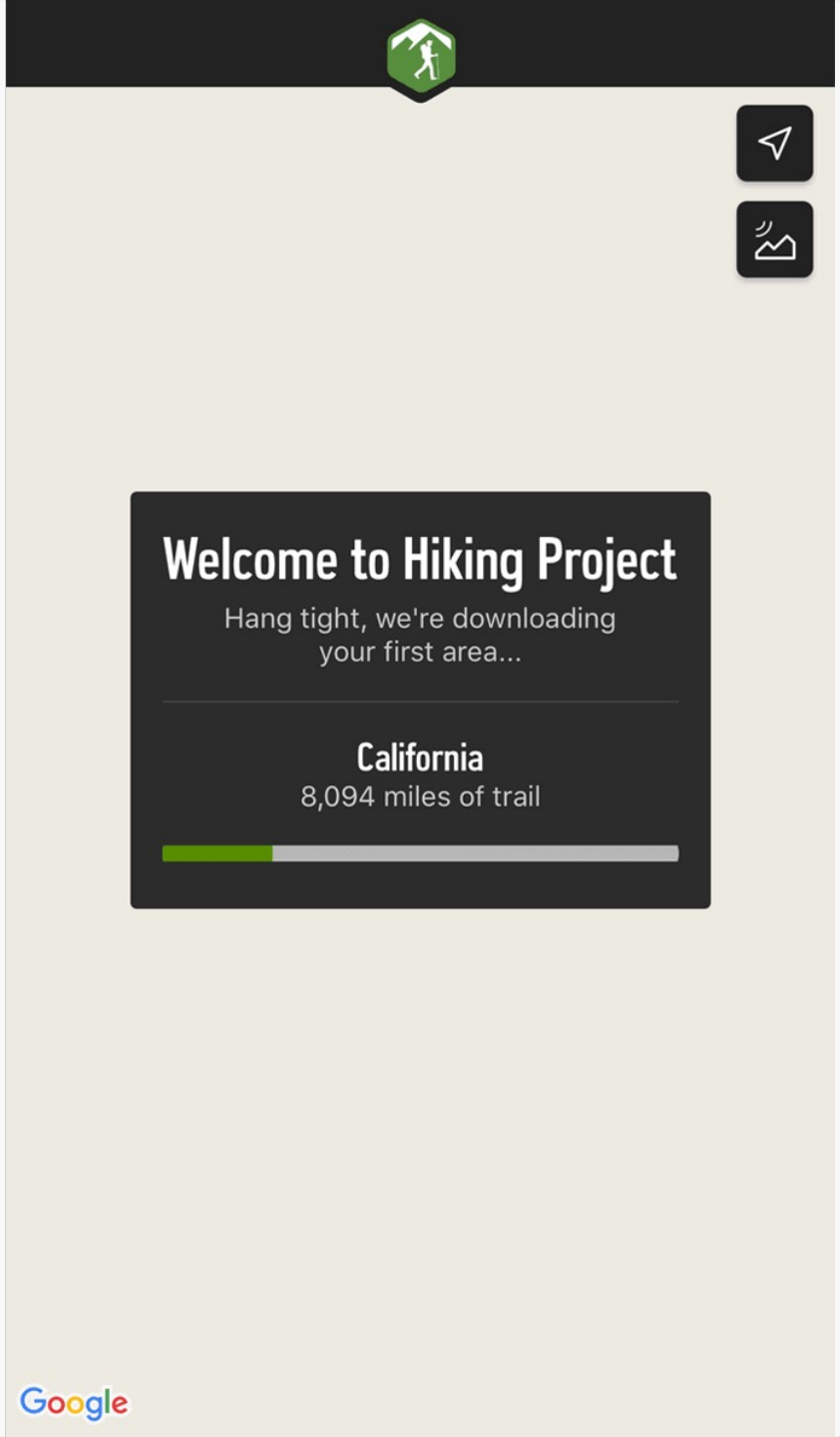
#1: Visibility of system status





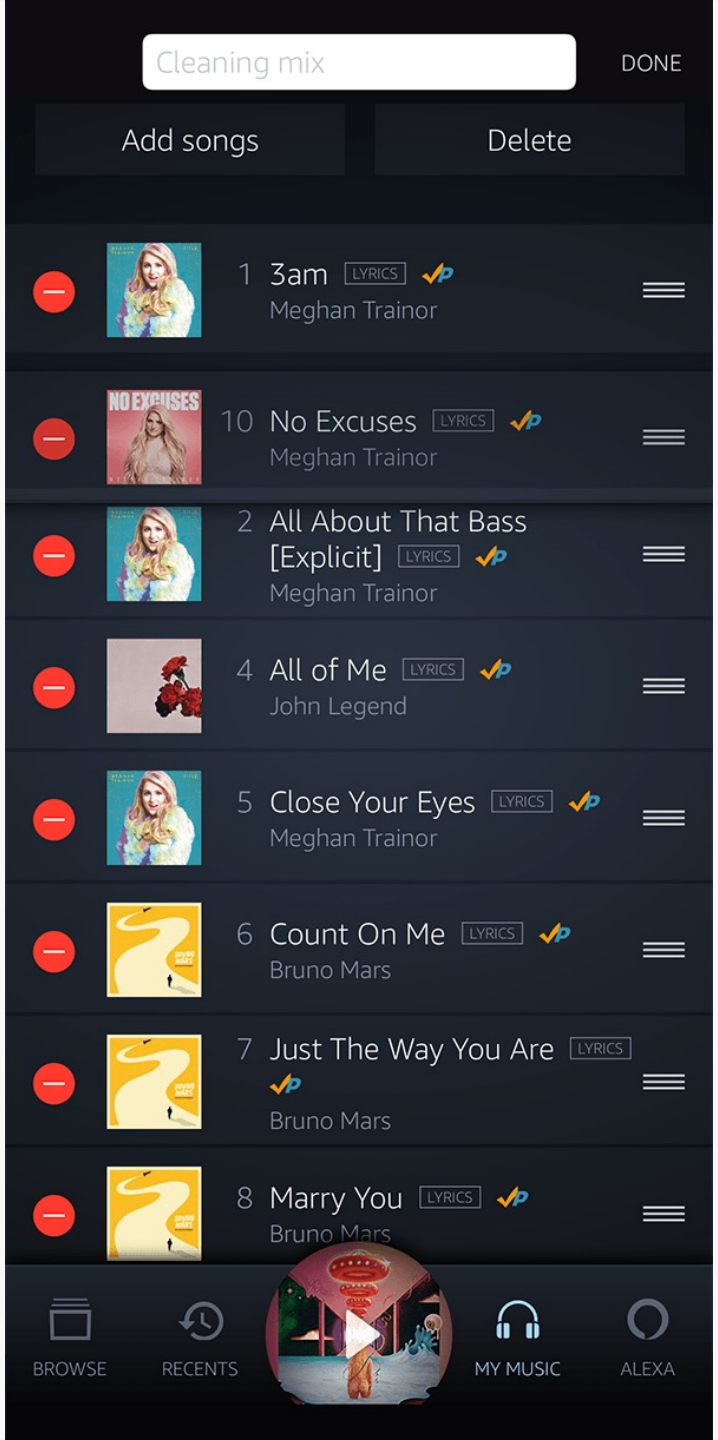
Some examples from: http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5





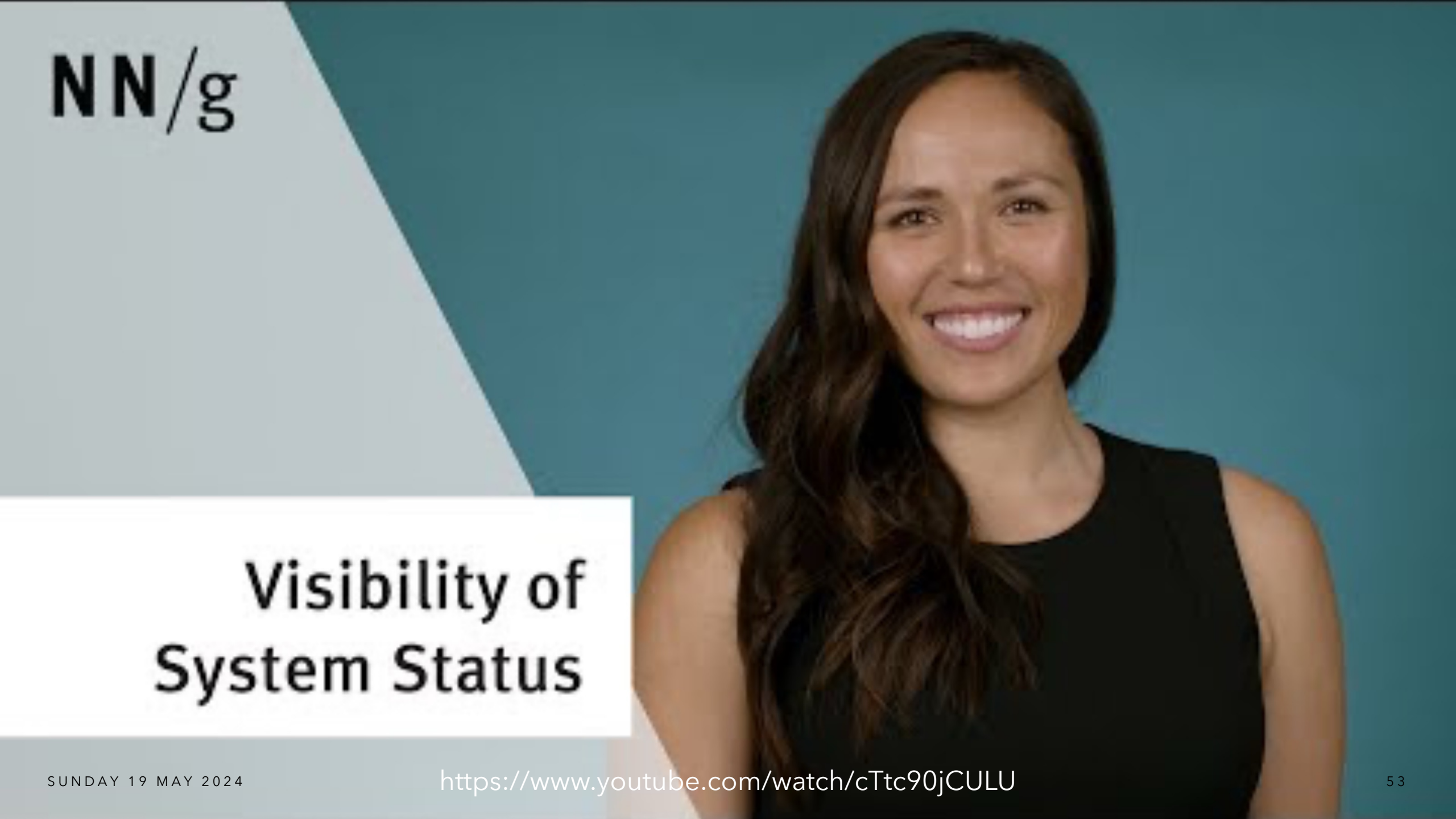
Changing the color and adding a checkmark to buttons on a selection screen communicates that the system has registered the user's choices (left).

Progress indicators reassure the user that a longer wait is normal, and that the system is still working (right).



The Amazon Music app on iOS allows users to directly manipulate the order of items within a playlist.

Users are aware of the system status at all times, and thus can easily identify and correct an error.



https://www.youtube.com/watch/cTtc90jCULU

Which Feedback?

- Time
  - Execution time for tasks
- Space
  - E.g., occupation of cloud storage
- Change
  - Ensure that the user is aware of changes that he requested (e.g., save, delete, send, …)
- Action
  - What is happening (running, stopped,…), in a redundant way
- Next steps
  - What will happen because of your action, and your possible next actions at this point
- Completion
  - Clarify when a task has been finalized

#1: Visibility of system status

- Communicate clearly to users what the system's state is – no action with consequences to users should be taken without informing them.
- Present feedback to the user as quickly as possible (ideally, immediately).
- Build trust through open and continuous communication.

