

# **Building User Interfaces**

# **Design Paradigms, Patterns, & Languages**

## **Professor Yuhang Zhao**

# Logistics

- Assignment React 2 Alpha: Friday -> Monday (Oct 25)
- Assignment React 2 Beta: optional (3 bonus credits), Saturday (Oct 30)
- Midterm: next week
- Prior quizzes

# What we will learn today?

- Design paradigms
- Design patterns
- Design languages

# Recap: What is interaction design?

# Interaction Design

**Definition:** Defining behaviors for a system that engages the full spectrum of its user's perception, cognition, and movements.

Differs from visual design in its closer and more complex relationship to user behavior and context.

*Example:* visual designers do not think about navigation models!

# Five Dimensions of Interaction Design<sup>1</sup>

1. 1D: Words
2. 2D: Visual representations
3. 3D: Physical objects and space
4. 4D: Time
5. 5D: Behavior

We talked about *visual design* and *navigation*, but how do we address all these dimensions?

## 5 DIMENSIONS OF INTERACTION DESIGN

The image shows a wireframe of a web application form. At the top, it says "Application Form". Below that is a text input field with placeholder text: "Please enter the description below:" followed by a long string of Latin placeholder text. At the bottom right of the form is a "SUBMIT" button.



INTERACTION DESIGN  
FOUNDATION

INTERACTION-DESIGN.ORG

<sup>1</sup> [Interaction Design Foundation](#)

# Interaction Design Paradigms

# What is a Design Paradigm?

**Definition:** An archetypal solution or an approach to solving design problems.

# Historical Interaction Design Paradigms <sup>9</sup>

1. Implementation-centric
2. Metaphoric
3. Idiomatic

<sup>9</sup> Cooper et al., 2014, About Face

## Implementation-centric Design

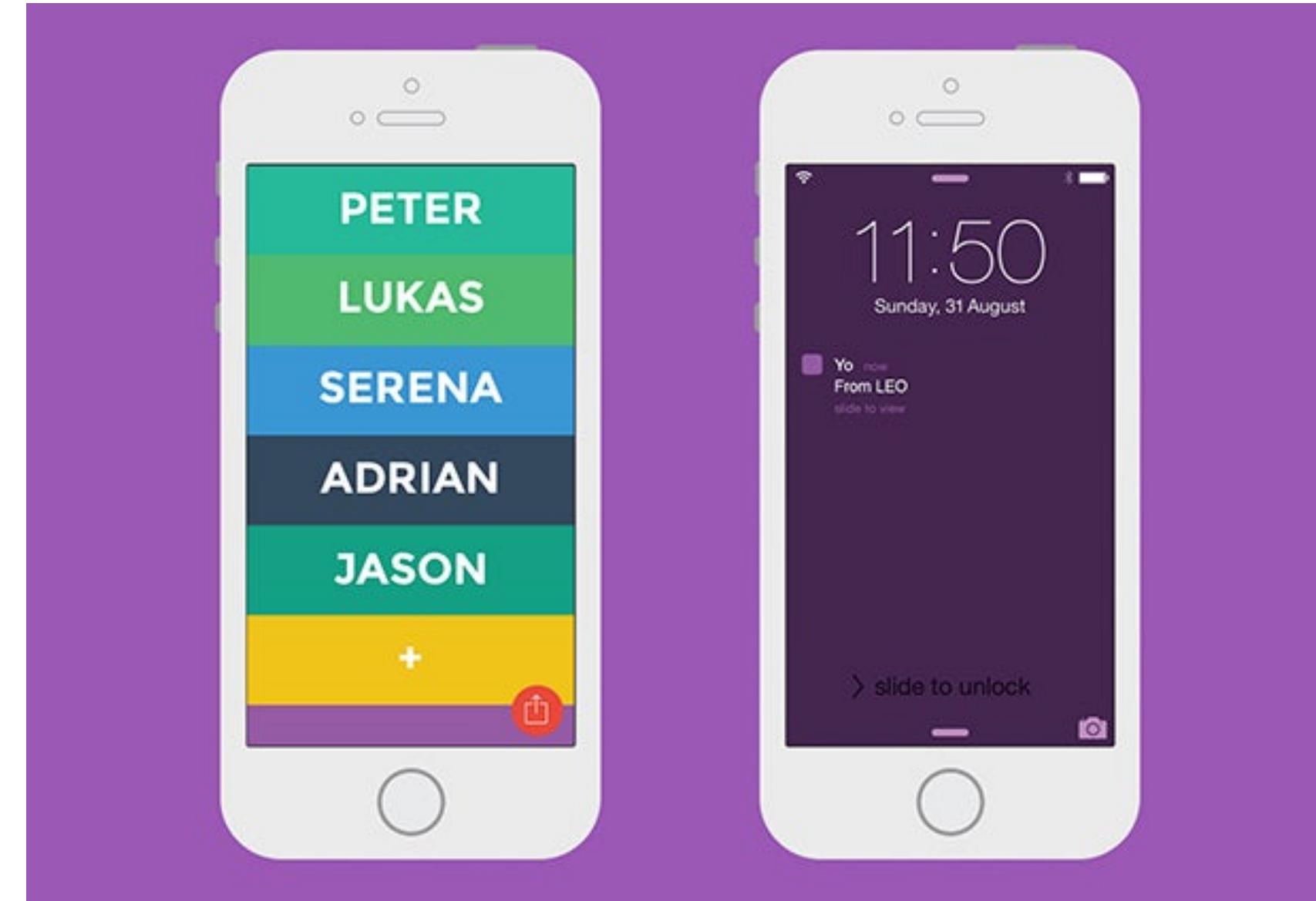
**Definition:** Interaction design maps directly to how system functions are implemented.

# Source<sup>2</sup> <sup>3</sup>



<sup>2</sup> Pintrest

<sup>3</sup> Entrepreneur Magazine



## **Pros & Cons of Implementation-centric Design**

### **Pros:**

1. Very easy to build, easy to debug, easy to troubleshoot

### **Cons:**

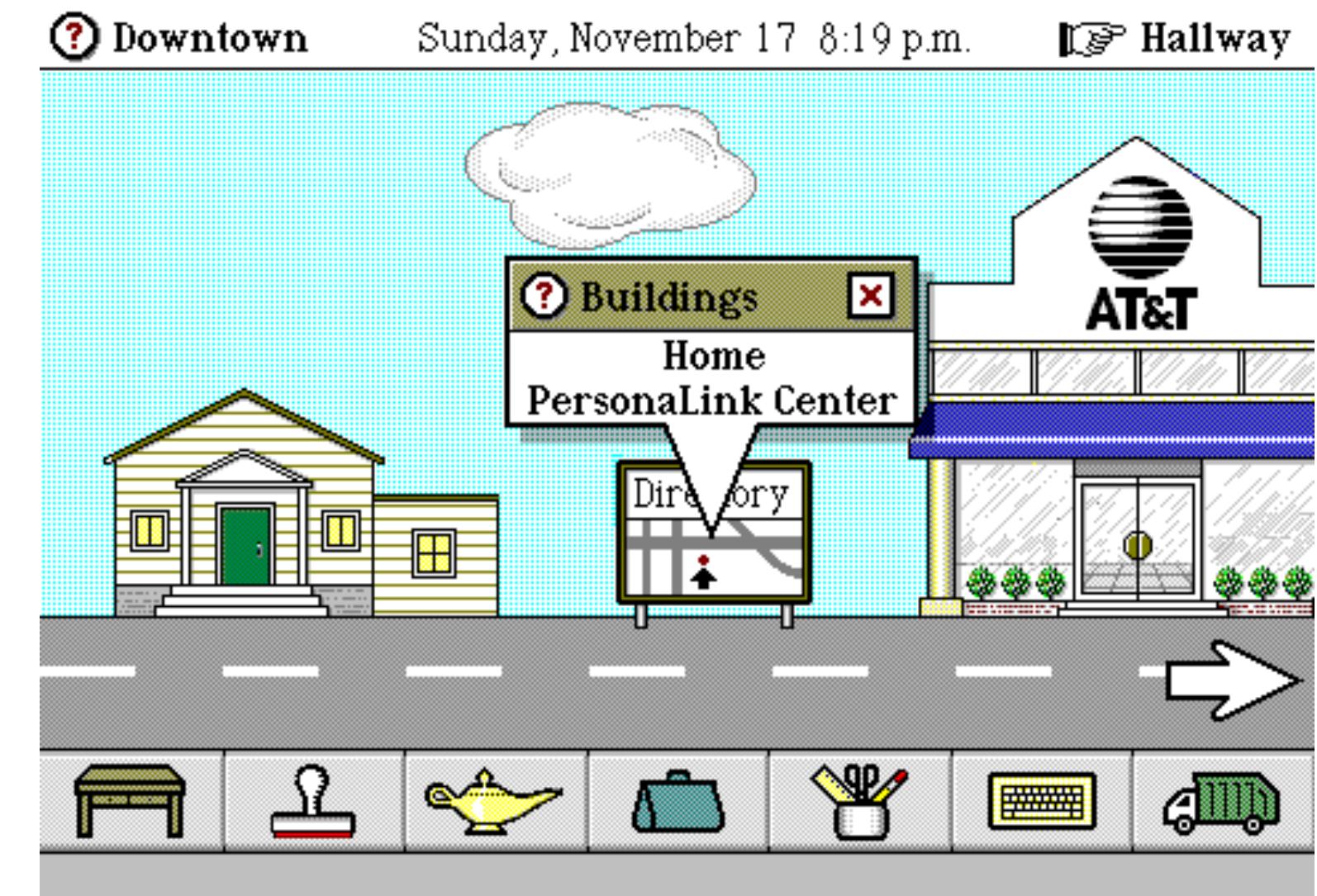
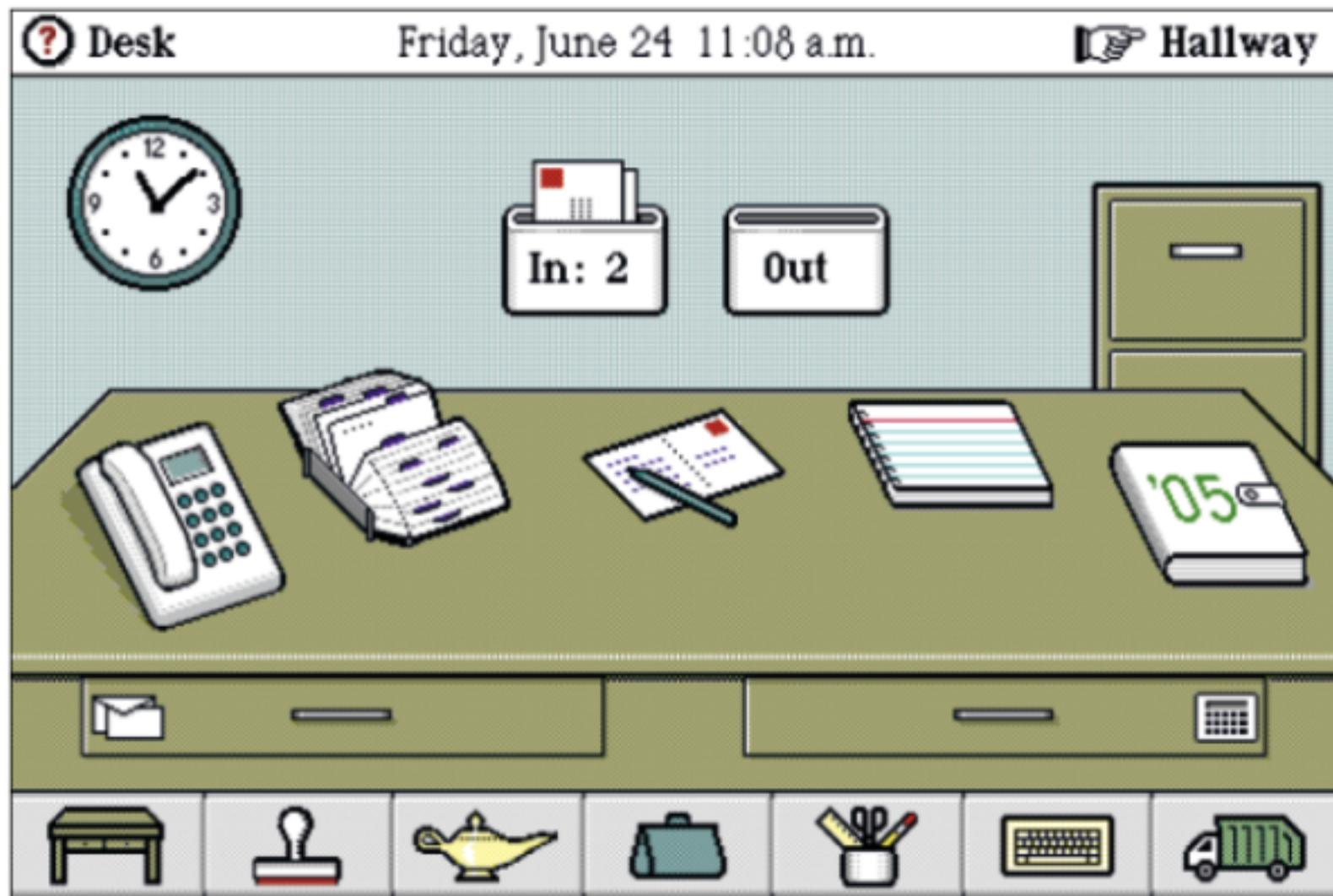
1. Requires learning how the functions work
2. Requires skills in using the functions
3. The system cannot perform high-level actions

