

**CSE 403: Software Engineering, Fall 2016**

[courses.cs.washington.edu/courses/cse403/16au/](https://courses.cs.washington.edu/courses/cse403/16au/)

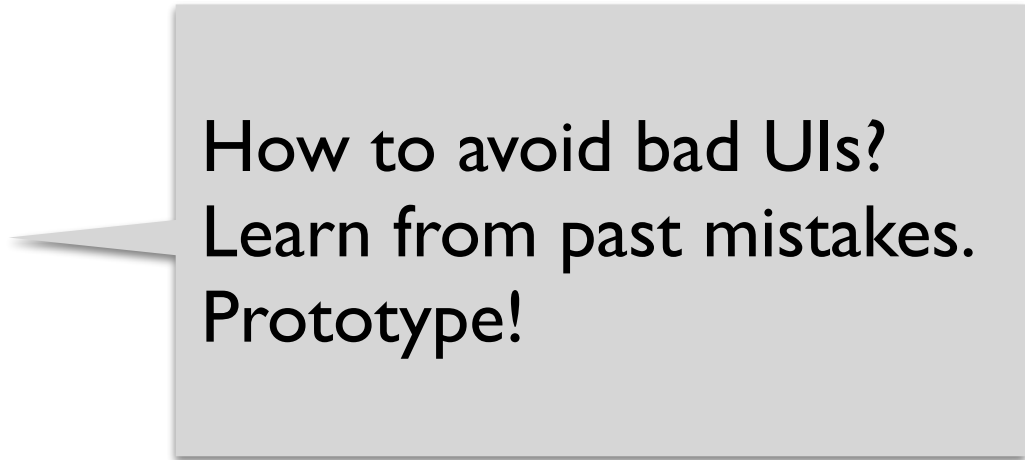
# **User Interfaces**

**Emina Torlak**

[emina@cs.washington.edu](mailto:emina@cs.washington.edu)

# Outline

- Usability
- Prototyping
- UI Design
  - Elements
  - Examples



How to avoid bad UIs?  
Learn from past mistakes.  
Prototype!

**usability, why it matters, and how to achieve it**

# What is usability?

The effectiveness with which users can accomplish tasks in a (software) system, as measured by

- **Learnability:** is it easy to learn?
- **Efficiency:** once learned, is it fast to use?
- **Safety:** are errors few and recoverable?

# Relative importance of usability dimensions

- Depends on the user
  - Novices need learnability.
  - Experts need efficiency.
  - But no user is uniformly a novice or an expert.
- Depends on the task
  - Missile launchers need safety.
  - Subway turnstiles need efficiency.



# Usability matters: the cost of getting it wrong

50% of all “malfunctioning” electronic devices returned to stores are in full working order, but users can't figure out how to operate them.

[Elke den Ouden, 2006]

# Usability matters: the cost of getting it wrong

50% of all “malfunctioning” electronic devices returned to stores are in full working order, but users can't figure out how to operate them.

[Elke den Ouden, 2006]



**Three Mile Island:** nuclear reactor meltdown caused by an ambiguous user interface


# A good user interface is hard to design ...

- You are not the user
  - Most software engineering is about communicating with other programmers.
  - UI is about communicating with users.
- Users are always right ...
  - Consistent problems are the system's fault.
- Except when they aren't
  - Users don't always know what they want.



# A good user interface is hard to design ...

- You are not the user
  - Most software engineering is about communicating with other programmers.
  - UI is about communicating with users.
- Users are always right ...
  - Consistent problems are the system's fault.
- Except when they aren't
  - Users don't always know what they want.



UI accounts for 50% of

- design, implementation, and maintenance time
- lines of code

# Achieving usability: best practices

- User testing and field studies
- Evaluations and reviews by UI experts
- Prototyping
  - Cheap, throw-away implementations
  - Low-fidelity: paper prototypes
  - Medium-fidelity: code prototypes

**Key to success:** good UI focuses on the user, not the developer or the system.

# **prototyping: what, why, when, and how**

# **What is prototyping? Why do it?**

# What is prototyping? Why do it?

- **Prototyping:** creating a scaled-down or incomplete version of a system to demonstrate or test its aspects.

