Comp 388/441 - Human-Computer Interface Design

Week 11 - 26th March 2015

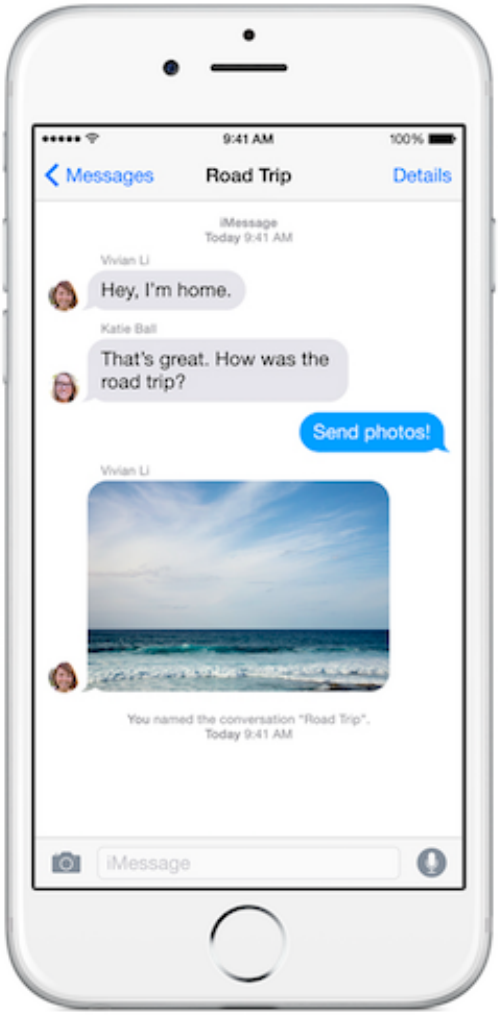
Dr Nick Hayward

Principles for usability - 7

Constraints

- apps and interfaces need to be designed and tested to prevent invalid states
    - *incorrect, invalid user interaction, invalid actions...*
- constraints may take various forms
    - *check correct relationships between elements and actions*
    - *check elements active only as needed*
    - *actions only performed when default data etc available*
    - *menu items active relative to contextual requirements*
    - *physical products often display such constraints*

