Mobile UI Design

Lecture 6

Goal of this lecture

- The mobile design mindset.
- Learn how to design and test easy to use systems:
  Process (User Centered Design, focus groups, usability testing, iterations), and
  Principles (GUI design principles, design patterns).
- NOT GUI programming, but how to design it from scratch.

Content

- Mobile UI mindset.
- Usability concepts.
- User versus developer perspectives.
- User centered design.
- Good and bad examples.
- High-level design principles.
- QA testing.

The Graphical User Interface

GUI is part of Human-Computer Interaction (HCI) which is the study, planning and design of how people and computers work together.

→User Interface (UI) is what users see, hear, touch, talk to or control and direct.

→UI is often "the product": what users see and may make their decision on.

UIs are everywhere

- PC
- Browsers
- Mobile
- TVs
- Phones
- Airplanes
- Cars
- Appliances

… So it must be critical

GUI versus CUI/CLI

CUI and GUI are both used in connection with computers, but:

- CUI is the precursor of GUI and users have to type on keyboard to proceed.
- GUI easier to navigate.
- Most modern computers use GUI.
- DOS is an example of CUI, whereas Windows is an example of GUI.

GUI

CUI

GUI translates in Usability

Usability refers to ease of use of the product, and includes everything the user comes in contact with:

- Documentation
- Installation
- Purchase, registration
- Support
- API and SW tools usability

For non-SW products:

- Packaging
- Assembly
- Ergonomics (physical)

E.g. smell of brand new original packaging

Importance of GUI

Increasingly critical for product success (e.g. Apple/Samsung patent battle)

- Users expect systems to work, but will chose those that are easier to use.
- Industry looking for SW engineers who can provide great "user experience" (that's you).

There are formal methods to asses usability of GUI!

What is good GUI, what is bad GUI?

To answer this ask yourself first:

→Who is the user?

→What are users' tasks and goals?

What is good GUI, what is bad GUI?

Reactions to bad GUI

If part of your work-related apps:

- Stress, fatigue, panic, frustration, annoyance,
- Low productivity, errors,
- Increased cost of doing business.

If for personal use:

- Frustration, annoyance,
- Return/abandon/uninstall, never use it again,
- Never buy product from that company!

Goal: Total User Satisfaction

Consider the total user experience: from "opening the box" to calling support:

- Features
- User Interface
- Response Time
- Reliability
- Easy to install/uninstall
- Help and docs
- Maintainability
- Graphics/attractiveness

Your Goal

Make systems that have all three Us:

1. Useful
2. Usable
3. Used

Example apps?

Benefits of good UI: Business

1. Application which involves users going through 4.8 million pages/year. Just 1 second savings per page saves 0.7 person/year of work.
2. Screen users in one study were 20% more productive with better screen design or performed transactions in 25% less time with 25% less errors.
3. A study on online shopping showed that for better designed sites task completion was 65% higher (= $$$)

Your experience?

Benefits of good UI: Mobile

Say you are buying a mobile gadget:

1. Product A: has a good price, tons of features BUT does not look cool, it's hard to activate, packaging is boring, feels cheap and shabby, has confusing UI.
2. Product B: cool, including packaging, delivery, easy to activate, feels good in your hand, easy to use by touch, hints and gestures BUT more expensive → most likely the winner.

Fundamental Observations of GUI Design

I. UI and usability involves people and is for people (GUI design principles, design patterns).

II. Careful design with user in the loop has to precede coding.

III. Design/Testing/evaluation has to involve human subjects other than designers.

IV. UI code is large, not algorithmically deep but very detailed. Its development/QA is often grossly underestimated and consists of many small simple steps.

V. Text/language on the screen is critical.

Perspectives on UI Development

User perspective

- Satisfies user needs
- Easy to learn, intuitive
- Permits user to focus on tasks and information
- Fast response
- Polished, attention to every detail
- Good text –focused, task oriented
- Works same on all devices, screens, resolutions

Developer perspective

- Built on time, within the budget
- Sells well
- Users are happy
- Easy to maintain
- Works/adapts to variety of delivery devices
- Easy to support

Mobile Interface Design Principles

Mobile is different.

Smartphones are more “powerful” than desktops in many ways:

- They are highly personal,
- always on,
- always with us,
- usually connected and directly addressable.
- fitted with powerful sensors that can detect location, movement, acceleration, orientation, proximity, environmental conditions and more.

Mobile Interface Design Principles

1. Mobile **mindset**

→Be focused: more is not better. You are going to have to leave features out.

→Be unique: know what makes your app different and amplify it.

→Be charming: mobile devices are intensely personal. Apps that are friendly, reliable and fun are a delight to use.

→Be considerate: app developers too often focus on what would be fun to develop, but you have to put yourself in your users' shoes if you ever hope to create an engaging experience.

Mobile Interface Design Principles

2. Mobile **context** – three major contexts:

→Bored (e.g., '*on the couch at home*'): immersive experiences geared toward a longer usage session are desired. Interruptions are highly likely so be sure your app can pick up where your user left off.

Examples: Facebook, Twitter, Angry Birds, web browser.

→Busy (e.g., '*running though the airport*'): accomplish micro-tasks quickly and reliably with one hand is critical. Remember that the user will have tunnel vision in this context, so huge targets and bold design are important.

Examples: TripIt, email, calendar, banking.

Mobile Interface Design Principles

2. Mobile **context** – three major contexts

→Lost (e.g., '*in transit, unfamiliar surroundings*', or interested in something unknown): sketchy connectivity and battery life are big concerns, so you should offer some level of offline support and be sparing with your use of geolocation and other battery hogs.

Typical examples: Maps, Yelp, Foursquare.

Mobile Interface Design Principles

3. Global guidelines – features that always matter

→Responsiveness – not speed, but real-time user feedback,

→Polish – the little details,

→Thumbs – design for the default typing fingers,

→Targets – easy to tap & avoid mistakes,

→Content – very intuitive in it's nature, don't add chrome,

→Controls – at the bottom of the screen (mouse vs finger),

→Scrolling – avoid it or animate.