# What is prototyping? Why do it?

- **Prototyping:** creating a scaled-down or incomplete version of a system to demonstrate or test its aspects.
- **Benefits of prototyping:**
  - aids UI design
  - help discover requirements
  - help discover test cases and provide a basis for testing
  - allows interaction with user to ensure satisfaction
  - team-building

# Some prototyping methods

- Code prototyping
  - implement a "quick" / incomplete version of a UI
- Prototyping with UI builders (e.g., Visual Studio)
  - draw a GUI by dragging/dropping UI controls on screen
- Paper prototyping
  - a paper version of a UI

# Why paper prototyping?

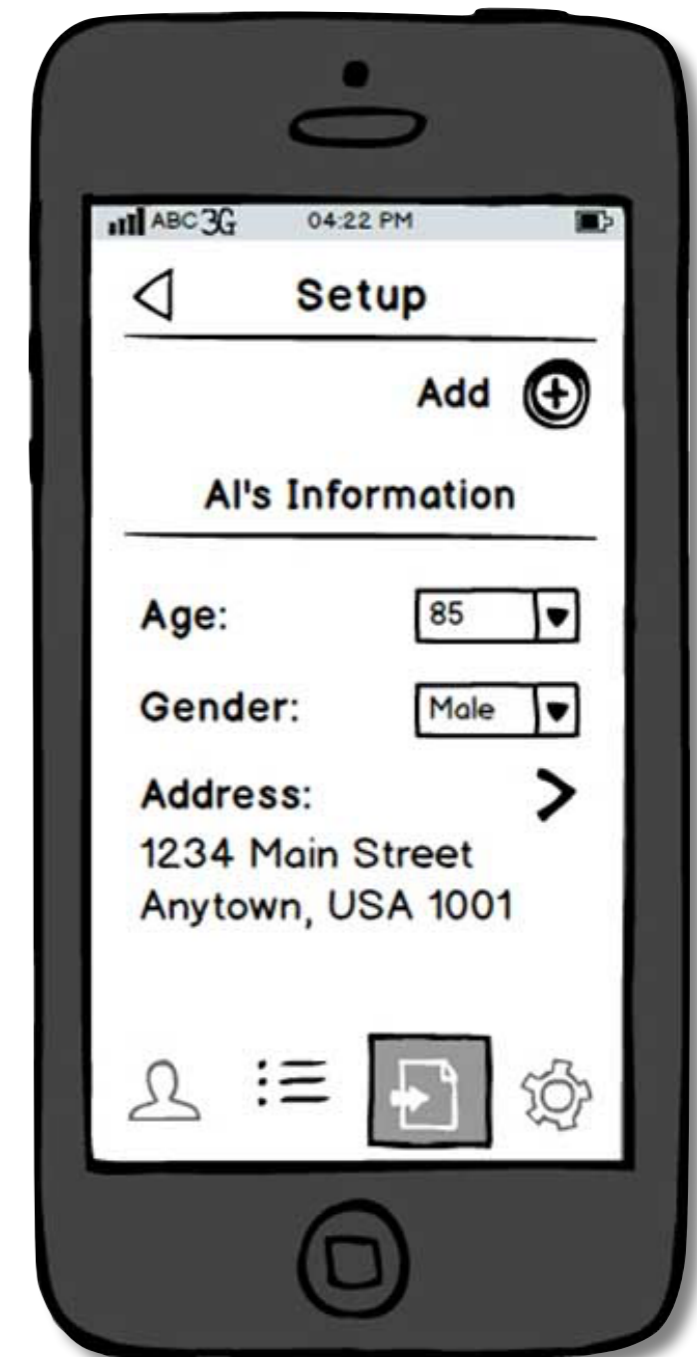
- Much faster to create and change than code
- More visual bandwidth (can see more at once)
- More conducive to working in teams
- Can be done by non-technical people
- Feels less permanent or final





# When to do (paper) prototyping?

Q. Requirements are the **what** and design is the **how**. Which is paper prototyping?



# When to do (paper) prototyping?

Q. Requirements are the **what** and design is the **how**. Which is paper prototyping?

## A. Prototyping

- helps uncover requirements and upcoming design issues
- during or after requirements but before design
- shows us **what** is in the UI, but also shows us details of **how** the user can achieve goals in the UI