#1: Visibility of system status
Rule of Thumb (Time)

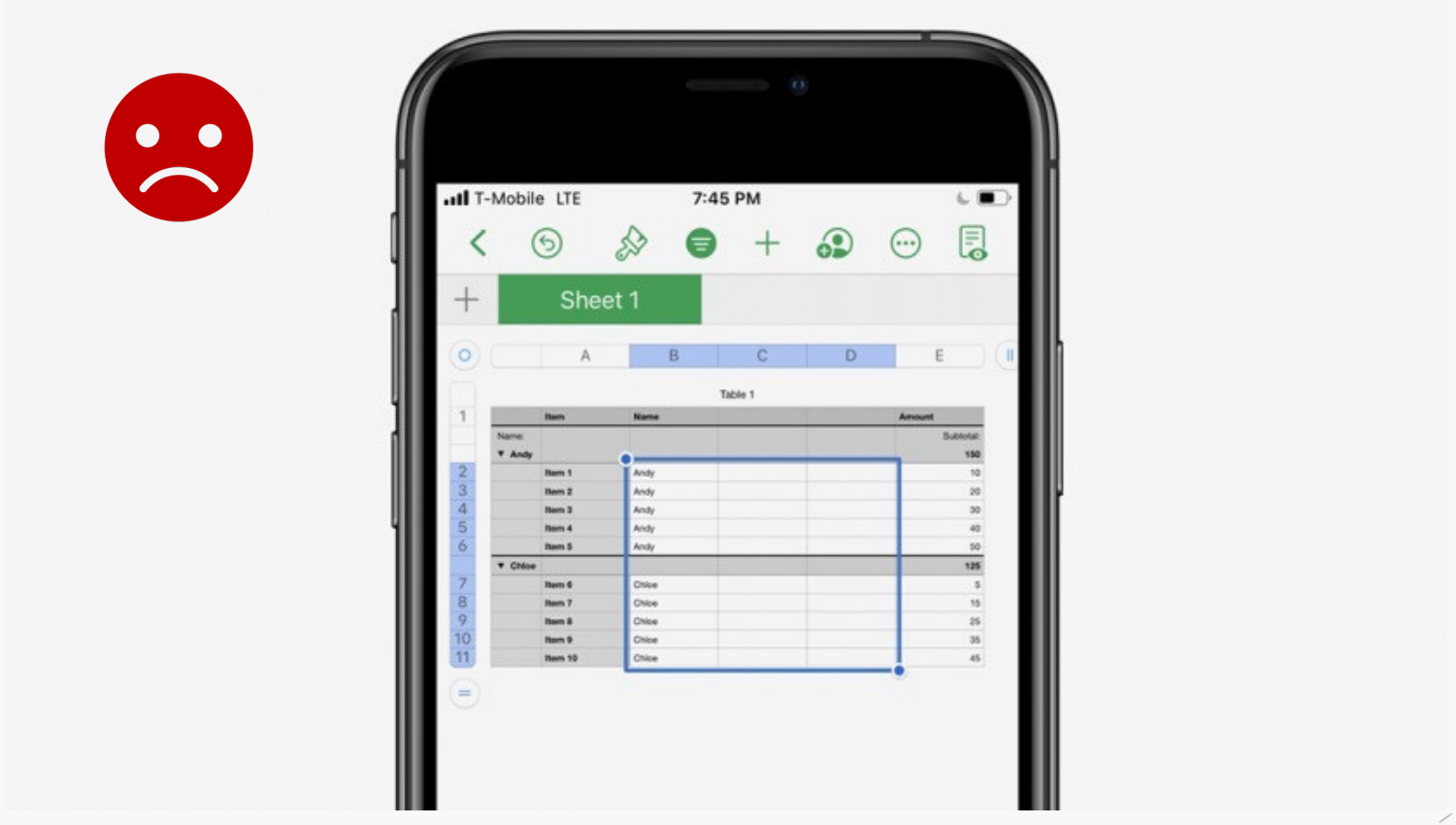
- If the execution time is…
- …Less than 1 second ⇨ just show the outcome of the action
- …Around 1-2 seconds ⇨ show feedback that the action is underway
- …More 2-3 seconds ⇨ show progress (percentage, estimated time,…)

#2: Match between system and the real world

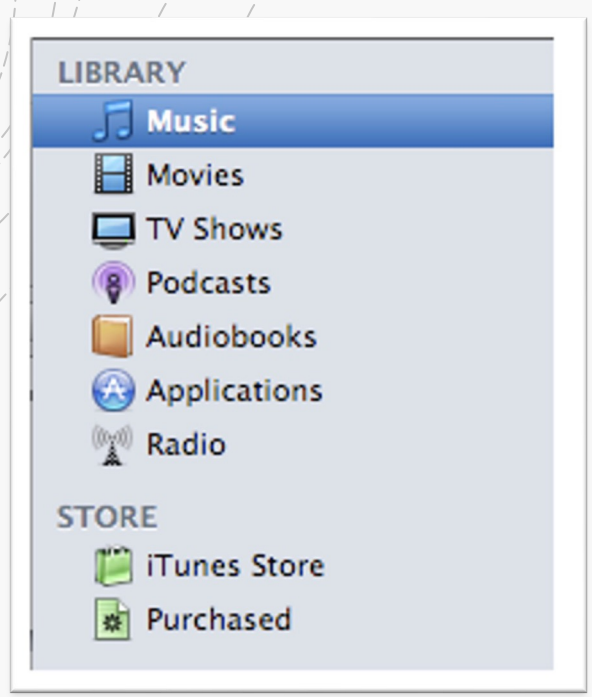
- The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.
  - Follow real- world conventions, making information appear in a natural and logical order.
  - Use familiar metaphors and language
- The way you should design depends very much on your specific users.
  - Terms, concepts, icons, and images that seem perfectly clear to you and your colleagues may be unfamiliar or confusing to your users.

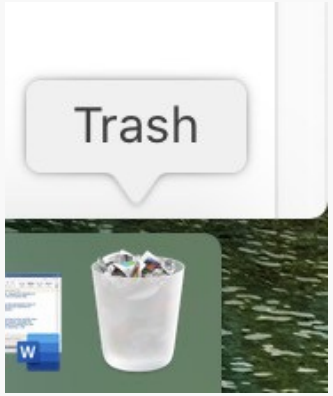
https://www.nngroup.com/articles/match-system-real-world/

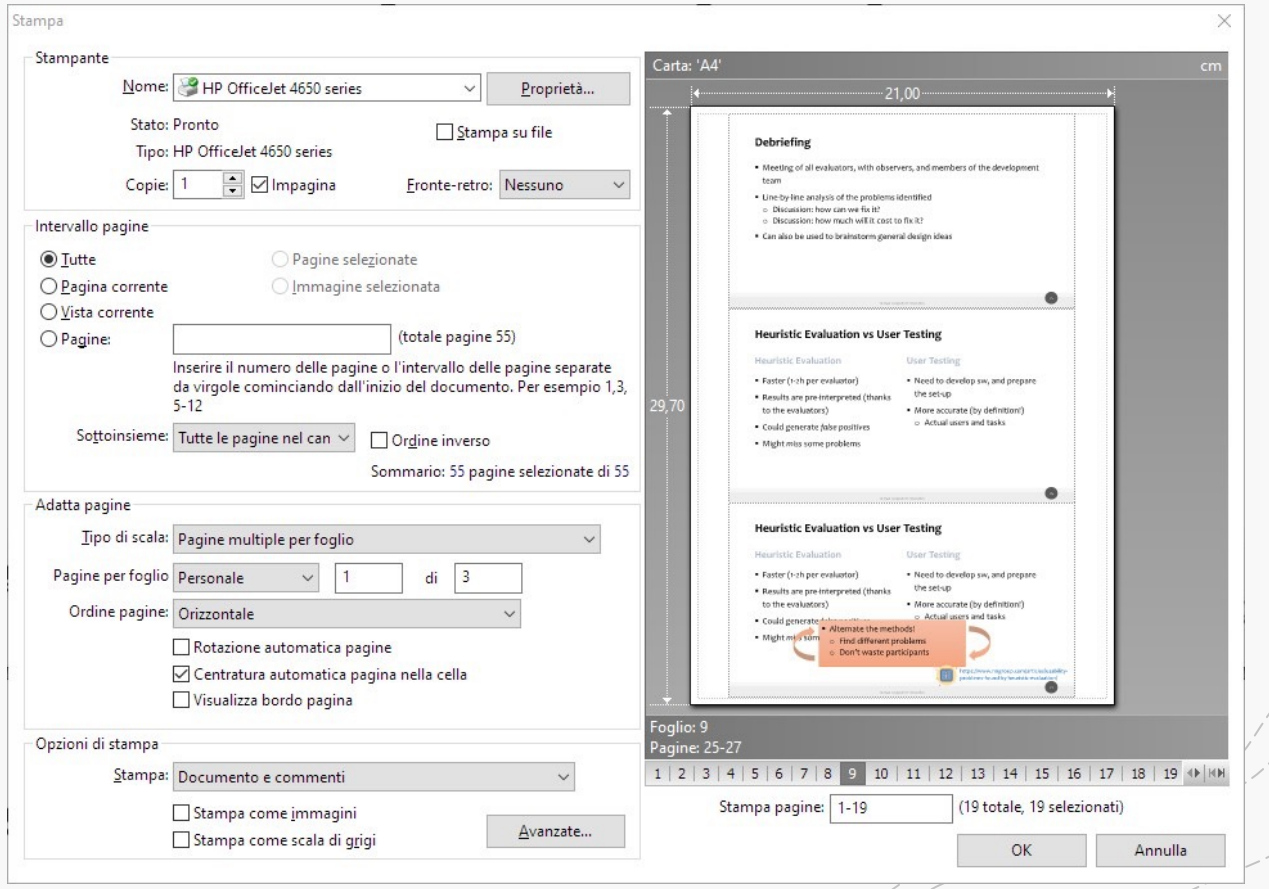
#2: Match between system and the real world



#2: Match between system and the real world



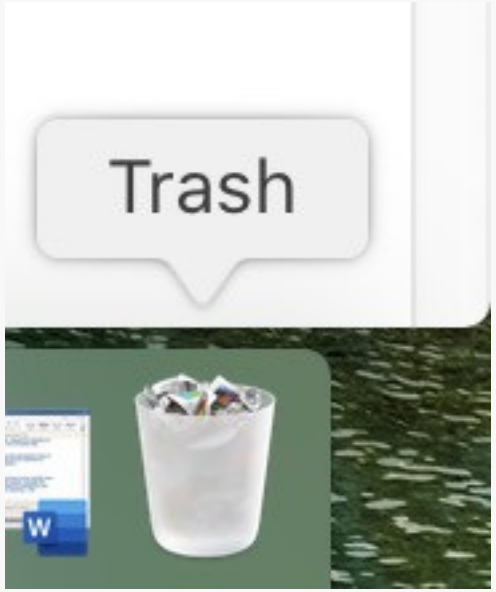


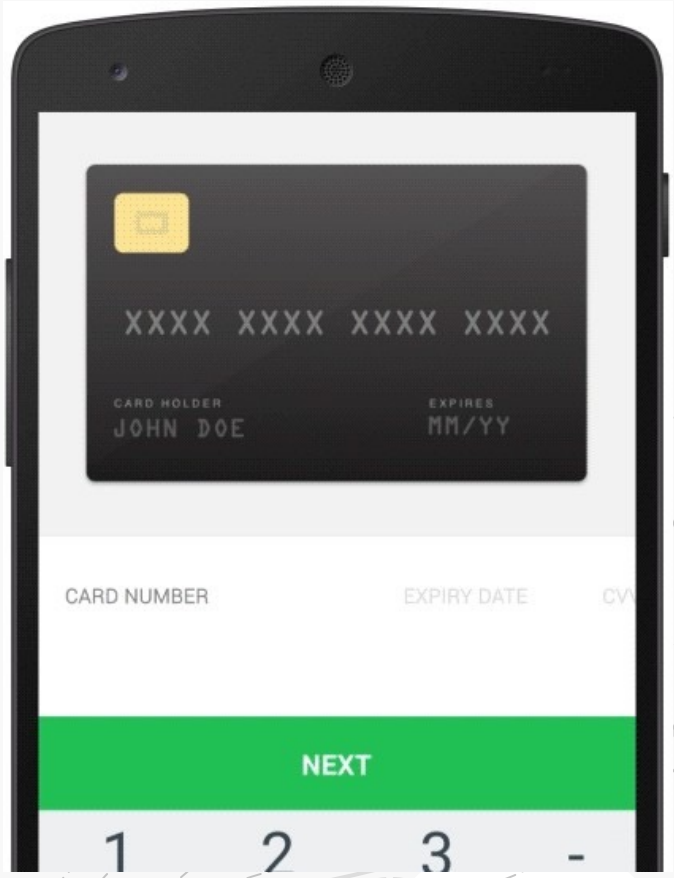


#2: Match between system and the real world
Exploit Familiarity

- Familiar Metaphors
  - Files, paper, folders, highlighters, …
- Familiar Language
  - Avoid jargon, acronyms, etc. that could be unknown to your users
- Familiar Categories
- Familiar Choices
  - E.g., explain the meaning of the error message (what happened, what are the consequences, what are the available options) in a simple way

#2: Match between system and the real world





NN/g

Match System & Real World

https://www.youtube.com/watch/0TAt9Pln51g

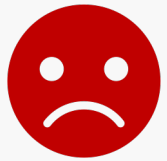
#2: Match between system and the real world Tips

- Ensure that users can understand meaning without having to go look up a word’s definition.
- Never assume your understanding of words or concepts will match that of your users.
- User research will uncover your users' familiar terminology, as well as their mental models around important concepts.