# Metaphorical Design

**Definition:** Following a real-world metaphor that users are expected to be familiar with.

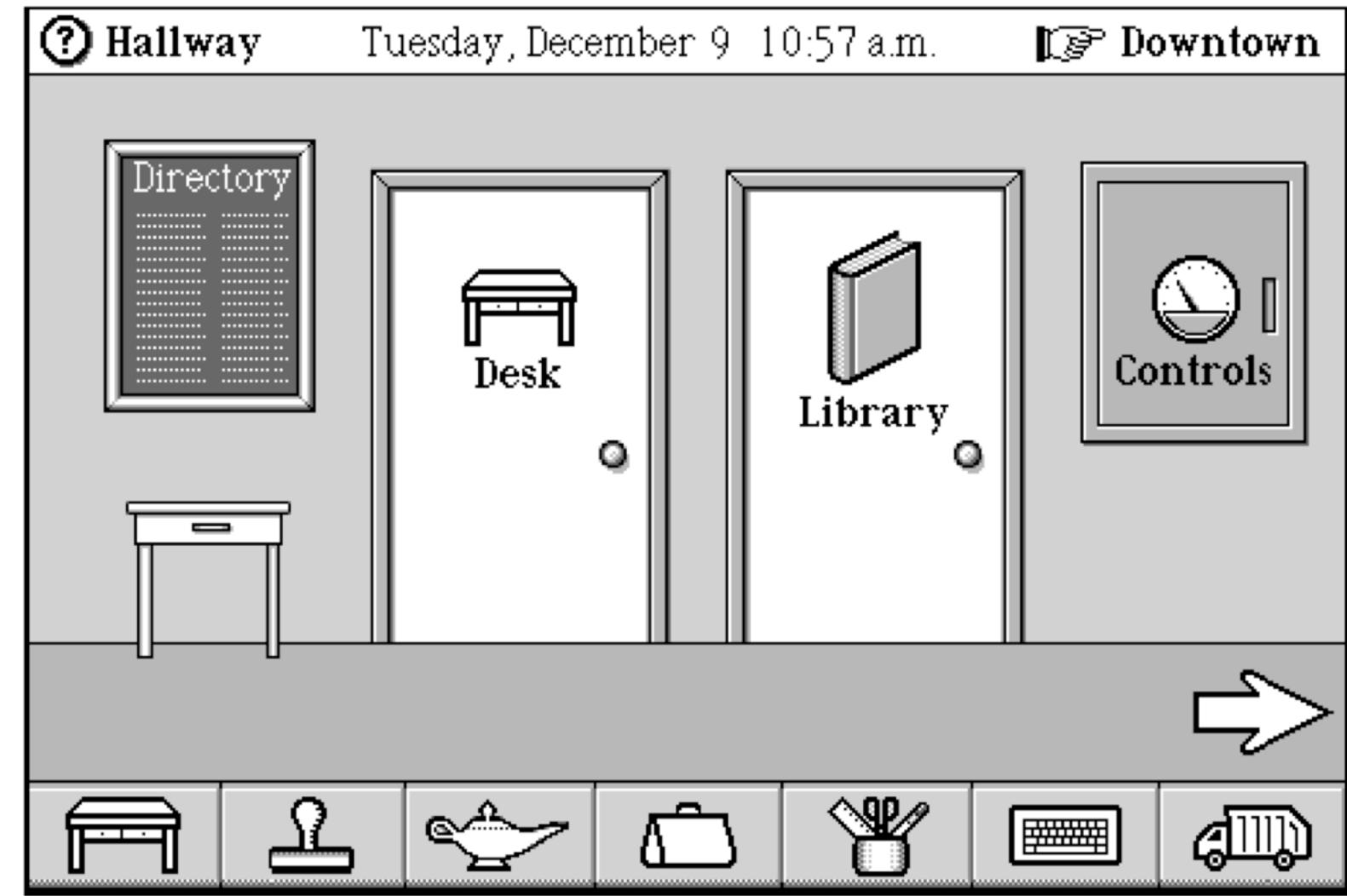
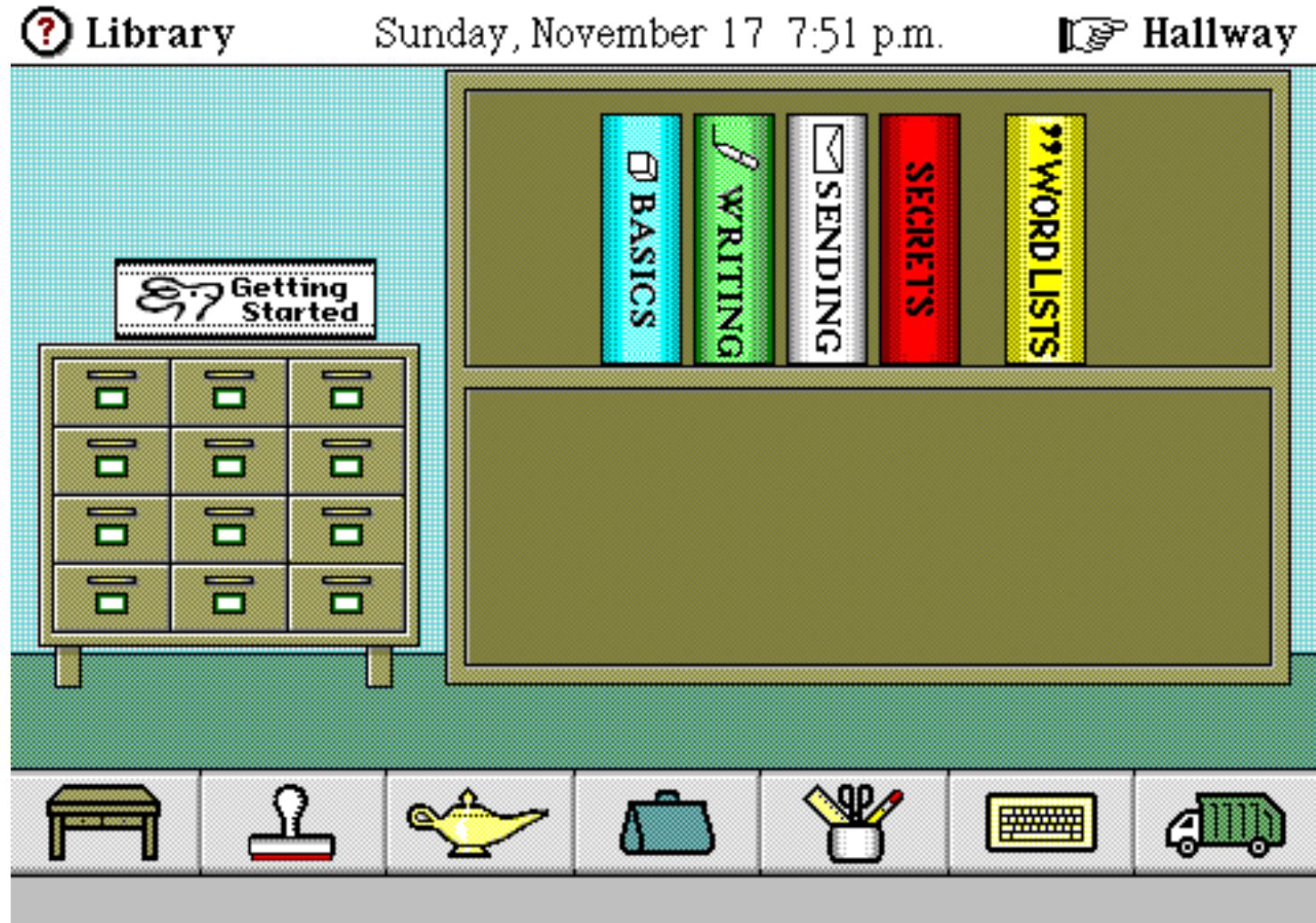
Metaphorical designs "jump-start" user mental models, rely on their existing knowledge of how things work in the real-world, and thus eliminate learning.

# Source<sup>4</sup>



<sup>4</sup> Wikipedia: [Magic Cap](#)

# Source<sup>56</sup>



<sup>5</sup> Wikipedia: [Magic Cap](#)

<sup>6</sup> NN Group: [The Anti-Mac Interface](#)

# Source<sup>7</sup>

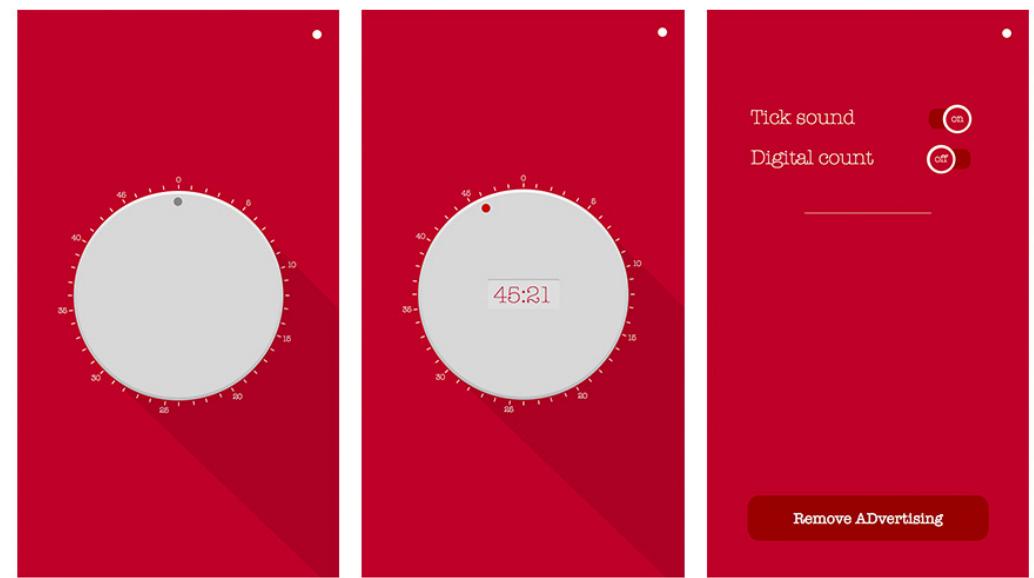
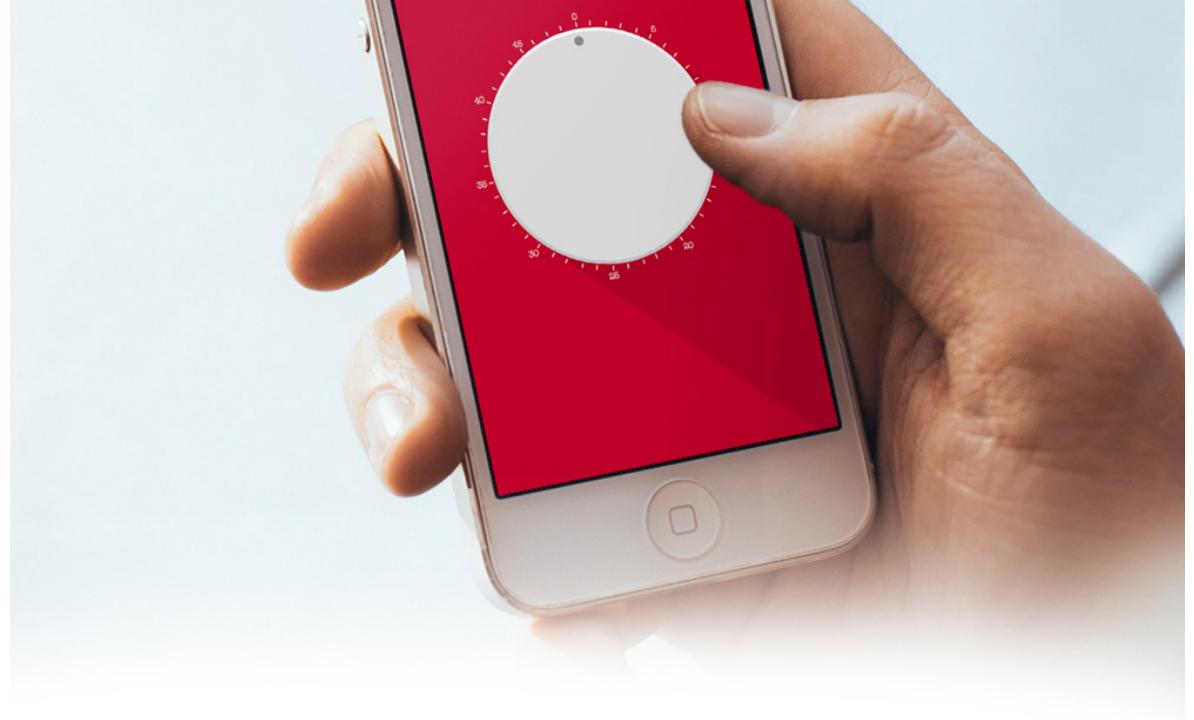
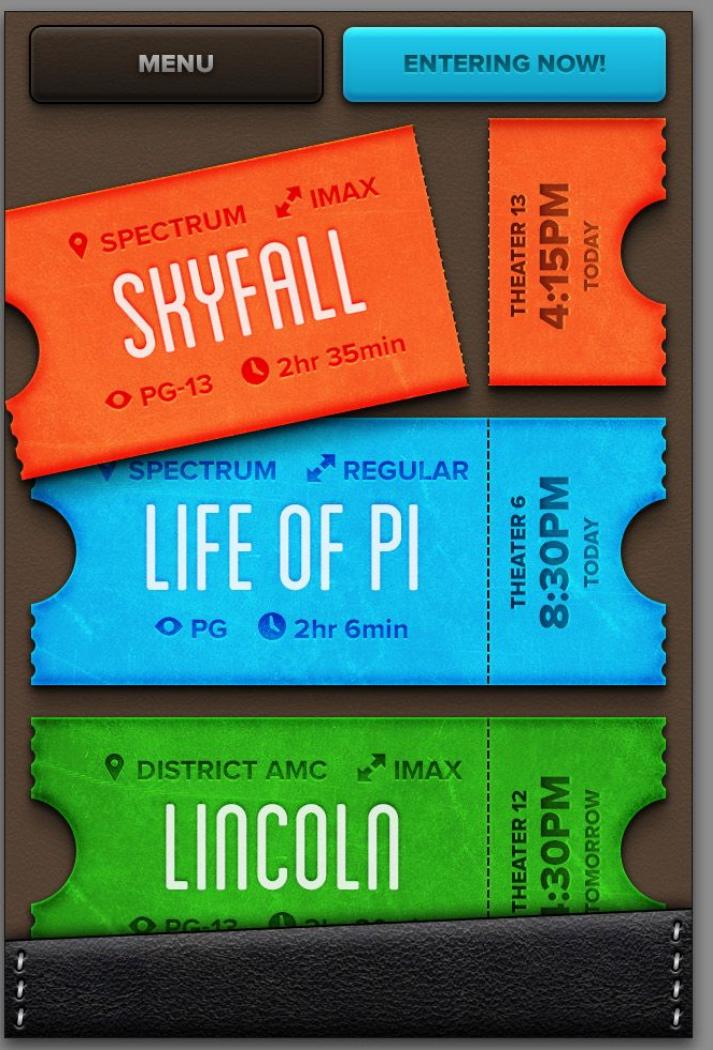


<sup>7</sup> UX Planet: [Metaphorical Design](#)

# Source<sup>8</sup>



<sup>8</sup> Apple App Store: [76 Synthesizer](#)



AND REDISEIGN FOR APPLE WATCH



*Pro Tip 1:* Metaphors use a familiar model from another domain (e.g., building vs. computer windows); analogues are similar to models in the same category (e.g., physical cards vs. e-cards).