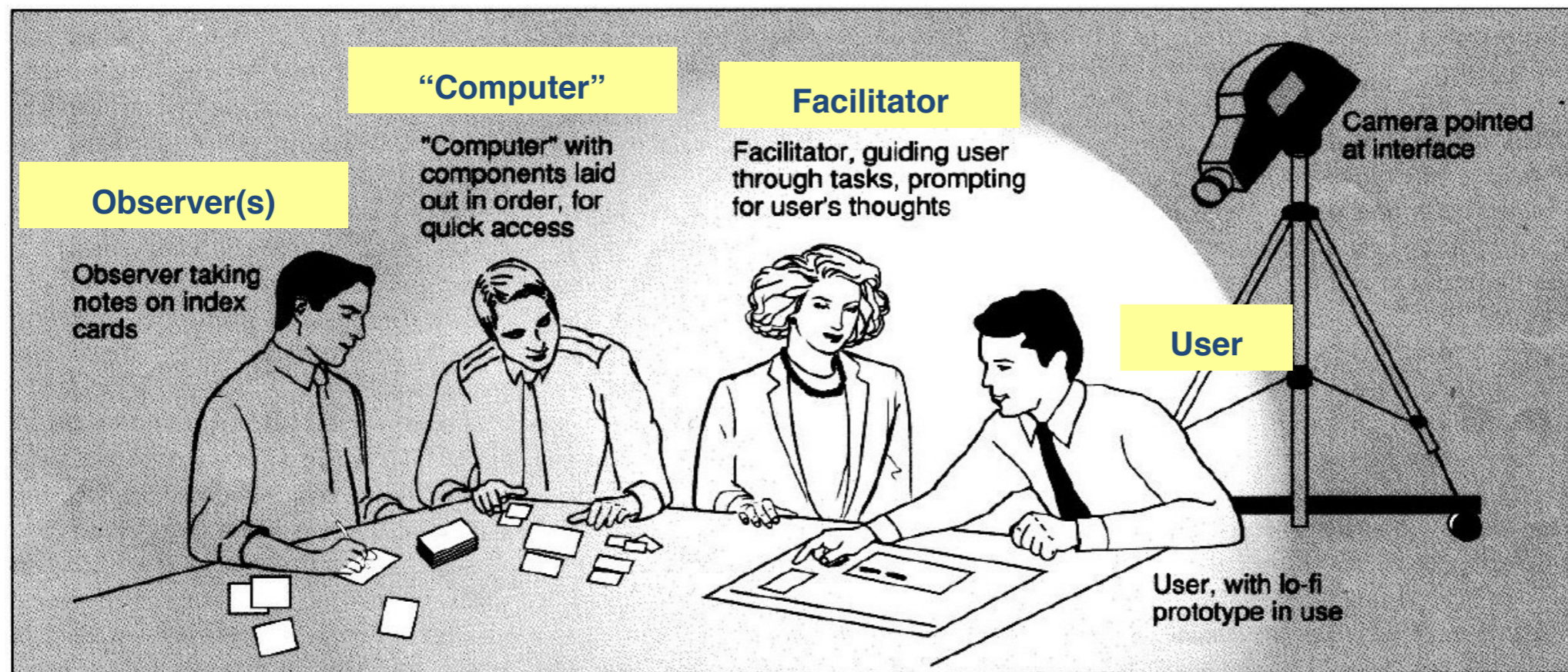


# How to build a paper prototype



# Paper prototype usability testing

- User gets tasks to perform on a paper prototype.
- Facilitator guides the user through tasks, prompting for feedback.
- Developer plays the computer.



