

## Final Practical 7-10 Deadline December 9 (Monday)

- Double check your P9 grades; this is the only practical due on this date with manual feedback
  - Submit a regrade request by Friday if you want feedback before the deadline!
- For P7/8/10, I'll only consider the autograder score
- As always you need 2/2 for credit
- I'll have 10-11 and 2-3 drop-in hours on the 9th to answer final questions (also project questions)

## Optional Re-Test

December 11-13 (Wednesday-Friday)

- Like the midterm, take online during any 2.5-hour period
- Only complete questions that you have not yet gotten full credit for
- Practice exams and exam 1 solutions available online

# Re-Test: DB Problem

(a) Consider the following user story:

As an engaged couple, we want to be able to set the date of our wedding prior to sending out invitations. Only **one** of the following models is necessary to implement the user story. Select it below:

- Couple    GuestHousehold    Guest

(c) You are provided the following partial normalized database schema for the following user story:

As an engaged couple, we want to organize our guests into households, so that we can send one save-the-date to each address.

Complete the `GuestHousehold` table; you may add additional rows if you need to, or leave rows empty. Make sure to specify the primary key (PK), the types of each column, and any foreign key (FK) constraints. **Either make it very clear from column names which primary key each foreign key refers to or draw an arrow from that row to the primary key.**

Table Couple		
PK	id	
Attribute	Type	Is FK?
id	int	yes/ <del>no</del>
name1	string	yes/ <del>no</del>
name2	string	yes/ <del>no</del>
weddingDate	date	yes/ <del>no</del>

Table Guest		
PK	id	
Attribute	Type	Is FK?
id	int	yes/ <del>no</del>
name	string	yes/ <del>no</del>
householdID	int	<del>yes</del> /no

Table GuestHousehold		
PK	id	
Attribute	Type	Is FK?
id	int	yes/ <u>no</u>
coupleID	int	<u>yes</u> /no
address	string	yes/ <u>no</u>



## Project Presentations December 14 (Saturday)

- Section A: 10:30AM-12:00PM
- Section B: 7:00PM-8:30PM
- 10-15 minutes, including the following (see website for details):
  - Description of your project (slides)
  - Polished demo (with deployed application)
  - Description of your process (slides)
- Anyone who hasn't presented **must** present; students **may** present multiple times
- You're expected to attend if you're on campus

## Project Deliverables December 15 (Sunday)

- *Pushing this deadline back to after the presentations*
- Like all sprints: tagged commit/completed user stories/working deployment/CI build passing/self and peer evaluation
- Unique to final sprint: final checkpoint, short team/individual writeups
- Send me .env files by email!

## Drop-In Hours

- I'll hold all of my standard drop-in hours next week
- I expect that course assistants **will not** hold their drop-in hours

## Practical/Assignment/Lecture Feedback

- Mean/median time spent was highest for PA3
- Assignment pain points: iterative nature/copying over code, lack of clarity on how the application should behave at each stage/what each test does
- Practicals: desire for a later testing practical, P8 too long, too much copy-paste
- Would prefer more focus on technical content during class time

## Standup Meeting Prompts

Each team member should discuss:

- What they did since the last class to help the team meet the Sprint Goal
- What they plan to do between now and the next class
- Any impediments that will prevent the team from meeting the Sprint Goal



## What does it mean to be a “Responsible Developer”?

What does it mean to be a responsible developer? Sometimes this is easy., Don't write apps that do illegal things. Sometimes it is less clear cut and based on your personal values.

How do you feel about: Building weapons? Companies that rely on paying minimal wages, here or abroad? Working for \_\_\_\_ (insert your troubling organization)?

There are no right answers here. You will have to decide for yourself what, if any, tradeoffs might exist for/around interesting projects (is the work interesting enough despite those tradeoffs...)

To give you a concrete example, some of you may have heard of Joe Redmond. He is one of our more prominent CS graduates. Developed the YOLO deep learning object recognition system which is now used in many applications. He walked away from the entire field of image recognition over concerns about the military applications

Technology is neither good nor  
bad; nor is it neutral.

Kranzberg's First Law of Technology

Technology is only as neutral as its creators, that is not “neutral” at all. We can't just build systems and then say, “How people use them is not our problem”. This class topic was motivated by studies of engineering (and other technical students) perceptions of ethics in their education and professional practice, and to be very transparent, our own curricular limitations as a department. As a department I think we were (are) behind where we needed to be on these questions. The creation of the “Responsible Computing” courses is a step towards addressing those needs.