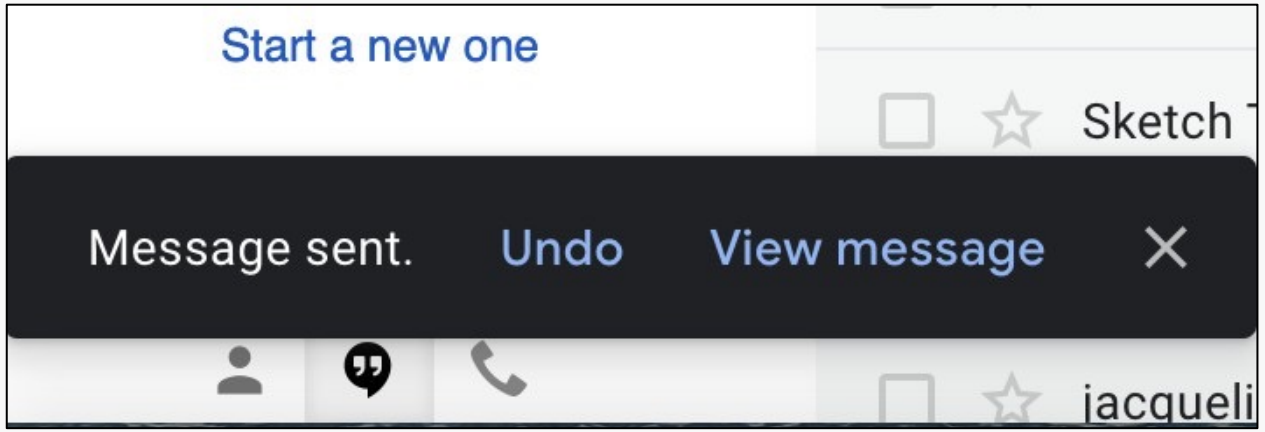
#3: User control and freedom

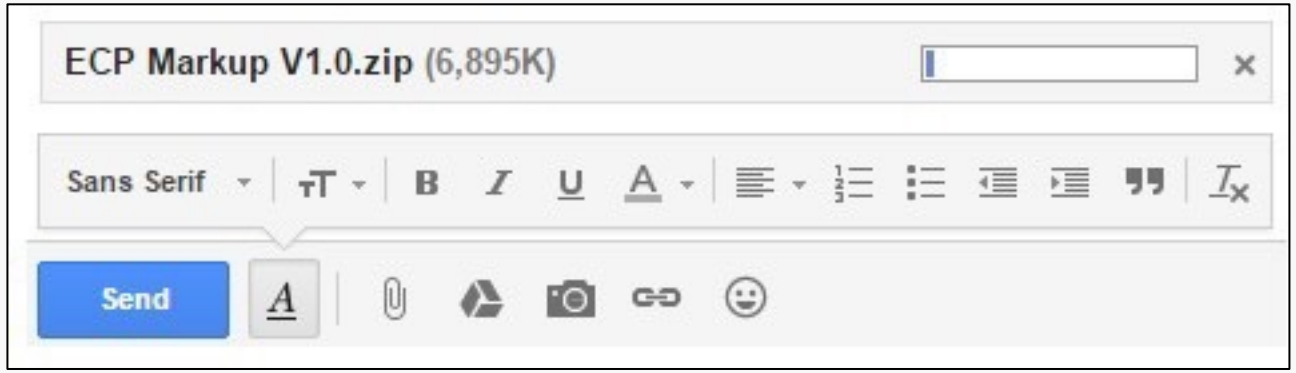
- Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

#3: User control and freedom

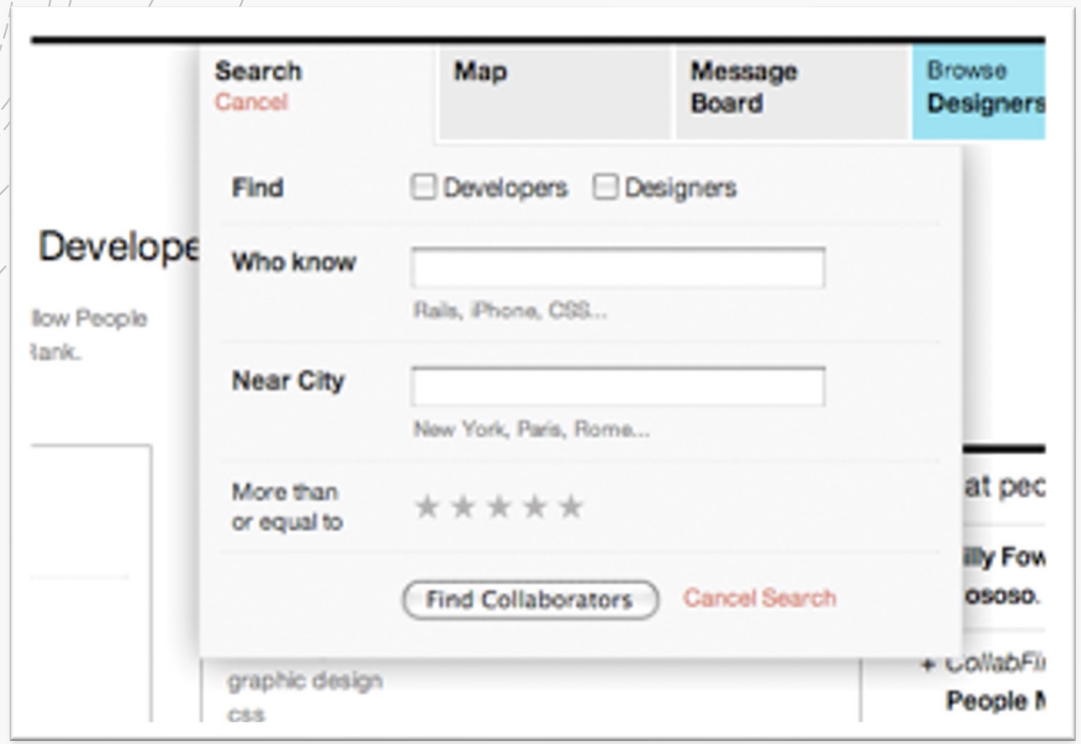


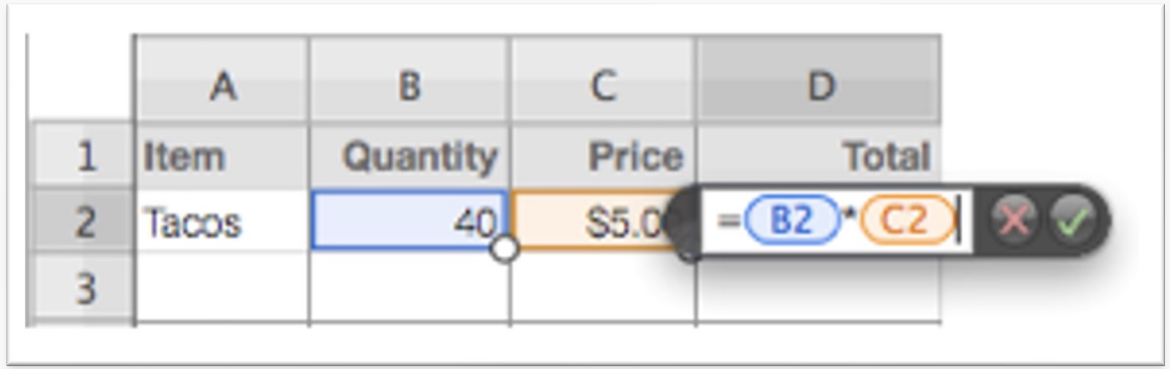
#3: User control and freedom

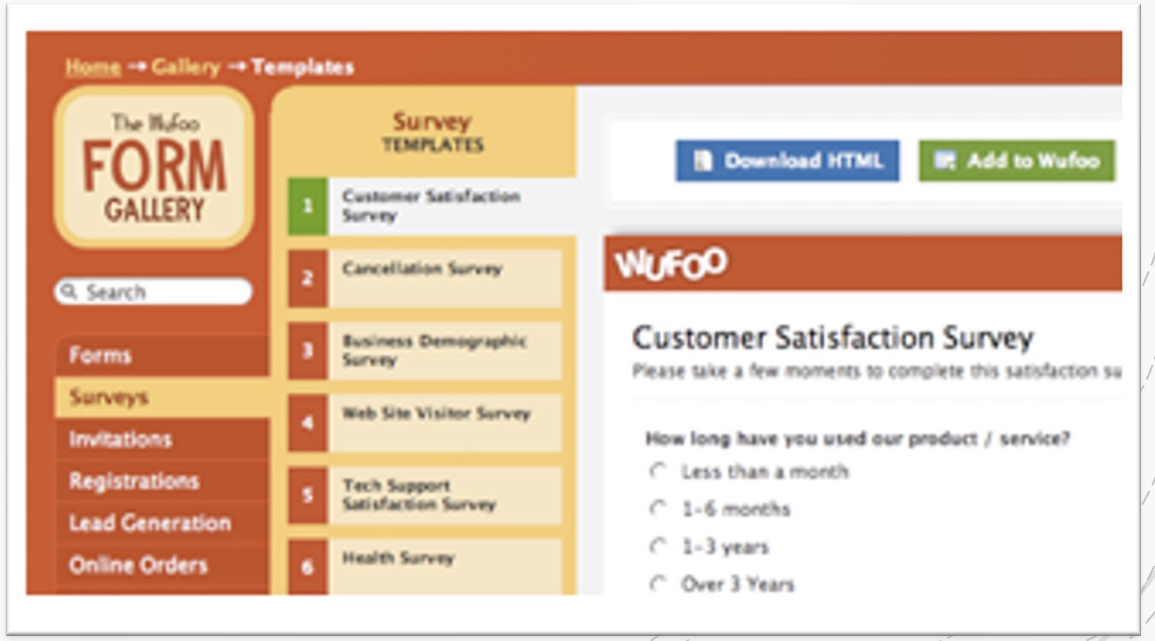




#3: User control and freedom







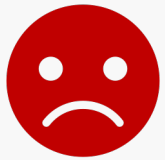
#3: User control and freedom Suggestions

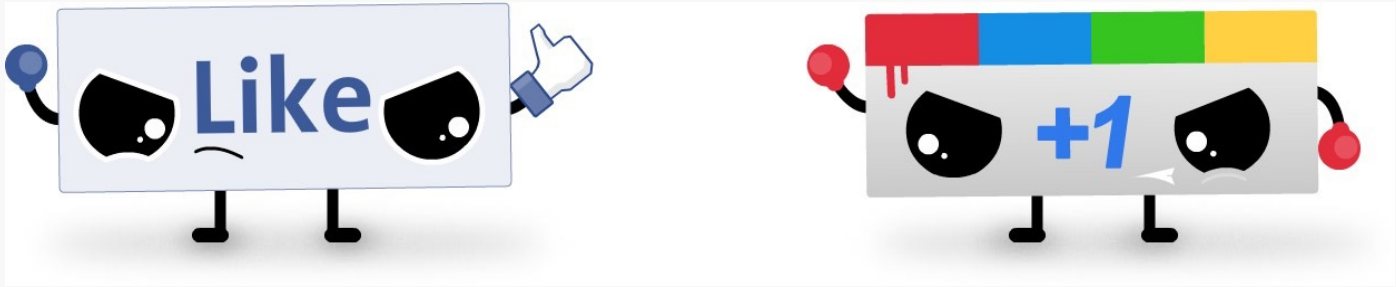
- Always provide a “back” (or equivalent) button
  - Support *Undo* and *Redo*.
- Allow users to “explore” different alternative paths
  - Except for one-shot wizard-like paths, aimed at novices or first-time users
- Show a clear way to exit the current interaction, like a *Cancel* button.
- Make sure the exit is clearly labeled and discoverable.

