SWE 432 -Web Application Development

Spring 2023

George Mason University

Dr. Kevin Moran

Week 13: Think-aloud Usability Evaluations & Site Design

Administrivia

- Midterm Exam Grades Posted!!
- *HW Assignment 3* - Working on Grades and Comments now.
- *HW Assignment 4 -* Out now, Due next week (April 28th)!
  - Extra Credit Opportunity!

HW Assignment 4

Step 1: Sign up on GitHub Classroom to Clone the Starter Project

Please follow the instructions for setting up this homework assignment in GitHub Classroom and deployment of your project via Heroku. The starter project includes code for a React Front-End. You may reuse your code from HW2 or HW3 if you would like to use a backend for this assignment, however, this is not required.

Click Here to View HW 4 Tutorial (Coming Soon!)

Step 2: Choose an Idea for an Interactive App

In this assignment, you are free to choose any idea for an app you'd like as long as it involves user interactivity, where the application is taking input from the user (e.g., clicking on buttons, entering text), updating the state of the application, and rendering new visual content from the new state. You might create a simple game, such as a number guessing game or checkers. You might create a data management app to, for example, create and browse recipes or track expenses. You should pick something that is interesting and exciting to you.

HW Assignment 4

Step 3: Implement your Interactive App

You'll implement your app as a front-end React app. Your app should satisfy the following requirements:

Requirements

- **React**
  - Create at least 5 separate React components.
  - Use conditional rendering to conditionally render visual content
  - Include handlers in your React components for at least 5 events
  - Create at least two controlled components, where input from an HTML control is bidirectionally synchronized with state in a React component
  - Create a list of child elements or components with unique keys
- **CSS**
  - Create at least one cascading selector which overrides another selector
  - Use at least two pseudo-classes
  - Center at least one element inside its container
  - Use the z-index and absolute or fixed positioning to display an element stacked on top of another element
  - Create an least one animation using transition
  - Specify at least one fixed size and one relative size
  - Use display grid to create a layout with multiple rows and columns
  - It is optional to use any styling libraries like Bootstrap or Material-UI, however, you must manipulate CSS as required above (for example, customizing the library with your own CSS files, or inline by setting style within your React components).

HW Assignment 4 - Extra Credit

I will be offering Extra Credit on this assignment. This extra credit will be applied toward your midterm exam grade, with a maximum of 10 potential points to be earned (out of 200). This is essentially amounts to half a letter grade.

- **Bonus (Extra Credit)**
  - Create at least 5 jest tests to test the functionality of your app
  - You should use at least 3 tests that use the `jest-dom` package and at least 2 "normal" Jest tests

Class Overview

- ***Lecture:*** Think-Aloud Usability Evaluations
  - Quick Lecture
  - Usability Study Activity

Usability Studies

Iterative Model of User-Centered Design

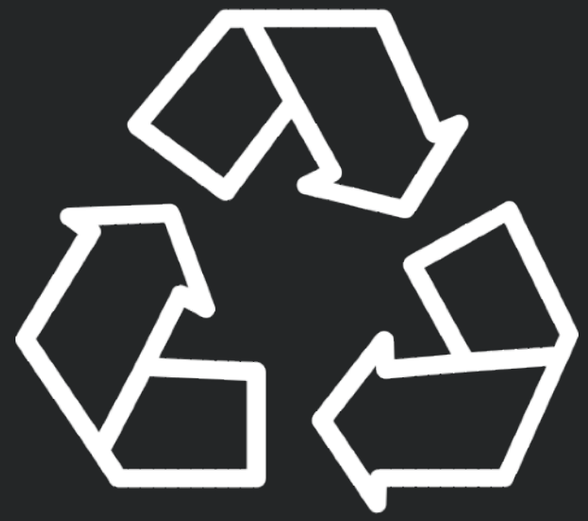
Observation

(Re)Define the Problem

Understand User Needs

Idea Generation

Brainstorm
what to build



Prototype

Build

Test

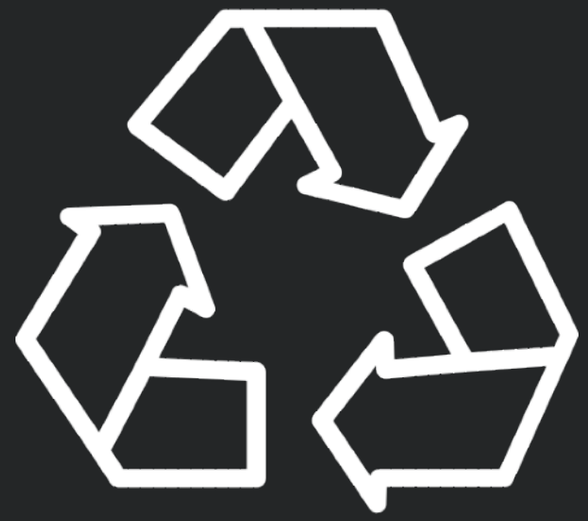
Evaluate what
you have built

Iterative Model of User-Centered Design

Observation

(Re)Define the Problem

Understand User Needs



Idea Generation

Brainstorm
what to build

Prototype

Build

Test

Evaluate what
you have built

Empirical

Usability Evaluation (Now)

Why Conduct Usability Studies?

- Evaluate interaction design with ***real*** empirical data, gathering ground truth of user performance
- Identify ***usability issues***

Think-aloud Usability Study

- Goal: observe users using app, identify usability issues
- Can use with
  - paper prototype
  - HTML prototype
  - Wizard of Oz study
  - actual app

Steps in a Usability Evaluation Study

- Formulate ***goals*** of study
- Design study protocol, tasks, materials, data collection, …
  - Pilot study design
- ***Conduct*** study
- ***Analyze*** data to assess task performance and identify usability issues

Formulate Study Goals

Study Goals

- Where are you in the design process? What feedback do you seek?
  - Exploring new design idea
  - Validating high-level approach
  - Identifying important usability issues
  - Evaluating a new feature just added or a particular corner case
  - Studying performance by specific users (e.g., expert users familiar with old version)
  - Comparing performance against competitors

Study Design

Selecting Participant Population

- Who will be the users?
- Goal: users representative of system's ***target users***
- Are there multiple ***classes*** of users (e.g., data analysts, site administrators)?
  - If so, which are appropriate given goals?
  - May choose several classes
- System ***novices*** or ***experts***?
- Might choose to include ***UX experts*** to help flag potential issues

Number of Participants

- More participants —> different participant interactions, more data
- Fewer participants —> faster, cheaper
- No right answer, as depends on potential diversity of interactions and users
- Nielsen & Morlich (1990) found that 80% of problems could be detected w/ ***4-5*** participants
  - Most serious usually detected with first few
  - Krug suggests 3

Informed Consent

- Important for participants to be told up front what they will do and provide affirmative consent
- Helps allay potential participant fears
- Make clear purpose of study
- Make clear that you are evaluating your design, **not** the user

Tasks

- What will users do?
- Goals for task design:
  - Provide specific goal: something that the user should accomplish
  - Comprehensive enough to exercise key features of your app
  - Short enough to minimize participant time commitments

Communicating Tasks

- Provide a scenario explaining the background of what users will be doing
- Provide a specific goal that the user should accomplish
  - But ***not*** how they should accomplish it
  - Don’t give away how you hope users will accomplish goal
- Communicate ***end criterion*** for task - how do they know they’re done?
- Provide maximum time limit after which they will be stopped

Recruiting Participants

- Many potential sources
  - Co-workers, colleagues, friends, family
  - Email, mailing lists, online forums
  - Announcement at related user groups
- Important to select sources that best match the background & knowledge of target users

Incentives for Participants

- Often (but not always) helpful to pay participants
- Most applicable when seeking participants with specialized expertise with whom you do not already have a personal or professional relationship
- Can also offer other incentives, such as gifts, coffee mugs, gift certificate; or free consulting, training, or software
- In some cases, just learning about future product can be incentive

Managing Participants

- Participants are valuable resource
  - Often finite resource
  - Think carefully about how participants will be used
  - Devise mechanisms for scheduling participants & reminders

Training

- Goal: ***avoid*** unless really necessary
- Training necessary when
  - Participants require specialized knowledge to act as target users
  - Target users will have access to specialized training materials before they begin study

Data Collection

- Think aloud
- Screencast
- Questionnaires interview questions to gather participant feedback

Questionnaires and Interviews

- Gather background or demographics about participants (if important)
- Supplement task performance data with subjective reactions
    - Perceptions of design, comments on potential issues, ideas for features
- Questionnaire - pre-defined questions, focused, less bias
- Interviews - more open ended, longer responses

Example Open-ended Questions

- What did you like best about the UI?
- What did you find most difficult or challenging?
- How might the UI better support what you're trying to do?