*Pro Tip 2:* Metaphors can be applied at different levels of abstraction.

*Pro Tip 3:* Mixed metaphors bring together models from different domains in a single design.

## Global Metaphor

**Definition:** A *global metaphor* provides a single, overarching framework for all the metaphors in the system (e.g., Magic Cap).

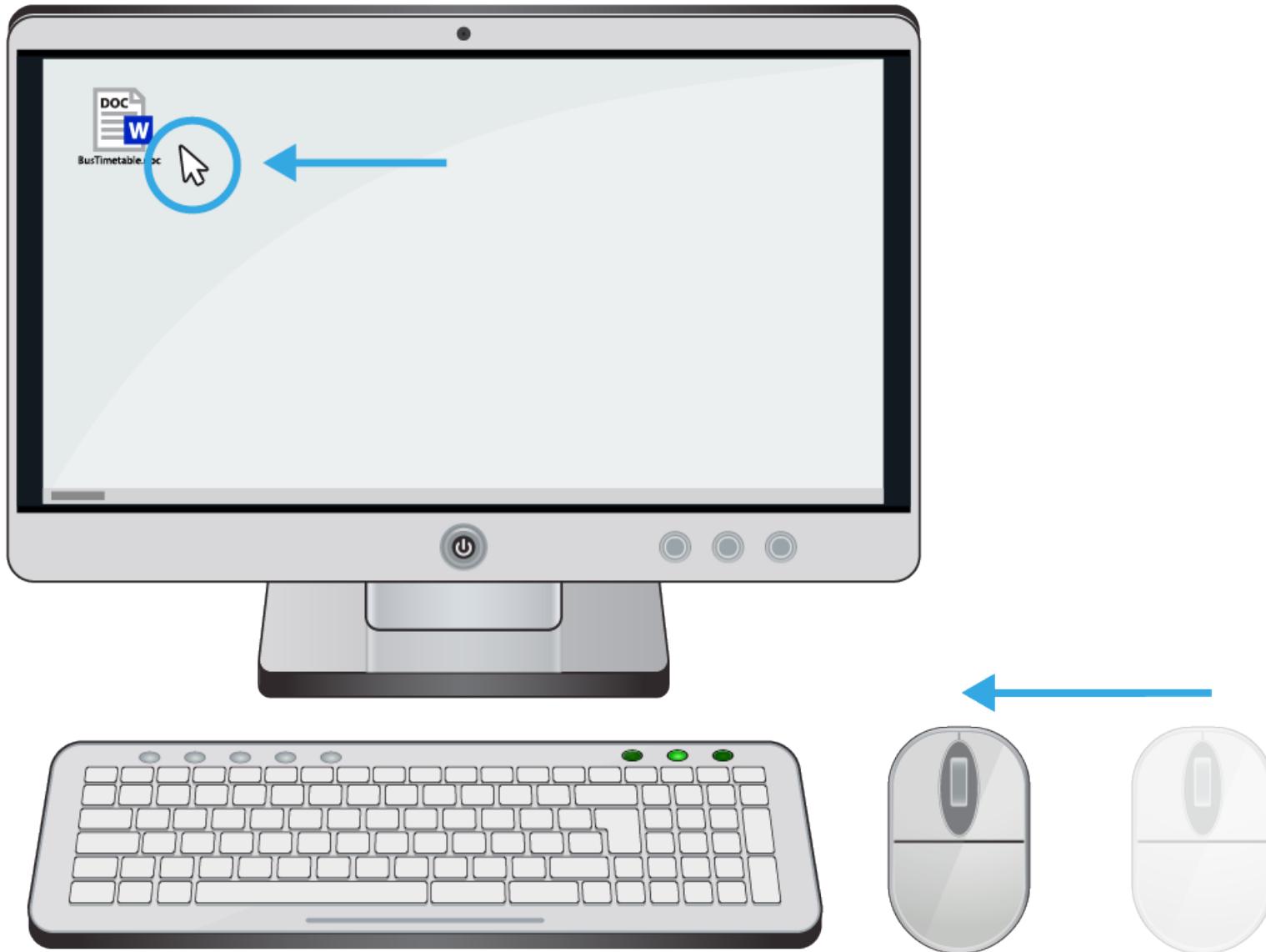
*Pros:* They work well in expert interfaces where the interface simulates a real-world system.

*Cons:* Inability to scale; lack of familiar real-world system for entirely new capabilities; cultural differences; inability to adapt as capabilities evolve.

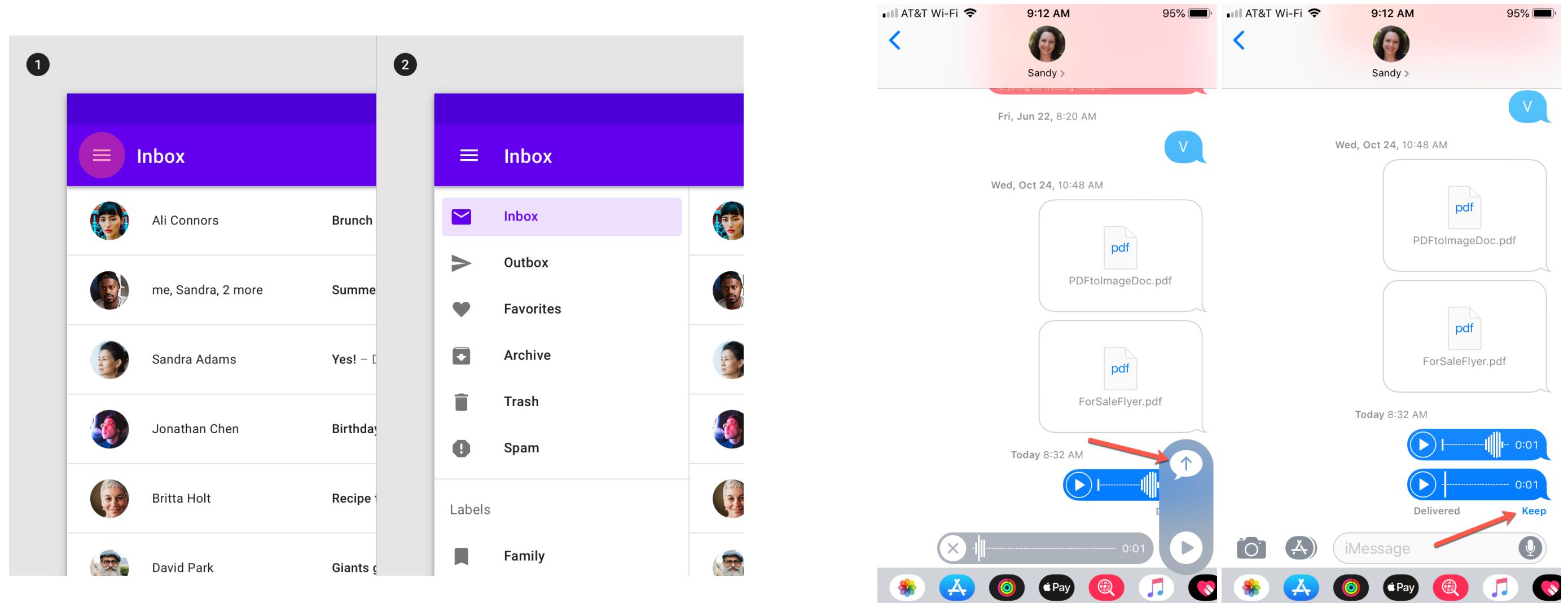
# Idiomatic Design<sup>10</sup>

**Definition:** Building dedicated, highly expressive interaction capabilities that users must learn.

Mapping cursor movements on a screen to mouse movements is an extremely successful example.



<sup>10</sup> [Image Source](#)



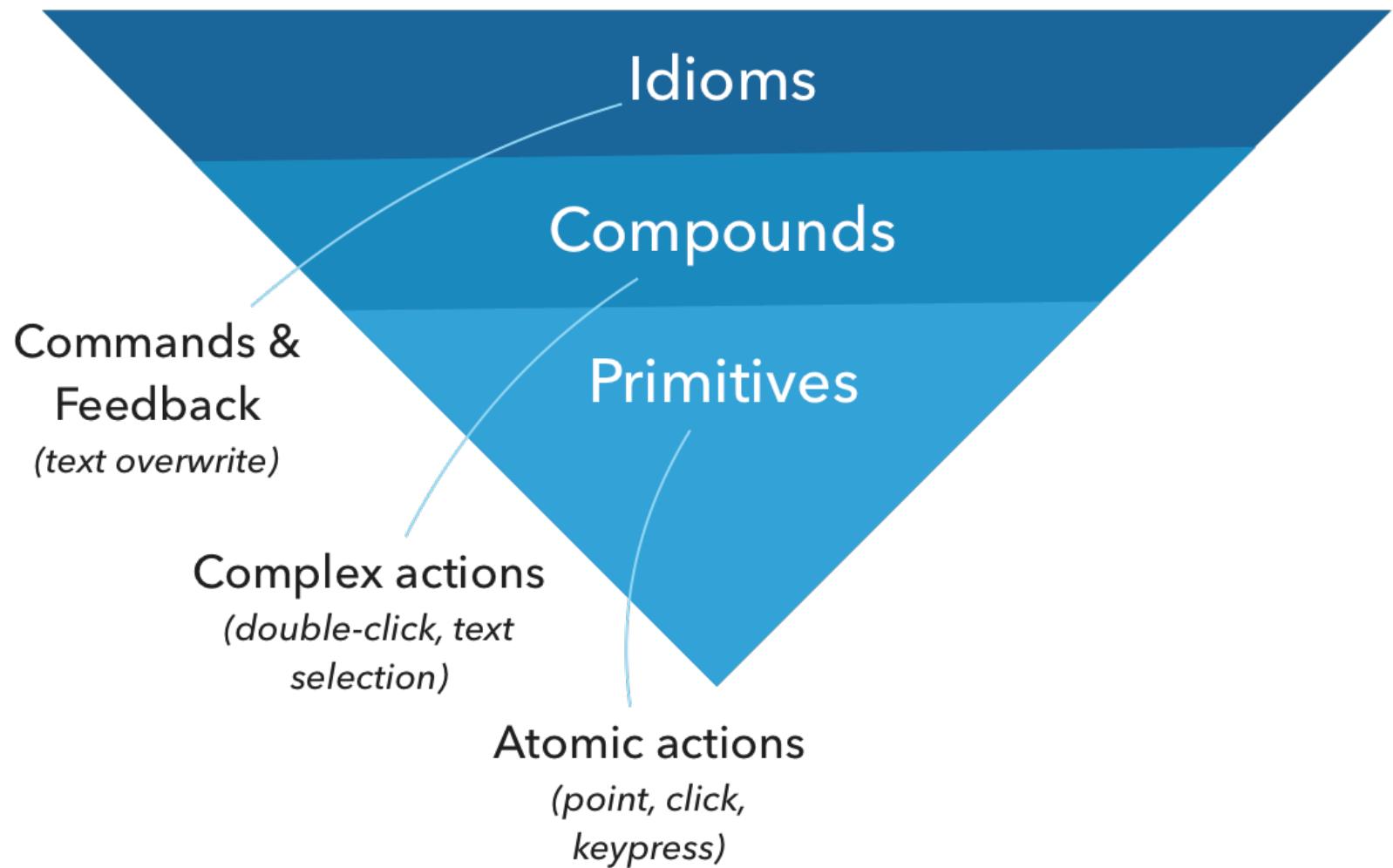
<sup>11</sup>Image Source

<sup>12</sup>Image Source

# Developing Idioms<sup>13</sup>

In designing idioms involve, three elements are established:

1. **Primitives**: atomic actions, e.g., point, click
2. **Compounds**: complex actions, e.g., double-click
3. **Idioms**: higher-level elements, e.g., deleting text



<sup>13</sup> Cooper et al., 2014, About Face

# Affordances

## Affordances

**Definition:** The perceived properties of a design element that give clues about how to interact with it. Designers have borrowed the concept from ecological psychology.

**Theoretical Roots:** James Gibson (1977, 1979) suggested that the human environment is structured in a way that communicates action possibilities through *affordances*.

Which environment affords *walking*?



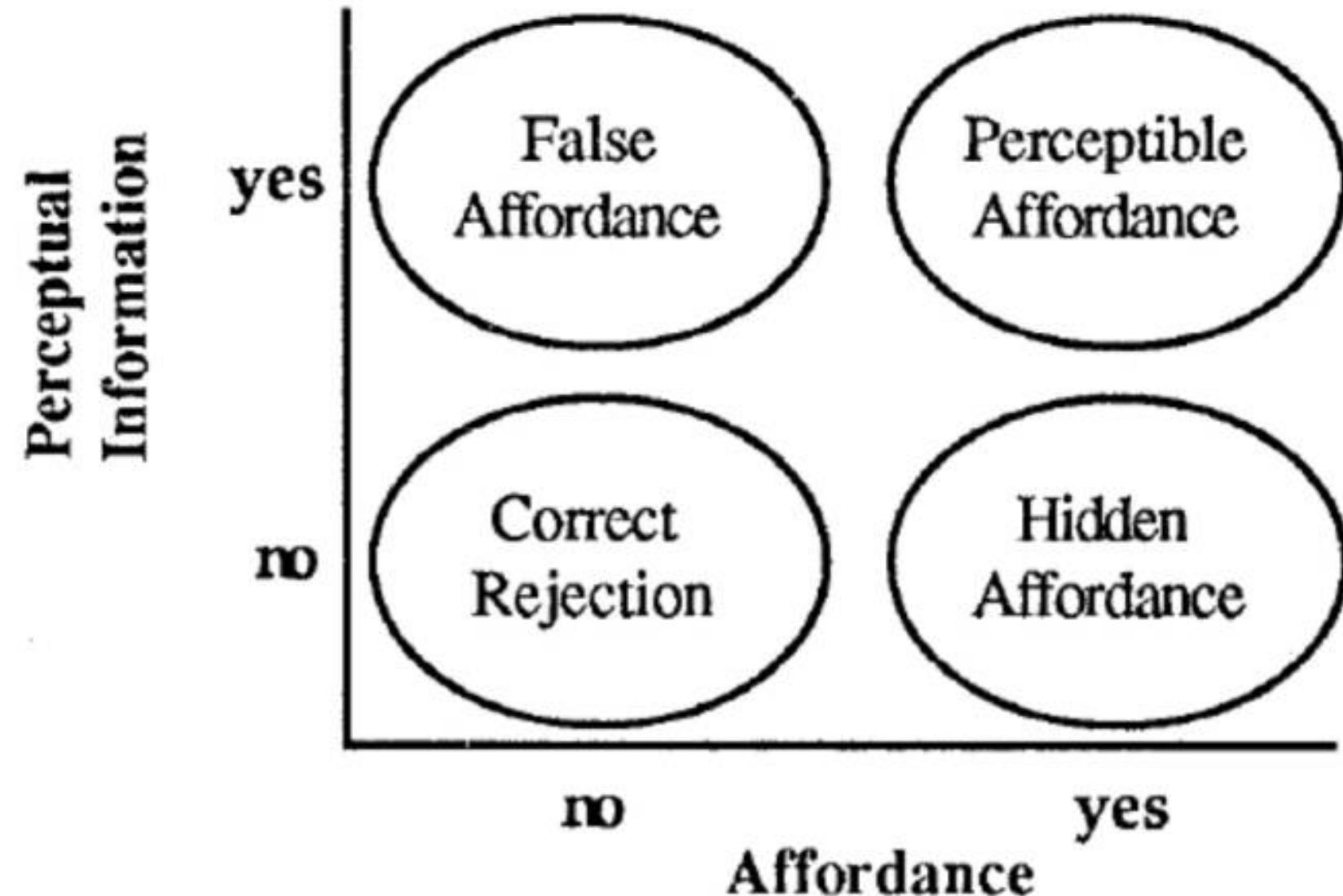
26



# Affordances in Design

*Perceptible affordances* enable users to intuitively recognize actions that are possible with interface elements.<sup>14</sup>

Affordances can also be *hidden* and *false*.