#4: Consistency and standards

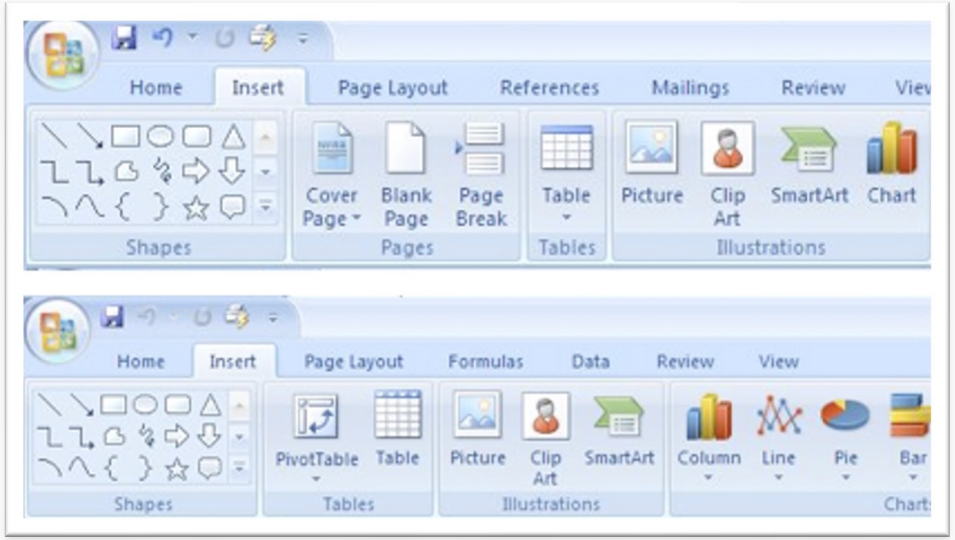
- Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Failing to maintain consistency may increase the users' cognitive load by forcing them to learn something new.

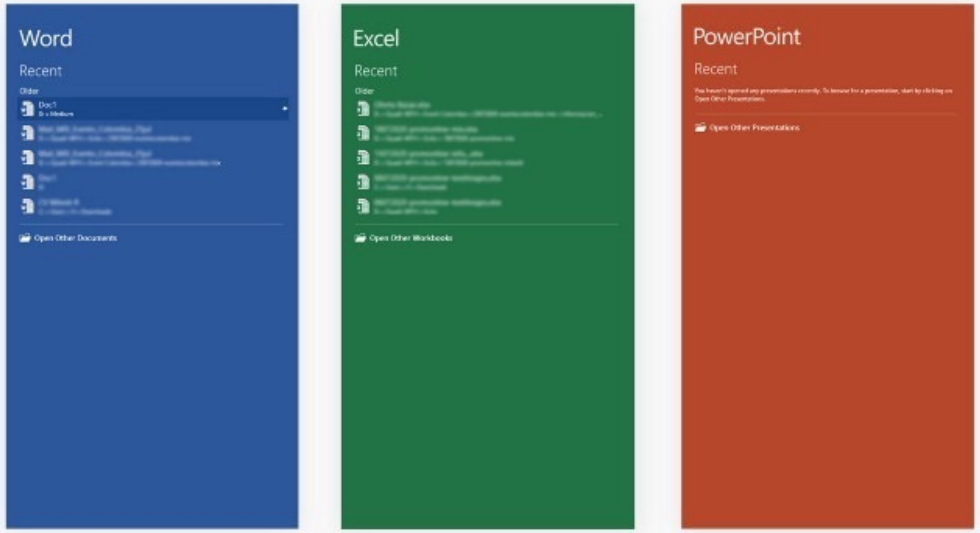
#4: Consistency and standards





#4: Consistency and standards





https://www.youtube.com/watch/Ibndy9KLOSQ

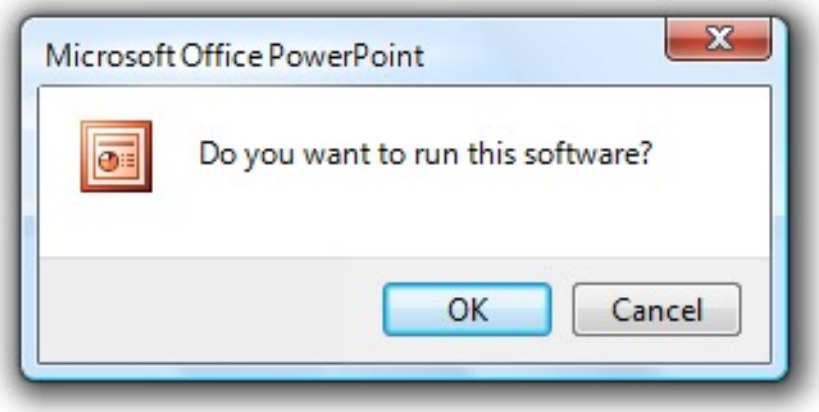
#4: Consistency and standards Suggestions

- Improve learnability by maintaining both types of consistency: internal and external.
- Maintain consistency within a single product or a family of products (internal consistency).
- Follow established industry conventions (external consistency).

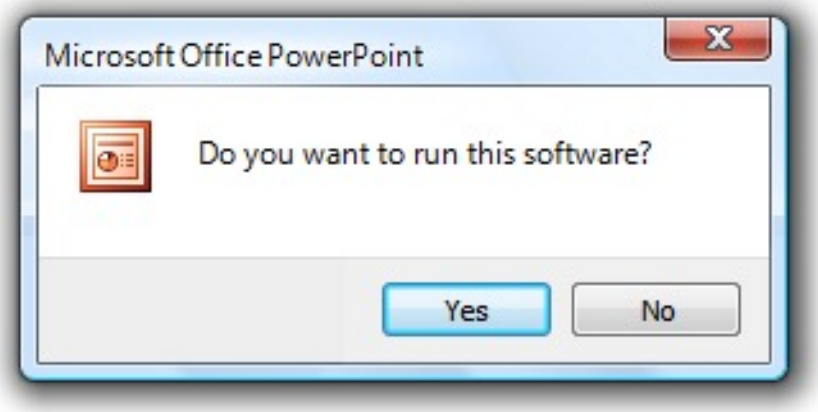
#4: Consistency and standards Suggestions

- Consistent layout for dialogs and forms
  - E.g., position of the navigation elements
  - E.g., position of the confirmation buttons
- Consistent meaning for Ok/Cancel, Yes/No choices
  - E.g., avoid: “Do you want to interrupt task?”
  - Still better, label buttons with the actual effect “Insert”, “Interrupt”, …
- Categories, lists of names, geographical regions, etc, should be taken from “standard” vocabularies

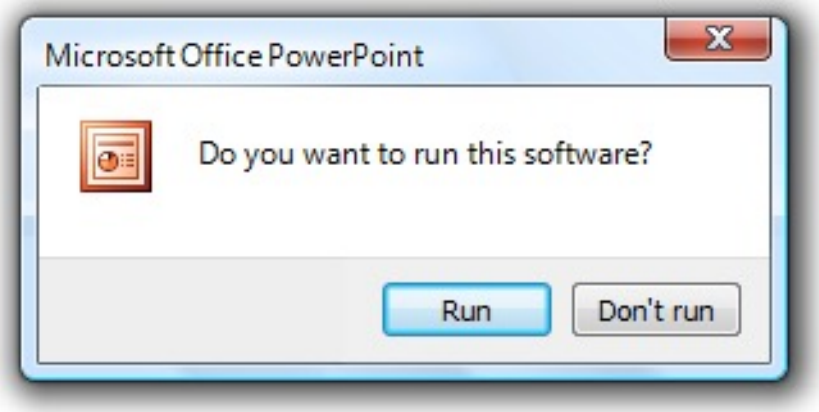
#4: Consistency and standards Examples



Bad



Acceptable



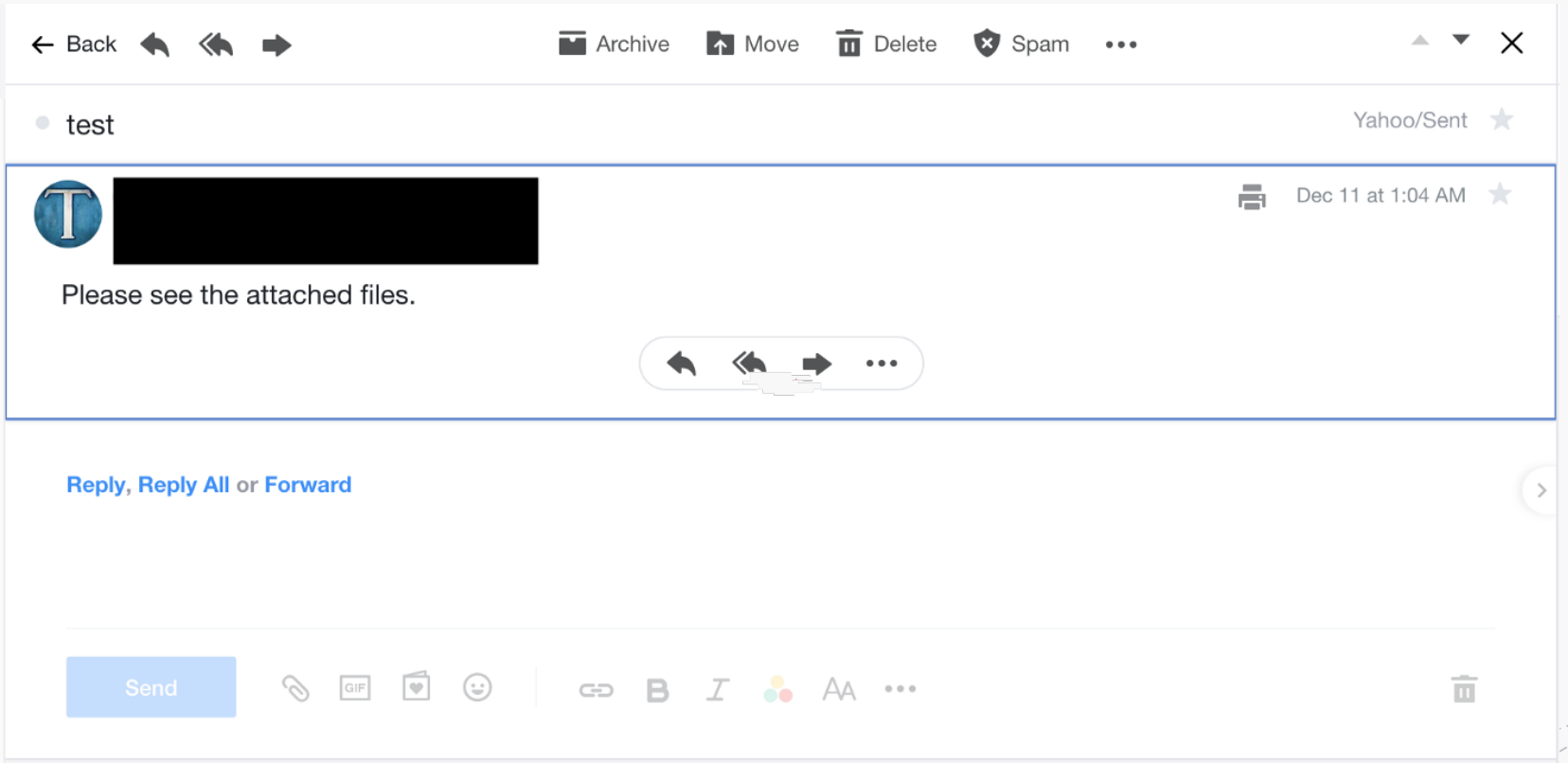
Better

#5: Error prevention

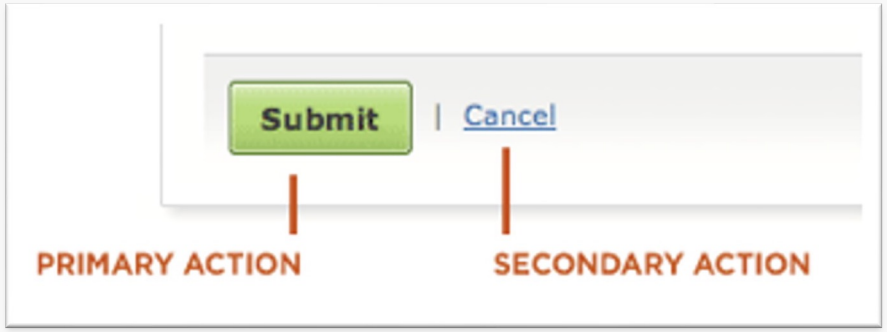
- Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