Mobile Interface Design Principles

4. Navigation models – getting around the app screens

→None: single screen utility apps

Example: Weather app on iPhone

→Tab bar: three to six distinct content areas

Example: Twitter for iPhone

→Drill down: List and detail content hierarchy

Example: Settings app

Mobile Interface Design Principles

5. User input – not very easy

→Keyboard variations (text, numbers, email, URL) for each input field.

→Decide which auto-complete features to be enabled for each input type: auto-correct, auto-capitalisation and auto-complete.

→Support landscape orientation for long typing fields.

Mobile Interface Design Principles

6. Gestures – power-user shortcuts.

→Invisible – decide on a clever way to reveal them,

→Two hands – always have controls to support one-hand operation (e.g. maps zoom in on Android vs iOS),

→Just 'nice to have' – not critical for most users,

→Not a replacement – no common user vocabulary, so have visible controls.

Mobile Interface Design Principles

7. Orientation

→Portrait is the most popular orientation, so optimize for this case first.

→Landscape is important for typing a lot.

→Add orientation lock in app if it gets confusing.

```
<activity android:name=".yourActivity"
        android:screenOrientation="portrait“
        ...
/>
```

Mobile Interface Design Principles

8. Communications – letting the user know what is happening

→Provide instant feedback for every interaction.

→Modal alerts are extremely intrusive; use them only in critical situations, or the ask user confirmation.

→Confirmations are much less intrusive; response to user action.

Mobile Interface Design Principles

9. Launching

→When a user goes back into your app after having used it previously, you should resume operations right where the user left off. This will give the illusion of speed and contribute to an overall feel of responsiveness.

→Launch screen should be a *content-less* image of your app, not inviting the user to interact.

Mobile Interface Design Principles

10. First impresions

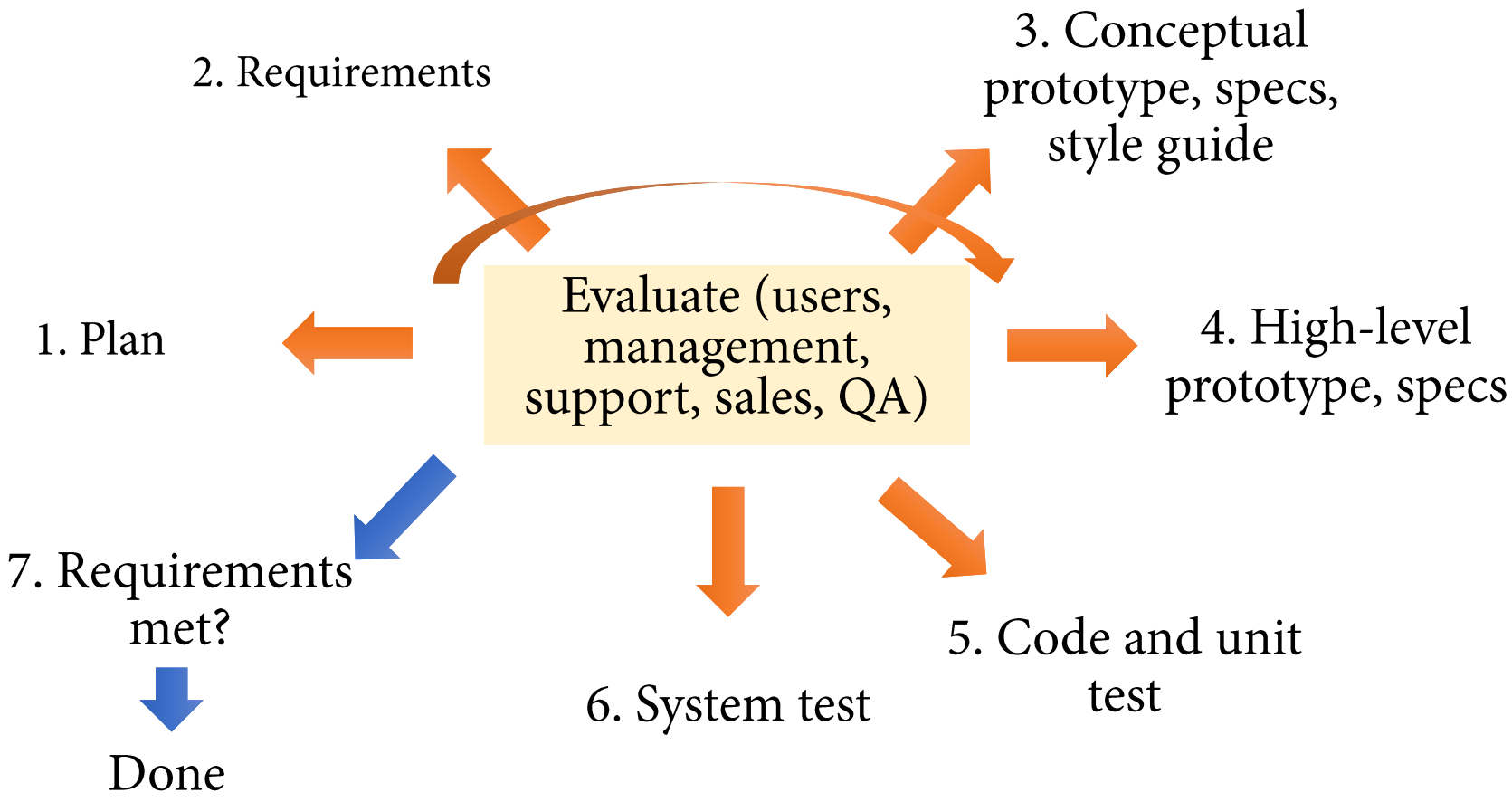
→ The app icon has to compete for attention in a sea of other icons.