Principles for usability - Messages on iOS 8



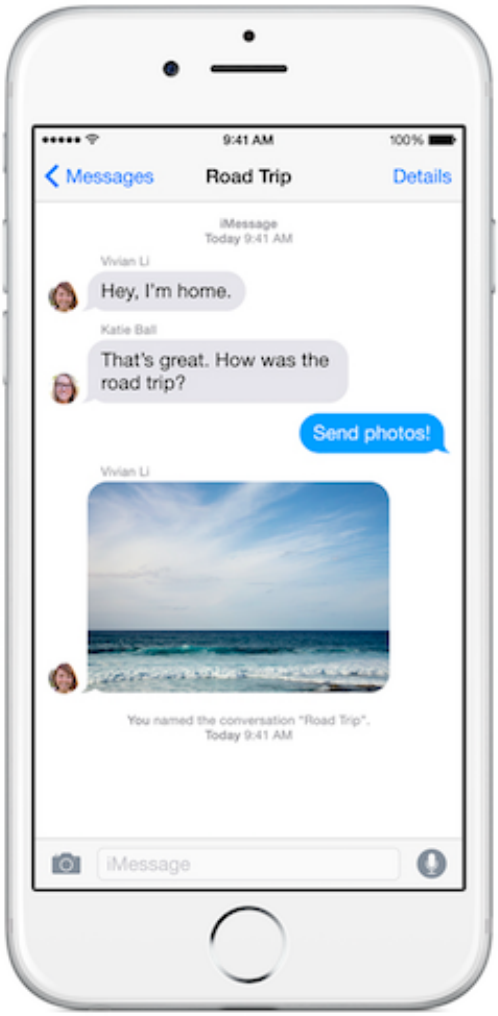
Source - Apple

Principles for usability - 8

Another consideration - naming

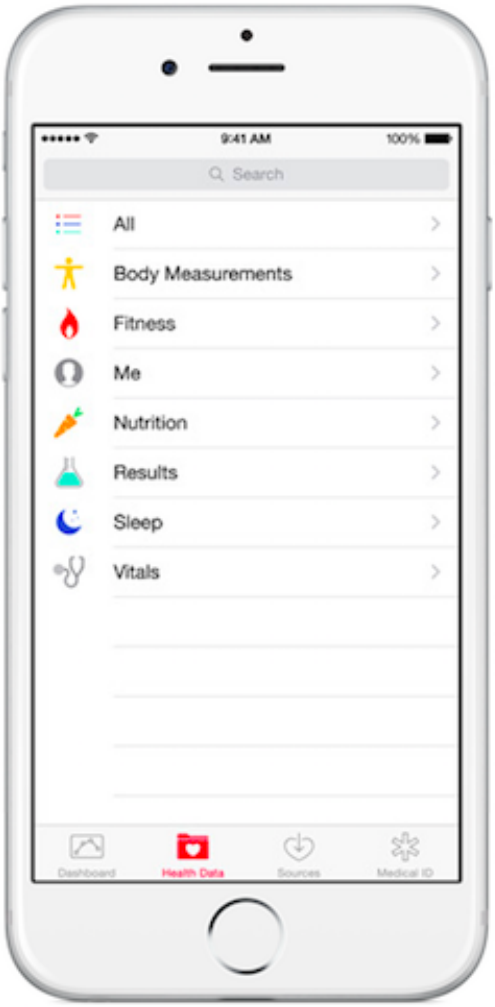
- names and labels key aspect of human communication, thought, understanding...
    - *also an important consideration in design*
- naming helps users understand the application
    - *their current location relative to navigation*
    - *the data and information they are viewing*
    - *action they can and cannot perform...*
- good naming helps a user form a correct mental model
- do not confuse naming with the use of technical jargon and terms
- precise, consistent naming helps us form unambiguous instructions, help, feedback...
- naming helps identify as well as differentiate between aspects of the design and functionality
- names should be unique relative to the context and the application
- namespaces are useful relative to application design and development

Principles for usability - Good(ish) naming (Messages for iOS 8)



Source - Apple

Principles for usability - Bad(ish) naming (Health for iOS 8)



Source - Apple

Principles for usability - 9

A few guidelines and thoughts on naming...

- does the name accurately reflect and describe its intended target?
    - *consider the action of the element relative to the name*
- is the name clear, concise, and free of ambiguity?
- use concise, easy to remember names
    - *better than longer, hard to remember descriptions*
- does the name inherently assume prior knowledge from the user?
    - *consider naming relative to perceived domain knowledge*
- acronyms are useful, but assume prior knowledge of the domain
    - *be careful when using acronyms, and consider cultural bias*
        - eg: VAT well known in Europe
- carefully consider capitalisation, and ensure consistency for chosen pattern
    - *eg: This Is Capitalised...This is Capitalised...This is not Capitalised (fully)...*
- users should be able to pronounce a name...not helpful if they have to check first

Principles for usability - Cultural naming concerns

Calpis Water and Pocari Sweat

	
Calpis Water	Pocari Sweat

Source - Calpis & Pocari Sweat

Principles for usability - Bad naming and icon

Foursquare icon

	
iOS 6 icon	iOS 8 icon

Source - Foursquare

Positive user experience

- we need to be able to identify traits of a positive user experience
    - *conversely, understanding a negative experience is also helpful*
- application allows a user to feel they are in control
- helps develop a sense of confidence and competence with the application
- helps encourage high productivity and efficiency
    - *enables and encourages our user to develop a sense of* ***flow***
- allows simple, routine tasks to be completed as quickly and easily as possible
- produces valid, useful output for the user
- user feels confident with the validity of produced results, calculations...
- considered aesthetically pleasing
- exhibits acceptable, sufficient performance to avoid unnecessary delays and waiting
- stable and reliable for the user...no *blue screen of death*
- makes it easy for a user to correct or modify any errors, mistakes...
- inspires trust and confidence in the user with logical, well-ordered design, navigation...