“Alpay's study (2013) confirms that first-year engineering students are already aware of the marginal status of ethics in their programs based on the lack of attention to the ethical aspects of their technical course content knowledge. A first-year undergraduate student acknowledged, “[in] a Computing degree we don't always appreciate the seriousness of what we are doing and the ethical implications of it” (p. 1463). The participant noted her engineering ethics course was at odds with other engineering courses that pay little attention to ethical issues. Hashemian and Loui's study (2010) shows that engineering students who did not take an elective engineering ethics class exhibited minimal sense of responsibility as they responded to an engineering ethics-related case scenario considering the posed problem as “not my business.”

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8660727/>

Modern humanity is distinguished by  
paleolithic emotions,  
medieval institutions,  
and god-like technology

E.O. Wilson

Increasingly large part of our experience of the world will be mediated through technology because it is everywhere

- Developing algorithms is all about breaking processes down to their component pieces and putting them back together again. We can't do this without having some understanding of the process.
- As a result, a key skill for programmers is the ability to break a problem down to its most fundamental components.
- Then we develop systems that we have complete control over.
- The result is a kind of hubris.
- We “rule” over these little worlds that we fully understand because we created them, and as a result we may think that we have the tools and/or ability to understand any problem.
- This creates blind spots about the things we don't understand, or don't want to pay attention to.
- We also start to think that this form of computational thinking applies to everything.

## Hippocratic Oath

- I swear to fulfill, to the best of my ability and judgment, this covenant:
- I will respect the hard-won scientific gains of those physicians in whose steps I walk, and gladly share such knowledge as is mine with those who are to follow.
- I will apply, for the benefit of the sick, all measures [that] are required, avoiding those twin traps of overtreatment and therapeutic nihilism.
- I will remember that there is art to medicine as well as science, and that warmth, sympathy, and understanding may outweigh the surgeon's knife or the chemist's drug.
- I will not be ashamed to say "I know not," nor will I fail to call in my colleagues when the skills of another are needed for a patient's recovery.
- I will respect the privacy of my patients, for their problems are not disclosed to me that the world may know. Most especially must I tread with care in matters of life and death. If it is given me to save a life, all thanks. But it may also be within my power to take a life; this awesome responsibility must be faced with great humbleness and awareness of my own frailty. Above all, I must not play at God.
- I will remember that I do not treat a fever chart, a cancerous growth, but a sick human being, whose illness may affect the person's family and economic stability. My responsibility includes these related problems, if I am to care adequately for the sick.
- I will prevent disease whenever I can, for prevention is preferable to cure.
- I will protect the environment which sustains us, in the knowledge that the continuing health of ourselves and our societies is dependent on a healthy planet.
- I will remember that I remain a member of society, with special obligations to all my fellow human beings, those sound of mind and body as well as the infirm.
- If I do not violate this oath, may I enjoy life and art, respected while I live and remembered with affection thereafter. May I always act so as to preserve the finest traditions of my calling and may I long experience the joy of healing those who seek my help.

Medicine can have its own "little god" problem, especially when you can literally save someone's life. A code of conduct, specifically the Hippocratic Oath, is intended in part to counteract that impulse. Typically, we think of this as "First, do no harm", but that phrase isn't in there, though it is implied (depending on the translation).

What is striking to me is how much of the Hippocratic oath is not about curing disease, <click>

## Hippocratic Oath

- I swear to fulfill, to the best of my ability and judgment, this covenant:
- I will respect the hard-won scientific gains of those physicians in whose steps I walk, and gladly share such knowledge as is mine with those who are to follow.
- I will apply, for the benefit of the sick, all measures [that] are required, avoiding those twin traps of overtreatment and therapeutic nihilism.
- I will remember that there is art to medicine as well as science, and **that warmth, sympathy, and understanding may outweigh the surgeon's knife or the chemist's drug.**
- I will not be ashamed to say "I know not," nor will I fail to call in my colleagues when the skills of another are needed for a patient's recovery.
- I will respect the privacy of my patients, for their problems are not disclosed to me that the world may know. Most especially must I tread with care in matters of life and death. If it is given me to save a life, all thanks. But it may also be within my power to take a life; this awesome responsibility must be faced with great humbleness and awareness of my own frailty. Above all, I must not play at God.
- I will remember that **I do not treat a fever chart, a cancerous growth, but a sick human being,** whose illness may affect the person's family and economic stability. My responsibility includes these related problems, if I am to care adequately for the sick.
- I will prevent disease whenever I can, for prevention is preferable to cure.
- **I will protect the environment which sustains us,** in the knowledge that the continuing health of ourselves and our societies is dependent on a healthy planet.
- I will remember that I remain a member of society, with special obligations to all my fellow human beings, those sound of mind and body as well as the infirm.
- If I do not violate this oath, may I enjoy life and art, respected while I live and remembered with affection thereafter. May I always act so as to preserve the finest traditions of my calling and may I long experience the joy of healing those who seek my help.