- Business card rather than art piece.
- Show what your app does.
- Keep text to a minimum.
- A polished icon suggests a polished app.

→First launch is critical to app success

- Avoid making new users get confused or frustrated
- Include a 'tips and tricks' overlay if the app is complex
- No usage manual instead of good UI!

User-Centered Design and Development



UCD: Iterative, evaluated all the time by multidisciplinary team

UCD process

1. Develop use cases
2. Define and prioritize functions
3. Create paper black & white mockups (no programming)
   Review mockups with internal users or domain experts and developers (internal focus groups) – revise^2
4. Create horizontal prototype – clickable UI but no back end
   Review mockups with users or domain experts and developers (interval focus groups) – revise^2
5. Start coding
   Get feedback, revise
6. Show to wider audience including limited external users
   Get feedback, revise

Usability testing and focus groups

Focus groups (internal or external) – combine usability testing AND design improvements: preferred in early stages

- 3-8 people in the room observe the product together and try use cases
- Discuss problems, come up with better design
- *Book keeping* to record the decisions/tasks (usually a lot of small details)
- Changes made – keep the list of what was *changed*
- Repeat and iterate (internally, then externally)

Example user feedback

I suggest having a page title like "Join the project: register as *Citizen Scientist*". This way we also introduce the key term "citizen scientist" which is totally not visible now (see more on this)

"Your screen name"

I would omit this text. Stay focused on Registration, by now We explained what Is this about (this should be In ABOUT)

I would start by saying" **Get involved, become Citizen Scientist by registering On this page.**

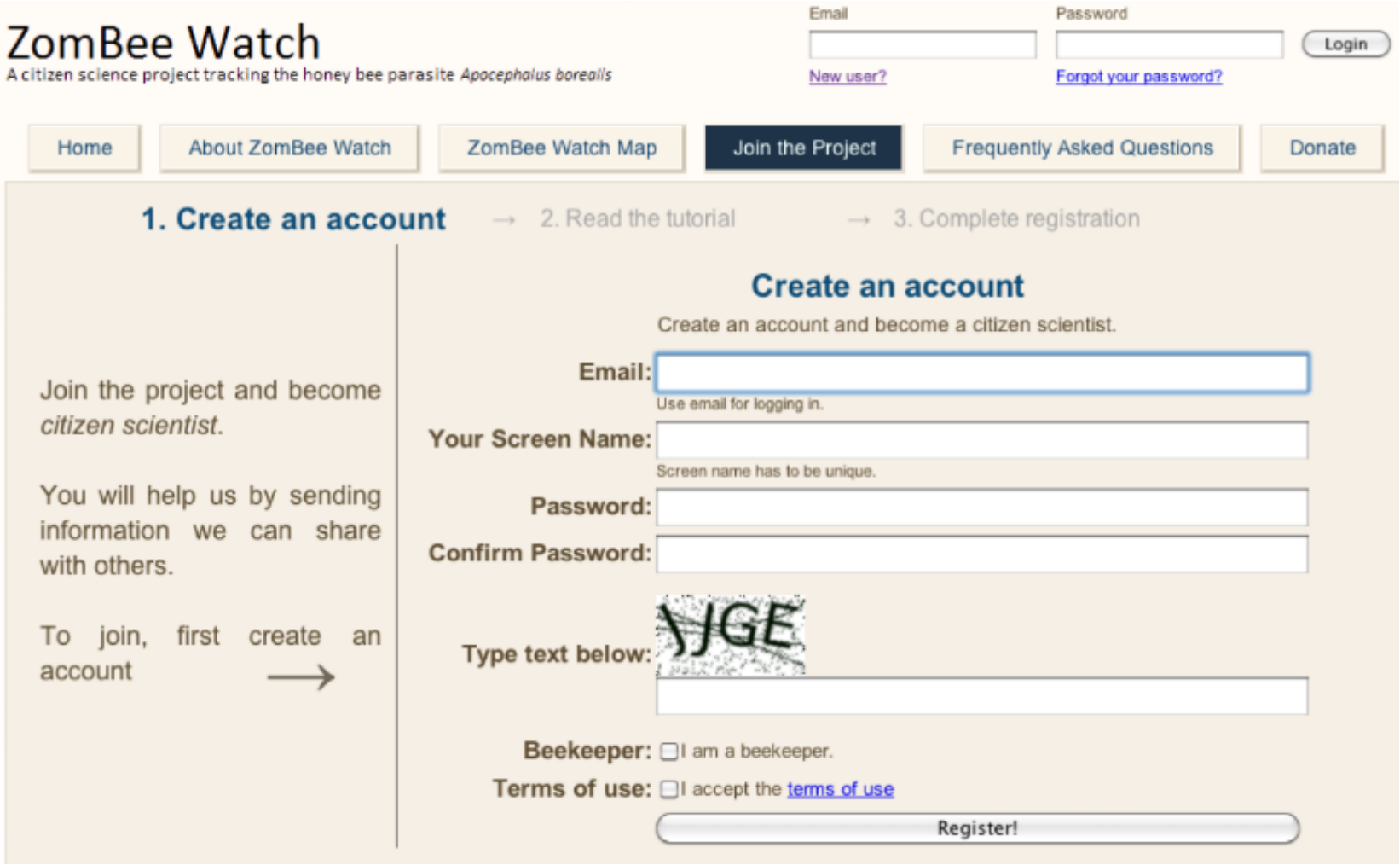
Pop up explain

Are we checking For uniqueness?

Align check buttons

I suggest: Create an account and become Citizen Scientist

Example after user feedback



Guiding UI Design

- Use GUI design principles and patterns for overall design –they are proven solutions. Some specific guidance is provided by some manufacturers (e.g. Apple, Microsoft).
- For implementation use available widget libraries as much as possible (tested for usability, portability and implementation: e.g. calendar, progress bar).
- Use UCD for overall SE process: test often and at every stage using typical users (not designers of GUI).