Negative user experience

- application leaves a user with a sense of feeling a lack of control
- overwhelming the user, creating a sense of incompetence and inadequate ability
- hinders the user from improving productivity and general efficiency
    - *prevents a sense of **flow***
- simple tasks and routine patterns prove overly complicated for the user
- output from the application is flawed, incorrect, poorly formatted...
- the app may produce unreliable results and calculations
- the UI design is aesthetically dis-organised, cluttered, unappealing...
- slow in performing tasks, and exhibits unnecessary delays and lags in performance
- unstable, buggy, and prone to crashing...
    - *user loses data due to poor performance*
- **excessive complexity** and difficulty in general functionality
- **too much work** involved to use the application in general
- design that conflicts with a user's perception of previous applications, iterations of a design, and competing products

Violating design principles

- issues that arise in usability
  - *consequence of poor interpretation, implementation, or misunderstanding general design principles*
- reconsider Norman's design principles
  - ***lack of consistency***
  - ***poor visibility***
  - ***poor affordance***
  - ***poor mapping***
  - ***insufficient feedback***
  - ***lack of constraints***

Designing an interaction concept - I

- app's **interaction concept**
  - *basic summary of our base, fundamental idea of how the user interface will actually work*
  - *describes presentation of the UI to the user*
  - *general interaction concepts that allow a user to complete tasks*
- inherent benefit is that it will often highlight initial usability issues
  - *including navigation, workflow, and other carefully considered and planned interactions*
- every aspect cannot be defined and outlined at the initial design stage
- follow a more agile approach instead of formal specification documents
- prototyping a particularly effective method for
  - *testing different design ideas*
  - *receiving feedback through peer reviews and associated usability testing*
  - *representing and communicating intended design to a client etc*
- lightweight written records as supplemental and supporting material

Designing an interaction concept - 2

Analysis of interaction concepts

- interaction styles
- information architecture basics, which often include the following
    - *a data model*
    - *a naming scheme, or defined glossary of preferred names and labels*
    - *a navigation scheme*
    - *a search and indexing scheme*
- an outline of a framework for interactions and workflow
- an outlined concept for transactions and any necessary persistency
- AND, a framework for the general visual design of the application

Designing an interaction style

- app's **interaction style**
  - *fundamental way it presents itself to a user to allow interaction with available functionality*
  - *many different concepts for interaction styles and overlap*
  - *many will employ a variety or combination of these interaction styles*
- an application might present the following styles to its users
  - ***menu driven options*** *- user is able to select options from menus, sub-menus*
  - ***forms*** *- user able to enter data, respond to queries by completing forms*
  - ***control panel options*** *- may show data visualisations, summaries, quick access options*
  - ***command line*** *- allows expert, power users to control the app using commands and queries*
  - ***conversational input*** *- user may interact in a back-and-forth dialogue or conversational style*
    - a sense of question asked and reply returned
  - ***direct manipulation*** *- direct user manipulation of objects within the app on the screen*
  - ***consumption of content*** *- app is simply a way to consume content*
    - eg: e-Book readers, music and video players...
- an app will normally use a combination of the above interaction styles

Designing an interaction style - mobile considerations

Source - Apple iPhone 6

Designing the information architecture - 1

- concerned with the organisation of information into a perceived coherent structure
- structure is considered comprehensive, navigable, and in many situations searchable
    - *eg: concepts, entitites, relationships, functionality, events, content...*
- designing such information architecture requires the following considerations and implementation
    - *data model*
    - *naming scheme or glossary*
    - *names and titles for identification of places*
    - *navigation and location awareness*
    - *navigation map and associated mechanisms*
    - *breadcrumbs and navigation notifications*
    - *presentation of such places*
    - *searching*

Designing the information architecture - visualisations...