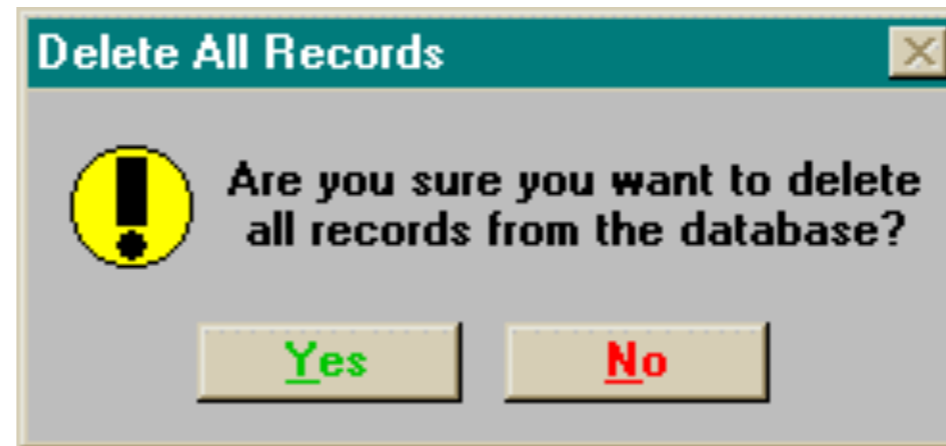
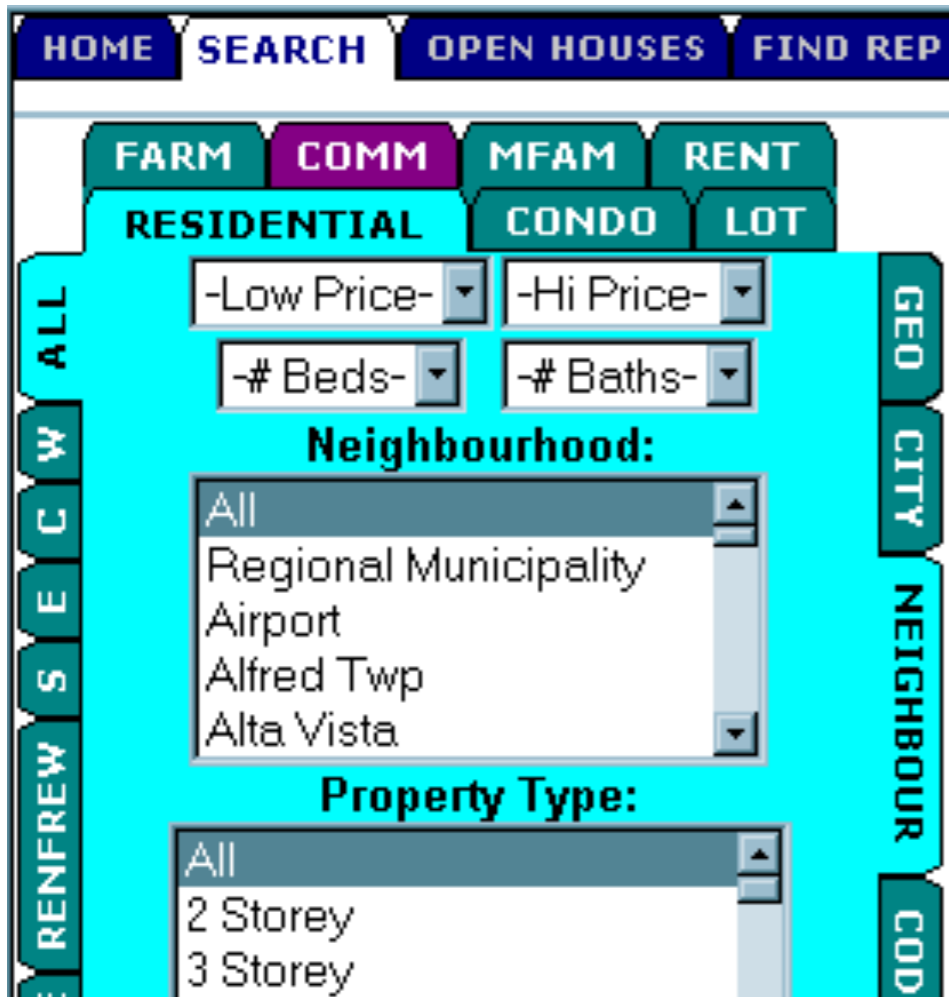
**UI design: elements and examples**