There is no single solution that is optimal!

Choose the right guidance

For example the iOS Human Interface Guidelines describes the guidelines and principles that help you design a superlative user interface and user experience for an iOS app.

Here are some links provided:

- iOS guidelines
- Android guidelines
- Material design guidelines
- Microsoft Fluent Design guidelines

High level Principles of UI Design

General principles to be followed for all UI designs.

- Derived from inherent cognitive/perceptual human properties.
- List detailed in alphabetical not priority order.
- All principles are important, BUT relative importance might depend on users and business objectives.
- Should be used as a check list to design/evaluate GUIs.

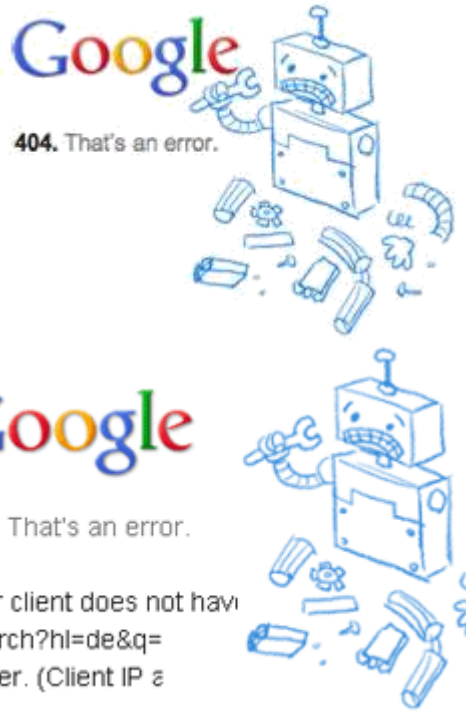
1. Aesthetically pleasing

- Contrast between elements
- Create groupings of related UI elements
- Align screen elements and groupings
- Provide uniform 2D/3D feel if possible
- Use colors and graphics appropriately
- Make sure your choice of colors and graphics renders on all devices, browsers, all screen and monitor color resolutions etc.)
- Examples of appealing designs

Often specified in “style guide”

2. Clarity –visual, conceptual, language

- (Distinguishable) visual elements
- (Clearly defined) functions
- (Intuitive) metaphors
- (Easy to understand) words and text



Error messages and text in general are often very confusing

Examples: Form filling, purchasing etc.

3. Compatibility

- With the user characteristics
- With user's mental concept of the application
- With task and job to be done
- With the product
- With rendering device (PC vs. handheld)

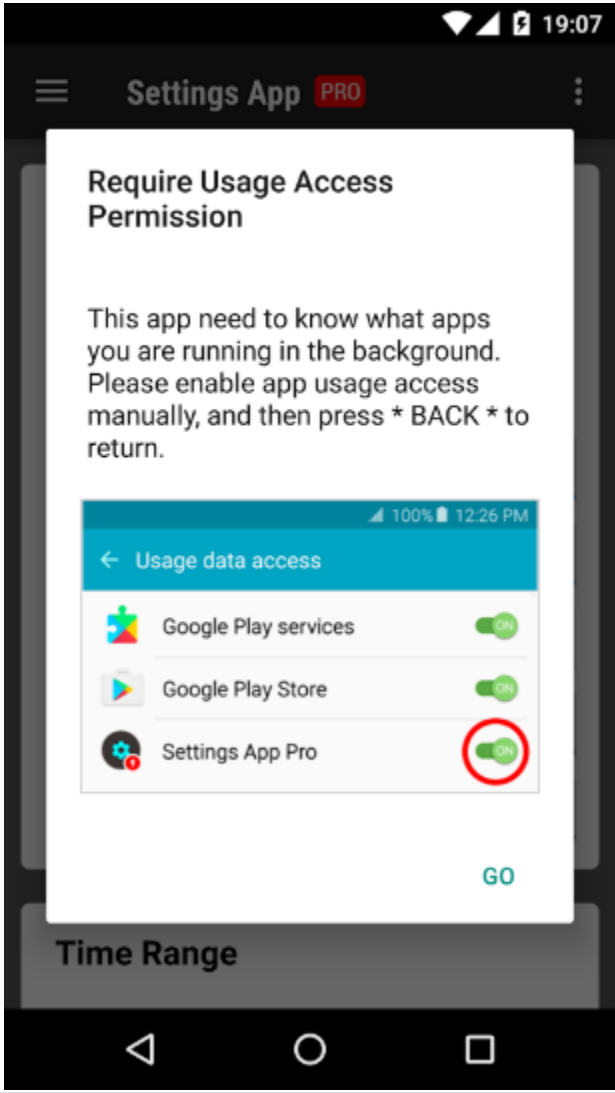
Down Dog:
Great Yoga
Anywhere

Do not assume all users **are alike** or that all are **like developers!**

User-centered design (UCD) helps here.

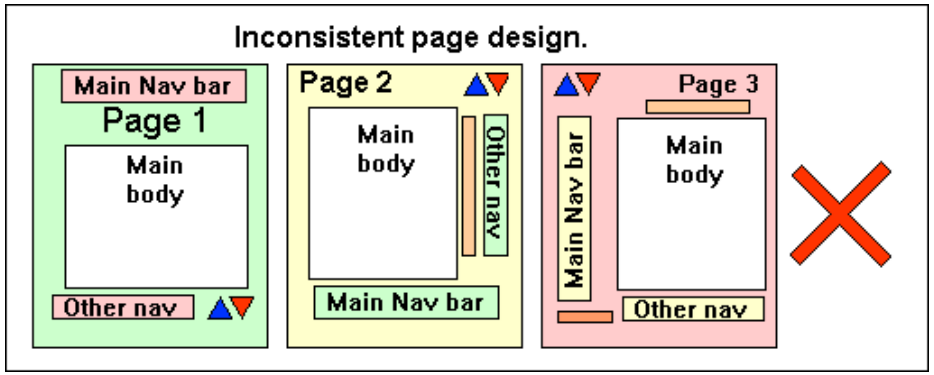
4. Configurability

- Allow personalization and configuration of settings
- Gives users sense of control