Piloting Study Design

- Dress rehearsal for conducting actual study
- Goals
  - Ensure software / prototype won’t “blow up”
  - Test tasks - ensure right length & difficulty
  - Test that materials are comprehensive and comprehensible
- As-needed piloting
  - Use first study session as pilot only if issues arise and must be addressed

Conducting the Study

Introduction (1)

- Greet participants, introduce yourself, thank them
- Build rapport, socialize
- Introduce them to the setup

Introduction (2)

- Give participant Informed Consent
- Answer any questions about study design
- Relieve anxiety and curiosity as much as possible
- Make clear evaluating design, not participant
- Let participants know you can’t answer questions about how to do task

Starting Session

- Give participants description of task
- Start any video recording
- Start encouraging participant to think aloud
- Begin observing participants work on task

Interactions During the Task

- Goal: listen, not talk
- Prompt participants to think aloud when necessary
  - e.g., What are you trying to do? What did you expect to happen?
- If show signs of stress / fatigue, let them take a break
- Keep participants at ease
  - If participants frustrated, reassure & calm participants
  - If so frustrated they want to quit, let them

Giving Help

- If participants totally off track, small reminder of goal might help
- Should ***not*** give participants information about how to complete the task
- What if user asks for help?
  - Direct them to think through it or work it out for themselves

Collecting Critical Incidents

- *Any action that does not lead to progress in performing the desired task*
- Often related to a gulf of execution or gulf of evaluation
- Generally does not include
  - accessing help
  - random acts of curiosity or exploration

Understanding a Critical Incident

- Important to understand in the moment what users goal is and what actions they are taking
- When a critical incident occurs, jot down
  - The time
  - What user was trying to do
  - What user did

Wrapping Up the Study Session

- Provide questionnaire (if applicable) / conduct interview (if applicable)
  - Probing into causes of behavior
- Answer any lingering questions the participant may have
- Thank the participant!!
- Provide any incentives (if applicable)

Reset Study Environment

- Make sure study environment is in the same state for all participants
  - Reset browser history / cache (if applicable)
  - Delete any user created content or materials

Analyzing Data

Critical Incident Analysis

- Identify critical incidents where something went wrong
- Easiest to catch in the moment - ***important to take good notes***
- Going back and looking at screencast can help you study context of issue in more detail

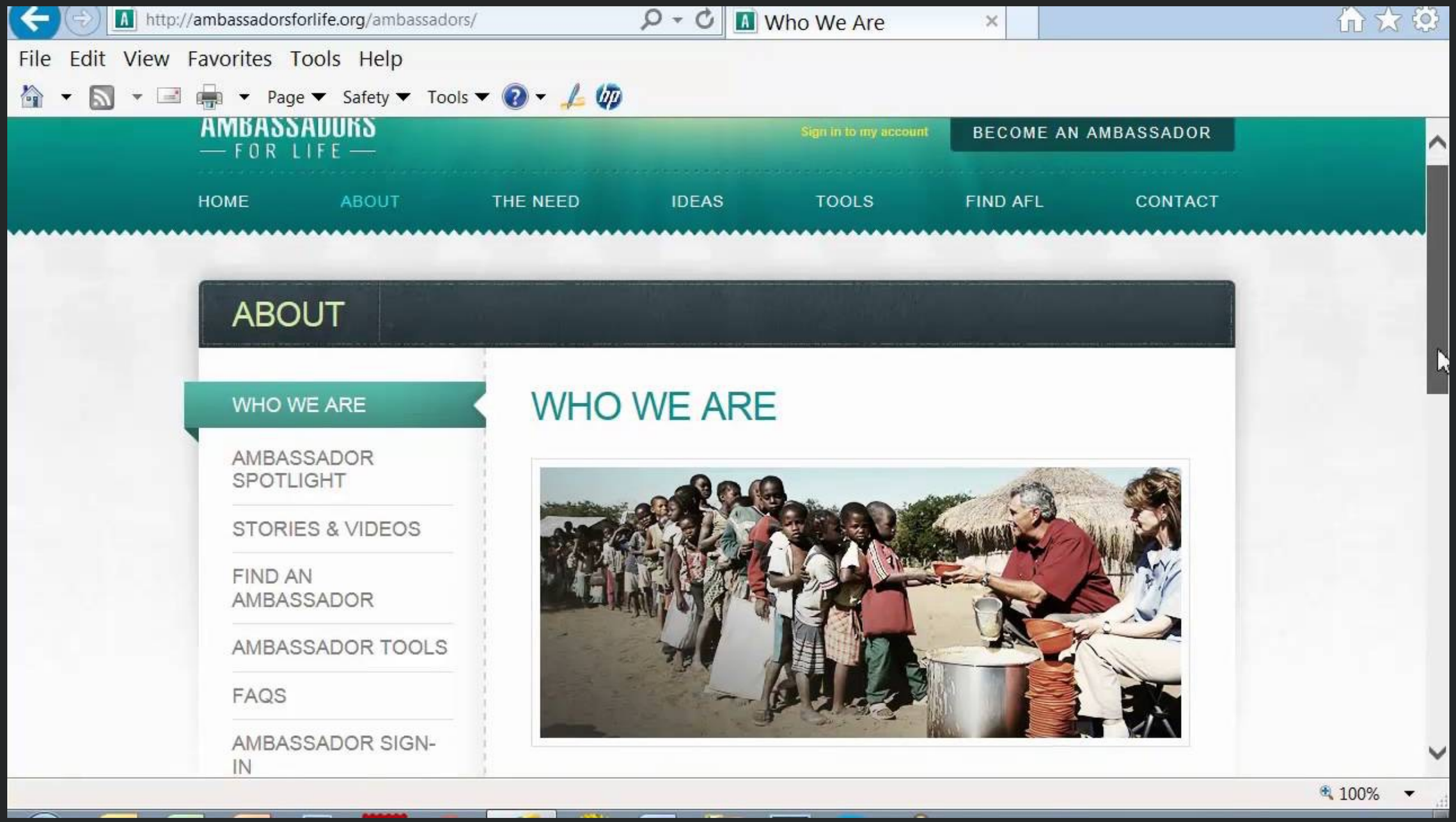
Reporting a Critical Incident

- Problem statement: summary of problem and effect on user (but not a solution!)
- User goals: what was user trying to do?
- Immediate intention: at the moment in time when problem occurred, what was the user trying to do
- Possible causes: speculate on what might have led user to take action they did

Critical Incidents → Usability Issues

- Group together similar incidents to form ***usability issue***
  - Match similar critical incidents within and across study sessions
  - Identify underlying cause
- Brainstorm potential fixes

Example of Thinking Aloud



In-Class Activity

Group Activity

- In groups of two
- Take turns conducting a usability study of an app of your choice
  - Try to think of a semi-difficult task that you might be able to improve
  - 5 mins to brainstorm 5-10 min task for each app
  - ~10 mins to conduct each study
  - Identify critical incidents (if any)

SWE 432 - Web Application Development

George Mason University

Instructor:
Dr. Kevin Moran

Teaching Assistant:
Oyindamola Oluyemo

Class will start in:

10:00

Overview of Site Design Principles

Exercise: How Should a Shopping App be Organized?

- Items organized into categories
- Shopping cart for collecting items you want to buy
- Secure way to enter payment information
- An easy way to search for items

Today

- What's a design space?
- How do you help users understand if it is possible to do what they'd like to do?
- How do you help users find what they're looking for?
- How do you balance tradeoffs between competing objectives in site design?

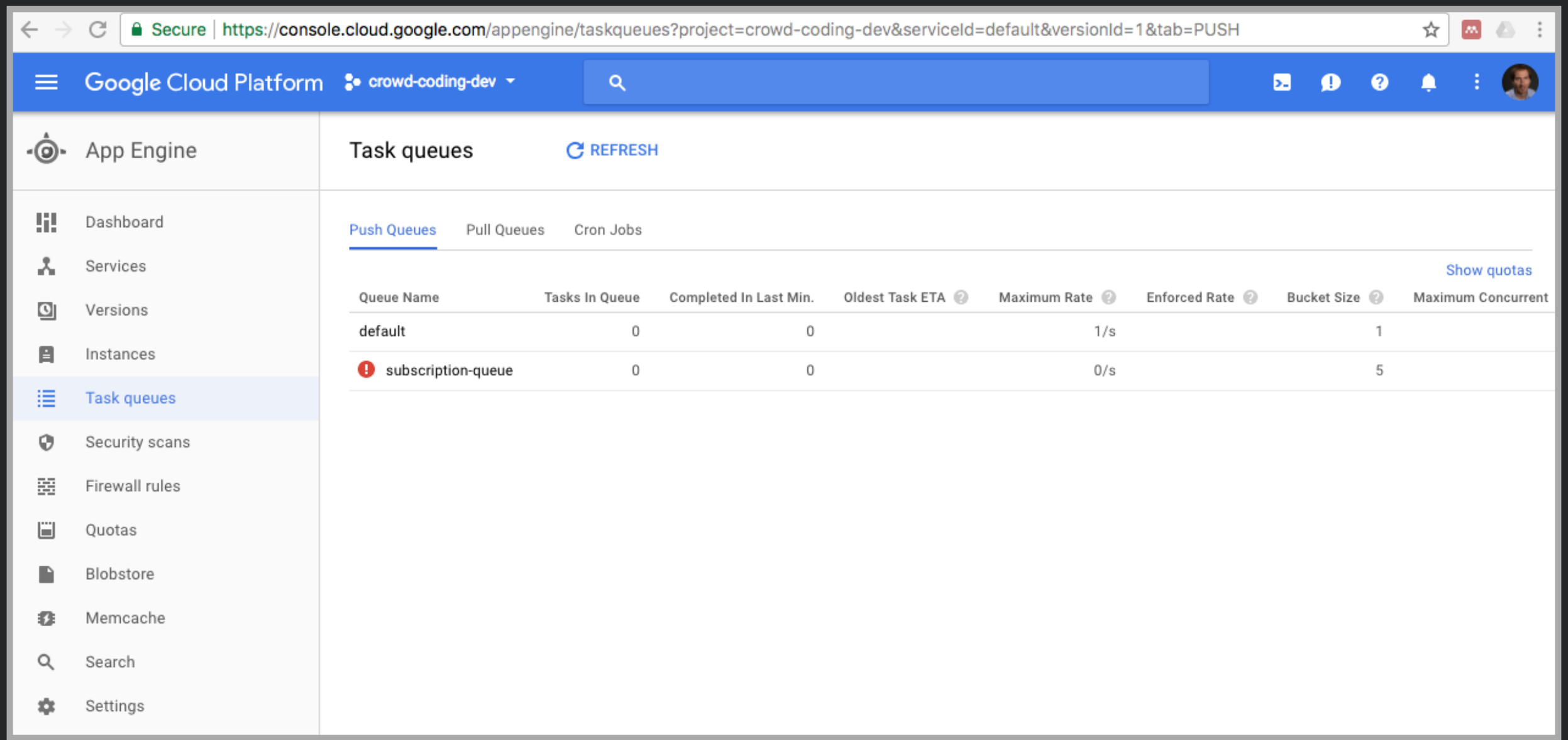
Design Space

- Space of **alternatives** that might potentially exist
  - All potential aspects of design (dimensions) that might vary
  - All potential choices for each design dimension
- Choosing a point in this space requires choosing ***design goals***
  - Thus far: task performance
  - Achieving this can often be decomposed into smaller design goals
    - e.g., minimize user errors, support more efficient navigation
  - And sometimes other design goals
    - Help users relax
    - Confuse users to teach them something
    - Encourage contributions to community
- Can use user-centered design to explore design space
  - Identify needs, sketch / prototype solution, evaluate
  - But large, so hard to enumerate every value for every variable

Interaction Techniques

- Way in which user interacts with user interface
- Examples
  - Search
  - Tabs
  - Progressive disclosure
  - Direct manipulation
- Represents a specific solution for a specific problem
  - May or may not be the best solution for a specific set of user needs and design goals
  - But helps reduce size and complexity of search space by offering standard choices

What can you do with this app?