# Golden rules of UI design

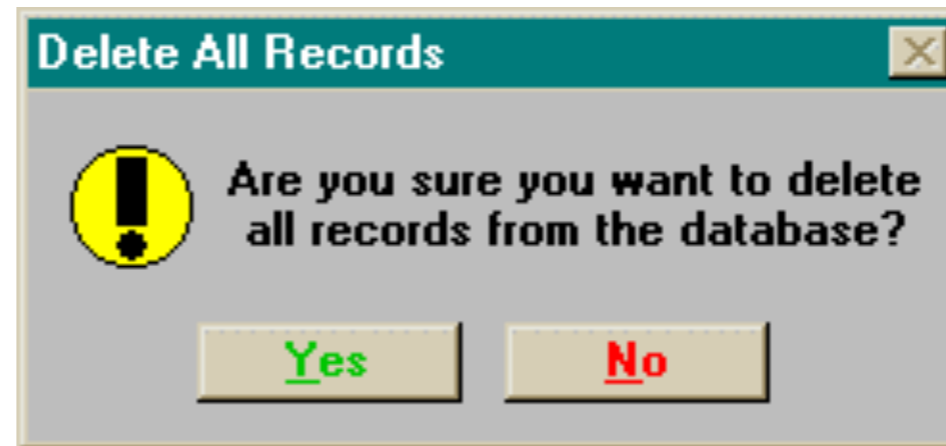
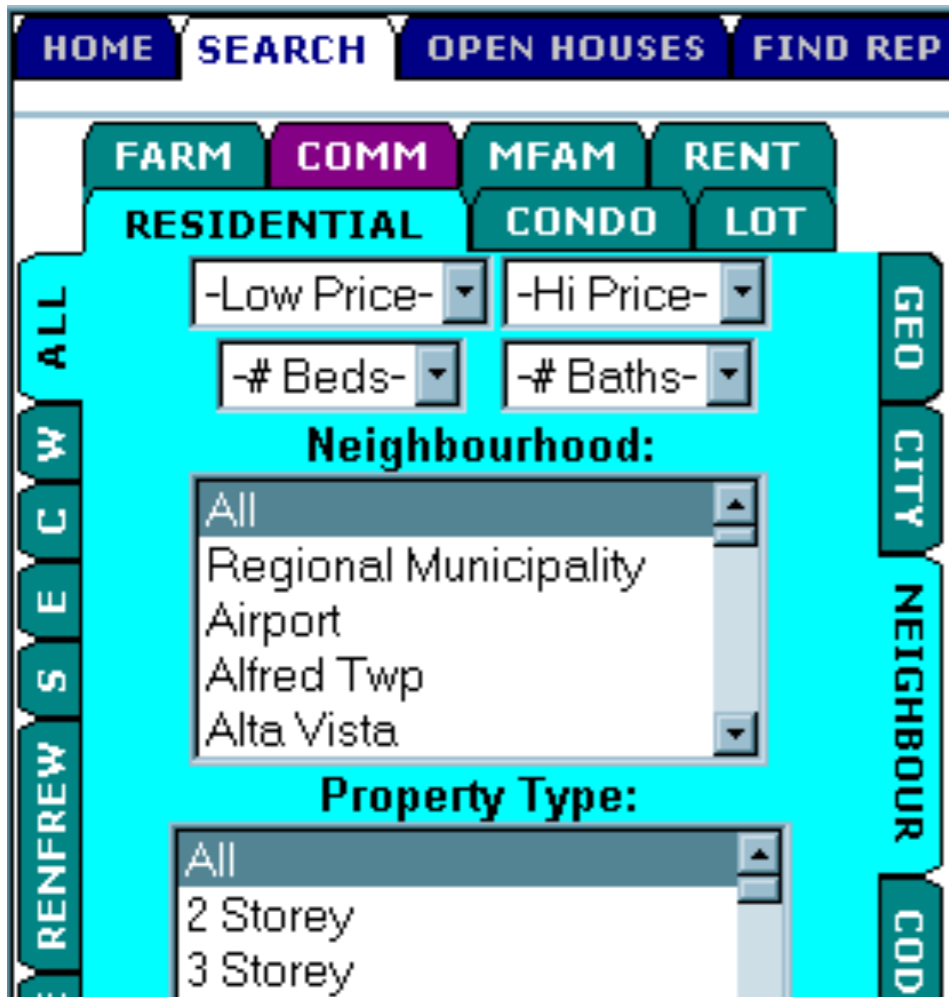
1. Strive for consistency.
2. Give shortcuts to the user.
3. Offer informative feedback.
4. Make each interaction with the user yield a result.
5. Offer simple error handling.
6. Permit easy undo of actions.
7. Let the user be in control.
8. Reduce short-term memory load on the user.

From *Designing the User Interface*, by Ben Schneiderman, noted HCI and UI design expert.