https://www.nngroup.com/articles/slips/

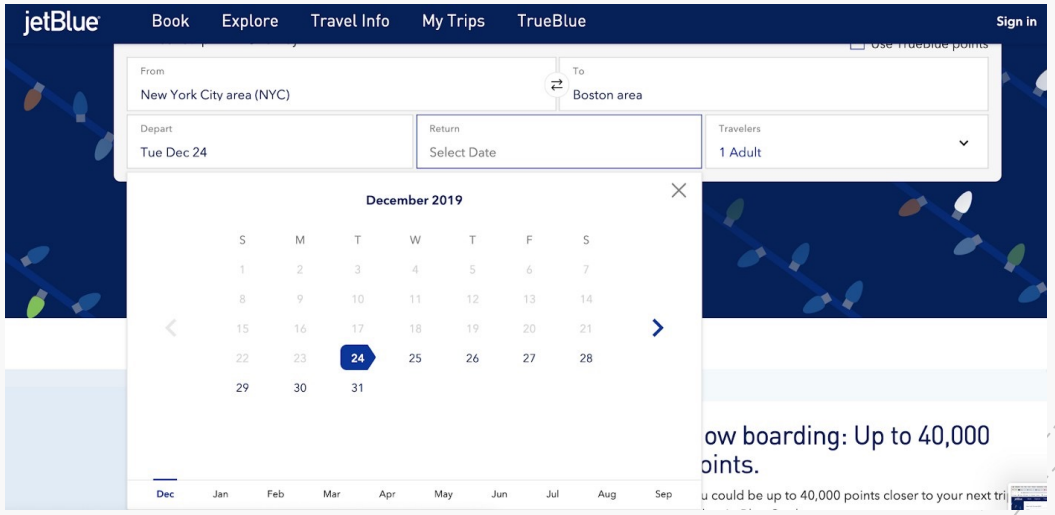
#5: Error prevention



#5: Error prevention



#5: Error prevention



https://www.youtube.com/watch/imS9s1DUY-I

#5: Error prevention - Suggestions

- Preventing data loss
- Prevent clutter
- Prevent confusing flow
- Prevent bad input
- Prevent unnecessary constraints (e.g., provide defaults for missing data)

#5: Error prevention - Suggestions

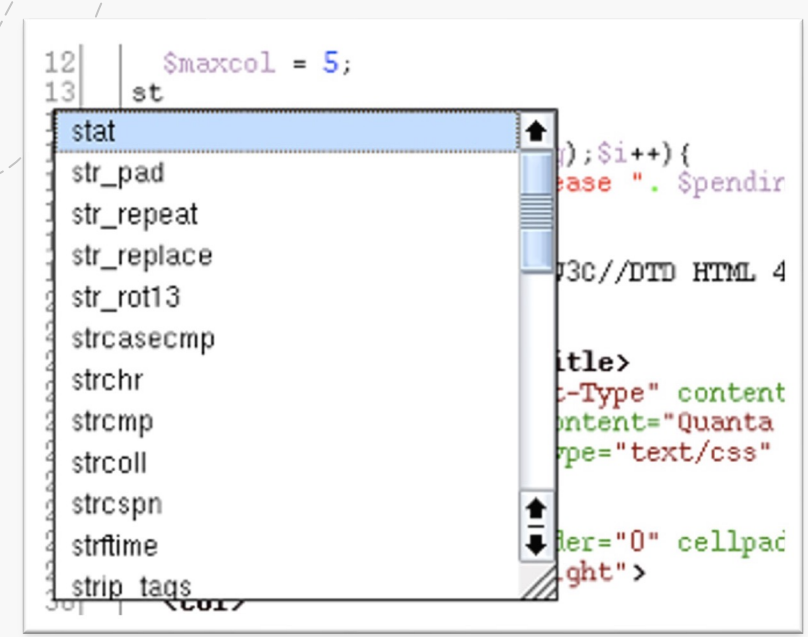
- Prioritize your effort: Prevent high-cost errors first, then little frustrations.
- Avoid slips by providing helpful constraints and good defaults.
- Prevent mistakes by removing memory burdens, supporting undo, and warning your users.

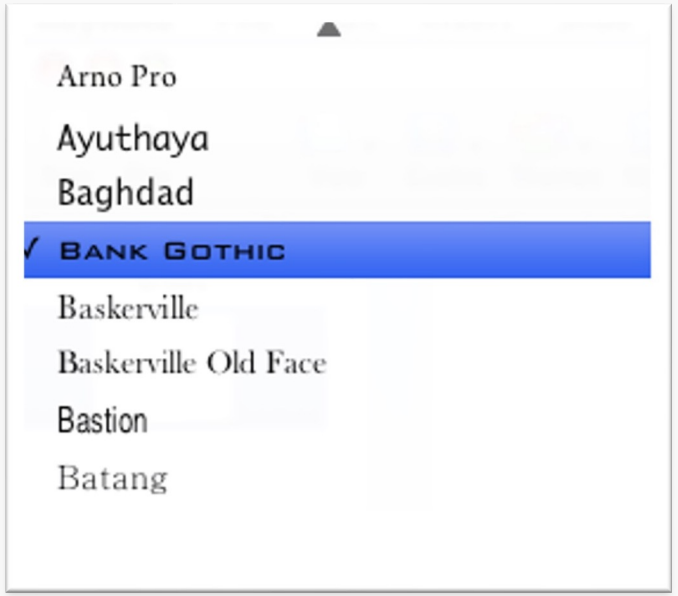
#6: Recognition rather than recall

- Minimize the user's memory load by making objects, actions, and options visible.
- The user should not have to remember information from one part of the interface to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

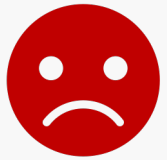
https://www.nngroup.com/articles/recognition-and-recall/