For example, consider these portions and in particular note the portions that are about being a human treating other humans. That can give us some ideas as we consider what it means to be a responsible developer, a human developing software for other humans.

## ACM Code of Ethics: General Ethical Principles

- Contribute to society and to human well-being, acknowledging that all people are stakeholders in computing.
- Avoid harm.
- Be honest and trustworthy.
- Be fair and take action not to discriminate.
- Respect the work required to produce new ideas, inventions, creative works, and computing artifacts.
- Respect privacy.
- Honor confidentiality.

<https://ethics.acm.org/code-of-ethics/>

There are some codes of conduct that govern computer scientists as well. The ACM is the Association of Computing Machinery, a professional society for Computer Science (and Computer Scientists). Here is initial part of the ACM code.

What if anything jumps out at you? Is there something you think is missing? Some things that I note:

- All people are stakeholders
- Avoid harm
- The emphasis on privacy and confidentiality

## ACM Code of Ethics: Professional Responsibilities

- Strive to achieve high quality in both the processes and products of professional work.
- Maintain high standards of professional competence, conduct, and ethical practice.
- Know and respect existing rules pertaining to professional work.
- Accept and provide appropriate professional review.
- Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.
- Perform work only in areas of competence.
- Foster public awareness and understanding of computing, related technologies, and their consequences.
- Access computing and communication resources only when authorized or when compelled by the public good.
- Design and implement systems that are robustly and useably secure.

<https://ethics.acm.org/code-of-ethics/>

In addition to those general ethical principles, the ACM defines a set of professional responsibilities and a set of professional leadership principles. Several items that I note:

- Give comprehensive and thorough evaluations of computer systems and their impacts, including analysis of possible risks.
- Perform work only in areas of competence.
- Design and implement systems that are robustly and useably secure

I think “Perform work only in areas of competence” speaks to the potential for hubris that we talked about earlier.

## ACM Code of Ethics: Professional Leadership Principles

- Ensure that the public good is the central concern during all professional computing work.
- Articulate, encourage acceptance of, and evaluate fulfillment of social responsibilities by members of the organization or group.
- Manage personnel and resources to enhance the quality of working life.
- Articulate, apply, and support policies and processes that reflect the principles of the Code.
- Create opportunities for members of the organization or group to grow as professionals.
- Use care when modifying or retiring systems.
- Recognize and take special care of systems that become integrated into the infrastructure of society.

<https://ethics.acm.org/code-of-ethics/>

One item that I noted was:

- Recognize and take special care of systems that become integrated in the infrastructure of society



## IEEE Code of Ethics

- We, the members of the IEEE, in recognition of the importance of our technologies in affecting the quality of life throughout the world, and in accepting a personal obligation to our profession, its members, and the communities we serve, do hereby commit ourselves to the highest ethical and professional conduct and agree:
- To hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment;
- To avoid real or perceived conflicts of interest whenever possible, and to disclose them to affected parties when they do exist;
- To be honest and realistic in stating claims or estimates based on available data;
- To reject bribery in all its forms;
- To improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems;
- To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;
- To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
- To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;
- To avoid injuring others, their property, reputation, or employment by false or malicious action;
- To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

<https://www.ieee.org/about/corporate/governance/p7-8.html>

The IEEE, or Institute of Electrical and Electronics Engineers, is another professional society relevant to Computer Science. They have promulgated a similar code of ethics. I note the items toward the end:

- To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression
- To avoid injuring others, their property, reputation, or employment by false or malicious action

**Don't Be Evil**

Google Code of Conduct


Originally this was in the Google prospectus and a bit of a slap back at competitors. It has since been phased out as a motto, though it is still found in the last line of the current code of conduct (<https://abc.xyz/investor/google-code-of-conduct/>) and is a focus on ongoing controversy and criticism.