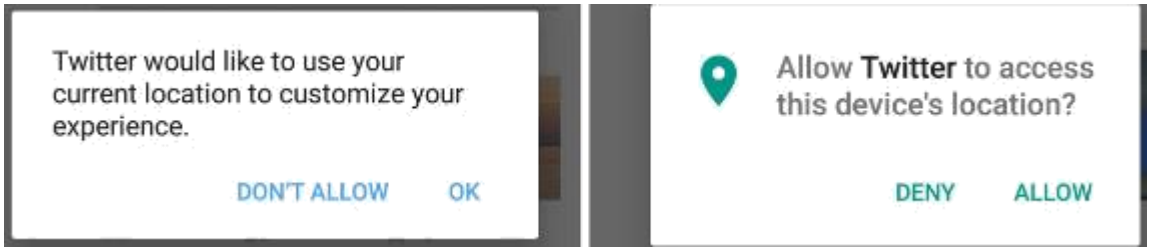
5. Consistency

- System should act, look and operate in consistent ways.
- Similar components should have similar look, similar uses and operate similarly.
- Same actions should always produce the same result.
- Function of elements should stay constant.
- Position of standard elements should not change.
- Use same font for similar level of abstraction.



6. Control

- User controls the interaction
- Do not let the system think it knows what is best for the user
- Do not perform "silent" actions
- Actions should be capable of interruption
- Maintain user-centered context
- Allow customization, always have defaults



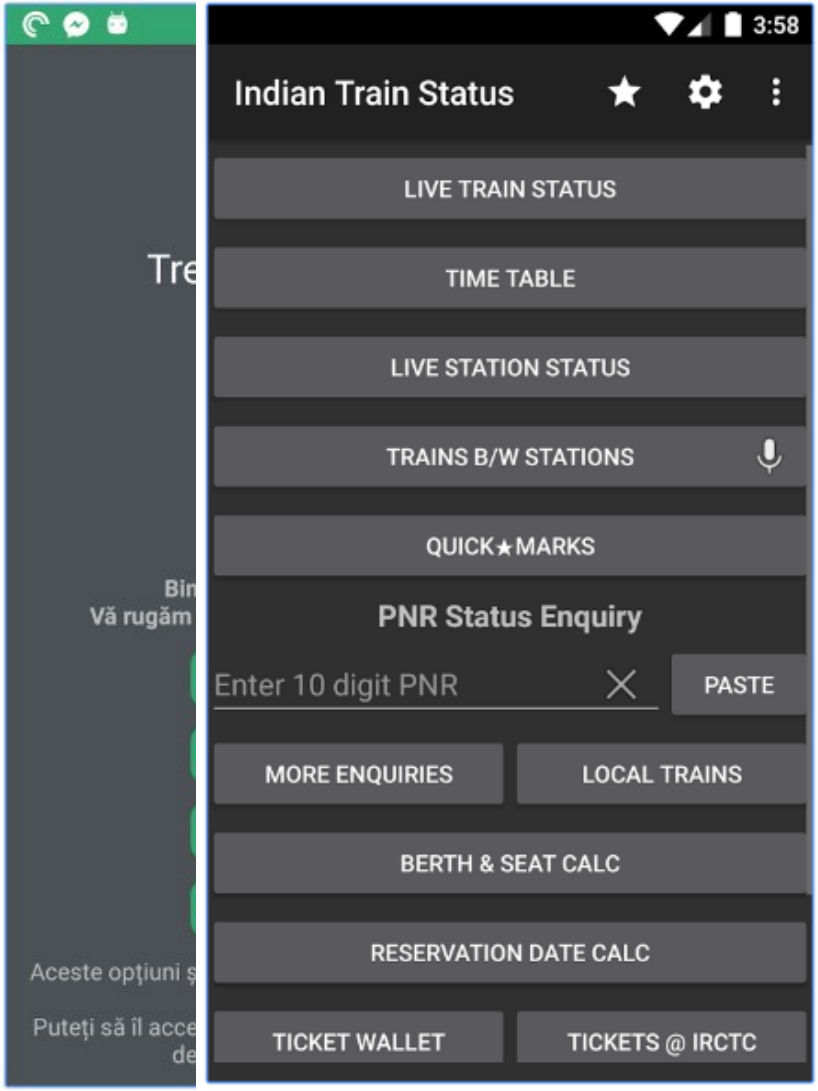
Bad example of implicit action after using the camera app

7. Context

- User should know the context all the time. This is especially challenging for web & mobile UIs
- User should never get lost
- Should always be able to easily go back, and go home
- For search: always know what you search for, where you applied the search for.
- For applications consisting of several ordered steps (shopping etc.): same as above, plus know what is the next step, easy to abort at any time.
- Desktop (multiple windows) vs. web/mobile (single screen) → context is harder to keep. Solutions?

8. Easy to understand

- What to look at?
- What to do?
- Why?
- When?
- How?
- Flow of actions should be easy to follow and learn, and it has to make sense



9. Efficiency

- Minimize eye and hand movements via proper screen layout.
- Anticipate user needs (but don't act on them)
- Minimize number of clicks and paths for each task

Fitts' law: Time-to-acquire-target = $a + b*\log_2(2D/W)$

where a, b are constants that depend on the input device

D is distance to center of object

W is width of the target along the axis of motion (tolerance)

Can result in huge savings, less user fatigue, increased productivity etc.

Fitt's Law (1954)

For designing interactive objects in UIs:

"As the size of an object increases the selection time goes down, and as the distance between the user's starting point and the object decreases so to does the time taken to make the selection.

Conversely, small objects, placed far away from the user's starting position take longest to select. Therefore, designers should aim to make objects as close to one another when they are used in the same sequence chain.