Analogy: Buying a Chainsaw

- You walk in to a hardware store to buy a chainsaw. What do you do?

Challenges in Site Design

- Sometimes large space for users to navigate to find information.
- No spatial sense of scale. 50 pages? 500 pages? 50,000 pages?
- No sense of direction. Which way did I just go?
- No sense of location. No spatial anchoring of where I am now and how that relates to where I could go.
- No place to check if something is *not* present or supported.

Site Design

- Some key design dimensions
  - Organization of content into pages / screens
  - Organization of content within pages / screens
  - Ways in which users navigate between pages / screens

- Key design goals
  - Reduce the time / cost for users to reach content
  - Reduce the irrelevant information users must read

Planning

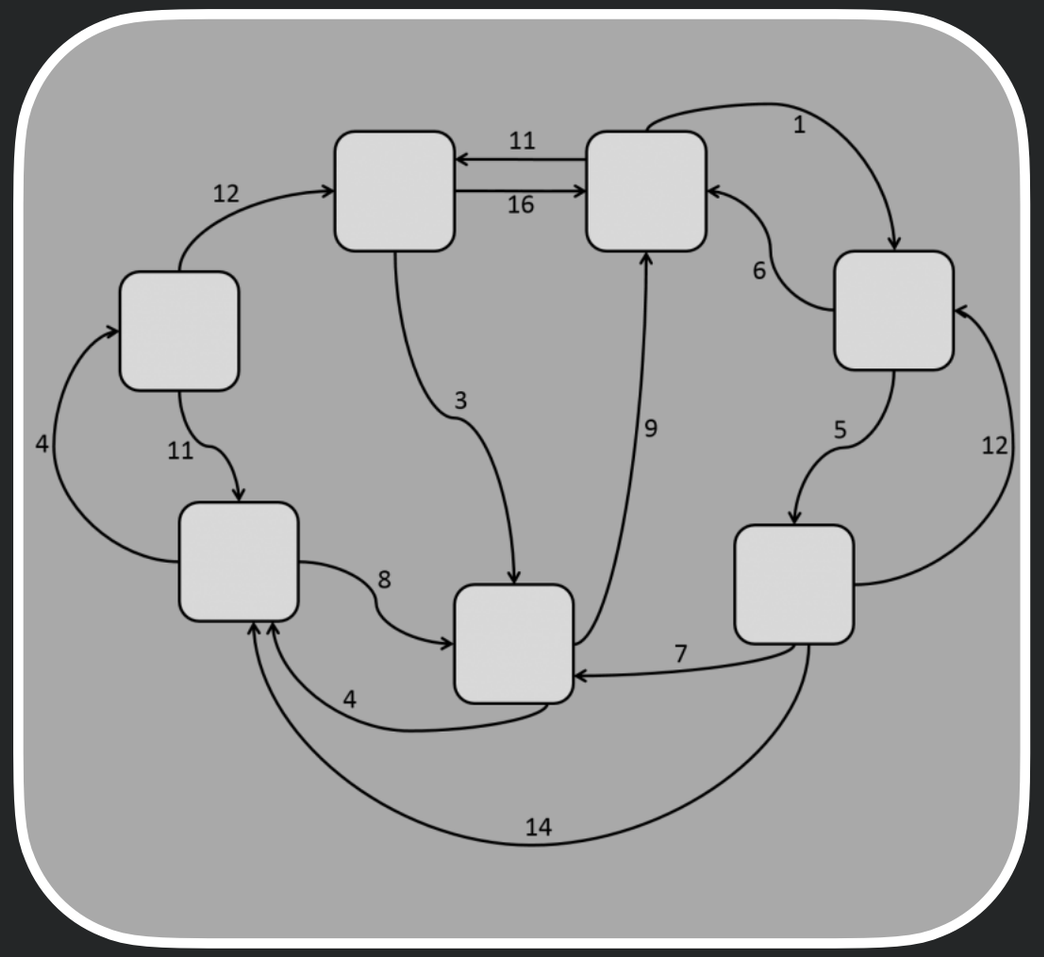
- Help users determine what they **can** do
  - Is this the right site for my goals? Is this the right page where I should spend my time?
- Support users in how they **determine** what to do
  - If this is the right place, how do I reach goal?

Information Foraging

- Mathematical model describing navigation
- Analogy: animals foraging for food
  - Can forage in different patches (locations)
  - Goal is to maximize chances of finding **prey** while minimizing time spent in hunt
- Information foraging: navigating through an information space (patches) in order to maximize chances of finding prey (information) in minimal time

Information environment

- Information environment represented as ***topology***
  - Information ***patches*** connected by traversable ***links***
- Examples
  - Web pages, connected by links
  - Menu options & dialogs connected by commands
  - Locations on map, connected by search, scroll, move interactions with map



Traversing Links

- Patch - a space in the environment where a user is located (e.g., a page, a dialog)
- Links - connection between patch offered by the information environment
- Cues - information features associated with outgoing links from patch
  - E.g., text label on a hyperlink
- User must choose which, of all possible links to traverse, has best chance of reaching prey

Scent

- User interprets cues on links by likelihood they will reach prey
- e.g., do I think that the “Advanced options" page is likely to have the option I’m looking for?

Simplified mathematical model

- Users make choices to maximize ***possibility*** of reaching prey per cost of interaction
- Predators (idealized) choice = max [V / C]
  - V - value of information gain, C - cost of interaction
- Don’t usually know ground truth, have to estimate
- Predator’s desired choice = max [E[V] / E[C]]

Design Implications of Information Foraging Theory

- Organize information into functionally ***related*** groups
  - If information required is already on same page, no need to go elsewhere
- Design effective ***cues***, helping users predict what will be found by traversing links
  - Better cues --> better ability to navigate to correct pages
- Match ***expectations*** of user's mental model
  - Cues are interpreted relative to mental model
- Provide ***search***
  - In large spaces, faster to search than traverse links

Search Increases Competition

- Users often enter sites through search engines, looking for site that will help accomplish goals

- Users form first impressions of sites rapidly

- Users will try another site if they perceive the value of continuing to forage in patch is low

Navigation

Common Navigation Usability Problems

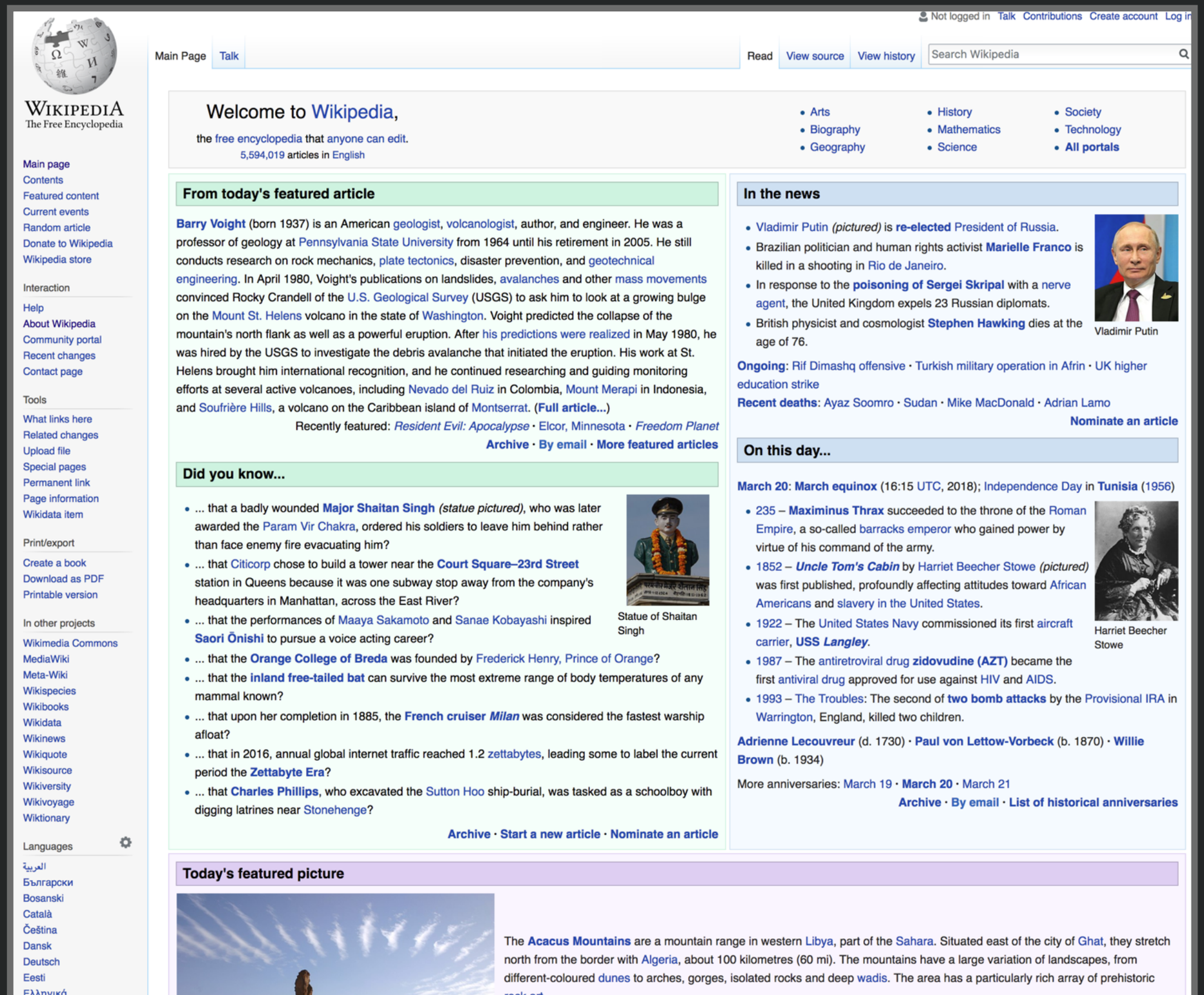
- User can’t find desired location
- User loses track of location
- User can’t remember information from another location