As an example of the type of conflicts that can arise when these codes meet business goals/desires, these resignations were in response to a project to use AI to speed up automatic classification of images of people and objects in military applications. In this instance Google engineers decided there was a line they would not cross.

**Why Uber's self-driving car killed a pedestrian**

*It was the first fatal accident of its kind*



ALEX DAVIES TRANSPORTATION 05.16.2019 05:46 PM

**Tesla's Latest Autopilot Death Looks Just Like a Prior Crash**

A Florida man was killed March 1 when his Tesla collided with a tractor trailer that was crossing its path. A government report says the Autopilot feature was activated.

Twitter Facebook LinkedIn Email Print

Fields like Civil Engineering, which due to historical failures and associated deaths, have an emphasis on professional responsibility (both generally and legally) and exists in a stricter regulatory environment than does Computer Science (in my view/perception). Perhaps because CS is a younger field, or perhaps because a perception that software is somehow not risky the same way. The reality is software is everywhere, including life and death applications. There is a responsibility to take this seriously — you are not just solving technological problems; you are creating products people will use (or products that will be used on them...).

We must ask, just because it can be done, should we? And when the stakes are high, how do ensure that we (as a field), exercise the requisite care?

An obvious response is that many of us will not be writing code for products where someone's life is on the line. Maybe you are just writing a simple application to allow Harvard and other college students to connect. Surely that will have no real impact on people's lives or society as a whole...

<https://www.economist.com/the-economist-explains/2018/05/29/why-ubers-self-driving-car-killed-a-pedestrian>  
<https://www.wired.com/story/teslas-latest-autopilot-death-looks-like-prior-crash/>

## Facebook data privacy scandal: A cheat sheet

by James Sanders | Dan Patterson  
in Security  
on July 24, 2019, 8:52 AM PST

Read about the saga of Facebook's failures in ensuring privacy for user data, including how it relates to Cambridge Analytica, the GDPR, the Brexit campaign, and the 2016 US presidential election.

### *How Misinformation 'Superspreaders' Seed False Election Theories*

Researchers have found that a small group of social media accounts are responsible for the spread of a disproportionate amount of the false posts about voter fraud.



### **Mob storms Capitol as Facebook, Twitter roles come under fire**

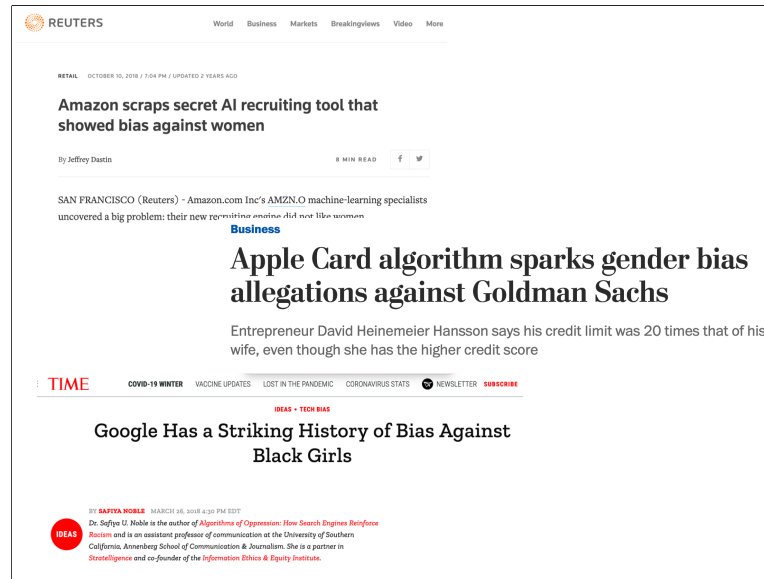
The social media companies face pressure to suspend Trump's accounts after the president used social media to push false claims of election fraud.