They should also try to make the interactive elements of the screen as large as is sensible with the amount of space available. Small objects spread apart take the longest time to select; avoid this combination of design features as much as possible."

Read more online.

10. Familiarity

- Employ familiar concepts and language
- Keep the UI natural as much as possible
- Use real world metaphors
- Follow adopted and de-facto standards (Windows, Mac, WWW)
- Follow common styles for each technical area (Database admin, systems management etc.)

→ "Good copy better than poor original"

11. Functionality

- Does the UI have all the right functions for major tasks?
- Are functions clear and non-overlapping?
- Major functions and advanced ones
- Effective functions for novices vs. power users
- New functions can be automated versions of common user tasks that could be done by other means

Examples: save and add contacts in Yahoo address book automatically; Apple Snap-back for new browsers;

12. Forgiveness & recovery

- Tolerate errors that are common (i.e. do not allow crashes)
- Prevent errors whenever possible
- Protect against catastrophic errors
- When error occurs, provide a constructive message
- Commands should be able to be abolished or reversed
- Work should never be lost in case of user or system error
- Warn the user in case the operation causes some dramatic change.

13. Predictability

- User should be able to anticipate natural progression in the task (via screen elements, cues)
- Important for applications where series of steps needs to be done (e.g. workflow, shopping)
- User should know all the time where he/she is, what is the next step, how to go back, how to abort etc.
- Context is important

14. Responsiveness

- Anything above 0.5 sec is not considered interactive and causes user frustration after long usage
- Response time should be consistent, logical and not vary considerably for similar functions
- Classic client systems are often faster than online ones but users seem to expect delays.
- Tricks can be used to avoid user problems for less responsive systems:
  - Show user something, anything but blank screen/frozen UI
  - Show some visual clues that the system is working (e.g. progress bar, progressively rendered images…)
  - Avoid periods of blank screens
- Generally, provide acknowledgement of user actions
  - Visual
  - Textual
  - Auditory

15. Simplicity

- Create as simple GUIs as possible. Simplicity sells.
- Use progressive disclosure
  1. Show common and necessary functions first
  2. Prominently feature most important functions
  3. Hide advanced functions
- Provide defaults
- Reduce screen complexity
- Make common actions simple to use, even at the expense of less common actions being harder to use

Examples: Google.com, Dropbox, WeTransfer, Evernote

Progressive Disclosure for Mobile Apps

- Pattern which refers to managing information complexity.
- Show only the information necessary at that point in the interaction
- Display more advanced functionalities of the app interface as the user interacts with it.
- Benefits:

  Important content is prioritized;

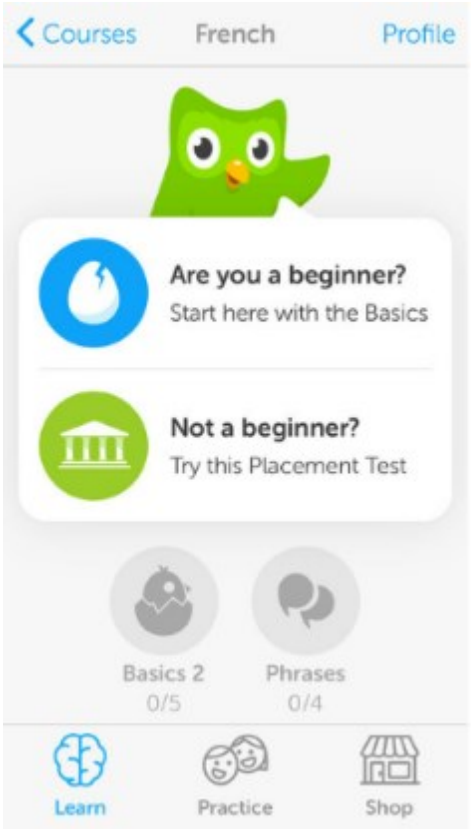
  Clean, simpler, more productive interfaces;

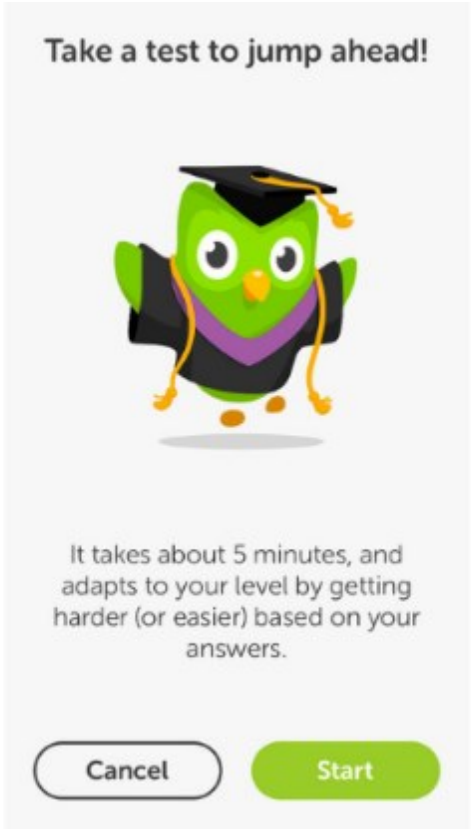
  Error rate is reduced

"show less, provide more": get the right split between initial and secondary features.

App onboarding example

- Duolingo





16. Transparency

- Permit the user to focus on the task.
- Mechanics of interface should not get in the way.
- "Workings" inside the app should not be visible to the user (e.g. server synch, GPS satellite selection, rescanning emails).
- No silent action (e.g. doing something important without users being informed).

17. Trade-offs

Bottom-line is that you will always need to design trade-offs

1. Final GUI will be based on trade-offs, often balancing conflicting design points.
2. User requirements always come above technical, development or engineering requirements.
3. Remember 2nd rule.

Ranking high-level design principles?