Hierarchy

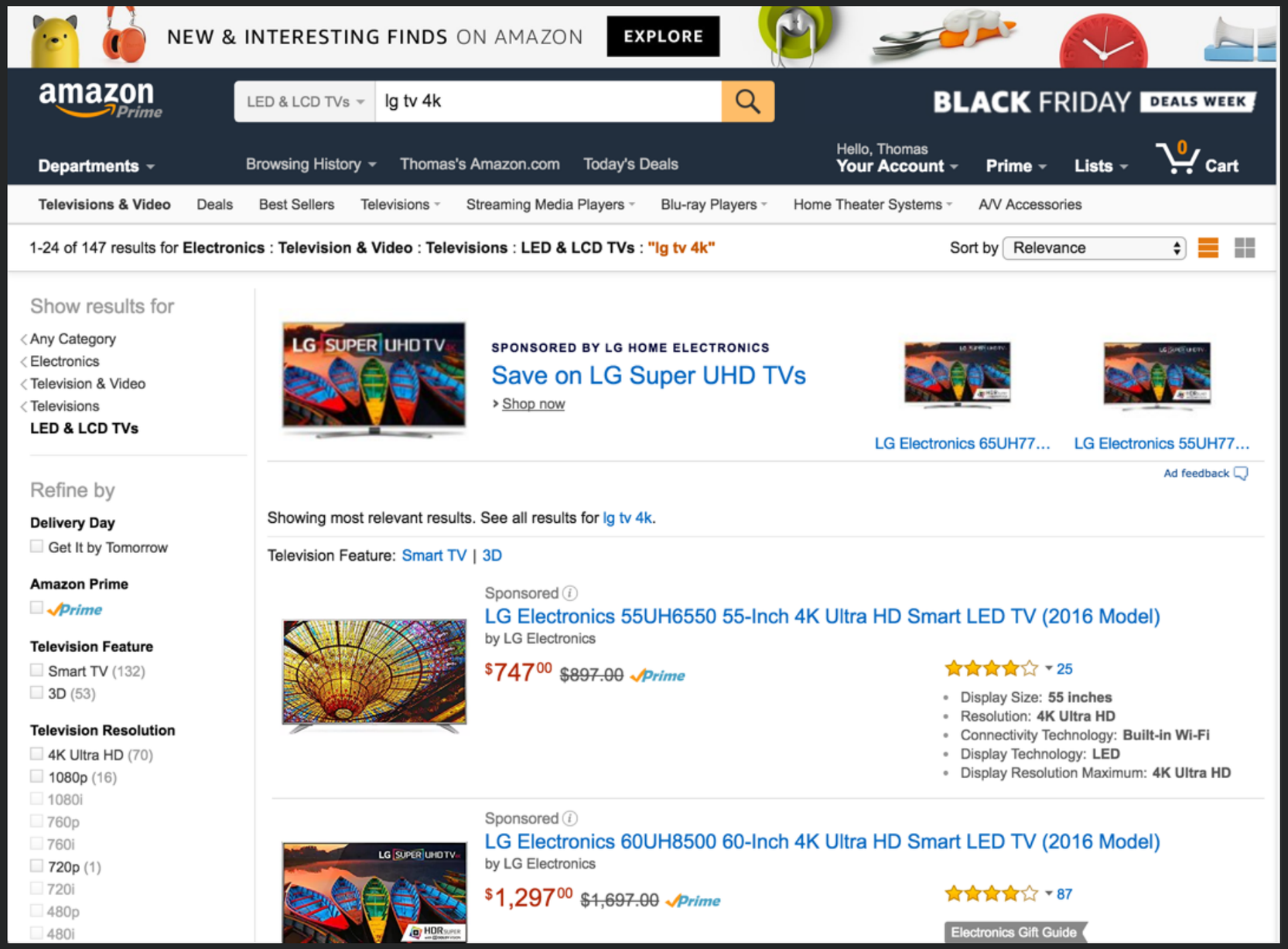
- Information in sites is hierarchical
  - Different pages at different levels of hierarchy
  - May be different navigation elements that lead into different subtrees

- Important to signal
  - what are hierarchies are present
  - which navigation elements are part of the same hierarchy
  - where the user currently is on each hierarchy

Example: Wikipedia



Web navigation conventions



…

Web navigation conventions

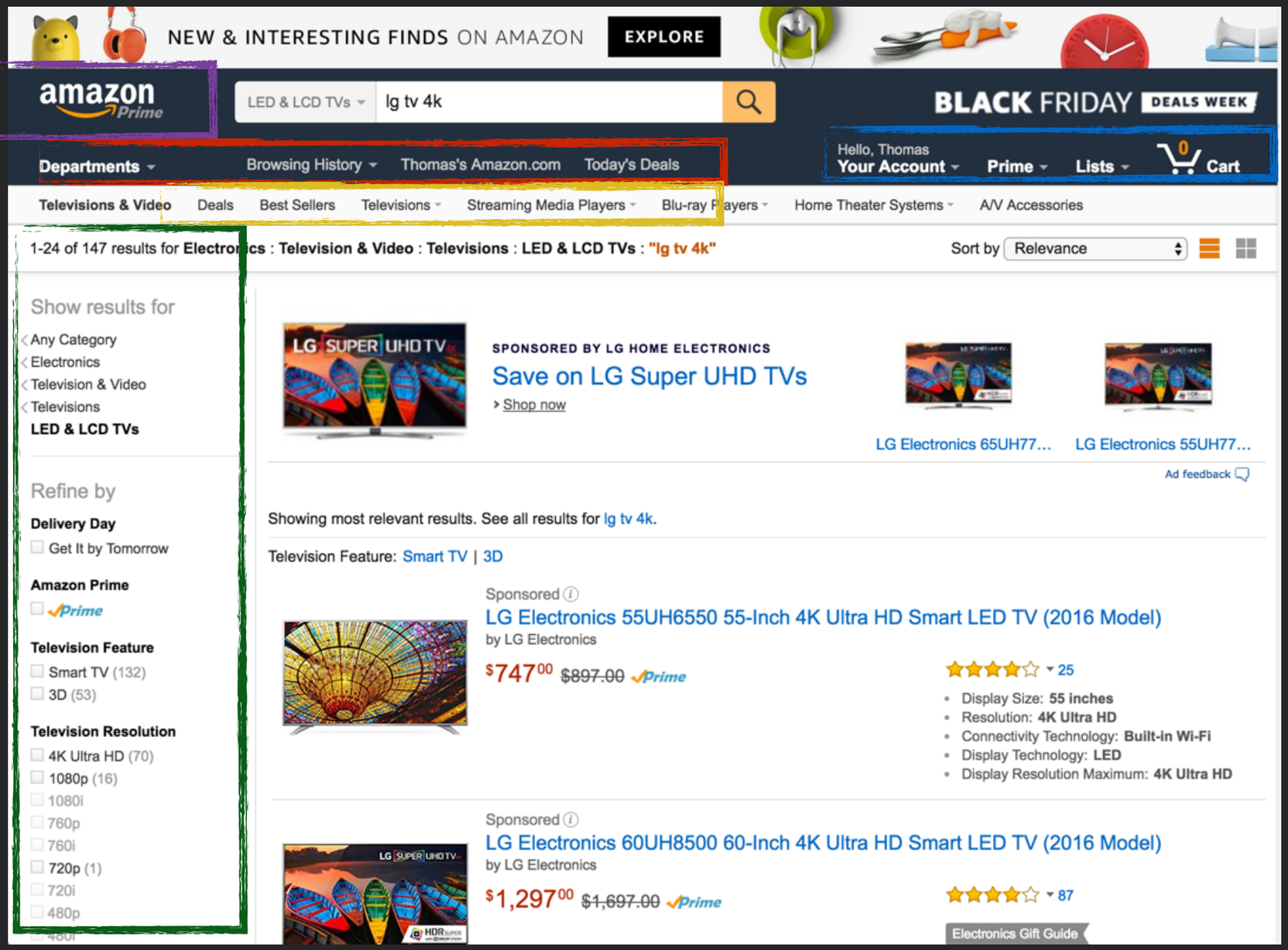
Site ID

You are here

Local navigation

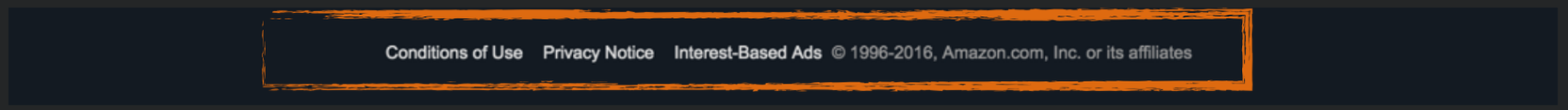
Utilities

Sections



…

Footer navigation



MS in Computer Science

Masters Students

- Advising
- FAQs
- Foundation Courses
- MS in Computer Science
- Core Courses
- Pre-approved Courses
- CS Course Descriptions
- Accelerated BS/MS Programs
- MS in Information Systems
- MS in Information Security and Assurance
- MS in Software Engineering
- Graduate Certificates

The MS in Computer Science prepares students for research and professional practice in computer science and related technologies. The program includes both fundamentals and advanced work in the areas of artificial intelligence and databases, programming languages and software engineering, systems and networks, theoretical computer science, and visual computing.

Degree Requirements

Students are required to complete 30 credits corresponding to 10 graduate courses. Courses are divided into **basic courses**, which have no graduate course prerequisite, and **advanced courses**, which have a graduate course as a prerequisite.