Source - Apple Health

Designing the information architecture - 2

Data model, naming scheme, naming places...

- identification and recording of the entities, attributes, and operations for each entity
- also includes identification of the relationships between the entities
- often argued that the data model is, in fact, part of the app's interaction concept
    - *perceived to help define the nature of the product*
- coherent and consistent naming scheme is important to aid user's mental model
- definition of official names for an app's key elements and processes
    - *can be formalised and recorded in the defined interaction concept*
- apps with specialised domains may require a glossary of names and labels
    - *helps define the official, preferred terminology*
    - *interaction concept may then link or reference this glossary*
- places within an app should be clearly named and labelled
    - *helps users determine what they are viewing and where in the app*
    - *helps users differentiate places and concepts within an app*
    - *clear naming of places helps define them in menus, instructions, help text...*
- user-defined place names are OK as well
    - *eg: a title of a document in an editing app*

Designing the information architecture - personal naming schemes

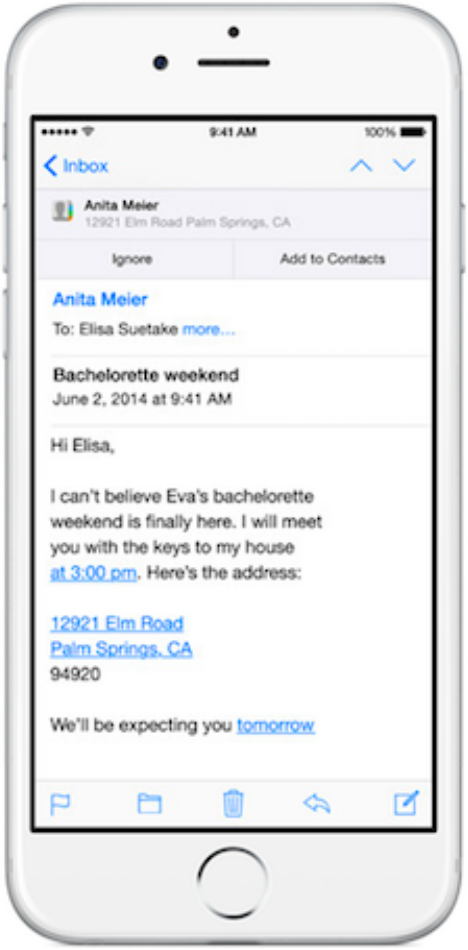
Source - Apple Photos

Designing the information architecture - 3

Navigation and places

- app design often references navigation relative to defined places
  - *eg: in a web app places may be defined as pages or screens*
- not all places need to be user accessible
- places may also refer to sub-divisions such as panels, tabs, sub-sections...
  - *sub-sections may also include dialogs, image presentations etc*
- for apps with many places, a design should help users determine and differentiate
  - *where they are currently located within the app*
  - *where they can go next*
  - *how to easily get where they want to go*
- in addition to naming places, we need to consider their actual presentation as well
  - *how do we present different places to our users*
  - *view multiple places at once, or page/navigate through single places*
  - *can these places be resized, moved and rearranged, opened, closed, hidden, removed entirely...*
  - *can we relate content from one place to another*

Designing the information architecture - determing places



Source - Apple Mail

Designing the information architecture - 4

Navigation map

- allow us to consider and define the places that may exist within our application
    - *the movements allowed from one to the other*
- beneficial if represented in a graphical manner within quick reference diagrams
- designing a complete navigation map at the design stage may be impractical and counter-productive
    - *initial map can always be expanded and modified as we develop the application.*
- some instances where a navigation map is simply impractical
    - *eg: dynamic applications, such as catalogues, wikis, some games...*
    - *many different links, pathways, and related material a user may generate*