# UI Hall of Shame



# UI Hall of Shame





# UI Hall of Fame

Google

Google Search

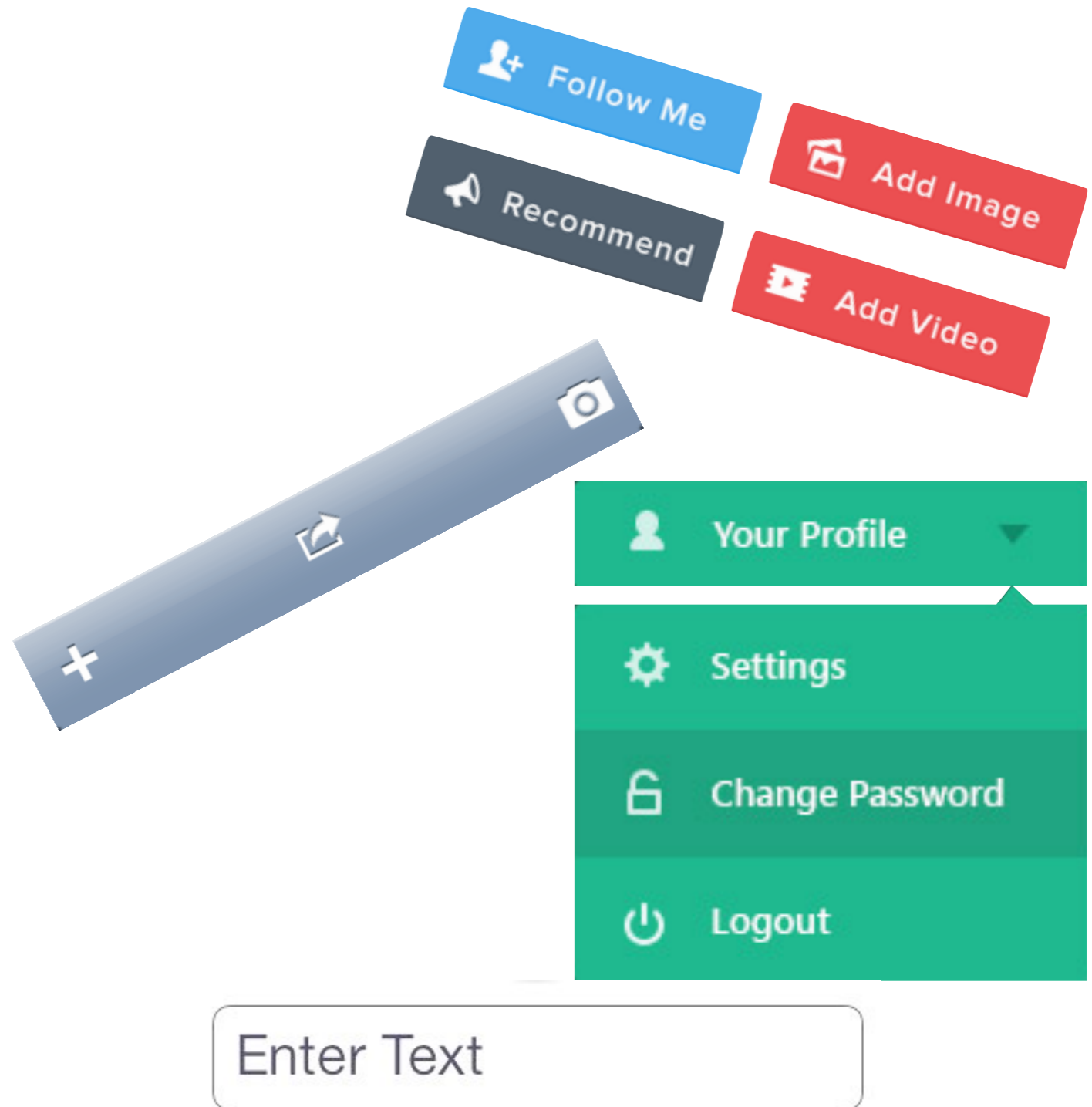
I'm Feeling Lucky



# UI design: components

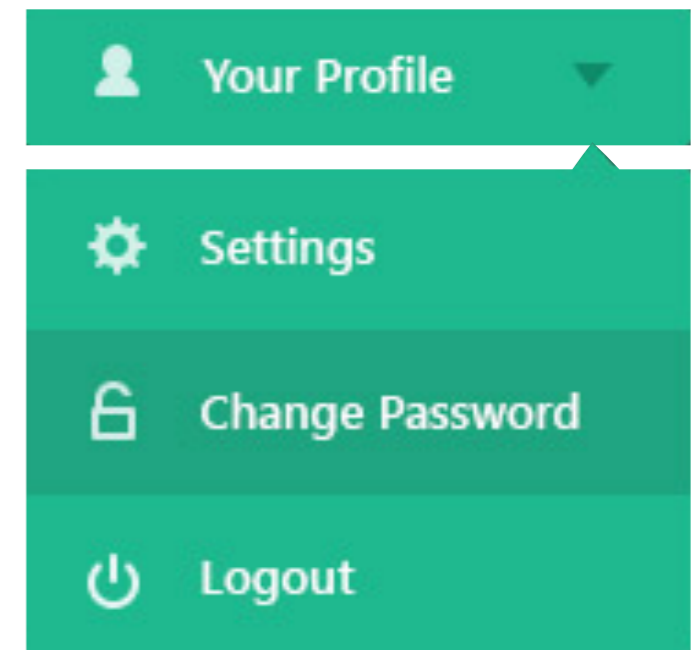
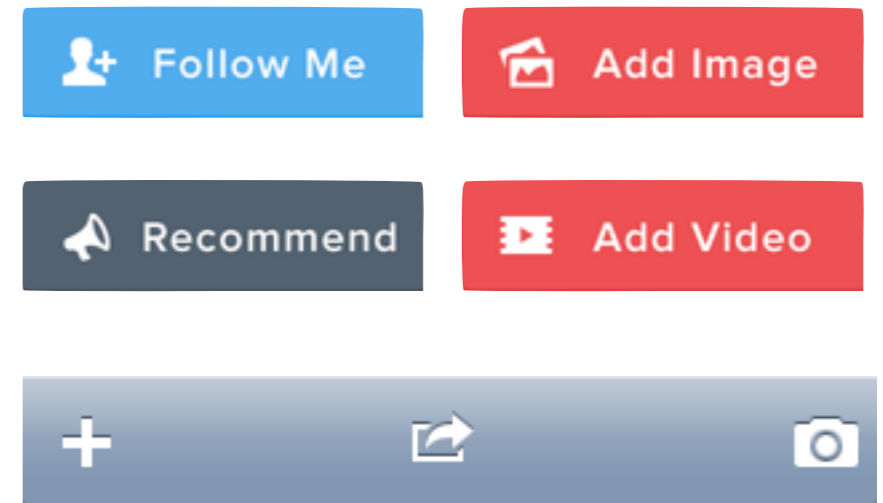
## When to use

- a button?
- a check box?
- a radio button?
- a text field?
- a list?
- a combo box?
- a menu?
- a dialog box?
- ...?



# UI design: buttons, toolbars, menus

- Use **buttons** for single independent actions that are relevant to the current screen.
  - Use button text with verb phrases such as "Save" or "Cancel", not generic: "OK", "Yes", "No"
  - Use Mnemonics or Accelerators (Ctrl-S)
- Use **toolbars** for common actions.
- Use **menus** for infrequent actions that may be applicable to many or all screens.