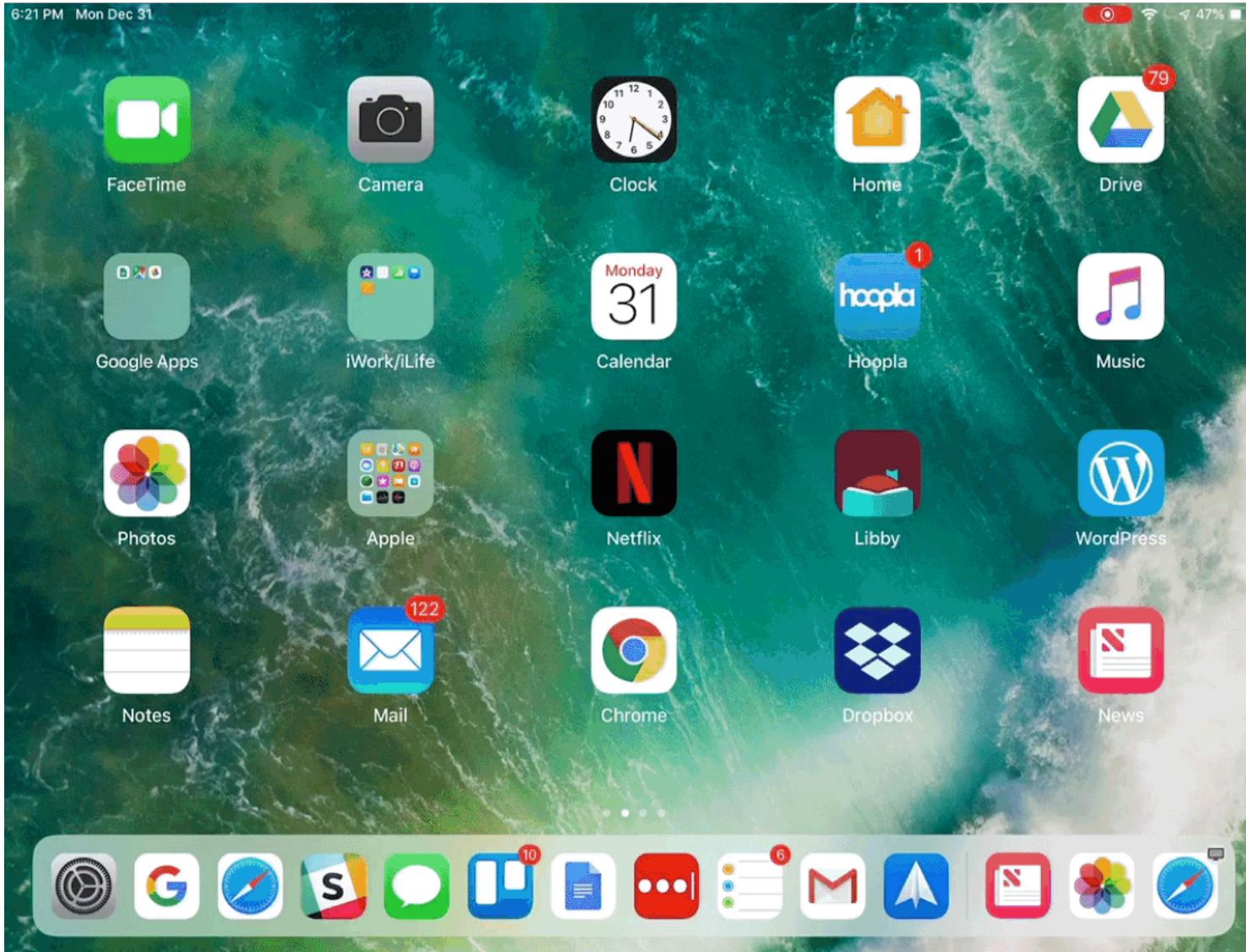


<sup>14</sup> Figure: Gaver, 1991, *Technology Affordances*

**False Affordances:** an action that is perceived by the user but in fact does not work as expected.

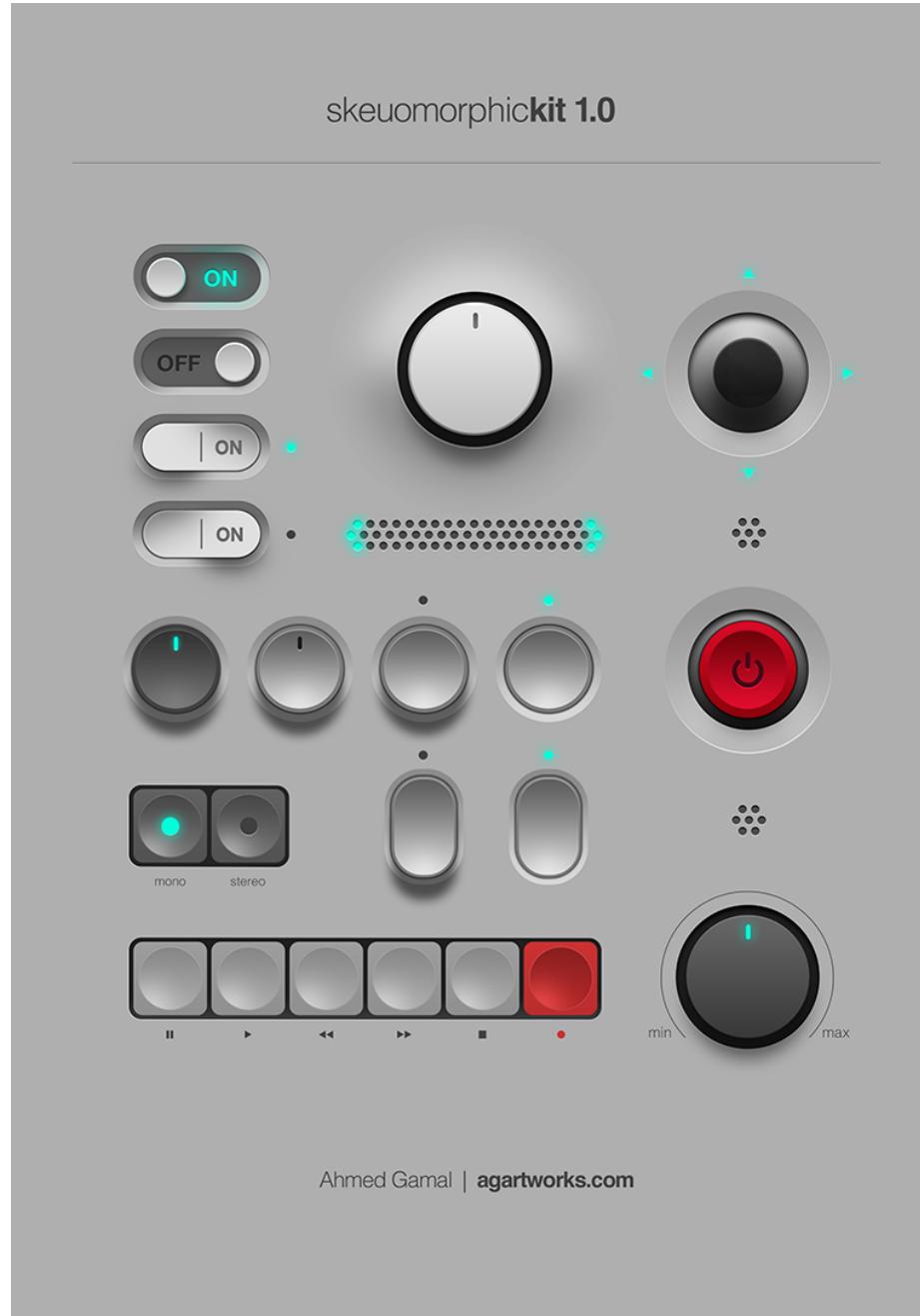


# Hidden Affordance: the affordance is not too obvious.





# Perceptible Affordances: an object's characteristics imply an action.



# In-Class Activity

## Metaphor & Affordance Deconstruction



Angle PRQ

$$\cos A = \frac{(9\sqrt{2})^2 + 13^2 - 5^2}{2(9\sqrt{2})(13)}$$

$$\cos A = 0.924$$

$$A = 22.38$$

6.  $f(x) = x^2 + 8ax + 4a^2, x \geq 0$   
 $g(x) = 6x - 2a, x \in \mathbb{R}$

(a)  $f(g(x)) = f(6x - 2a)$

$$= (6x - 2a)^2 + 8a(6x - 2a) + 4a^2$$

$$= [36x^2 - 12ax - 12ax + 4a^2]$$

$$= [36x^2 - 24ax + 4a^2] + 48ax - 16a + 4a^2$$

$$= 36x^2 + 24ax - 8a^2$$

IF  $a = 4$

(b)  $f(g(2))$

$$= f(6(2) - 2(4))$$

$$= f(4)$$

$$= 4^2 + 8(4)(4) + 4(4)^2$$

$$= 208$$

(c)  $f^{-1}(x) : x^2 + 8(4)x + 4(4)^2$

$$y = x^2 + 8(4)x + 4(4)^2$$

$$y = x^2 + 32x + 64$$

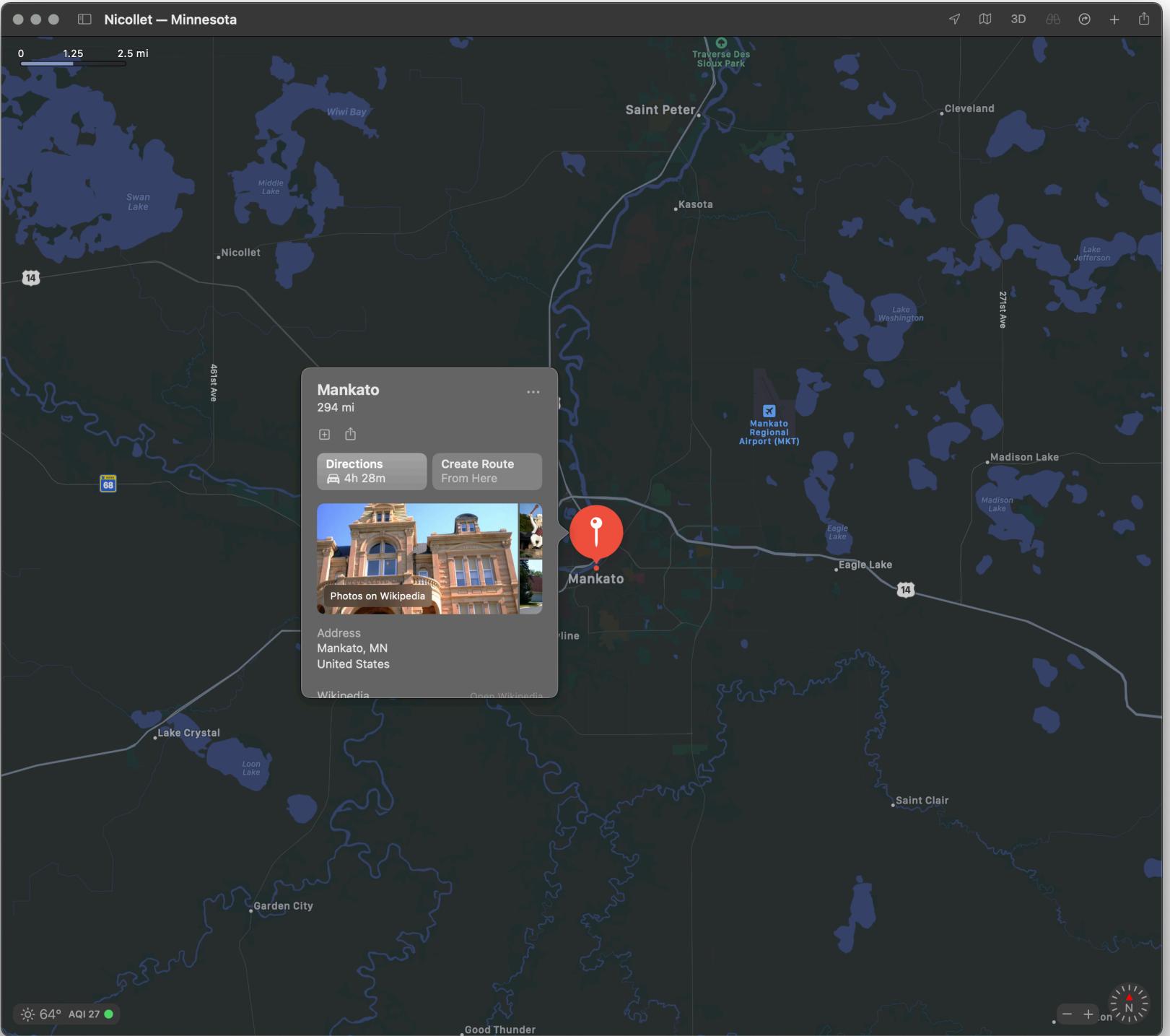
$$y = (x+16)^2 - 192$$

$$y + 192 = (x+16)^2$$

$$\sqrt{y+192} = x+16$$

$$-16 \pm \sqrt{y+192} = x$$

so  $f(x) = -16 + \sqrt{x+192}$



# Design Patterns

# Design Patterns

**Definition:** A design pattern is a general, reusable solution to a commonly occurring problem within a given context.

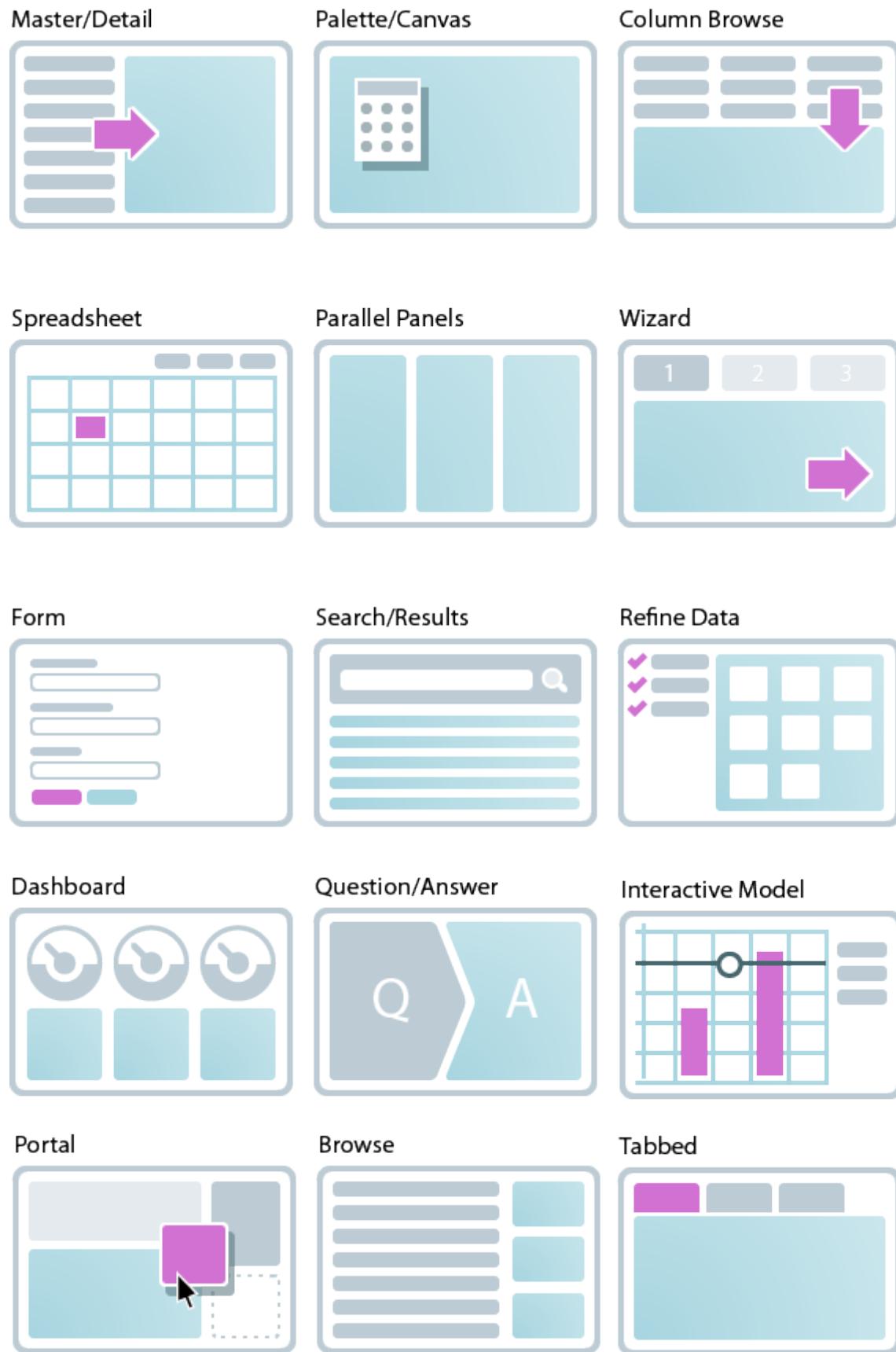
Originally developed by Christopher Alexander (1977; *A Pattern Language*) to address problems in architecture and city planning.<sup>15</sup>



<sup>15</sup> [Smart Cities Dive](#)

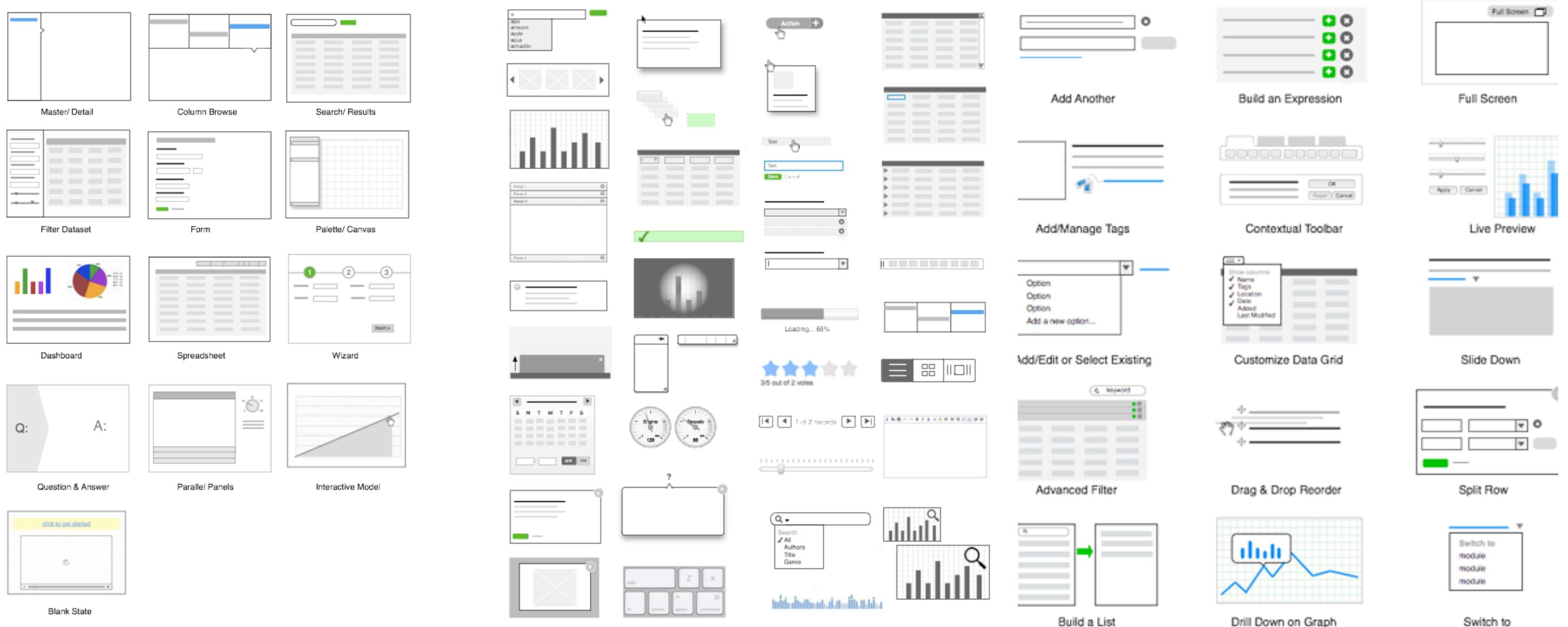
# Design Patterns in UX

In the last decade, designers have also developed and refined patterns for overall structure and organization, components and controls.<sup>16</sup>



<sup>16</sup> Neil, 2010, 12 Standard Screen Patterns

# Source<sup>17</sup>



<sup>17</sup> Neil, 2010, 12 Standard Screen Patterns

# Pros & Cons of Design Patterns

## Pros:

1. Reducing design time and effort
2. Improving the quality of design solutions
3. Establishing familiarity across systems
4. Providing a baseline or state of the art

## Cons:

1. Not every design problem will warrant a pattern
2. Patterns may not exist for new design spaces