1. Aesthetically pleasing
2. Clarity
3. Compatibility
4. Easy to understand
5. Configurability
6. Consistency
7. Control
8. Context
9. Efficiency
10. Familiarity
11. Functionality
12. Forgiveness, recovery
13. Predictability
14. Responsiveness
15. Simplicity
16. Transparency
17. Design trade-offs

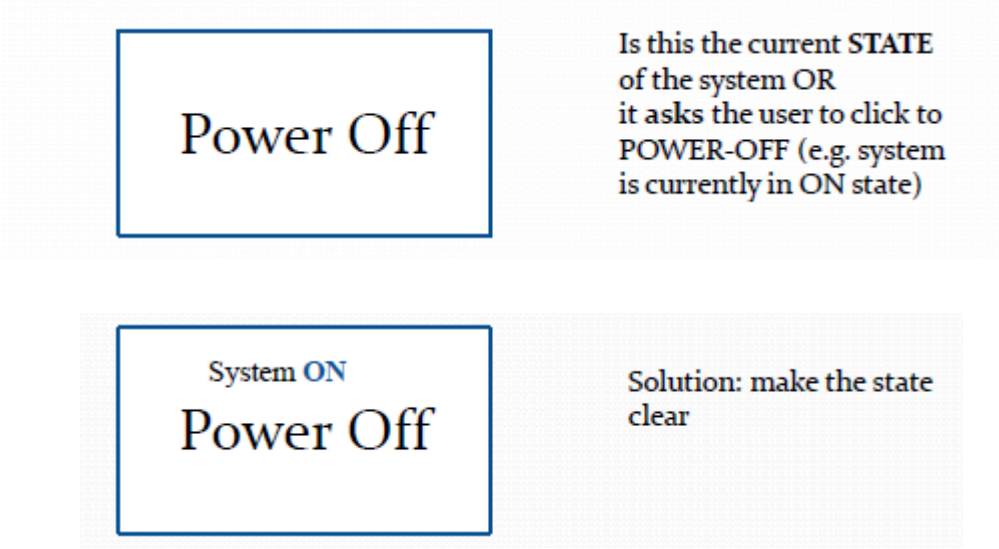
Dragutin's top

Dragutin Petkovic, Prof. @SFSU, "Usability, GUI Design and Principles"

- Simplicity
- Functionality
- Context
- Consistency
- Responsiveness

Some bad examples

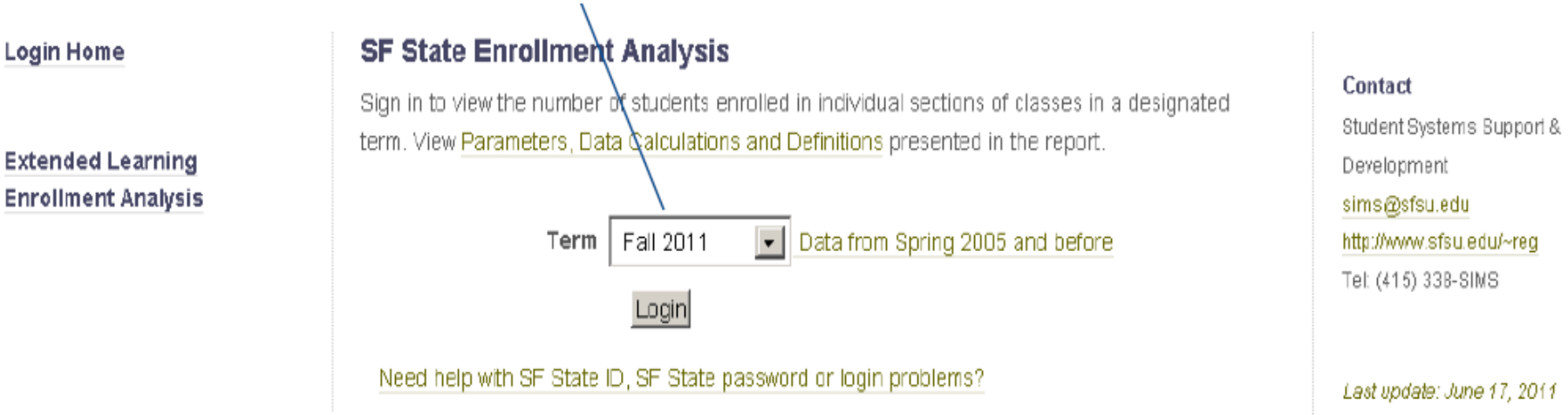
Confusion between state and action



Some bad examples

Wrong* flow of actions: first select from drop-down list, then login

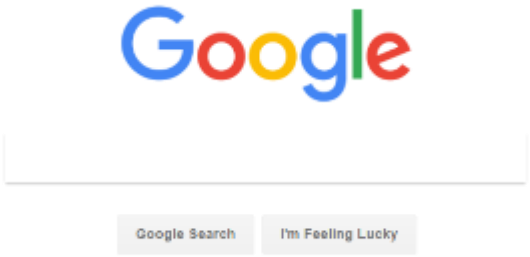
- If login fails you have to reselect the option
- If the option is wrong you have you sign out and reselect