  - Users hate menus! Try not to rely too much on menus. Provide another way to access the same functionality (toolbar, hotkey, etc.)



# UI design: check boxes and radio buttons

- Use **check boxes** for independent on/off switches.
- Use **radio buttons** for related choices, when only one choice can be activated at a time.

1. Do you have pets?

Yes

No

2. Which pets do you have?

Dog

Cat

Lizard

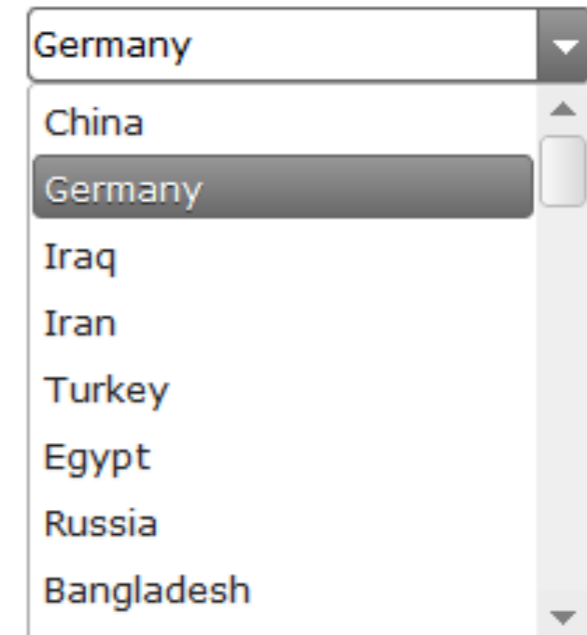
Bird

# UI design: text fields, lists, combo boxes, sliders

- Use **text fields** (usually with a label) when the user may type in anything they want.
- Use **lists** when there are many fixed choices (too many for radio buttons); all choices visible on screen at once.
- Use **combo boxes** when there are many fixed choices; don't take up screen real estate by showing them all at once.
- Use a **slider** or **spinner** for a numeric value.