# Design Languages

## The Problem with Patterns

**Problem 1.** Can I piece together different patterns to make a complete design? **No**, as this eclectic design would lack coherence.

**Problem 2.** How do I choose which pattern to use? Are patterns interchangeable? **No**, there has to be a *principle* to the selection of patterns.

**Problem 3:** Pattern languages help you create a design that is consistent vertically. How do we create a system that is consistent *horizontally*? I.e., how do we achieve visual and behavioral consistency in designs?

## Enter Pattern Languages

**Define:** A complete and hierarchical collection of patterns for a family of design problems.

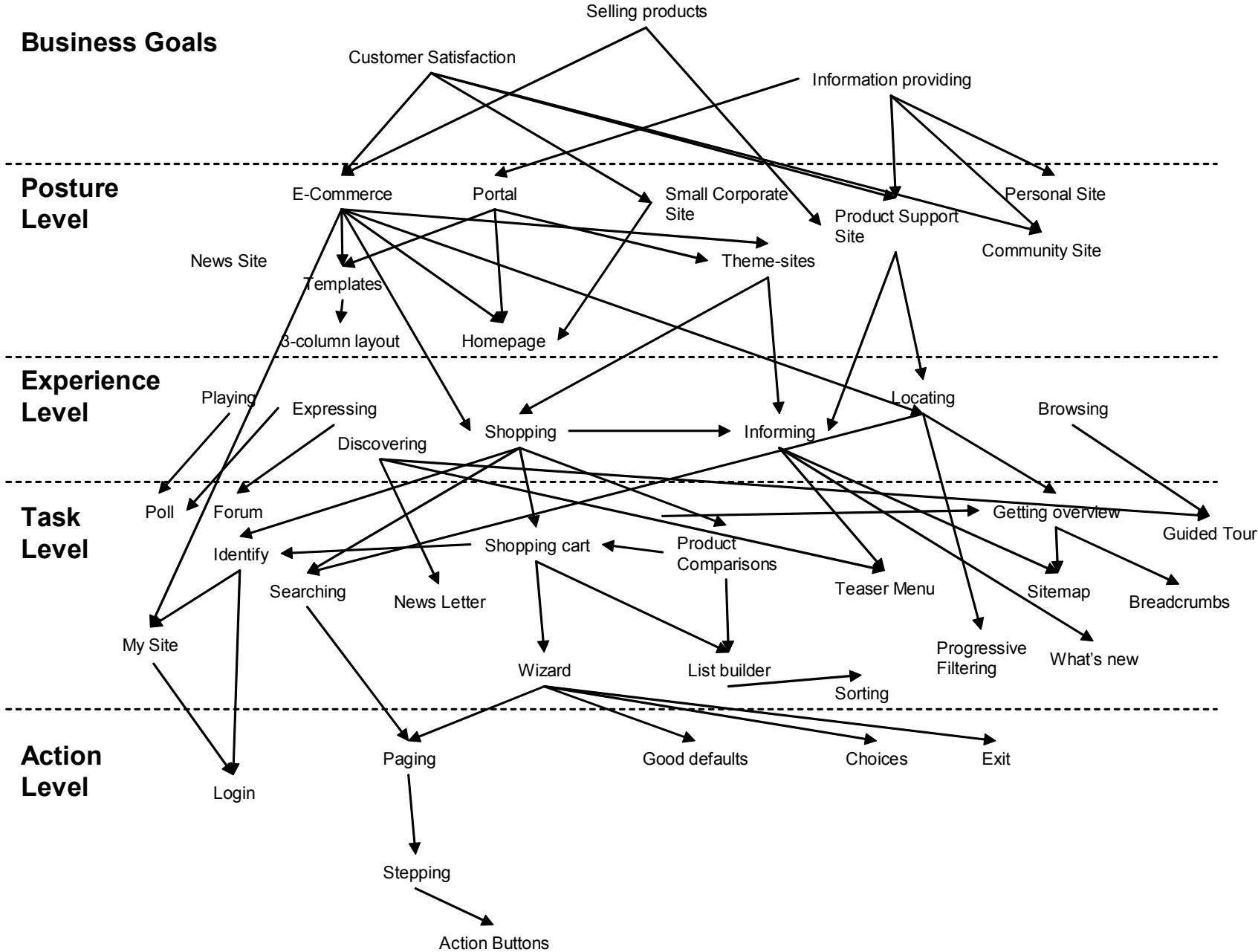
Patterns are *words* (e.g., a component) that are connected with grammar rules to make *sentences* (e.g., a screen) and eventually *language* (e.g., user experience).<sup>18</sup>

The pattern language can be thought of as patterns being applied at different *levels*. Let's see an example.

<sup>18</sup> Kruschitz & Hitz, 2009

# Source<sup>19</sup>

The screenshot shows the homepage of orange tree imports.com. At the top left is the logo featuring an orange tree in a pot. The main navigation menu includes links for SHOP, COOKING SCHOOL/EVENTS, WEDDING REGISTRY, ABOUT OTI, a shopping cart icon, and a user profile icon. A search bar with the placeholder "search products" and a magnifying glass icon is located at the top right. The background is teal. In the center, there is a large image of three smiling plastic spoons (red, white, and green) with faces drawn on them. Below this image, the text "Fun, Functional, Fantastic" is written in a large, cursive, black font. At the bottom of the page, there is a welcome message: "Welcome to Madison's favorite specialty shop! Here you'll find ideas for fun gifts, seasonal delights, and functional tools to make your kitchen a happy place. Whether you're visiting us in person in Madison, Wisconsin or online, our friendly staff looks forward to making your shopping experience fantastic." Another message below states: "We are now able to have a limited customers at a time come into the store to shop from 11:00-5:30, Monday-Saturday. Please note that we require that staff and shoppers wear face covering, and use hand sanitizer. We want everyone to stay well!"



<sup>19</sup> van Welie & van der Veer, 2003

## Posture-Level Patterns

**Definition:** The *structure* that an application follows, i.e., what *type* of application it is, e.g., "an e-commerce app," "a social media app," or "a personal homepage site."