Courses are grouped in the following five broad areas:

- Artificial Intelligence and Databases
- Programming Languages and Software Engineering
- Systems and Networks
- Theoretical Computer Science
- Visual Computing

All the following requirements should be satisfied for the MS in CS degree:

- CS 583 - Analysis of Algorithms (from the Theoretical Computer Science area) and two additional core courses from two other areas must be successfully completed with a grade of B- or better.
- At least four courses (12 credits) must be chosen from the **advanced courses** in the list of preapproved courses from at least three different areas.
- At least six courses, including two advanced courses, must be designated CS.
- At least eight courses must be taken from the list of preapproved courses. Up to two computer science-related courses that are not on the list of preapproved courses may be taken with the approval of the Computer Science Department.

Project/Thesis (optional):

Three to six credit hours of the advanced classes may be replaced by a project or thesis. The project or thesis must be guided and approved by a committee of three appropriate faculty members and presented at an appropriate forum. The thesis must meet relevant university requirements.

For additional information on the degree requirements of the MS in CS:

- The MS CS section of the Mason Catalog is the **official source** for the degree requirements of the program.
- These slides from the orientation for new MS students provide an overview of the program, as well as additional useful information.

Academic Advising

A plan of study form for the MS degree should be completed and submitted by the student soon after admission to the program. This serves as a planning guide for the student. This plan should be kept up to date by regular consultation with the academic advisor. A final signed version of the plan must be included when the student submits a graduation application.

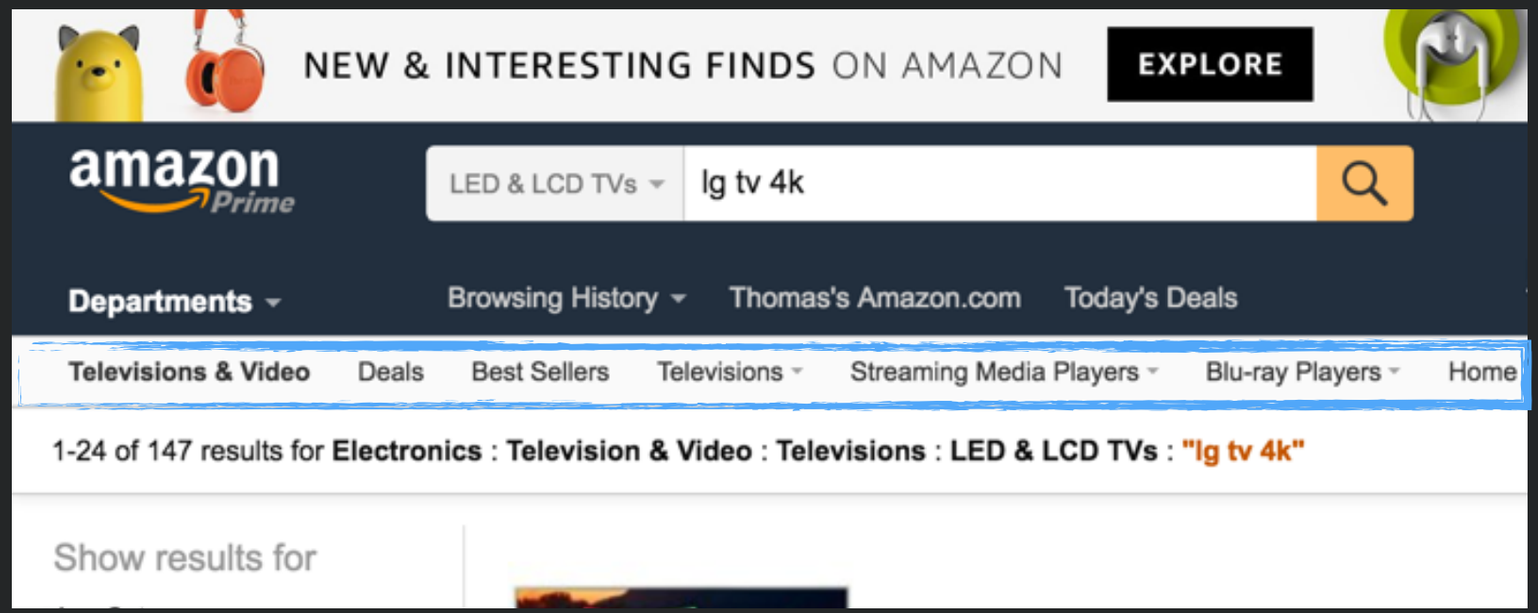
Plan of Study forms for all the MS degrees offered by the CS department are available at this web page.

For more information, please see the academic advising pages and the FAQ for Masters students.

Persistent Navigation

- Forms a common idiom users already understand
- Gives instant confirmation that still on the same site
- Supports consistency and standards
  - If *all* of your pages function same way, users know how to do actions & what to expect
  - Ok for specialized page like forms that are clearly different to not follow conventions.

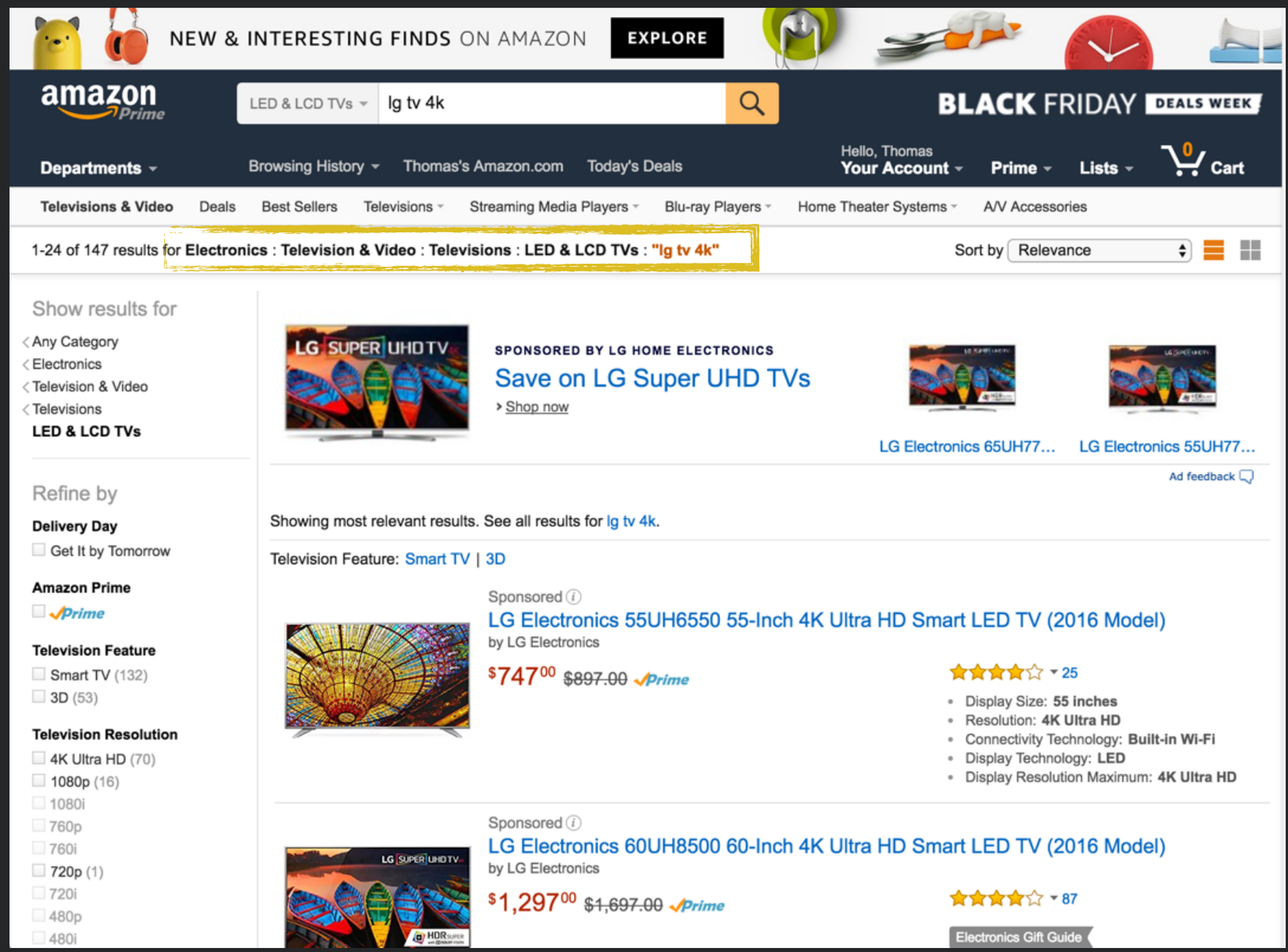
Tabs



- Example of a metaphor: tab dividers in a three ring binder or folders in a file drawer
- Partition into sections
- Advantages
  - Easily understood and self-evident
  - (Usually) hard to miss