** counter-intuitive, less efficient*

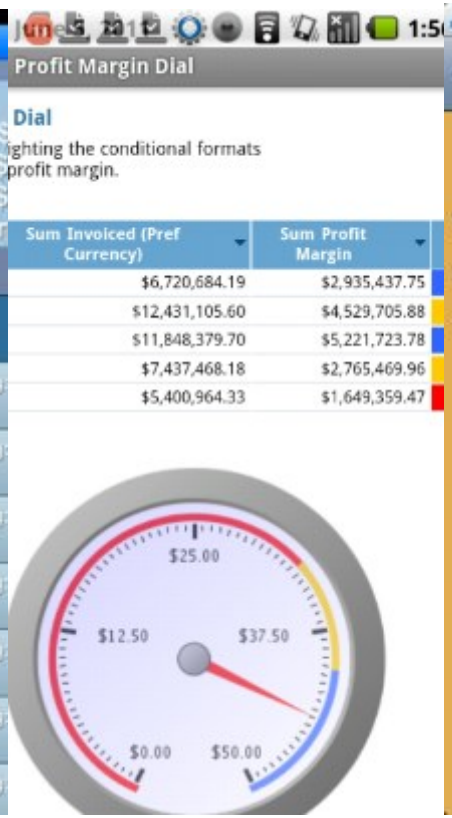
Some bad examples

Simplicity first



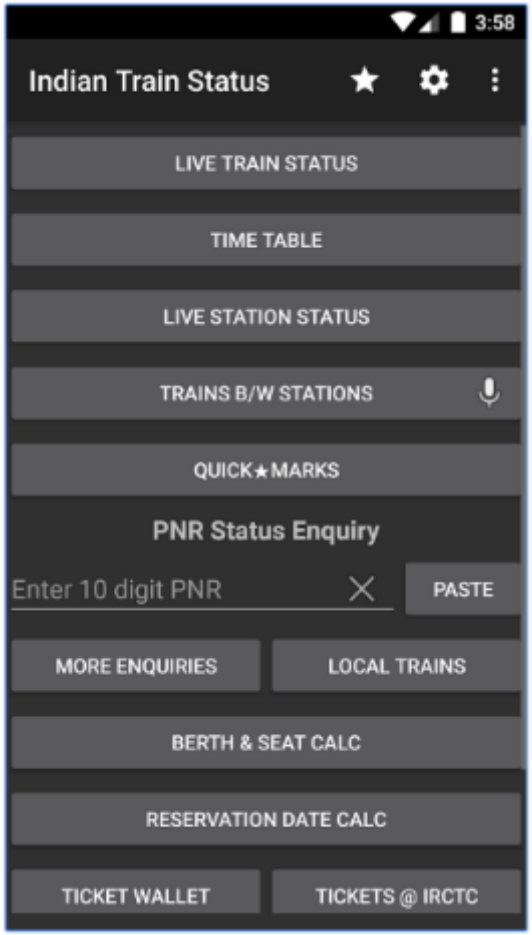
Some bad examples

Aesthetics



Some bad examples

Information concentration and consistency



Human Characteristics related to GUI: Perception

Awareness and understanding of elements and objects through sight, sound (etc.) influenced mostly by experience

- Proximity: objects relate to each other
- Similarity: similar objects should relate
- Unity: objects forming closed shape perceived as groups
- Balance and symmetry: vertical and horizontal
- Subjective perception of motion, image, video, audio quality
- Context: color backgrounds influence color perception etc.
- Simple designs are easier to remember

Human Characteristics related to GUI: Memory

1. Sensory memory: buffer that stores sensory input. Forgotten easily.
2. Short-term memory: holds limited amount of data for several minutes (holds up to seven concepts)
3. Long term memory: large storage capacity, but can be unreliable

- Recall: getting information from long term memory (hard)
- Recognition: selecting from multiple choices or recognizing a pattern (easier)

Human Characteristics related to GUI: Vision

- Visual acuity: capacity to resolve details
- Foveal and peripheral vision
- Relative acuity is halved at 2.5 degrees from the center: influences character spacing on the screen etc.

1. Design patterns so as not to cause jitter and visual distraction
2. Busy periphery can distract main foveal vision
3. Acuity drops with age: font type, contrast and size to be adjusted for older users

Human Characteristics related to GUI: Interactivity

Response time versus user reaction

<0.1 sec → Instantaneous, no interruptions

<1.0 sec → Delay noticed, but no break in thought stream

>10 sec → Interruption, user switches to another task

Human Characteristics related to GUI: People

We are all different: motor and perception abilities, skills, education, learning abilities, age etc.

- How to design for wide population?
  1. Lowest common denominator?
  2. Provide varying levels of GUI that can be adjusted to various people
  3. Allow people to customize

Human Characteristics related to GUI: People

Knowledge and Experience:

- Computer Literacy – digital know-how on using a smartphone
- System experience – being accustomed to a specific platform/OS
- Application experience – knowing what a (popular) app does
- Task experience – having done similar tasks before (e.g. map search)
- Education
- Reading level
- Typing skill
- Native language or Culture

Types of users

Novice users

→Depend on system's features to assist (help etc.)

→Restricted vocabularies

→Simple tasks

→Need learning time

→Sometimes high "startup" curve that may lead to dropping the app

Novice users are never "stupid". Systems must be designed for them, as they can be your key market segment in early stages.

Types of users

Expert users

→Leverage experience

→Use recall (fast) rather than recognition

→Fast

→Need less help and feedback

→Like to reduce keystrokes, have shortcuts etc.

Allow experts to leverage their expertise. Systems also need to be designed to accommodate them

GUI Quality Assurance

- QA finds bugs in terms of specs (different from usability testing)
- QA is done by traditional QA departments –passing QA does not mean the app is usable
- GUI QA is time intensive: try all buttons, all functions, test all paths of usage, test for all browsers, resolutions and use cases
- QA Automation tools for GUI QA exist
  - Robotium
  - UI Automator
  - Appium

QA Automation Tools

Robotium

- Library written in Java
- Free, all versions, “selenium for Android”
- Overhead in writing tests
- Unsuitable for interaction with system software
- No unlock, no record/play, no screenshots

QA Automation Tools

UI Automator

- Developed by Google for Android 4.1+
- Interacts with all kinds of (system) apps
- Can utilize external buttons of a device
- Generates informative and detailed reports
- One of the best tools available

QA Automation Tools

Appium (Studio)

- Free framework working with Java, C#, Ruby
- Can control Chrome and Safari browsers
- Poor, insufficient reports

QA Summary

About 80% of bugs reproduce on all platforms.

→Perform mobile testing on a popular platform to discover up to 80% of defects.

→The other 20% will be found on all the other platforms.

→It is better to test software products thoroughly on fewer platforms than hastily on numerous.