Queenie Wong, Andrew Morse, Carrie Mihalick  
Jan 7, 2021 10:12 am PT

There is a saying, if you can't figure out how a company makes money, you are probably the product, not the customer... Some issues we need to grapple with are quite hidden. I'm sure that Zuckerberg, et al. never thought that their ideas would lead to them to testifying before congress

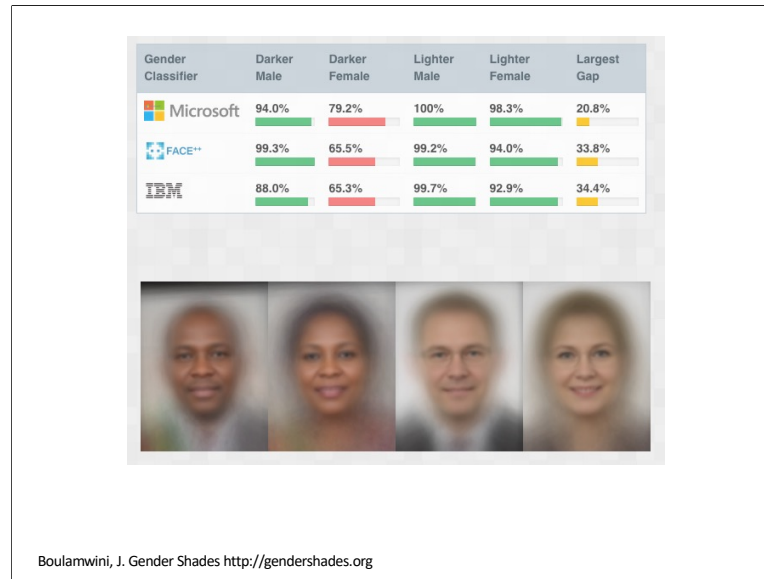
They are now in a balancing act, trying to figure out how to moderate content, without silencing anyone, while maximizing profits. Profits are at their highest when people are attracted to the site and stay there. And what is more attractive than outrage?

They are developing tools to automatically vet content. Both out of practicality and the tendency to trust automated systems because they are free from human bias...



Except they are not. Computational systems are not inherently neutral. As the authors of Gender Shades, a project we will talk about in a moment, wrote: “Automated systems are not inherently neutral. They reflect the priorities, preferences, and prejudices [of the developers].” This isn’t to suggest that the developers are actively trying to generate discriminatory outcomes, but instead that is the unintentional result of the choices the developers make (for training data, for evaluation benchmarks, etc.).

For example, a couple of years ago, Amazon shut down an AI tool they used for recruiting when it emerged that it penalized applications with the word “women’s” in them (<https://www.theverge.com/2018/10/10/17958784/ai-recruiting-tool-bias-amazon-report>). Apple had issues because it seems that women were getting lower credit limits on the Apple card. Googling “black girls” used to bring up pages of pornography.



We just mentioned the Gender Shades project by Joy Buolamwini. In that project Joy evaluated commercial gender prediction tools (predict gender from images). Here are the results. Notice the difference in performance for darker vs. lighter skin.

The problem with all of these examples is the "Algorithmic Bias" inherent to the system. Sometimes, these are part of the algorithm itself (e.g., something the developer just didn't/doesn't notice). More often, though, they are inherent to the data used to train the system. Most facial recognition systems were trained using data that was not representative of the population at large. For example, the Amazon hiring tool was trained using the employees that were successful at Amazon. Looking at a population that skewed male, it tried to replicate that skew.

There is also the problem of releasing a program, in the first place, that claims to be able to label gender. For some of us, being misgendered could be amusing or just another "automation" quirk we ignore. Some could find it more offensive or even devastating depending on the result.

## Universal design example: Designing for gender diversity and inclusion

- Give people a really good reason for asking
- Make it private, safe, and anonymous
- Always make it optional
- Ask for pronouns instead
- Be ready for a complex answer
- Consider internationalization
- Just don't ask

Gender\*  
 Male  Female

**Gender**  
Check one or more options that reflect your gender.

- woman
- man
- non-binary
- transgender
- intersex
- Two Spirit (see below for more information)
- gender non-conforming
- Other:

Sabrina Fonseca,  
<https://uxdesign.cc/designing-forms-for-gender-diversity-and-inclusion-d8194cf1f51>

Thinking about the user can be more fraught than you perhaps first realize. These are some suggestions for design specifically related to gender. We are used to seeing the form to the top right and may not think much about it. But this kind of question can reduce engagement and lead to false conclusions based on bad data. The author articulated some recommendations (shown at the left) and highlighted other approaches, like the form to the bottom right which allows someone to check multiple responses.