Designing the information architecture - 5

Navigation mechanisms

- many different ways for a user to switch places and content. A few defined examples include
    - ***bookmarks***
    - ***buttons***
    - ***events*** *- triggered by a user action or application process can show a notification or message window*
    - ***flow diagrams*** *- visualise steps and outcomes relative to the current complex process or workflow*
    - ***hierarchical structures*** *- eg: trees used to display hierarchical depth of data...*
    - ***history***
    - ***links***
    - ***maps*** *- data points represented geographically, or conceptual map of data, app domain...*
    - ***menus***
    - ***searching*** *- simple act of searching by keyword, selecting from a faceted list of terms...*
    - ***switching*** *- move between multiple places currently available within the UI*

Designing the information architecture - 6

User location

- clearly identify a user's current location
- acts as a quick reminder to the user
    - *also creates a familiar contextual placeholder within the app*
- indicate the user's current location in a number of different ways
    - *clearly display the title or name of the current place with any associated contextual name*
    - *highlight the current place name or title on a visual map or flow diagram*
    - *include a representation of location on a visual flow diagram for a process of series of tasks*
    - *locate a current place within a defined hierarchical structure*
        - such as a tree representation of the current document or data...
- breadcrumb trail useful for hierarchical data representations
    - *benefit of acting as both location indicator and simple form of navigation*

Designing the information architecture - user location

Source - Apple Keynote

Designing a common interaction framework - 1

Considerations

- identify core sets of features, tasks, actions, operations, and processes
- consider series of use cases that follow and share similar patterns of interaction
    - *editing application may allow user interaction with many disparate tools and actions*
        - common menu structure, tools...variance is the selected tool itself
        - interaction will be able to follow a similar pattern
    - *we can also see this type of example with games*
        - many different levels, challenges, opponents
        - similar interaction concepts from level to level
- create an initial list or breakdown of these similar tasks or features
    - *then start to design an interaction framework to describe perceived commonalities*
        - such as the presentation and behaviour of the user interface
    - *this list allows us to*
        - understand how the application will fundamentally behave
        - ensure consistency across such similar tasks
            - allowing users to develop correct mental models
    - *by simply documenting the commonalities between such tasks*
        - saves us from re-documenting the same aspects for individual tasks for our overall specs
- framework also useful for the development of the overall design and its technical underpinnings

Designing a common interaction framework - 2

Issues

- how tasks are started or triggered
    - *eg: user selecting an item on a menu...*
- required authorisations
- when and how tasks can be activated and any given cases where tasks may be disabled
- how and when the task is considered complete
- does the start or end of a task signal a change in any status, mode etc...
- what are the effects of the task on the system's data
    - *eg: is data saved automatically, does it persist or is it temporary*
    - *what happens if the task is abandoned*
    - *what happens if an error breaks the task...*

Designing a common interaction framework - 3

Data and persistency - part 1

- need to consider data transactions and persistency in an application
    - *eg: what, if any, of the application's data needs to saved or stored...*
- for the interface and interaction concepts
    - *consider how the actual saving of data works in the application*
    - *is the data generated by user interactions saved in a persistent store?*
    - *is the data saved in a temporary memory cache?*
    - *consider how such data saving and persistency is relayed to the user*
    - *are they aware that the data is being saved?*
    - *is it an explicit act in the interface design?*
    - *is it part of an auto-save option running as a background process?*

Designing a common interaction framework - 3

Data and persistency - part 2

- consider standard data design patterns that include validations of the data
    - *also consider accompanying error and notification messages*
- for the interface and interaction designs
    - *carefully plan how error messages are presented*
    - *whether the validation occurs on the client or server side*
- consider whether partial data for incomplete tasks is saved
- in the interface design, clearly identify potential *save points*
    - *helps correct notification to the user*
    - *we can also offer suggestions, reminders, completion estimates...*
    - *save points allow us to track current data*
    - *has it been saved recently?*
    - *is it a version or a re-write of saved data...*
        - is it a persistent save or cached?