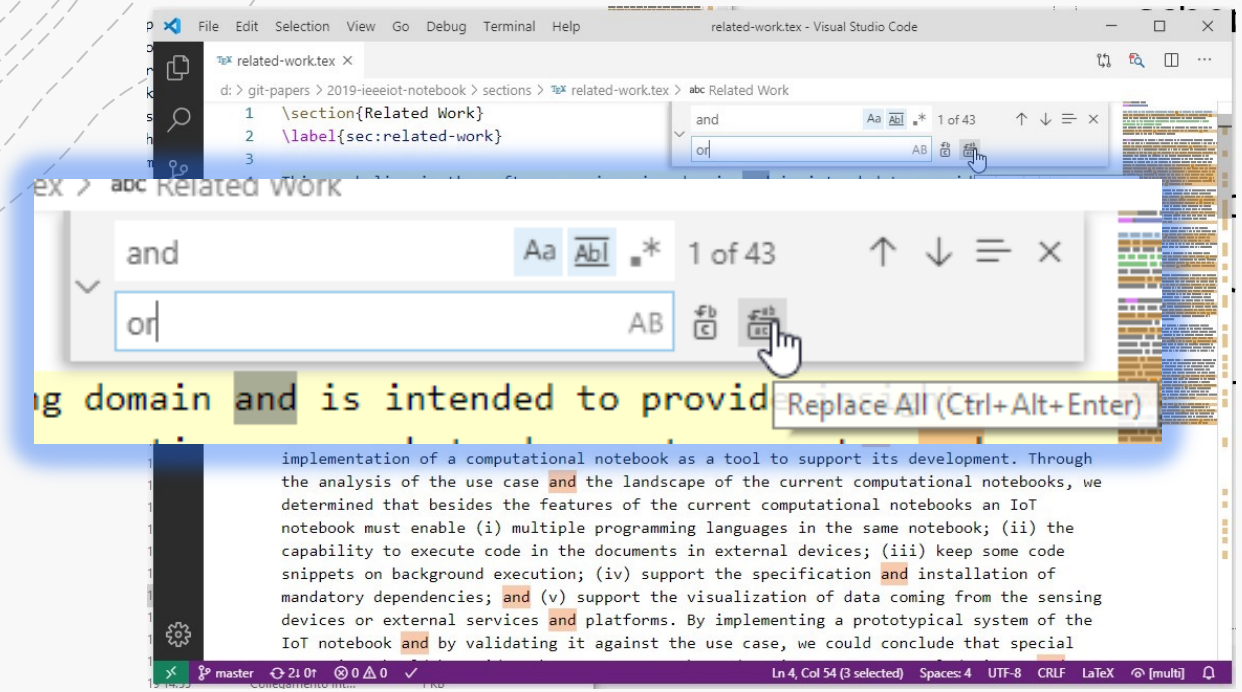
#6: Recognition rather than recall

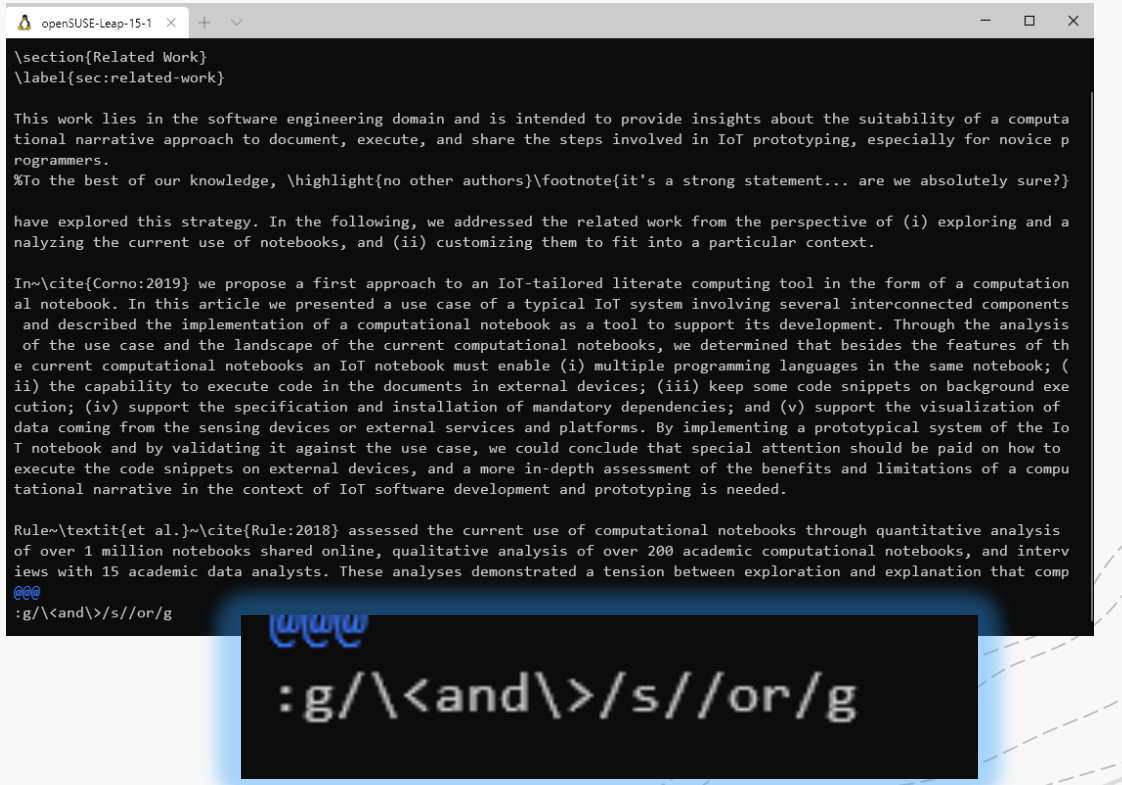




#6: Recognition rather than recall

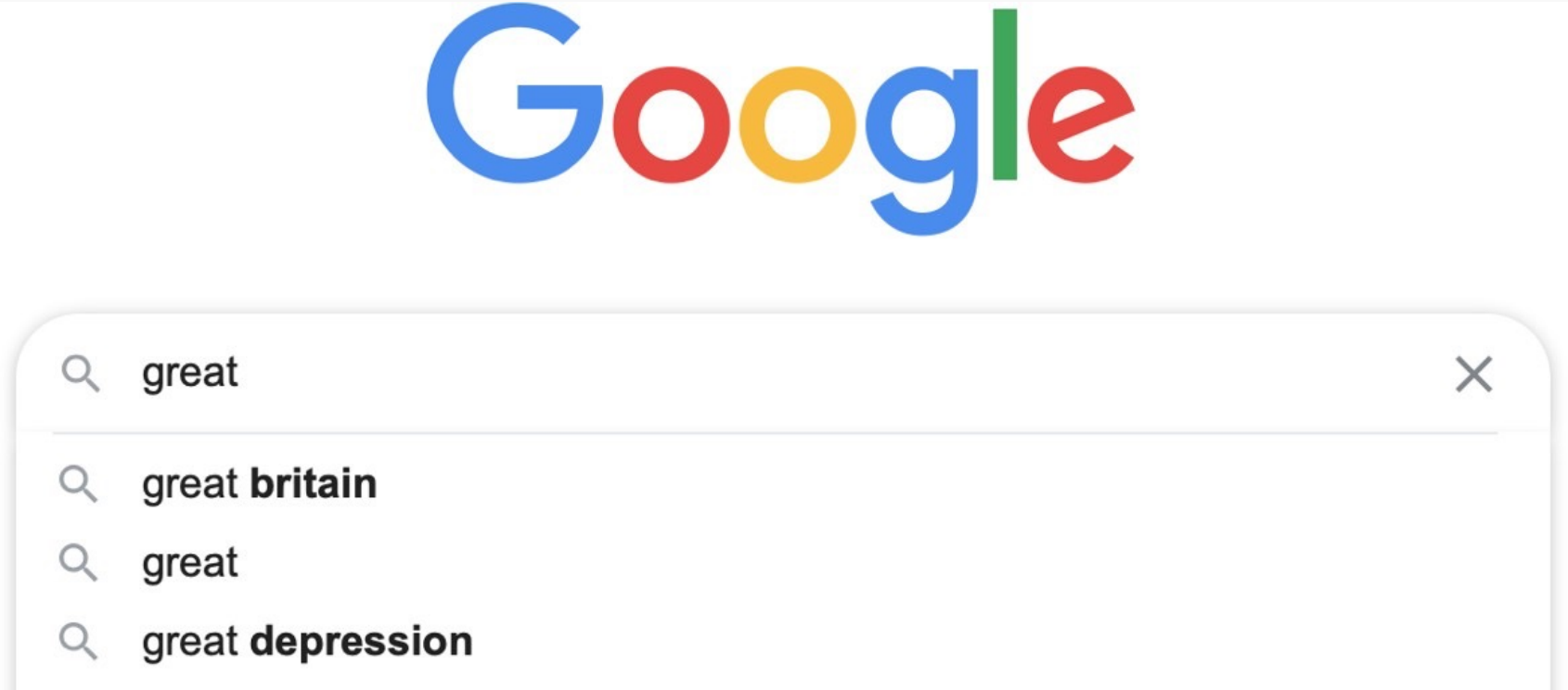






Nielsen's Usability Heuristics

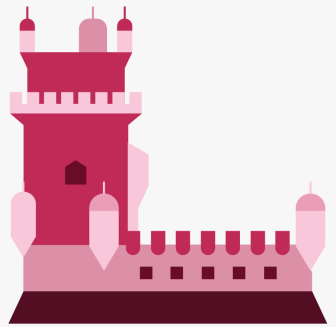
- **#5 Recognition rather than recall**



https://www.youtube.com/watch/6glQPp6q4Jc

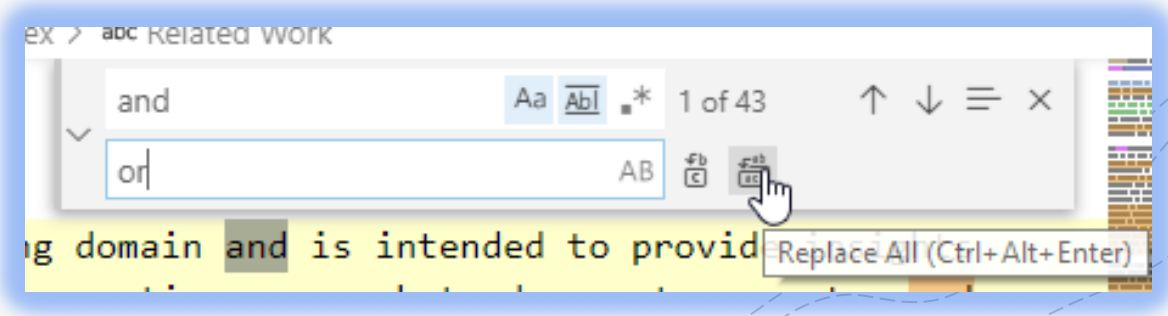
#5 Recognition rather than recall Suggestions

- Let people recognize information in the interface, rather than forcing them to remember (“recall”) it.
- Offer help in context, instead of giving users a long tutorial to memorize.
- Reduce the information that users have to remember.



#5 Recognition rather than recall
Suggestions

- Avoid codes (use explicit names)
  - e.g., L, VL, EL, EA, …???
- Avoid extra hurdles
  - e.g., asking for unnecessary (or premature) information
- Provide previews
  - Code completion
  - Page preview
  - Order summary
  - Itinerary
  - …



#7: Flexibility and efficiency of use

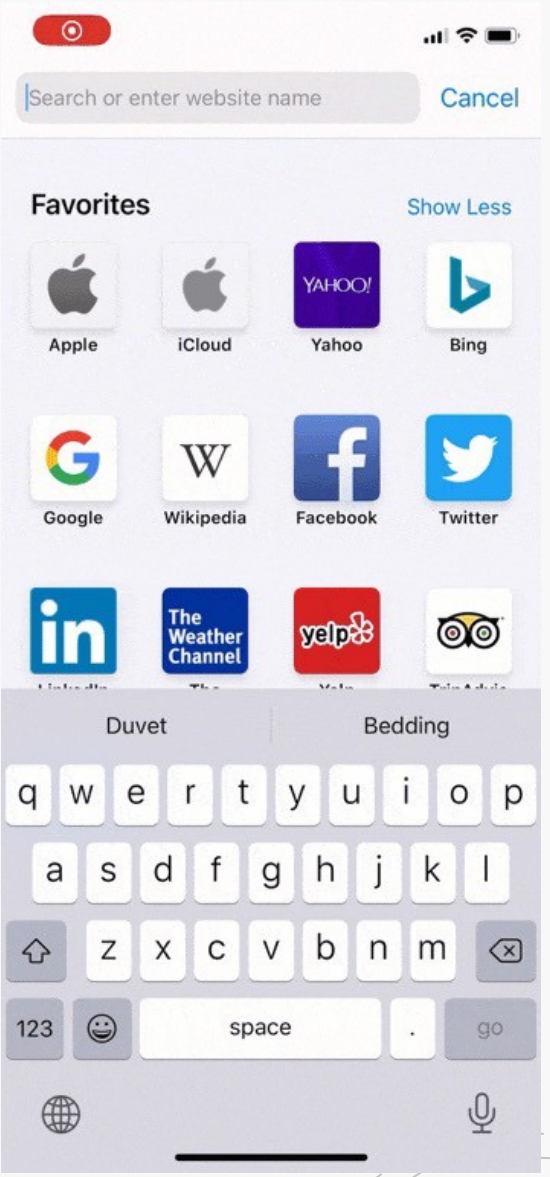
- Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- Shortcuts – hidden from novice users – may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
- Provide personalisation by tailoring content and functionality for individual users.

#7: Flexibility and efficiency of use

Common Shortcuts

Add Action	Return
New Window	⌘N
Synchronize with Server	^⌘S
Clean Up	⌘K
Planning Mode	⌘1
Context Mode	⌘2
Inbox	⌥⌘1
Quick Entry	^⌥Space

Quick Entry's shortcut can be customized in Preferences



https://www.youtube.com/watch/LoTdRTBB8BQ

#7: Flexibility and efficiency of use Suggestions

- Flexibility = Default + Options
  - E.g., present some popular choices, but let the user enter a custom one (train ticket machines)
- Exploit background information for providing more information
  - E.g., weather forecasts in a calendar interface
- Novice and Expert Users Have Different Needs
  - Support proactivity, personalization, and different interaction techniques!
- Recommendations
- Provide relevant information, only

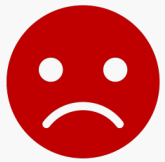
#7: Flexibility and efficiency of use Suggestions

- Provide *accelerators* like keyboard shortcuts and touch gestures.
- Provide *personalization* by tailoring content and functionality for individual users.
- Allow for *customization*, so users can make selections about how they want the product to work.

#8: Aesthetic and minimalist design

- Interfaces should not contain information which is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.
- Prioritise the content and features to support primary goals.

#8: Aesthetic and minimalist design



rand of Gate Openers and Operators, Elite, Viking, Doorking, Power Master, Ramset, Allstar, FAAC, Apollo, SEA. We are also manufacturer of Custom gates in Aluminum, Iron, Steel our Ornamental gates are second to none. For your Driveway automatic electric gates entrance we offer a full line of Access
ent, Telephone entry system, intercom, keypad and gate accessories and safety devices, loop detector, safety loop, photo cell. Offering complete Custom decorative fencing matching designs and style, 100's of fence pictures to choose, picket fence, deck, pool, garden, estate, modern, we have it all at gre

Welcome To: Gates N Fences

L.A. Ornamental Corp
3708 N.W. 82nd Street
Miami, Florida 33147
Phone: 305-696-0419
LAOrnamental@Aol.com

Designed to Enhance the Entry of your home with Custom Ornamental Decorative Driveway Gates while bringing Safety, Security and convenience.

Home
Driveway Gates
Modern Driveway Gates
Custom Driveway Gates
Privacy Driveway Gates
Garden Gates
Modern Garden Gates
Privacy Garden Gates
Fencing
Fencing_2
Railings
Modern Balcony Railings
Openers - Operators
Ramset
FAAC

Search

Top Brands of Gate Openers and Operators, Commercial, Residential, Industrial, Swing, Slide, Rack & Pinion, Barrier

- BFT Gate Openers
- PowerMaster Gate Openers

All of our Aluminum or Wrought Iron Gates, or Fences are designed and manufactured to withstand a range of outdoor conditions. Our commitment to our customers and dedication to produce quality gates has earned us thousands of satisfied customers.