In this and many of the design problems we encounter, we may not realize what we don't know. That is why diverse voices "in the room" are so critical.



## Think of an application that felt like it wasn't designed for you...

Think of an application you needed to use but felt like it wasn't designed for you...

- Why did it feel like it was designed for someone else?
- What changes would you have made?

Not just bad design (e.g., Banner) but seemed to be expressly designed for someone else...

A common example is names... Other past examples are applications that assume prior addresses, jobs.

There is a famous blog post “Falsehoods Programmers Believe About Names” that has 40 false assumptions, such as “People have exactly N names, for any value of N.” There are a wide range of naming approaches in world, of which most people will only encounter a small subset. If you build just what you know your technology will not work for everyone.

The author writes about an example of individual with a hyphenated last name “...complaining about how a computer system he was working with described his last name as having invalid characters. It of course does not, because anything someone tells you is their name is — by definition — an appropriate identifier for them. John was understandably vexed about this situation, and he has every right to be, because **names are central to our identities**, *virtually by definition*.

Having your name is rejected is more than just frustrating. “Authentic name” policies can be discriminatory (e.g., Native American names are rejected, as has happened) or if using your “real” name would expose you to harm. “In September 2014, hundreds

of transgender people and drag queens had their Facebook accounts shut down after they were reported as fake. As individual users started to realize that this was happening to people across their communities, they began to organize, lodge complaints and protest. Though Facebook apologized to the lesbian, gay, bisexual and transgender (LGBT) and drag queen communities after the issue garnered media attention, as of today there has not been meaningful change in Facebook's policy. Rather, the company continues to require users to provide onerous documentation to verify names that are flagged by users or Facebook as potentially unauthentic. This is problematic for a range of communities, including trans and gender non-conforming people, drag queens, survivors of violence, Native Americans and more."

<https://www.kalzumeus.com/2010/06/17/falsehoods-programmers-believe-about-names/>

[https://media.business-humanrights.org/media/documents/files/documents/Sarah\\_Gunther\\_Facebook\\_Real\\_Name\\_Policy.pdf](https://media.business-humanrights.org/media/documents/files/documents/Sarah_Gunther_Facebook_Real_Name_Policy.pdf)

<https://www.userinterviews.com/blog/design-failure-examples-caused-by-bias-noninclusive-ux-research>

**Don't Be Evil**

Google Code of Conduct

Let's return to "Don't be evil". Whatever you think about Google, I think this can be a good starting point.

To some extent this can be boiled down to the unofficial Hippocratic oath: 'First, do no harm'. For me, it is important to never lose sight of the user. What are the contexts of use? What are the risks to the user? What are the risks to society?

More than “How do we DO NO  
EVIL with technology?” We ought  
to consider “How do we DO  
GOOD?”

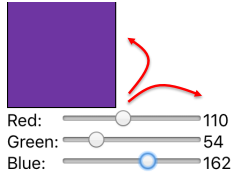
Ge Wang, Artful Design

Ge Wang says, this isn't enough. “Don't be evil” sets a pretty low bar.

Wang says to design from your values in the first place. Develop tools that make the world a better place. Maybe it is something that solves one of the world's ills. But it can be quieter than that. Maybe it brings a little art into the world, maybe it makes someone laugh, or evokes the feeling of satisfaction you get when you use a tool made by someone who really cared about the details and made sure the Statue of Liberty had hair.

To me, this is what a responsible developer is. Someone who thinks about users and society and consequences and incorporates their values into the work they do. The work we do in here is not somehow separate from the world, but very much a part of it. Thus, it can both reflect society's challenges, however you define them, *and* be a vehicle, a powerful one, for effecting change. I am not trying to encourage you to make any particular choice or choices. The choices you make will and should be personal. Instead, I ask you to remember that technology is not neutral and will – can – reflect your values. So don't put your values down when you pick up your keyboard.