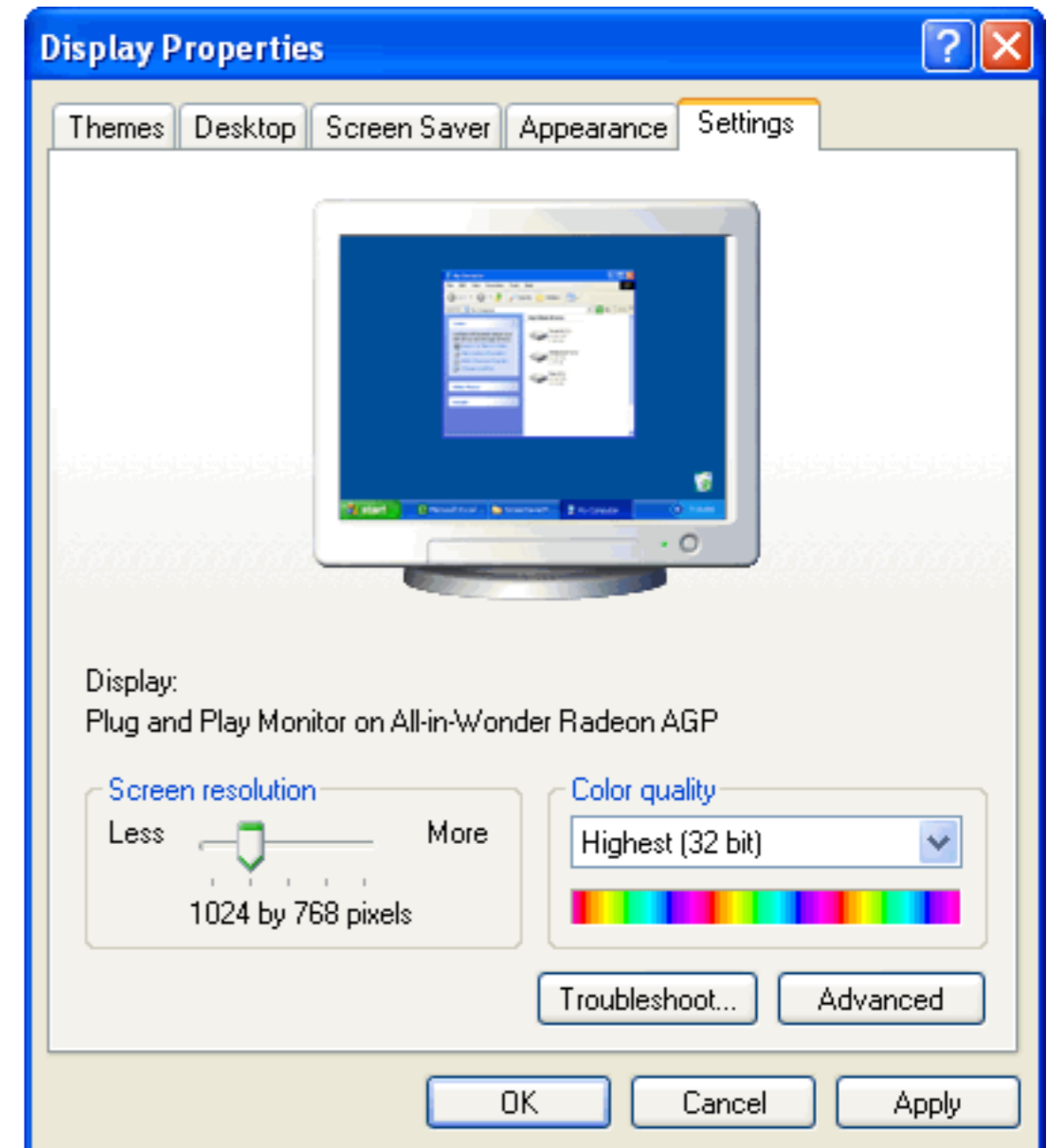


Enter Text




# UI design: dialogs and panes

- Use a **tabbed pane** when there are many screens that the user may want to switch between at any moment
- Use **dialog boxes** or **option panes** to present temporary screens or options
  - “modal” dialog box prevents any other action




# UI design: an example

**LibSys Search**

Choose collection  

Word or phrase

Search by  

Adjacent words  YES  NO

Good UI dialog?  
Assume there are  
20 collections and 3  
ways to search.

# Summary

- Usability
  - Learnability, efficiency, safety
- Prototyping
  - Paper and code prototypes
- UI Design
  - Know which elements to use
  - Strive for simplicity