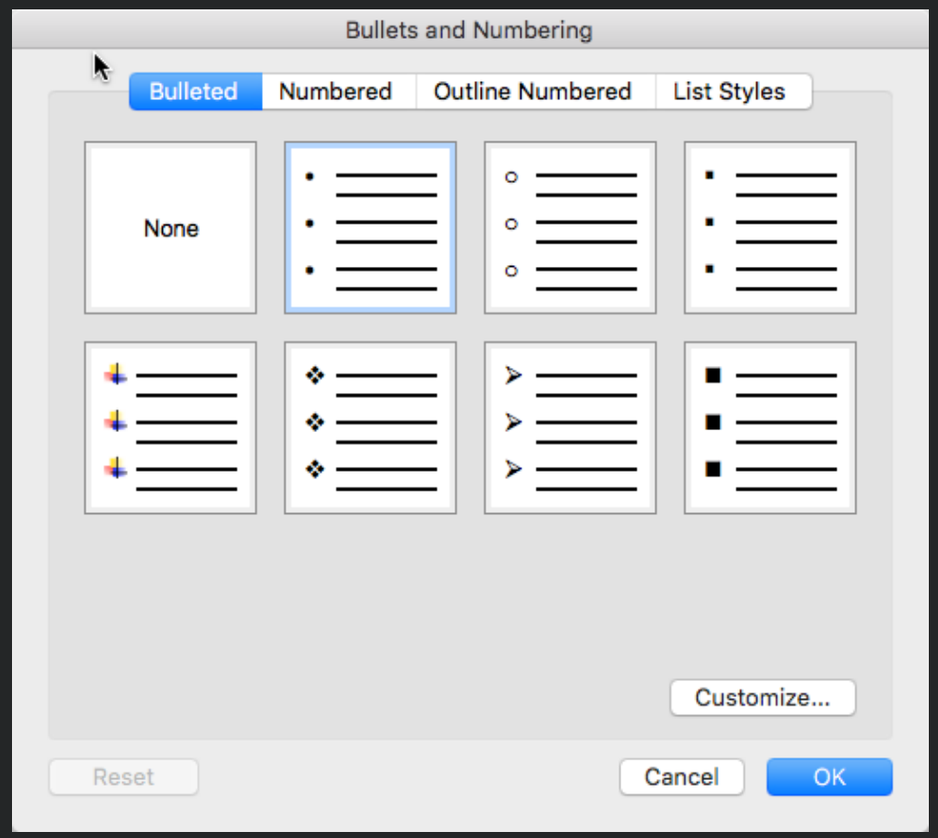
Breadcrumbs

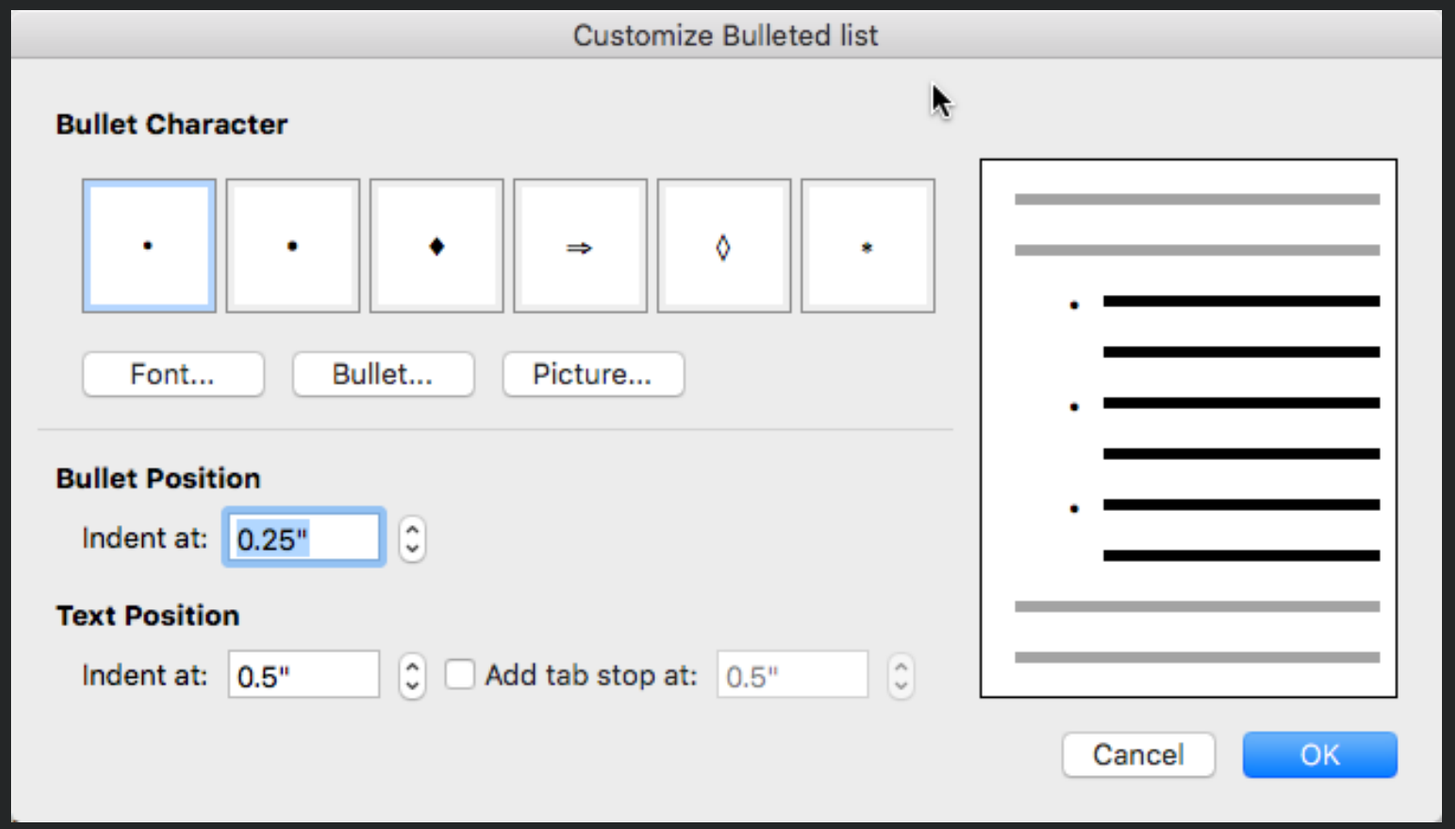
- Offer trail of where the user has been and how they got there
- Shows hierarchy of information space
- Shows current location



Progressive Disclosure

- a.k.a. details on demand
- Separate information & commands into layers
- Present most frequently used information & commands first





Effective Site Design

- Answers to the following should be obvious for a good site design
  - What site is this? (Site ID)
  - What page am I on? (Page name)
  - What are the major sections of this site? (Sections)
  - What are my options at this level? (Local navigation)
  - Where am I in the site? (“You are here” indicators)
  - How can I search?

Metaphors & Idioms

Metaphors

- One way to communicate what interface can do is through metaphors to the real world
- Uses existing mental models from the real world

Metaphors - Advantages

- Leverages understanding of familiar objects & their functions
  - File cabinets, desks, telephones
- Provides ***intuitive*** understanding of possible affordances & eases mapping tasks to actions
  - Open a folder, throw file in trash, momentum scrolling

Metaphors - Disadvantages

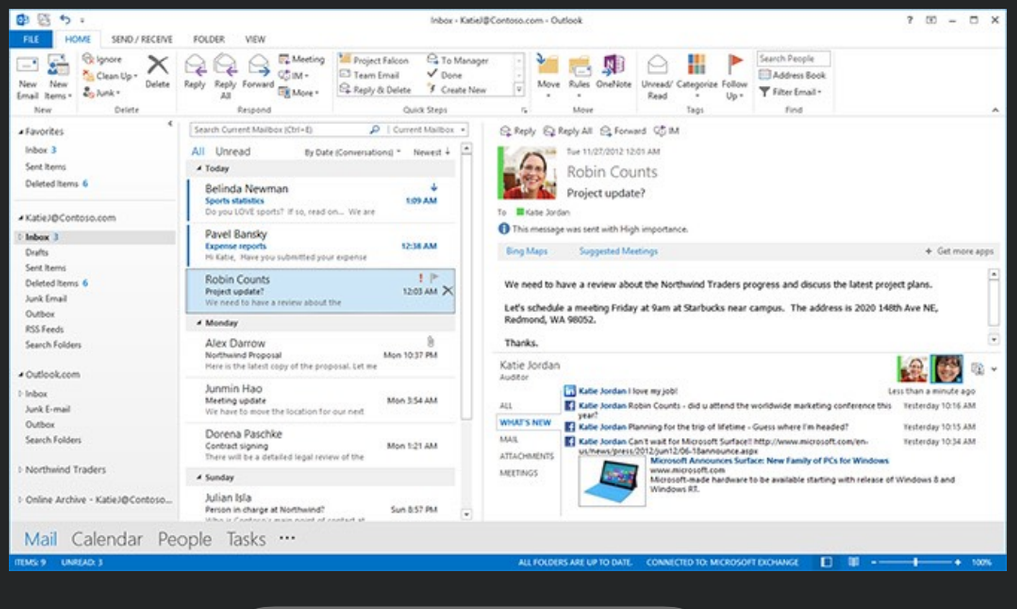
- Tyranny of metaphor: ties interactions closely to workings of physical world
- Adds useless overhead in extra steps, wastes visual bandwidth
- Taken literally, becomes non-sensical
  - e.g., nesting folders 10 levels deep

Alternative - Idioms

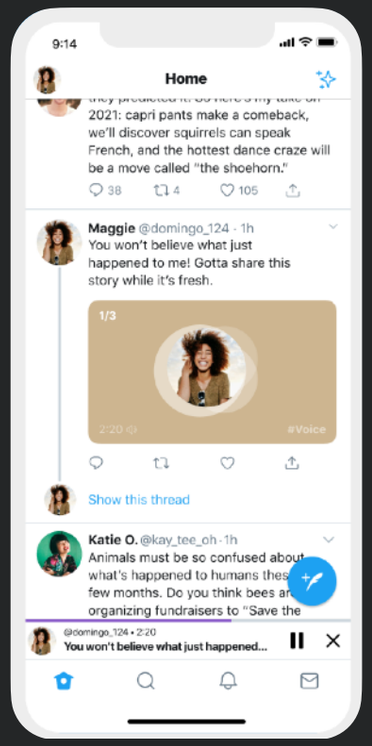
- A consistent mental model of how something works
  - e.g., Files: open / close / save / save as
- Offers intuitive understanding of affordances & interactions
- Provides consistent vocabulary for describing interactions
- Only have to learn it ***once***
- Might have originated in real world, but thought of in terms of mental model for UI interactions