## In the beginning...

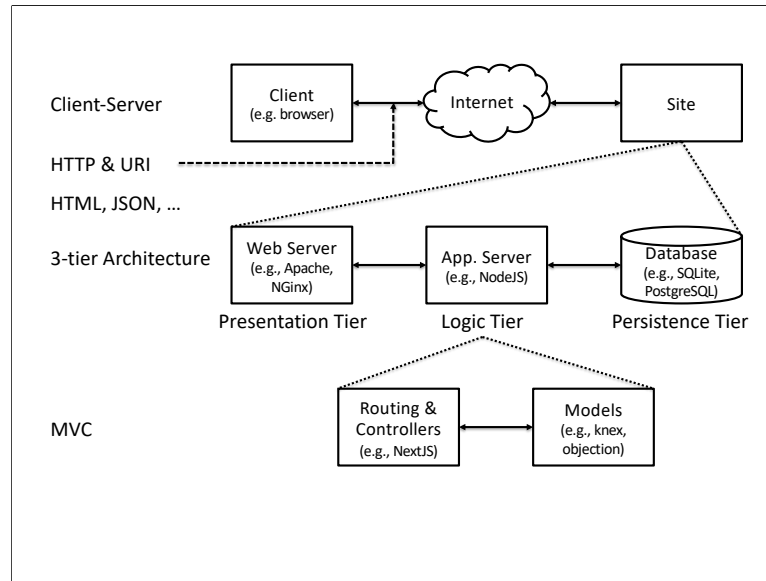


```
<div class="blue-slider">
  <div class="color-label">blue: </div>
  <input type="range" id="slider-b" .../>
  <span id="value-b"></span>
</div>
```

```
// Set oninput callback for each slider
sliders.forEach((slider) =>
  slider.addEventListener("input", update));

const update = function() {
  colorBox.style.background =
    `rgb(${sliders[0].value}, ${sliders[1].value}, ${sliders[2].value})`;
  sliders.forEach((slider, index) =>
    labels[index].innerHTML = slider.value);
};
```

In the beginning, I suspect this was new to many of you... But we have now implemented this kind of interactivity and much more. In fact,...



Over the semester, we have learned about, used and often implemented components in every one of these boxes, from the JavaScript running on the front-end (i.e., the client) to the route handlers on the server, the database schema and more...

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.

## Callbacks and more callbacks!

```
const wrapValue = (n) => { // function(n) {  
  let local = n;  
  return () => local; // function () { return local; }  
}  
  
let wrap1 = wrapValue(1); // () => 1  
let wrap2 = wrapValue(2); // () => 2  
console.log(wrap1()); // What will print here?  
console.log(wrap2()); // What will print here?
```


Along the way we have gained familiarity with JavaScript (perhaps a new language) and its emphasis on closures and callbacks. The idea of functions as values (functions as 1<sup>st</sup> class objects), and asynchronous execution hopefully now feels familiar.

```

function ColorPicker() {
  const [red, setRed] = useState(0);
  const [green, setGreen] = useState(0);
  const [blue, setBlue] = useState(0);

  const color = {
    background: `rgb(${red}, ${green}, ${blue})`
  };
  return (
    <div>
      <div className="color-swatch" style={color} />
      <LabeledSlider label="Red" value={red} setValue={value => setRed(value)} />
      <LabeledSlider label="Green" value={green} setValue={value => setGreen(value)} />
      <LabeledSlider label="Blue" value={blue} setValue={value => setBlue(value)} />
    </div>
  );
}

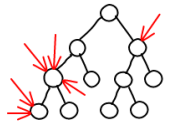
```



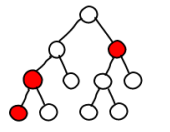
**Single source of truth!**

**Props down! Callbacks up!**

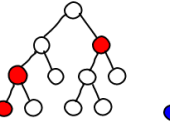
setState



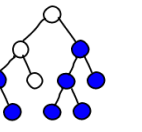
Dirty



Dirty



Re-rendered




We have made extensive use of React, and some its key principles:

- Maintain a single source of truth
- Props "flow" down
- Callbacks "flow" up

React implements a design pattern for highly interactive UIs. Your responsibility as the programmer is to define what you want rendered on the screen for a given state of the application and how you want to update that state in response to user actions. React fills in the "last piece" of the cycle by efficiently re-rendering the UI as the state changes.



Route	Controller Action
POST /api/films	Create new movie from request data
GET /api/films/:id	Read data of movie with id == :id
PUT /api/films/:id	Update movie with id == :id from request data
DELETE /api/films/:id	Delete movie with id == :id
GET /api/films	List (read) all movies



```
router.get(async (req, res) => {  
  const films = Film.query().withGraphFetched('genres');  
  res.status(200).send(films);  
});
```

We learned about writing backend servers using Javascript to create RESTful APIs. RESTful APIs are built around the idea of actions on resources, i.e., the film resource in the film explorer example. The routes are typically implemented as queries to our persistence layer, in this example to a RDBMS via the Objection.js ORM. This particular query is using the "one-to-many" relation defined on the Film Objection.js model to automatically fetch the Genres associated with each of the films.