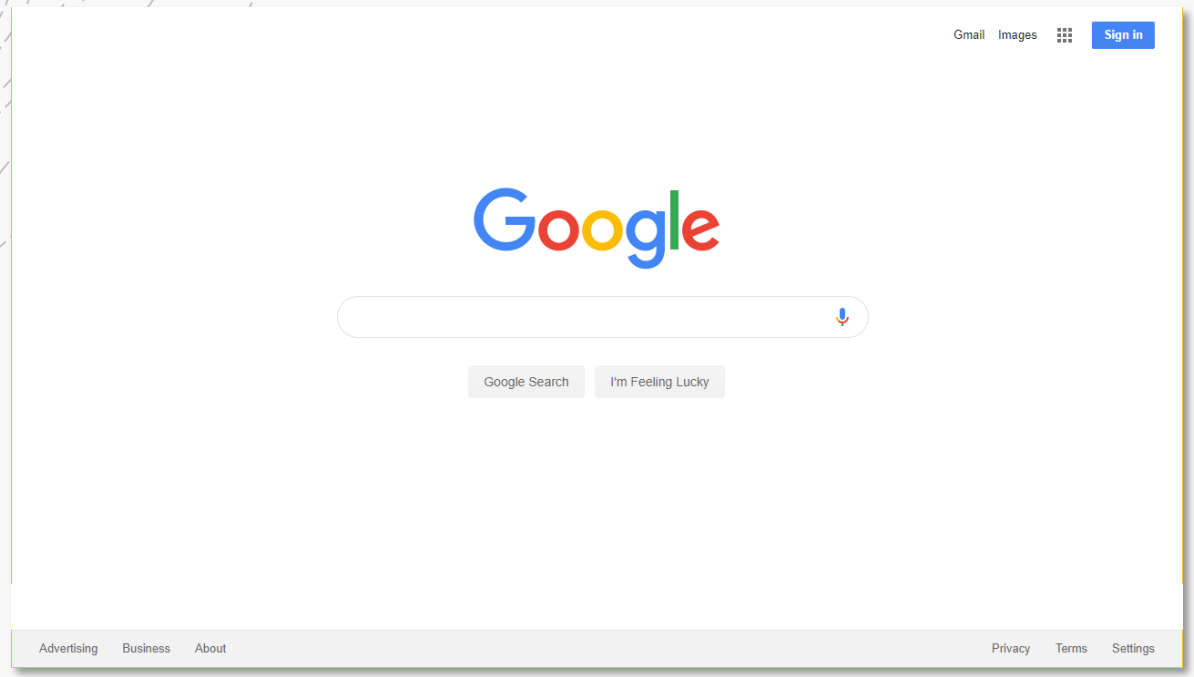
Although we offer a wide selection or Ornamental Designs or Decorative Designs, we can design and manufacture any style in aluminum or wrought iron metals. L. A. Ornamental & Rack Corp also offers Fences, Garden or Walk Thru Gates to match your driveway gates. With over thirty five years of experience in manufacturing and designing elegant, custom, or exotic Aluminum Driveway Gates and Fences, our past and future customers can have peace of mind that they are receiving quality workmanship. We are a Fence Company that gives our customers 110% of dedication to manufacture quality driveway gates and fences.
For a quote please send an e-mail to LAOrnamental@aol.com

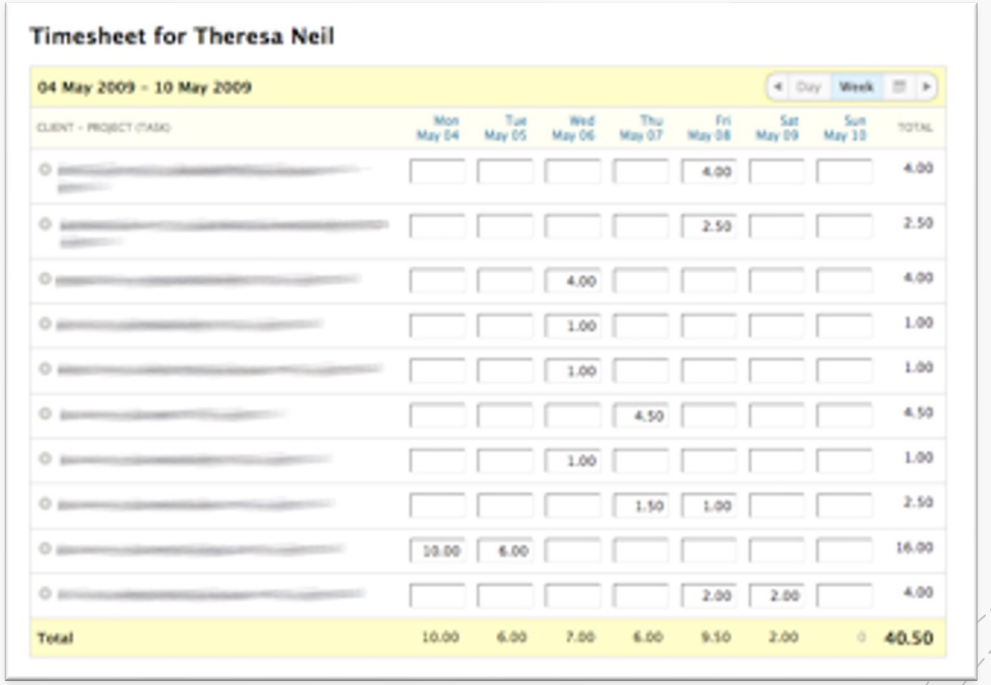
If your looking For Privacy with your Driveway Gates ,Garden Gates, or Walk Thru Gates, we offer a Solid Backing with your choice of Aluminium, Steel, Plexiglas or Plastic. All solid backing are offered in many different colors to choose from. Privacy Gates

We offer a large selection of Gate Openers and Gate Operators for Residential Driveway Gates, Light or Heavy Commercial Gates, or industrial locations. If your not sure the style or size of the Gate opener / gate operator you need, please e-mail or contact us so we can gladly help guide you to the correct choice. We offer all type of Gate Openers / Gate Operator, Sliding Gate Openers / Gate Operator, Swing Gate Openers / Gate Operator, Hydraulic Gate Openers / Gate Operator. We also have a wide selection of replacement Main Circuit Boards for all brands, and Remote Controls for Visors or Keychains.

Railings - L. A. Ornamental Rack Corp offers top quality Balcony Railings, Front Porch Railings, Deck Railings in Metal, Aluminum, or Wrought

#8: Aesthetic and minimalist design





https://www.youtube.com/watch/ZgbRmeWDgd0

#8: Aesthetic and minimalist design Suggestions

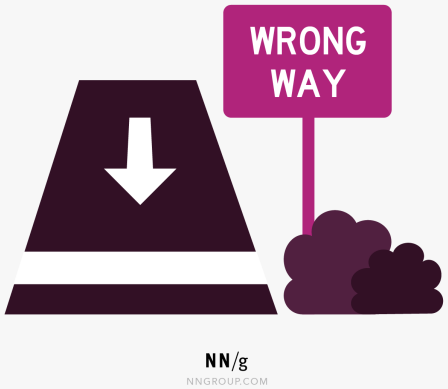
- Key information must be “above the fold”
  - Especially on low-resolution devices
- Keep high signal-to-noise ratio
  - Colors, fonts, backgrounds, animations, …
  - Borders, dividers, …
- Minimalistic login experience
- Accept redundant ways of entering information
- Prune features that are outside the “core” functionality

#8: Aesthetic and minimalist design Suggestions

- Keep the content and visual design of UI focused on the essentials.
- Don't let unnecessary elements distract users from the information they really need.
- Prioritize the content and features to support primary goals.

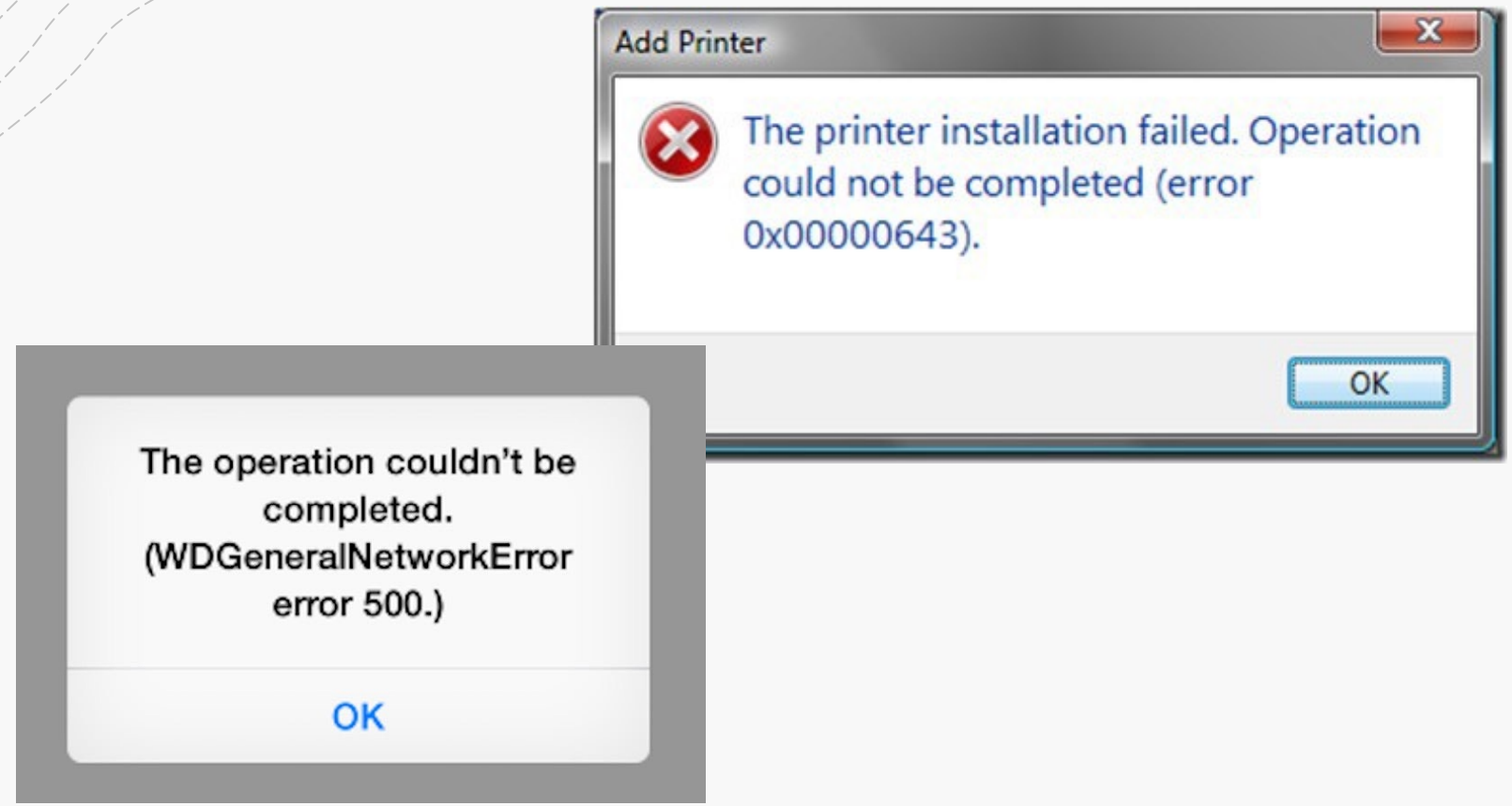
#9: Help users recognize, diagnose, and recover from errors

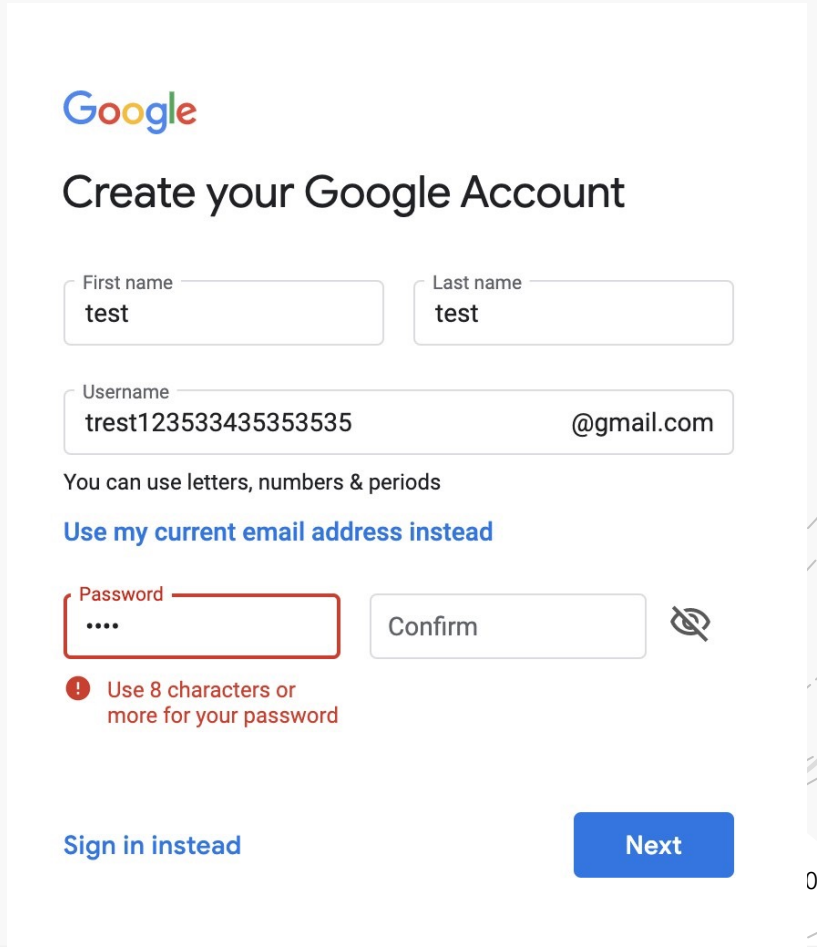
- Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.