Design and specification - I

- consider various techniques for designing and specifying user aspects of application's design
- how can we communicate options, choices, design concepts...
- prototypes and mockups act as an important part of this process
    - *may include highly detailed visual representations or low fidelity examples*
    - *choice often reflects development priorities and application complexity*
    - *iterative nature of prototypes may aso be useful for more complex development*
- designs and specifications relative to clear distinction between
    - *application's appearance*
    - *application's intended or expected behaviour*

Design and specification - 2

Application appearance

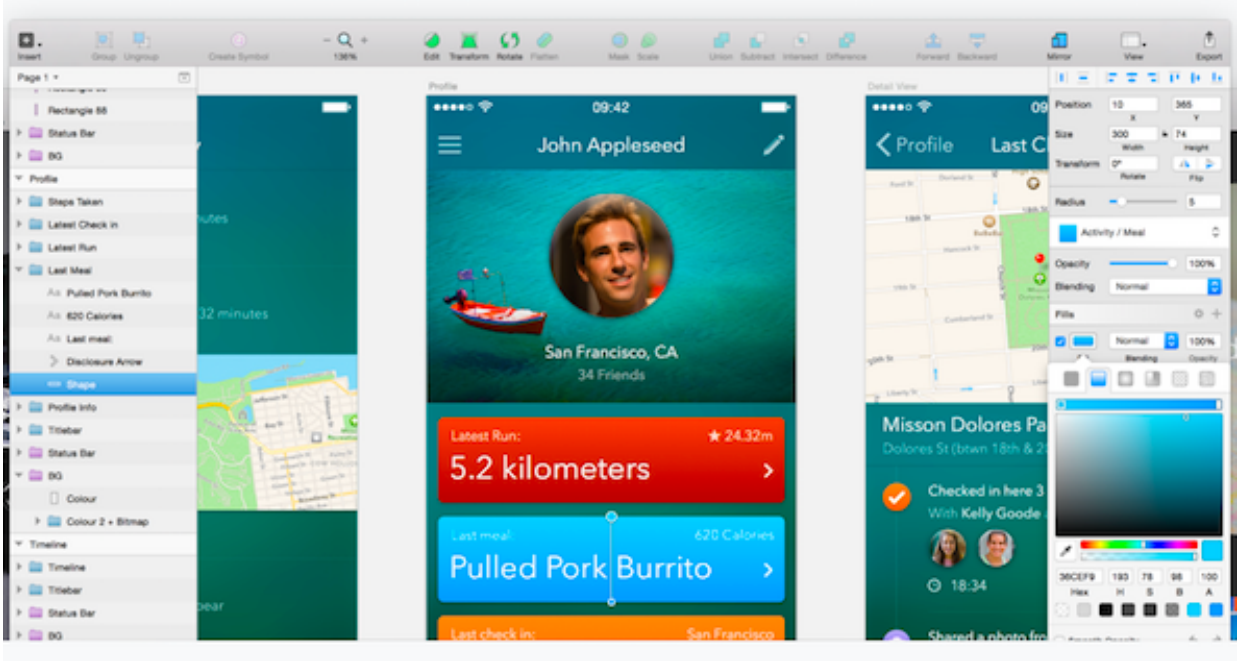
- prototype or mockup helps us plan and visualise an application's appearance and interface
  - *could be high fidelity or low fidelity*
  - *choice often reflects state of the application and intended purpose of the mockup or prototype*
  - *eg: sales/funding demo vs design for development*
  - *perceptual difference between mockup and prototype*
  - *static mockups do not specify behaviour*
    - rely upon additional interaction and behavioural specifications
  - *prototype designed to demonstrate an application's intended behaviour*
  - *prototype perceived as an interactive piece of software in its own right*
  - *not considered fully functional, finished product*
  - *may only represent small components of the application*
  - *intended to show sample scenarios, interactions...*