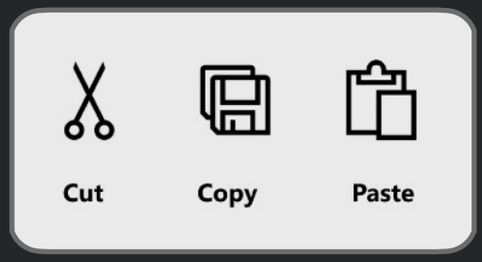
Examples of Idioms

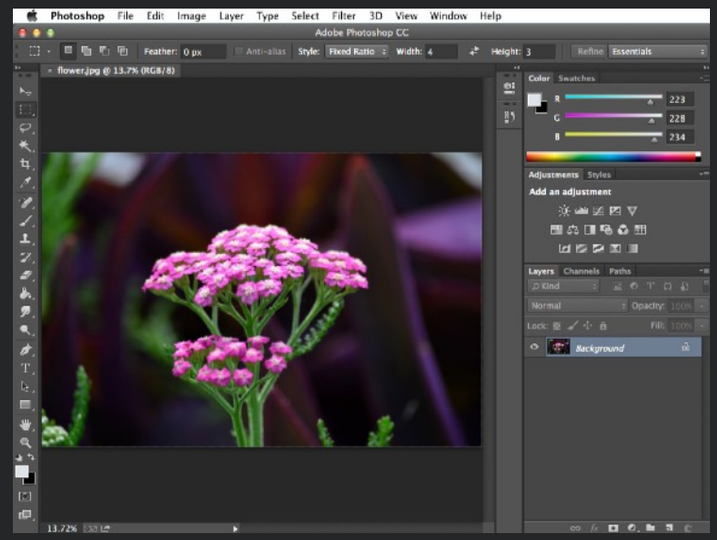
- Email
- Clipboard: cut / copy / paste
- Format painter



- Newsfeed
- Follow item







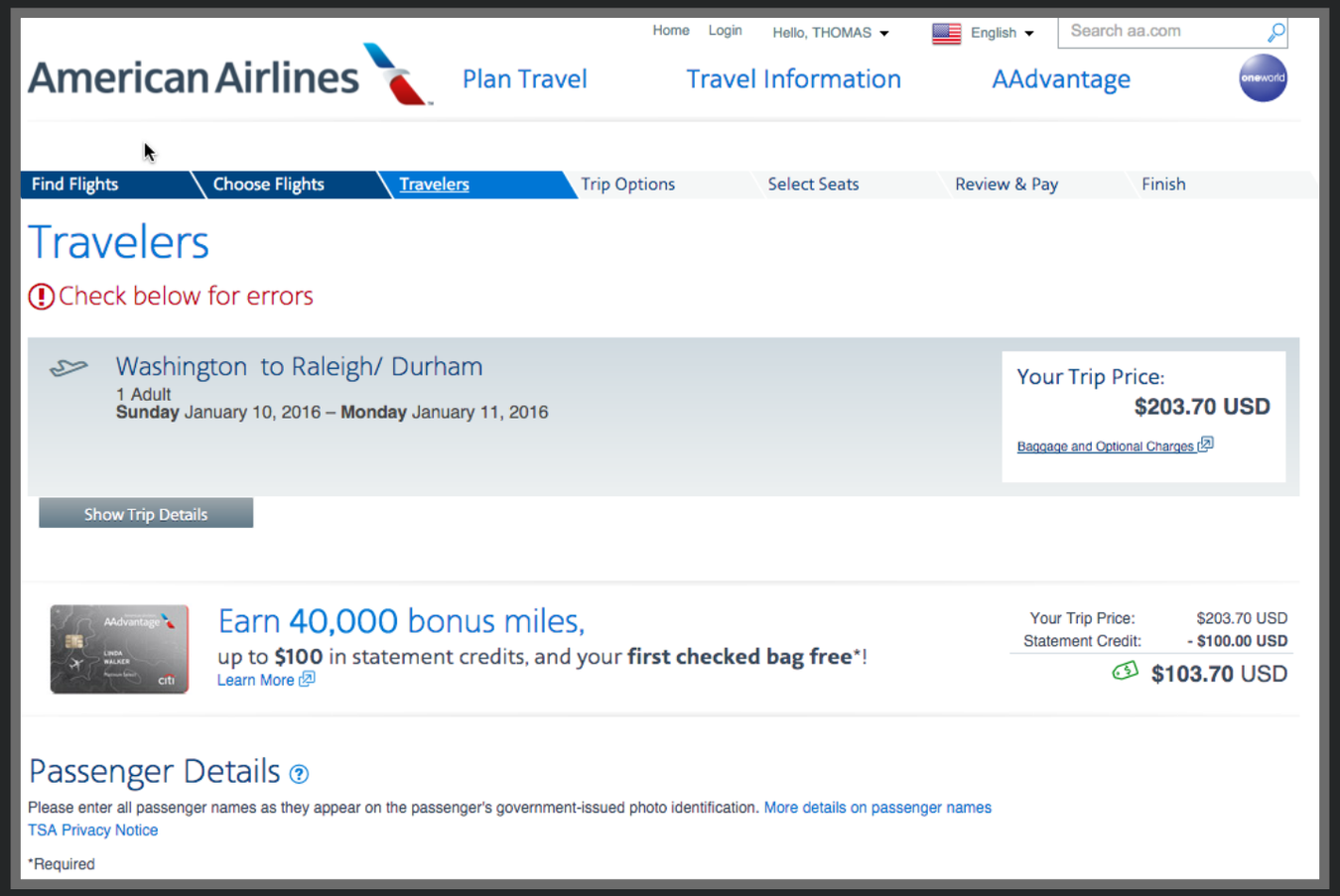
Ordering User Actions

Task Structure

- In some cases, users must take actions in specific sequence
- Must input some information before being able to access subsequent information
  - e.g., must select a shipping method before seeing a final price
- To the extent possible, want to leave users in control of task (user control and freedom)
- But also do not want to distract users by making unrelated decisions in random order (flexibility and efficiency of use)
- And do not want to overwhelm users with too many options at a time (minimalist design)
- Good designs need to balance tradeoffs

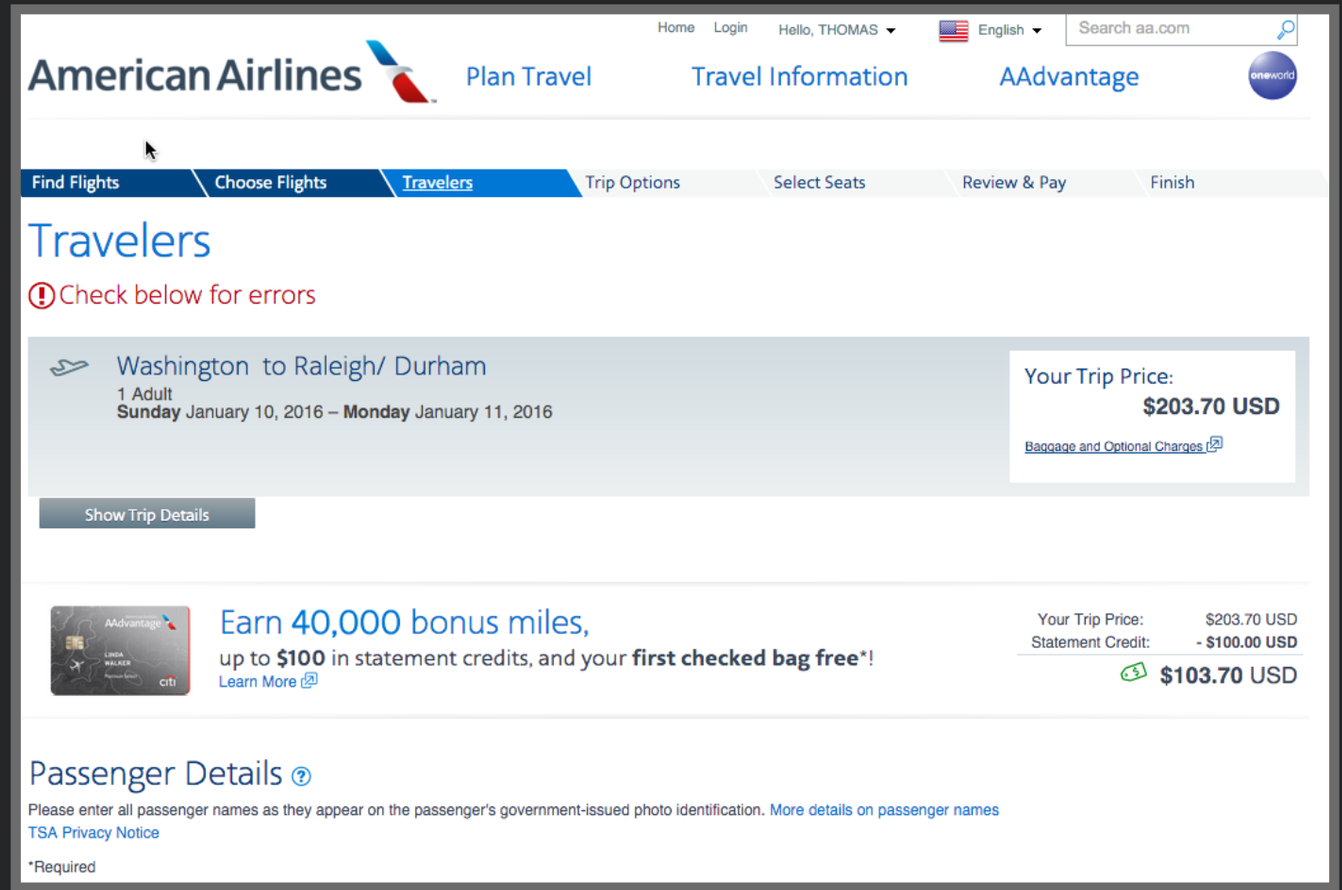
Separate long tasks into sequences

- Reduce short term memory demands by having user only work on one aspect of larger task at a time
- Don’t interrupt users in the middle with unrelated tasks
- Provide closure of each subtask at the end