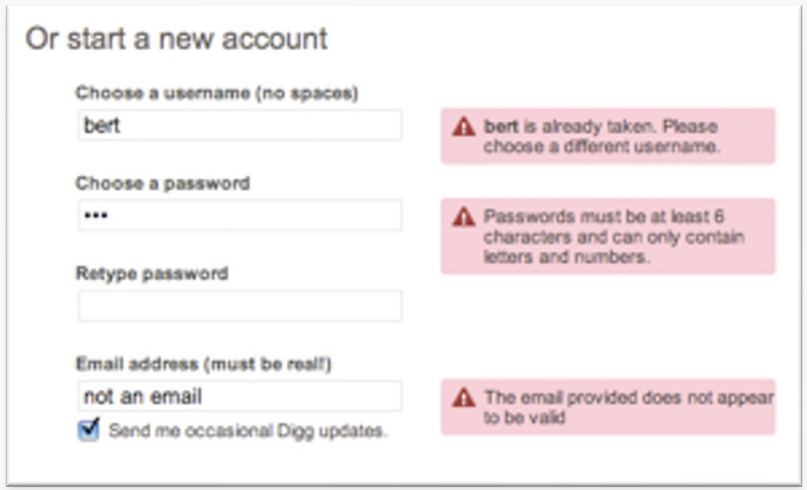
#9: Help users recognize, diagnose, and recover from errors

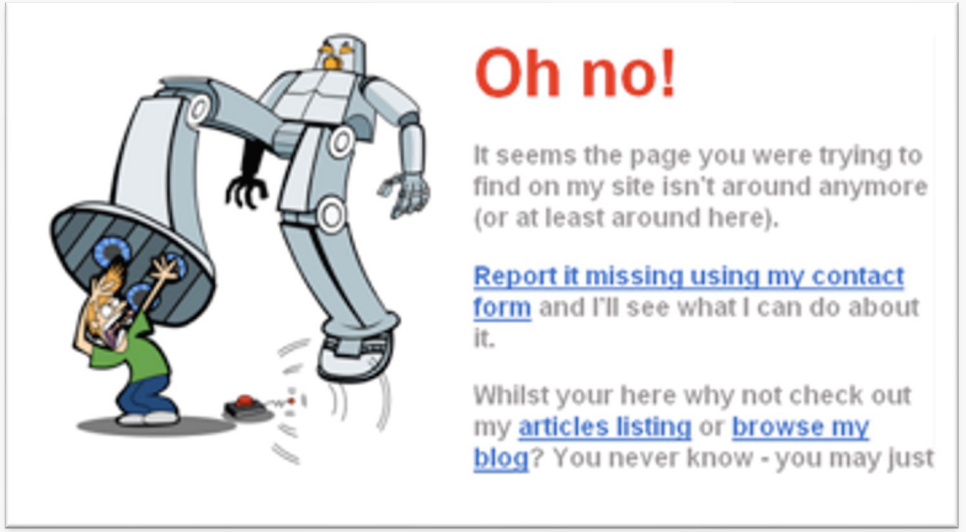
#9: Help users recognize, diagnose, and recover from errors





#9: Help users recognize, diagnose, and recover from errors





NN/g

User Errors

https://www.youtube.com/watch?v=cCun-ReLTFI

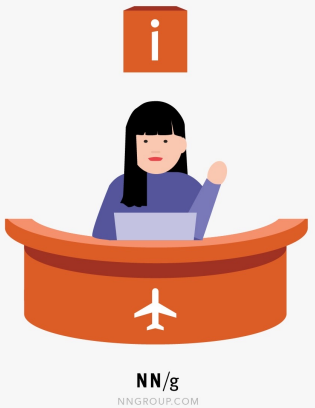
#9: Help users recognize, diagnose, and recover from errors - Suggestions

- Use traditional error-message visuals, like bold, red text.
- Tell users what went wrong in language they will understand – avoid technical jargon.
- Offer users a solution, like a shortcut that can solve the error immediately.

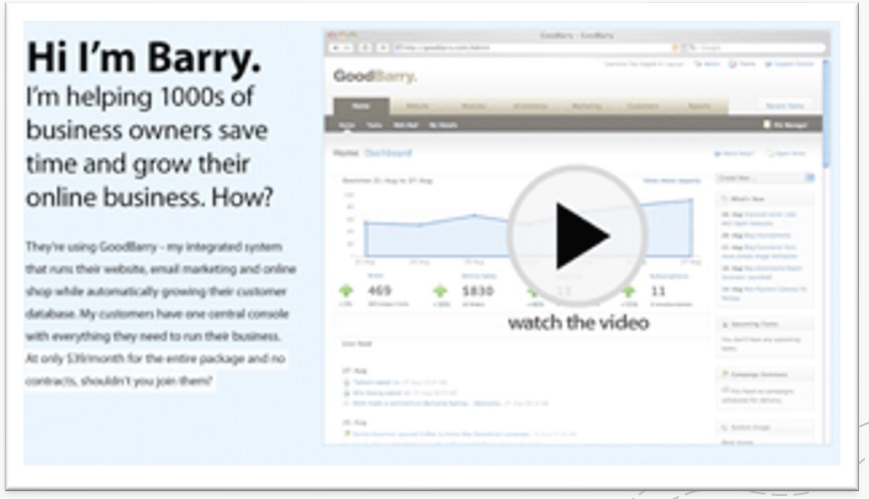
- Make errors easy to identify
  - Colors, fonts, …
- Make problem clear
  - Problem cause
  - Problem location
- Provide a solution
  - Give a suggestion
  - Show a path forward
  - Propose an alternative

#10: Help and documentation

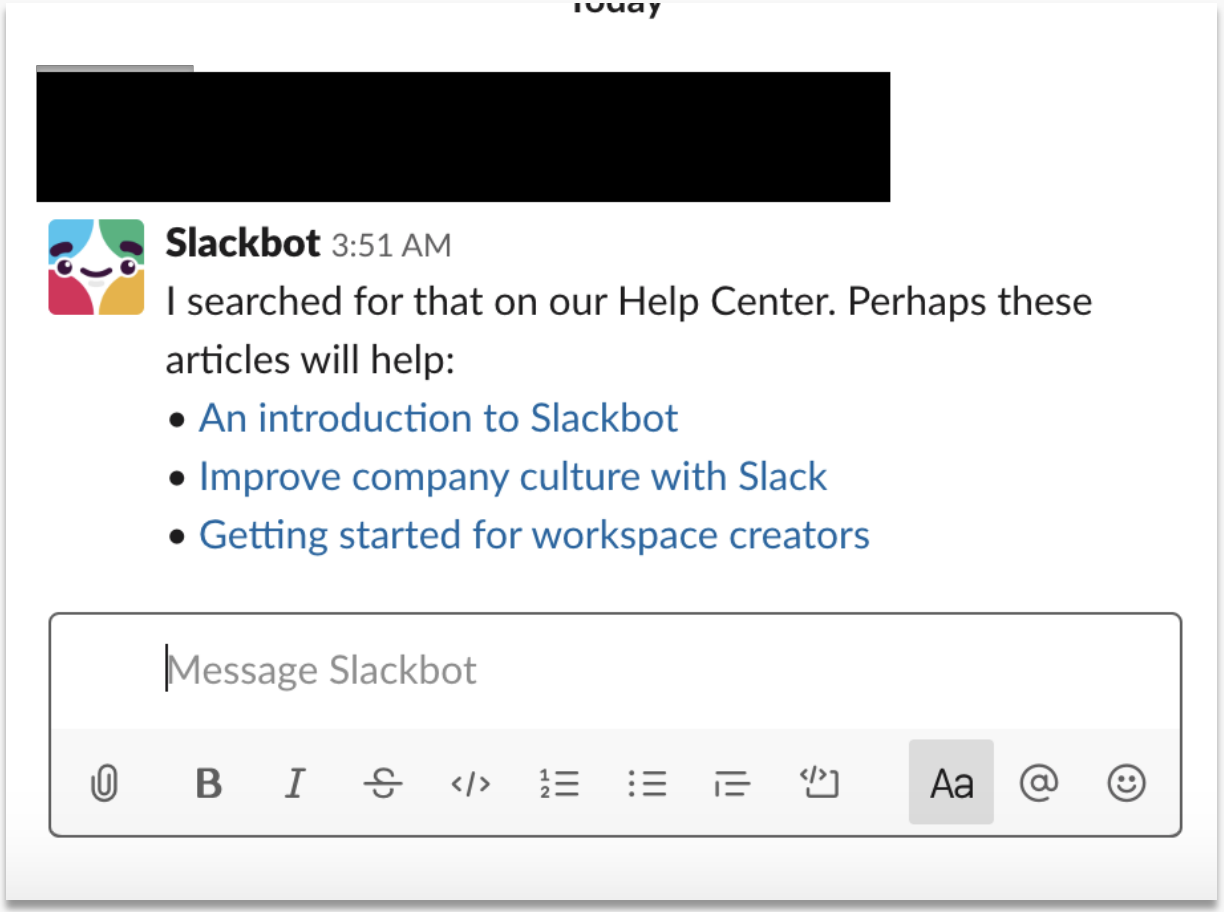
- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation.
- Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.



#10: Help and documentation



#10: Help and documentation



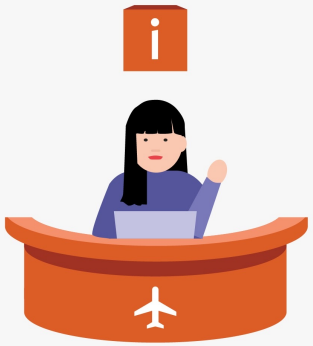
NN/g

Help & Documentation

https://www.youtube.com/watch/ilQVRzatb50

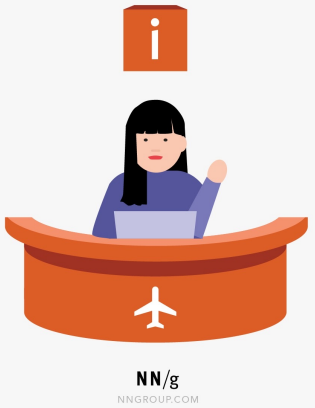
#10: Help and documentation - Suggestions

- Ensure that the help documentation is easy to searc
- Whenever possible, present the documentation in c the moment that the user requires it.
- List concrete steps to be carried out.



#10: Help and documentation - Suggestions

- Provide examples
  - In documentation
  - In complex choices
- Help the user understanding the error gravity
  - E.g., printing outside margins
- Provide ‘tips’ for showing new actions or steps
- Use pop-overs to point to changes in UI (or for first usage)
- Avoid too-opaque “terms and conditions” (summarize, if possible)



References and Acknowledgment

- Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale: Human Computer Interaction, 3 rd Edition
  - Chapter 9: Evaluation Techniques
- Ben Shneiderman, Catherine Plaisant, Maxine S. Cohen, Steven M. Jacobs, and Niklas Elmqvist, Designing the User Interface: Strategies for Effective Human-Computer Interaction
  - Chapter 5: Evaluation and the User Experience
- COGS120/CSE170: Human-Computer Interaction Design, videos by Scott Klemmer, https://www.youtube.com/playlist?list=PLLssT5z_DsK_nusHL_Mjt87THSTlgrsyJ

License

- These slides are distributed under a Creative Commons license "**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**"
- **You are free to:**
  - **Share** – copy and redistribute the material in any medium or format
  - **Adapt** – remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** – You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** – You may not use the material for commercial purposes.
  - **ShareAlike** – If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** – You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/

Vietnam National University of HCMC
International University
School of Computer Science and Engineering

THANK YOU

Dr Vi Chi Thanh - vcthanh@hcmiu.edu.vn
https://vichithanh.github.io