## Elements of a Posture-level Pattern

Once we determine the posture of an application, it gives us guidance on:

- Structure
- Components
- User experience
- Alternatives/competitors

**Structure:** Central canvas with supporting panels<sup>22</sup>

**Components:** Canvas, dashboard, score panel, data summary

**UX:** Measurement during the activity, review later

**Competitors:** Strava, RunKeeper



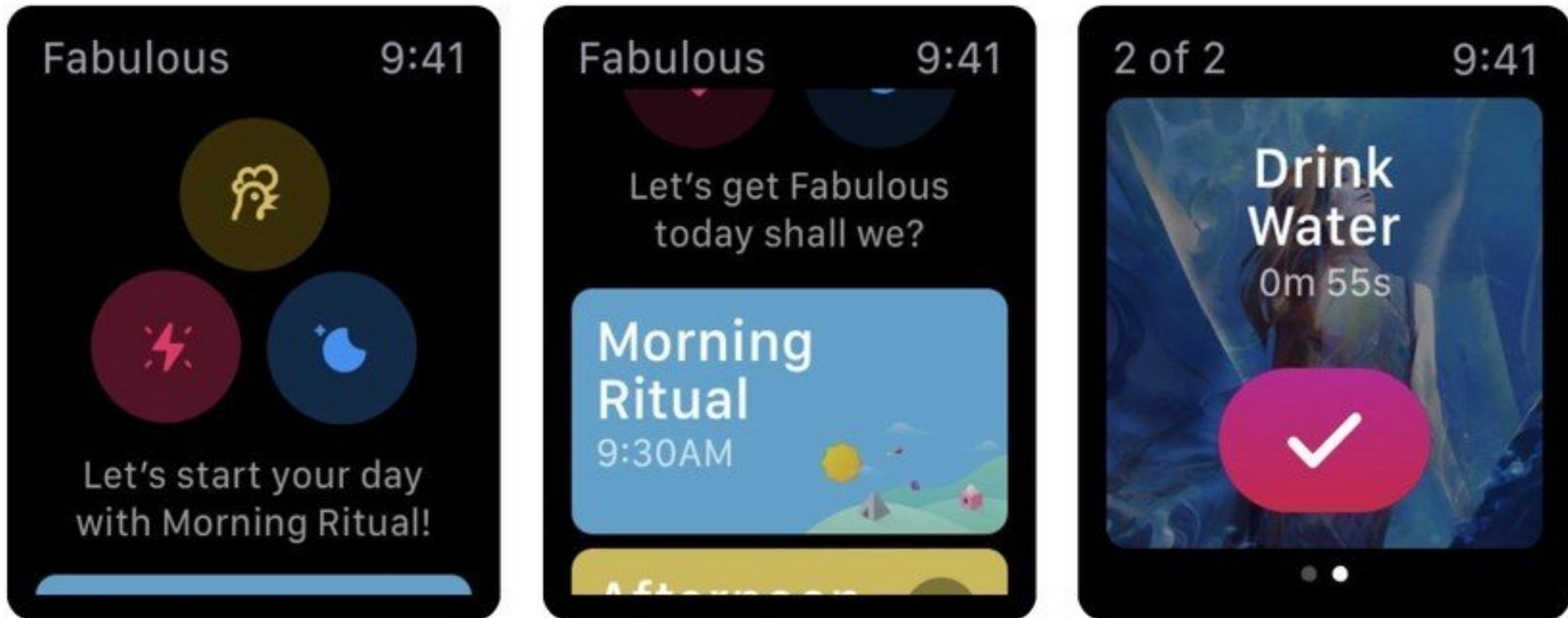
<sup>22</sup> [Image source](#)

## Experience-Level Patterns

**Definition:** The *user goals* that make up the *user experience* that the application supports, e.g., activity tracking, coaching, and reviewing.

Experience-level patterns can also capture the *quality* of the user experience, e.g., *motivational* coaching.

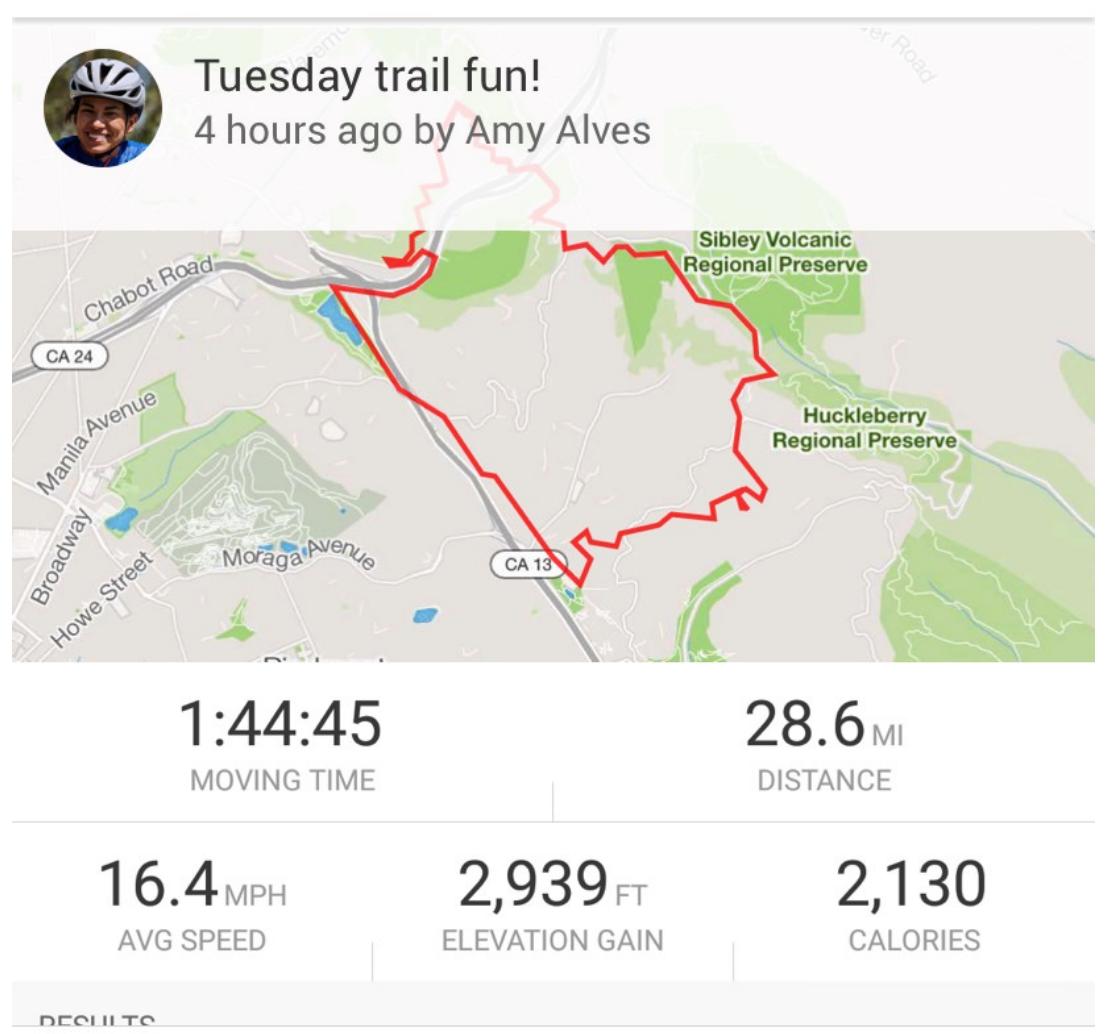
# Source<sup>23</sup>



<sup>23</sup> Image source

## Elements of an Experience-Level Pattern<sup>24</sup>

- Primary goals, e.g., activity tracking
- Secondary goals, e.g., community building



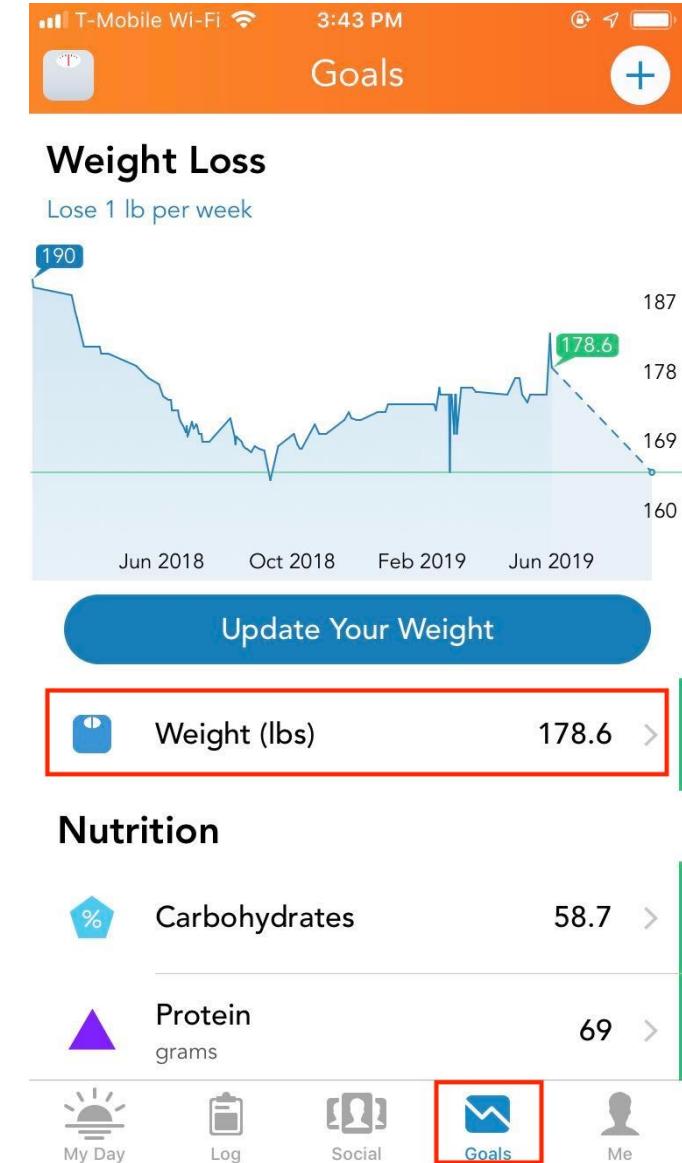
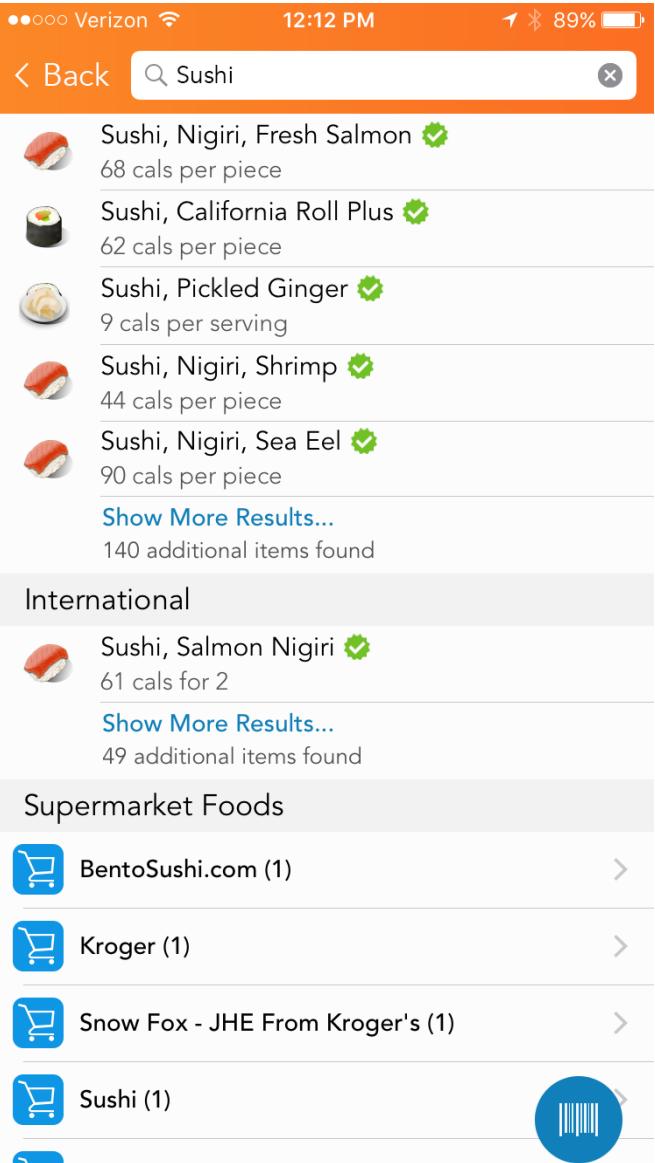
<sup>24</sup> [Image source](#)

## Task-Level Patterns

**Definition:** Design solutions that help users accomplish sequences of actions that make up user tasks, e.g., logging a meal, capturing a run, or completing a workout.