Design and specification - 3

Hi-Fi mockups

- intended to act as a realistic approximation of an application's design
- allows us to represent and visualise the appearance of the user interface
    - *often used for demonstration purposes, such as attracting funding, sales contracts...*
- allows us to test colour schemes, design layouts, patterns...
- hi-fi mockups normally designed as static images with no actual interaction
- Adobe's Photoshop, Illustrator, In-Design...often popular tools for creating such mockups
    - *offer detailed, relatively quick mockups to help visualise an application*
- HTML, CSS...also popular options for creating quick, hi-fi mockups
    - *can be used for a variety of application mockups*

Design and specification - Hi-Fi mockup



Source - Sketch

Design and specification - 4

Hi-Fi prototypes

- prototype intended to act as an interactive application
    - *not intended as fully functional application*
    - *a concise working simulation*
- prototype intended to create a rapid, working example of functional components of an app
- code often sufficient to simulate and replicate results for a given action and scenario
    - *often will not include a database or persistent data storage*
    - *may simply simulate and demonstrate action of saving the data*
- important to create a prototype of the interface and user interaction
    - *not backend logic and implementation*
- prototypes normally limited in their breadth and depth of functionality
    - *should not be shallow in its implementation*
    - *demonstrate and evaluate an app's specified details in depth*
    - *shows careful, well-planned concept and design for each aspect of your app*
- **NB:** high fidelity prototypes can be time consuming to produce correctly

Design and specification - Hi-Fi prototype

Framer

- many examples available at the Framer website
  - *OK Google*
  - *Android Lollipop*
  - *Carousel*
    - best demo at the moment...

Design and specification - 5

Low-Fi mockups and prototypes

- low-fi mockups often seen as a **rough sketch** or outline
- often referred to simply as **wireframes**
- their simplicity can offer an inherent utility and speed of creation
- not trying to recreate the exact look and feel of an app
- often more interested in layout of visual components and elements
    - *offers a quick reference point for further development*
- easily sketched on paper, or use formal tools such as
    - *Adobe's Photoshop, Illustrator...*
    - *The Gimp - an interesting open source alternative*
    - *could even use a simple tool like Google Drawings*
    - *many mobile drawing apps as well*
- inherent benefit of low-fi mockups is quick creation
    - *quick to modify and update*
- low-fi prototypes often seen as a series of linked low-fi mockups
    - *simple interaction leads to mockup sketches*
    - *again, not aiming for pixel accurate representations of app*

Design and specification - Low-Fi mockup

Source - Flinto

Design and specification - 6

Rapid prototyping

- provides quick examples of an application's design
  - *helps promote and encourage development and iterative design*
- iterative design helps encourage feedback early in the design process
  - *continues throughout the design process as well*
- we might consider the following as we develop our prototypes
  - *consider what needs to be prototyped early and often*
  - *how much do we actually need to prototype at each stage?*
    - consider the most common design elements and interaction
    - checking how something will work and not prototyping a full application
  - *work out how different places in the app are connected*
    - connection between interactions, places...
    - consider the patterns that exist within the app
    - example pathways for a user through the app to achieve a given goal
  - *choose your iterations for prototypes*
    - helps us avoid the temptation to prototype the whole application at once
  - *different fidelity for different iterative stages*
    - low-fi mockups for initial design layout and elements
    - low-fi prototypes for many initial interactions
    - hi-fi prototypes as we approach the final product

Design and specification - Tools

A few example tools for mockups and prototypes

- HTML, CSS, JavaScript, Bootstrap...
- Adobe Photoshop, Illustrator
- Sketch3
- Proto.io
- Flinto
- framer
- mirror.js (useful for Android)
- Google Drawings
- XCode Interface Builder
- Apple's Keynote (useful for iOS)