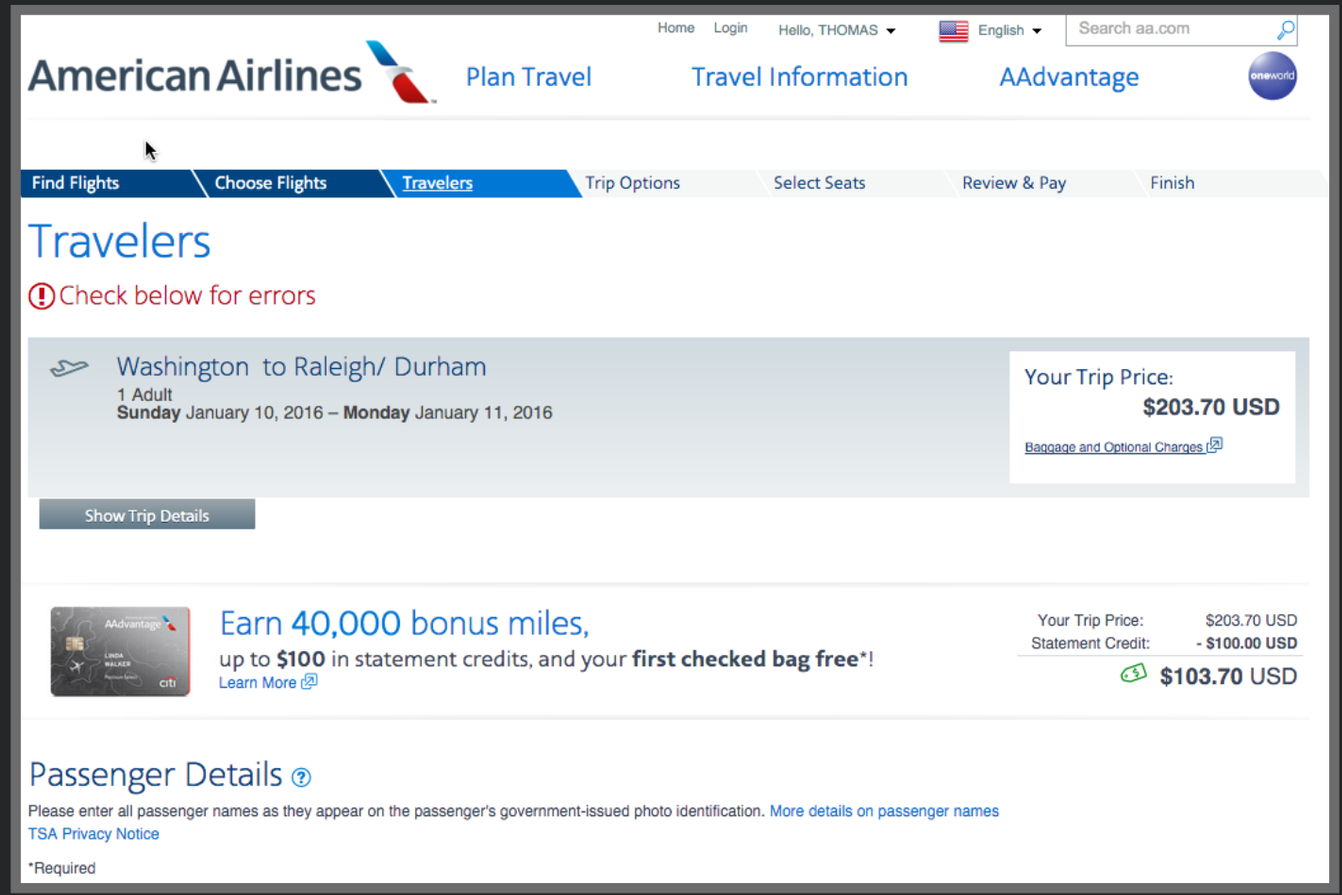
Design for flexibility & efficiency

- Users may take paths never envisioned by designer
- Using studies to identify different task flows, design flexible support for each



Keep users in control

- Important users do not feel constrained
- Want users to feel that they can do things the way they want to do them, not as software dictates to them



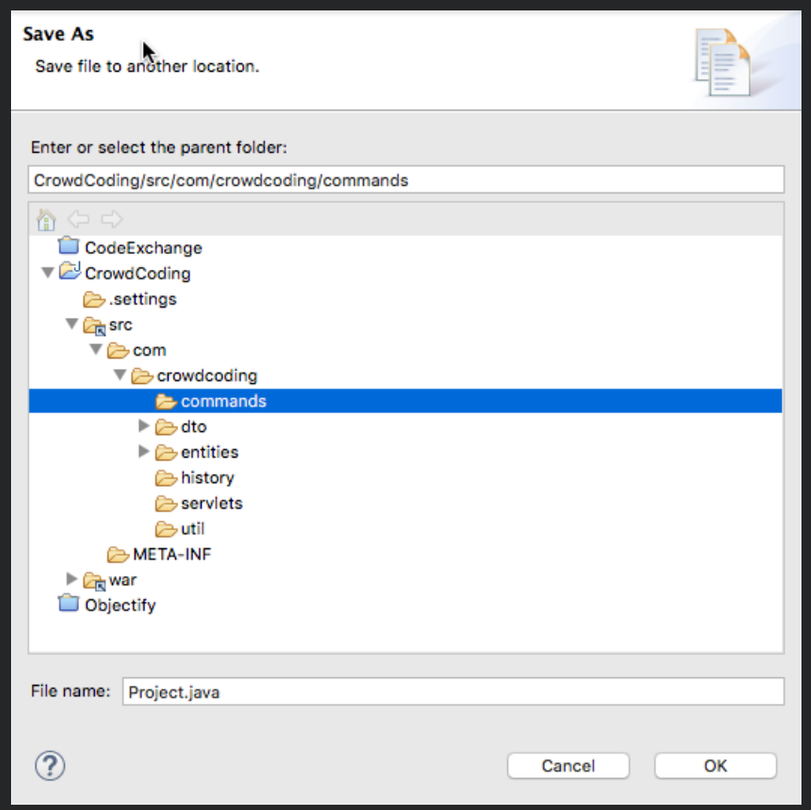
Orchestration & Interaction Flow

- Interaction flow - the next thing the interface wants to do is exactly what user expects
  - Follow users' mental model
  - Let user direct software
  - Keep all related tools available
- Surprises interrupt interaction flow
- Interfaces should be invisible

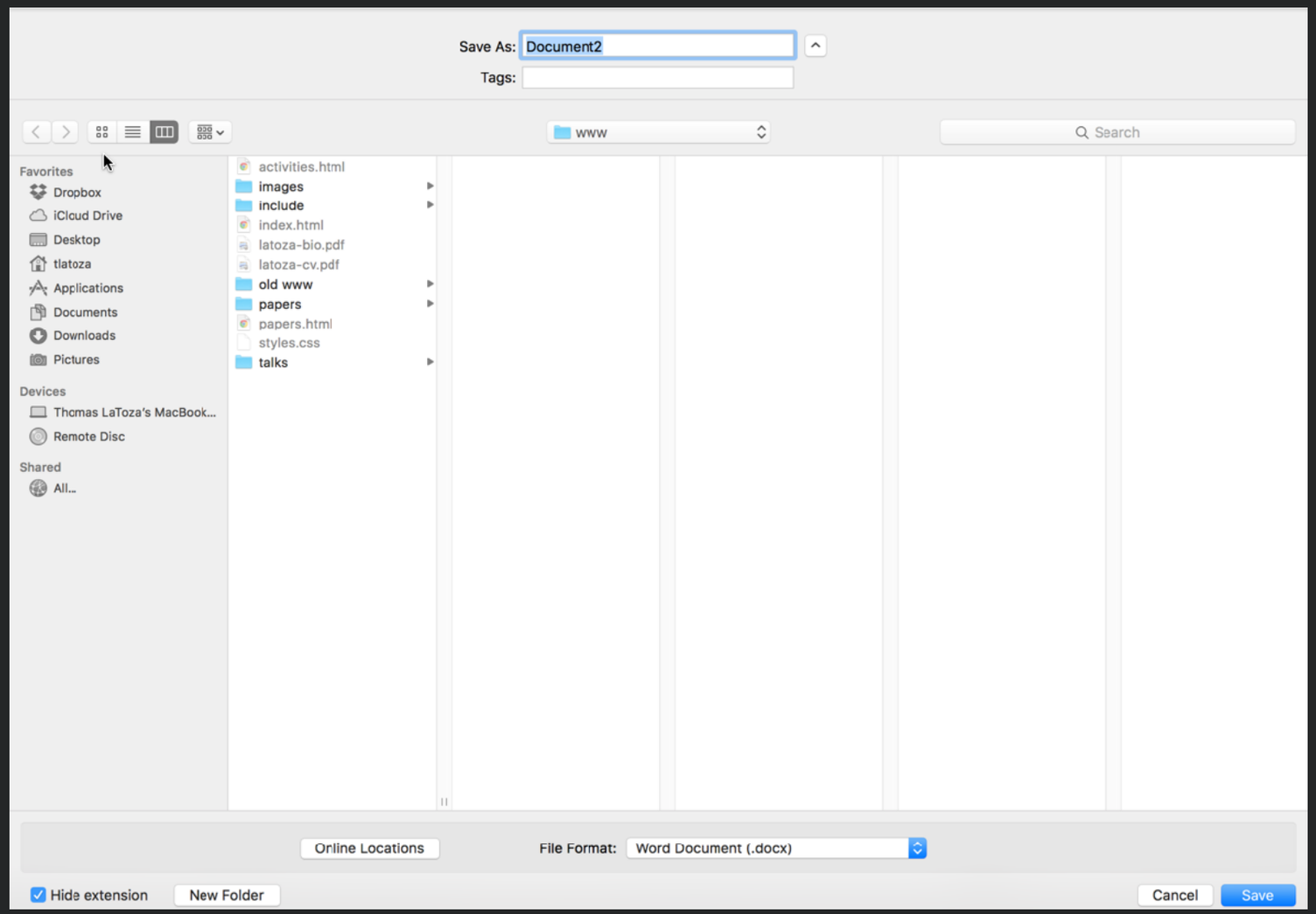
Anticipate Likely Next Actions

- Based on typical observed task flows, surface options for user to take likely next steps

What if folder does not exist?



VS.



Interaction Flow Guidelines

- Don’t use dialogs to report normal behavior
- Separate commands from configuration
- Don’t ask questions, give users choices
  - Give users default input, show possible options
- Make dangerous choices hard to reach
- Design for the probable, provide for the possible

Group Activity

Acknowledgements

- Slides adapted from Dr. Thomas Latoza's SWE 432 course

Acknowledgements

- Slides adapted from Dr. Thomas Latoza's SWE 432 course