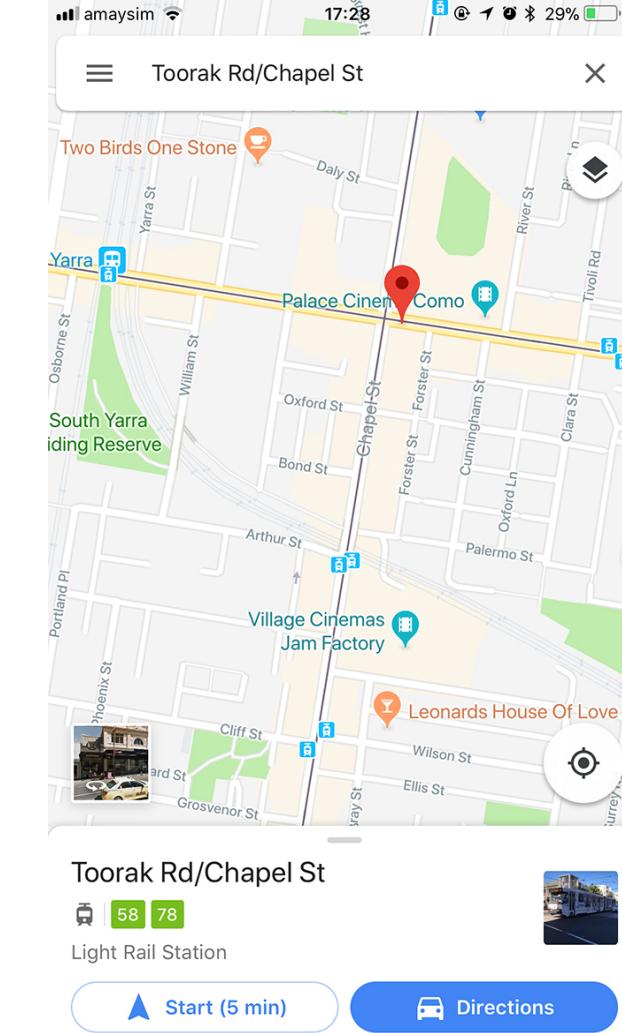
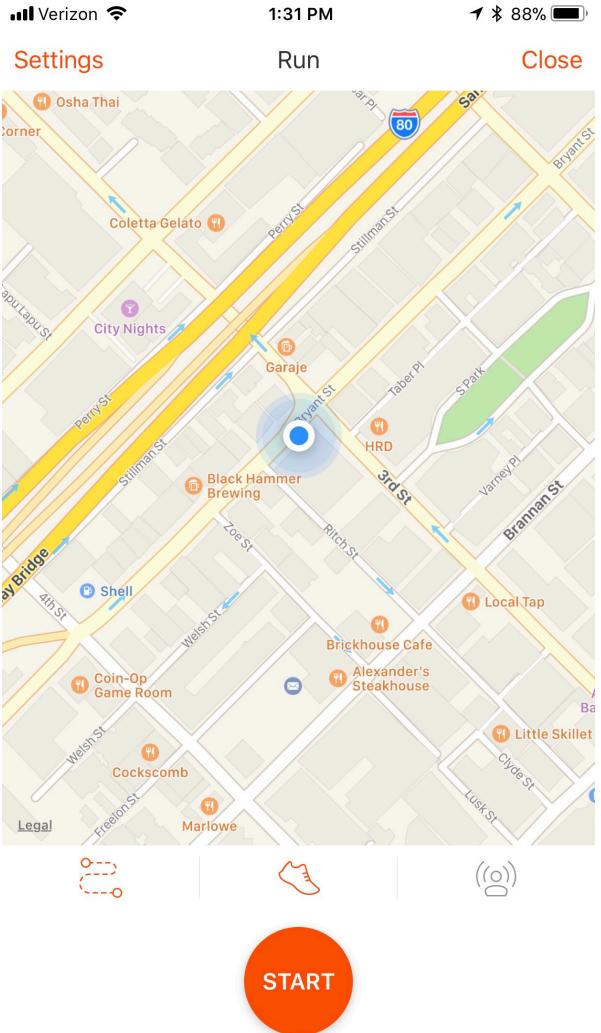
*Tasks* point to specific layout patterns. E.g., meal logging can be done through a "search-and-results" pattern, capturing a run can be done through a "dashboard" pattern.

# Source<sup>25</sup>



<sup>25</sup> Image sources: [left](#), [right](#)

Task-level patterns can be domain independent. Business goals and posture-level patterns set the context for these patterns.<sup>26</sup>



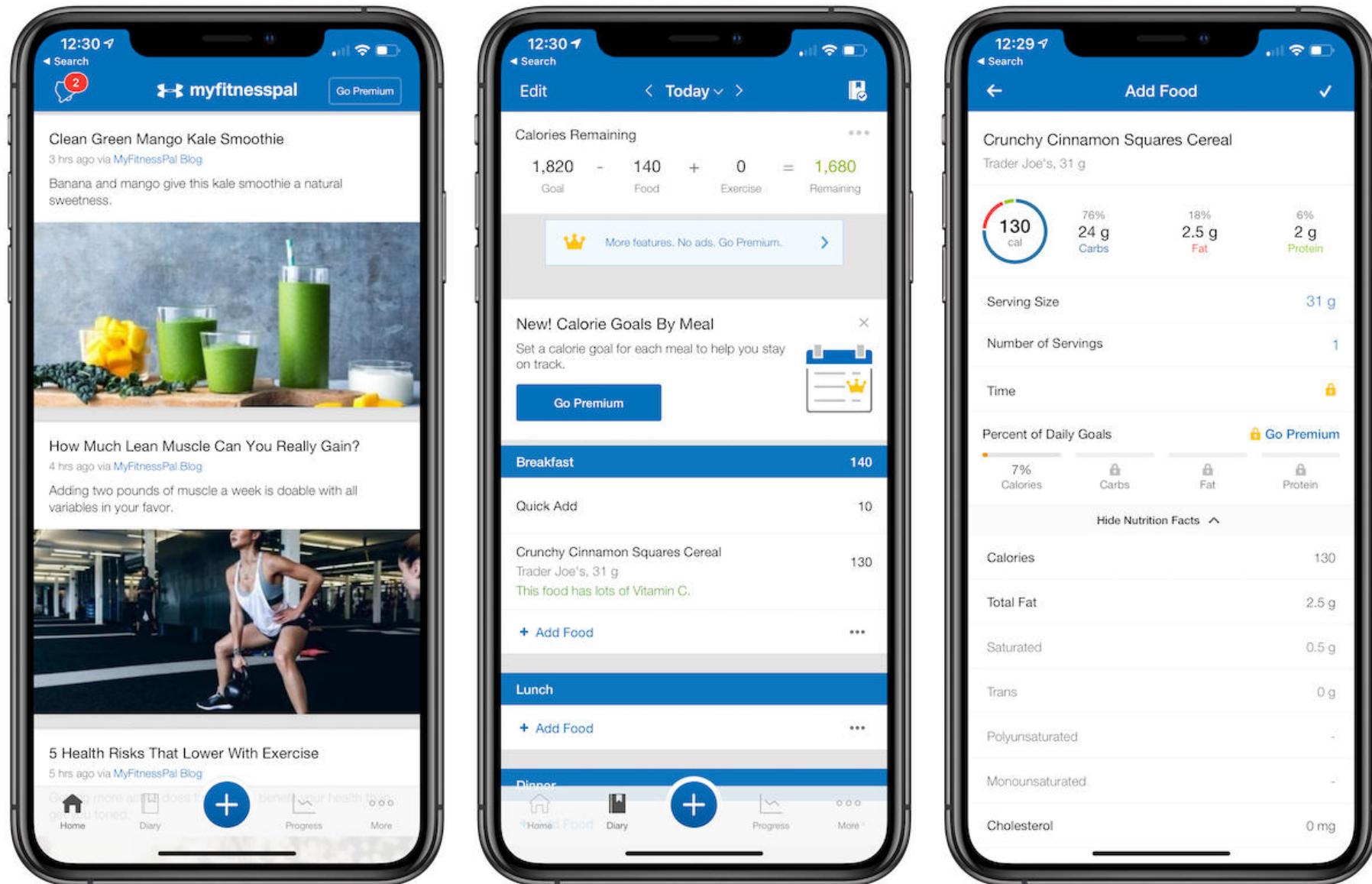
<sup>26</sup> Image sources: [left](#), [right](#)

## Action-Level Patterns

**Definition:** Design solutions that support the actions taken to complete the steps(s) of the user's task, e.g., a "start" button to initiate activity tracking, a selectable list entry for a food item.

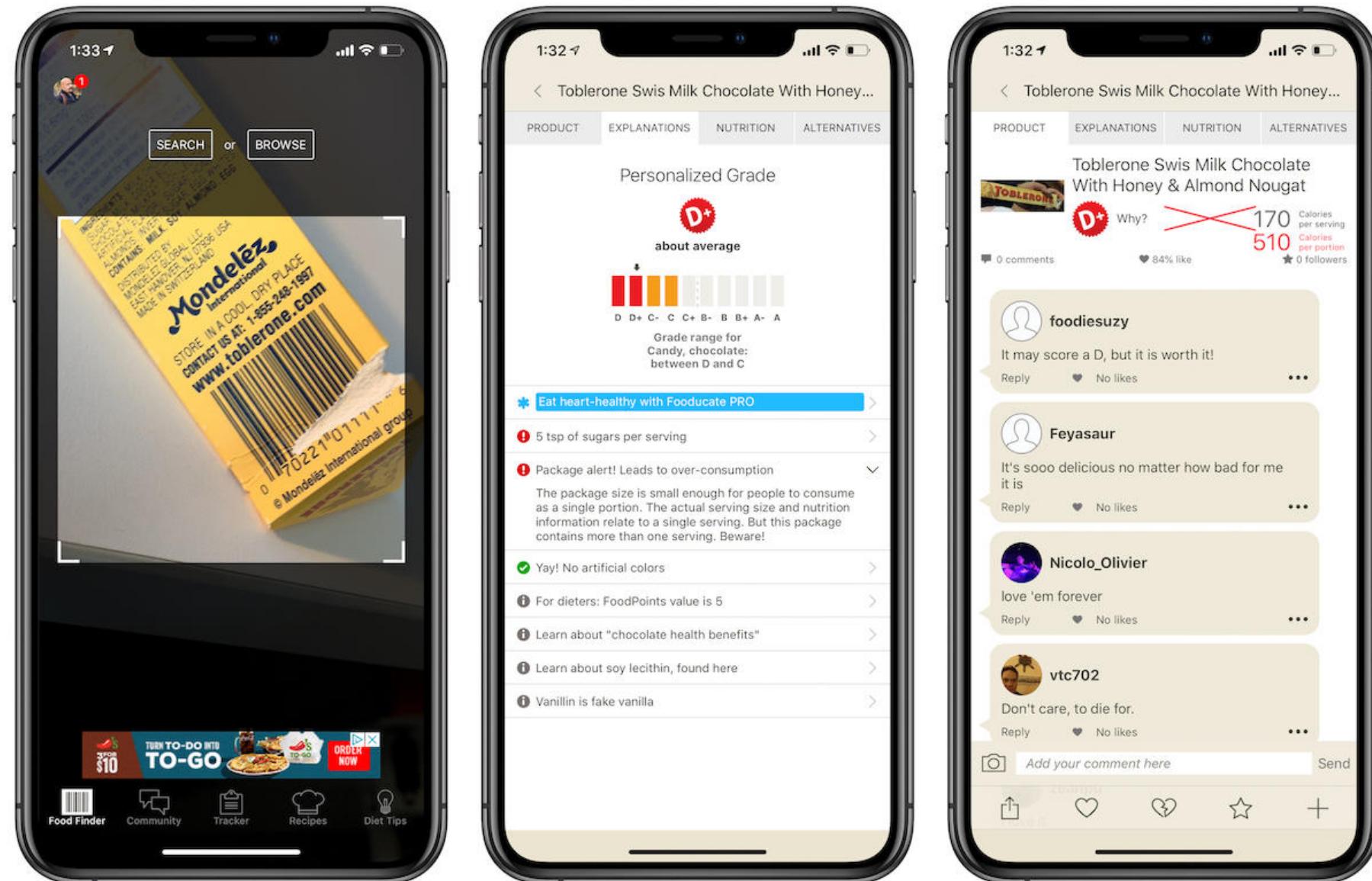
Action-level patterns are the lowest level of building blocks for a design. They are often called *widgets* or *components* (as in React).

# Action-level patterns for a *food tracking app*:<sup>27</sup>



<sup>27</sup> Image source: [My Fitness Pal](#)

# Action-level patterns for a *food education* app:<sup>28</sup>



<sup>28</sup> Image source: [Fooducate](#)

# In-Class Activity

## Pattern Language Deconstruction

# Source<sup>36</sup>

The screenshot shows the Cronometer website homepage with an orange header. The header includes the Cronometer logo, navigation links for HOME, BLOG, FORUMS, EMAIL, and a green LOGIN button. Below the header, a large white banner features the text "Track Your Nutrition, Fitness, & Health Data" and "Log your Diet, Exercise, Biometrics and Notes". A green "SIGN UP FOR FREE" button is centered below the banner. At the bottom of the page, there is a screenshot of the Cronometer mobile app interface showing a diary entry for April 19, 2016, with various food items and their nutritional values.



<sup>36</sup> Image sources: [left](#), [right](#)

## Business Goals

*Mission of the application*

## Posture Level

*"Type" of application*

## Experience Level

*User goals*

## Task Level

*Task sequences*

## Action Level

*User actions*

## Track Your Nutrition, Fitness, & Health Data

Log your Diet, Exercise, Biometrics and Notes

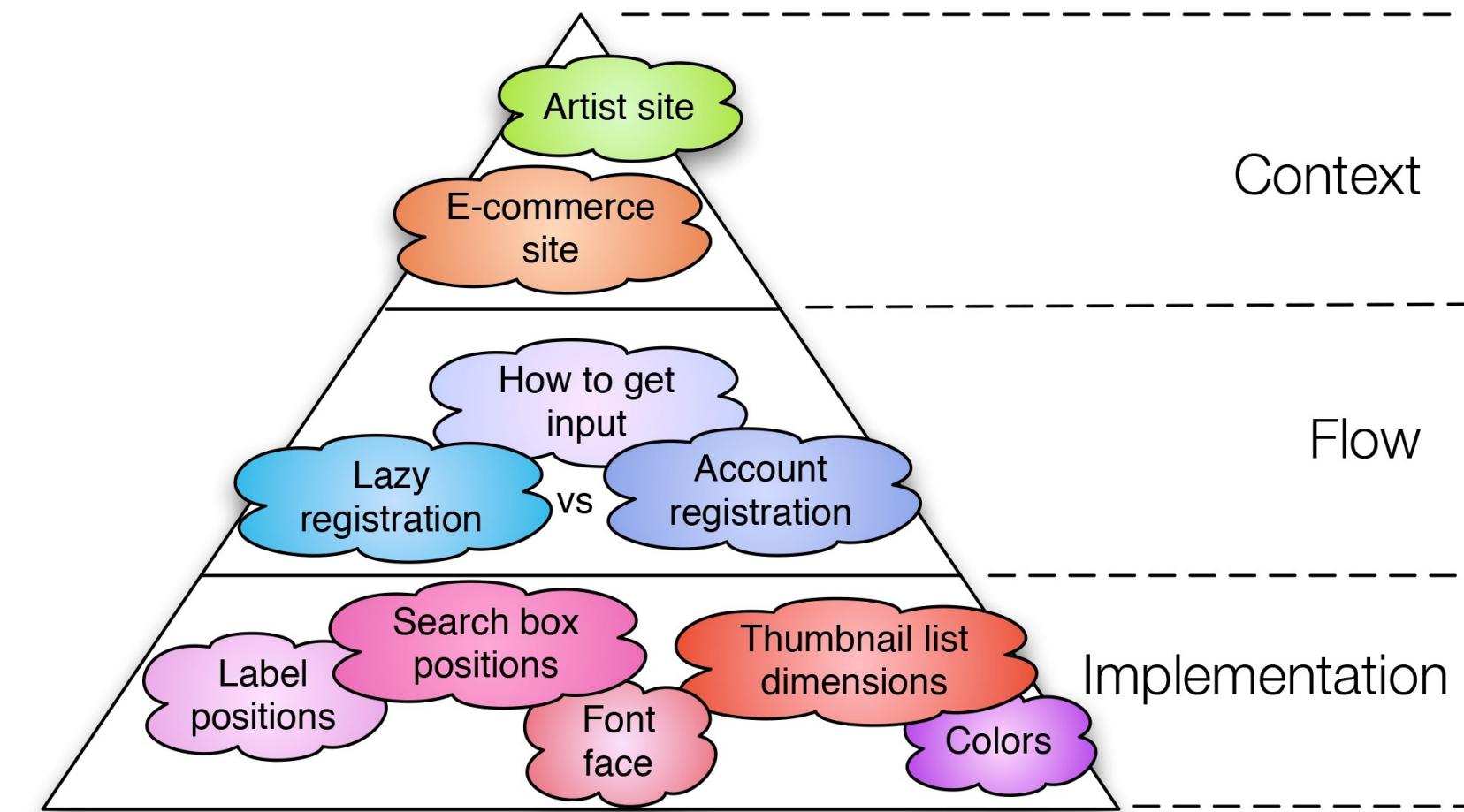
SIGN UP FOR FREE



# A Simplified Model<sup>29</sup> <sup>30</sup>

Three-levels of patterns:

1. **Context:** Type of app
2. **Flow:** Components that support specific functions
3. **Implementation:** The visual/behavioral elements that implement the functions



<sup>29</sup> Anders Toxboe

<sup>30</sup> More on the three-levels of patterns by Jerry Cao

## How do we use patterns?

**Common practice:** Patterns in the higher levels are defined informally, and the task- and action-level patterns are adopted through experimentation and trial and error.

**The problem:** Ineffective (e.g., lack of coherence across different levels) and inefficient (wasted effort in experimentation).

**The solution:** Defining patterns top to bottom will "generate" the design when patterns are available across all levels.<sup>31</sup>

<sup>31</sup> van Welie & van der Veer, 2003

# Where do we find patterns?<sup>32</sup>

Task- and action-level patterns are organized into catalogues/collections based on functional similarity.

## User Interface Design Patterns

Getting input	Navigation	Dealing with data	Social
<b>Forms</b> WYSIWYG Password Strength Meter Input Feedback Captcha Calendar Picker Structured Format Fill in the Blanks Expandable Input Keyboard Shortcuts Input Prompt Drag and drop Autosave Forgiving Format Morphing Controls Inplace Editor Good Defaults Preview Undo Settings <b>Explaining the process</b> Wizard Steps Left Completeness meter Inline Help Box <b>Community driven</b> Vote To Promote Pay To Promote Wiki Rate Content Flagging & Reporting	<b>Tabs</b> Navigation Tabs Module Tabs <b>Jumping in hierarchy</b> Notifications Breadcrumbs Modal Fat Footer Home Link Shortcut Dropdown <b>Menus</b> Vertical Dropdown Menu Horizontal Dropdown Menu Accordion Menu <b>Content</b> Carousel Tag Cloud Progressive Disclosure Cards Event Calendar Adaptable View Article List Continuous Scrolling Archive Categorization Tagging Thumbnail Favorites Pagination <b>Gestures</b> Pull to refresh	<b>Tables</b> Table Filter Alternating Row Colors Sort By Column <b>Formatting data</b> Dashboard Copy Box Frequently Asked Questions (FAQ) <b>Images</b> Slideshow Gallery Image Zoom <b>Search</b> Autocomplete Search Filters	<b>Reputation</b> Collectible Achievements Leaderboard Testimonials <b>Social interactions</b> Friend list [Mini] Activity Stream Follow Auto-sharing [Mini] Chat Friend Reaction Invite friends
			<b>Miscellaneous</b> <b>Shopping</b> Product page Pricing table Coupon Shopping Cart <b>Increasing frequency</b> Tip A Friend
			<b>Onboarding</b> <b>Guidance</b> Walkthrough Blank Slate Playthrough Coachmarks Guided Tour Inline Hints <b>Registration</b> Lazy Registration Account Registration Paywall

<sup>32</sup> [Image source](#)

# Online Pattern Libraries

- [UIPatterns.io](#)
- [UI-Patterns](#)
- [Mobbin](#)
- [UI Garage](#)
- [Welie](#)

# Design Style Guides

**Definition:** A vocabulary of design elements that are repeatedly applied to interaction design problems. These are task- and action-level interface components that follow a consistent look and feel in appearance and behavior.

*Non-digital example:* NASA Graphics Standard Manual.<sup>33</sup>



<sup>33</sup> NASA

#### NASA Uniform Patches

Personnel identification is an important facet of the NASA identification program. An embroidered patch incorporating the logotype is available for application on a wide variety of uniforms and clothing. Two patch designs, shown to the right, are available.

For general personnel, a white patch with a NASA Red logotype is available. This achieves the simplest and most effective identification on various types and colors of clothing that may include other badges or name tags. The patch is applied on the right front side of the garment approximately 1½" (3.8 cm) directly above the breast pocket or in a comparable position on garments without pockets. On a blazer (fig. e), the top edge of the patch aligns with the left breast pocket.

A few specific color recommendations are made for NASA uniforms: royal blue for flight suits; white for lab coats, hardhats, and helmets. A 7" wide (17.8 cm) logotype may be embroidered in NASA Red centered on the back of a white lab coat (fig. d). On a white hardhat or helmet, a 5" wide (12.7 cm) NASA Red decal of the logotype may be centered on the front (fig. g).

To distinguish emergency/security personnel (security guards, firemen, etc.) a distinctive NASA Red patch with a white border, white logotype and the installation identification in black is available. The name of the emergency/security service (i.e. Fire Department) appears in white centered within a smaller black patch that is positioned ¾" (.9 cm) under the red patch. This configuration is worn on both shoulders of the uniform, on both shirts (fig. f) and outer-jackets. A light blue shirt and hat with dark blue trousers or skirt is recommended.

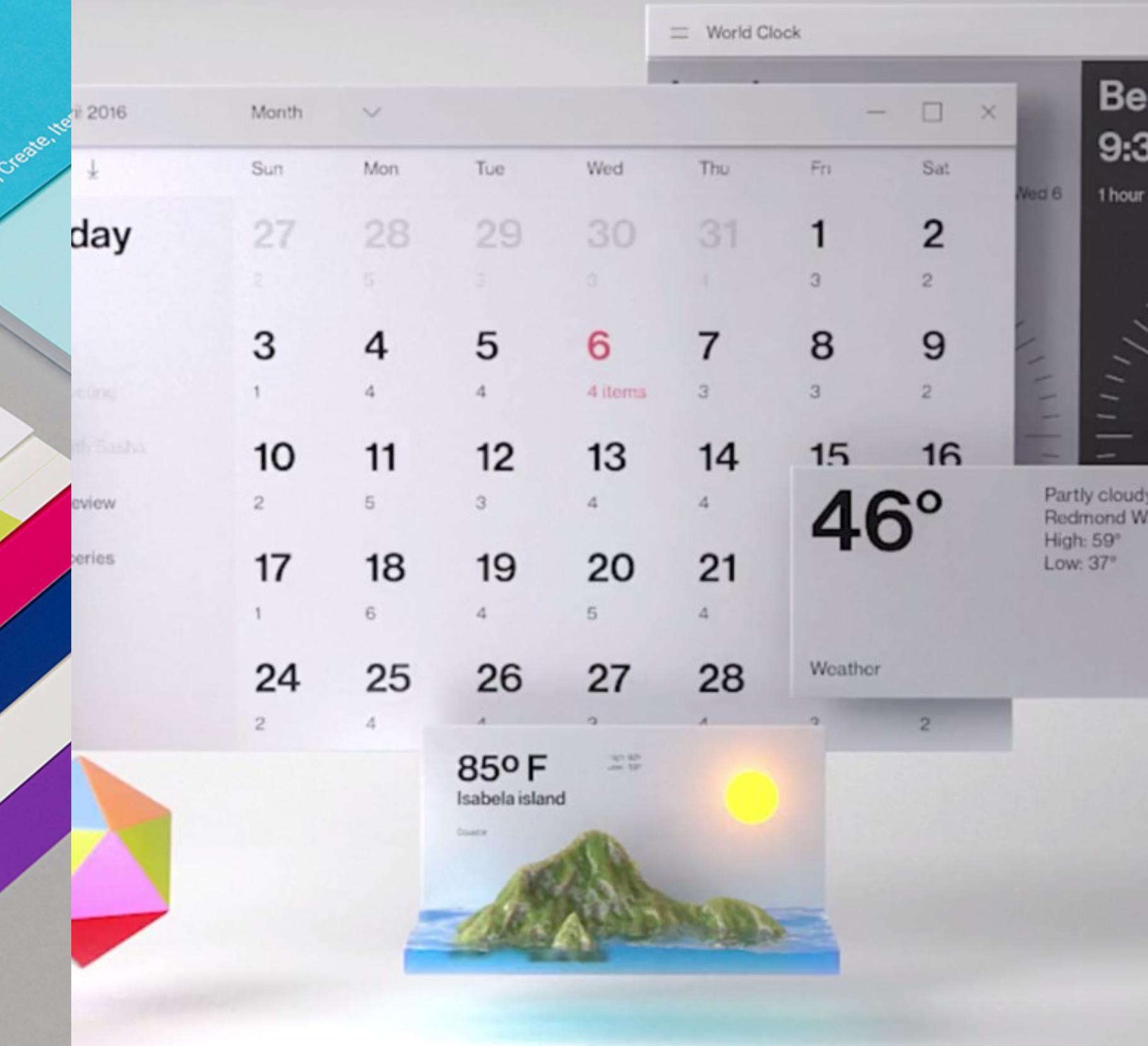


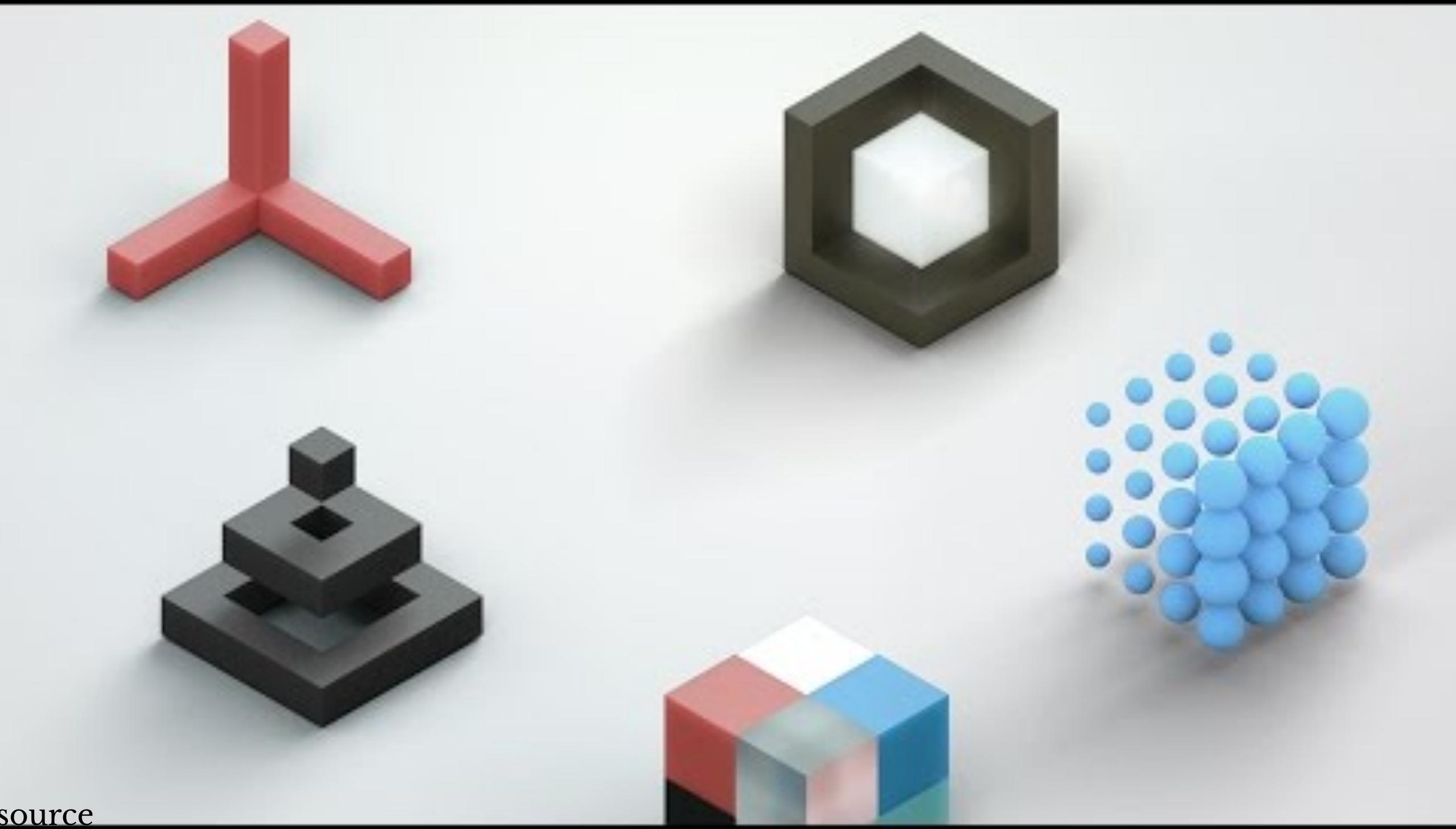
# Source<sup>3435</sup>



<sup>34</sup> Left: Google Material Design

<sup>35</sup> Right: Microsoft Fluent Design System





source

# Commonly Used Design Style Guides<sup>20</sup>

- Material Design
- Fluent Design System
- Materialize
- Ant Design
- Grommet
- Flat Remix



<sup>20</sup> Image source

# Case Studies of Design Language Use

- Material studies examples
- Fluent design case studies

# What did we learn today?

- Design paradigms
- Design patterns
- Design languages