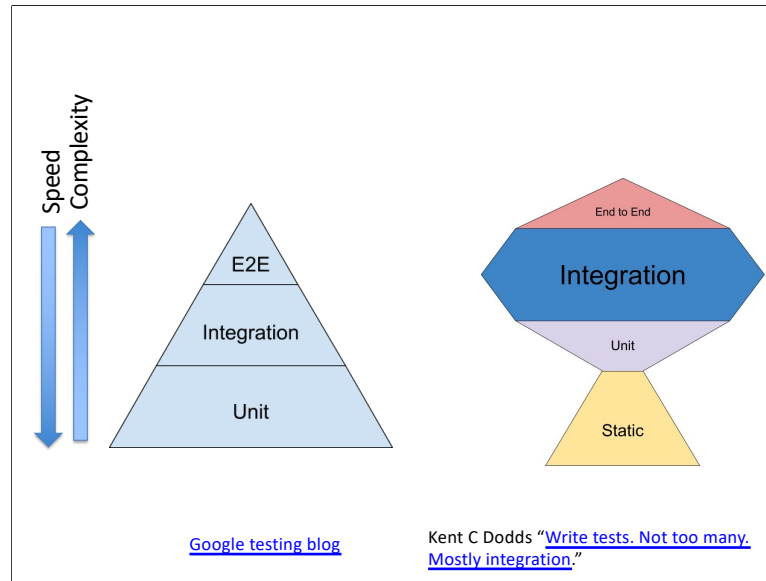
	Relational (RDBMS)	Non-Relational
<b>Data</b>	Table-oriented	Document-oriented, key-value, graph-based, column-oriented, ...
<b>Schema</b>	Fixed schema	Dynamic schema
<b>Joins</b>	Used extensively	Used infrequently
<b>Interface</b>	SQL	Custom query language
<b>Transactions</b>	ACID	CAP

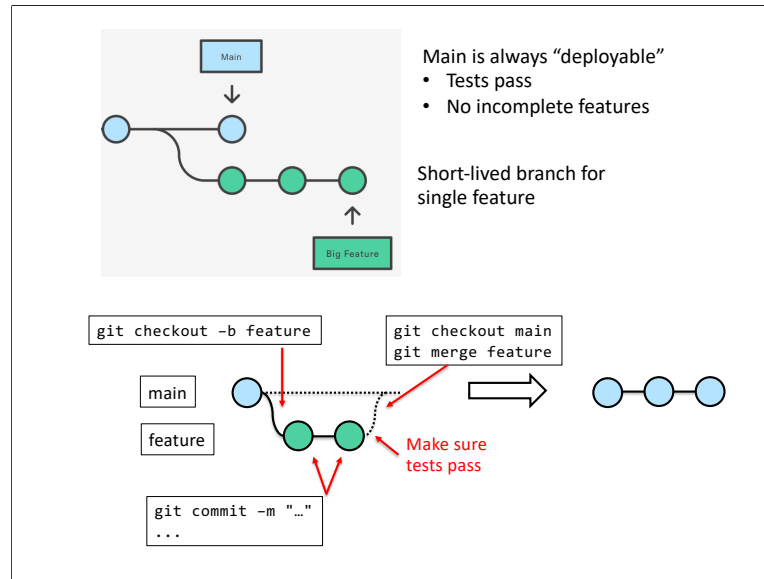
```
SELECT * FROM people
WHERE age > 25;
```

```
db.people.find(
  { age: { $gt: 25 } }
)
```

We learned about different approaches to data persistence on the backend with a focus on relational databases, e.g., SQLite and PostgreSQL.



We learned about (but maybe didn't always practice...) test driven development (TDD) and concepts like unit testing and integration testing. We also talked about the use of behavior driven development (BDD) to move us from user story to scenario to test to implementation.



We learned about using git to manage development, and the value of continuous integration (CI). CI rigorously tests (we hope) every integration in production-like environment, the motivation is to:

- Prevent development-production mismatch
- Test multiple browsers, etc.
- “Stress test” code for performance, fault-tolerance, etc.

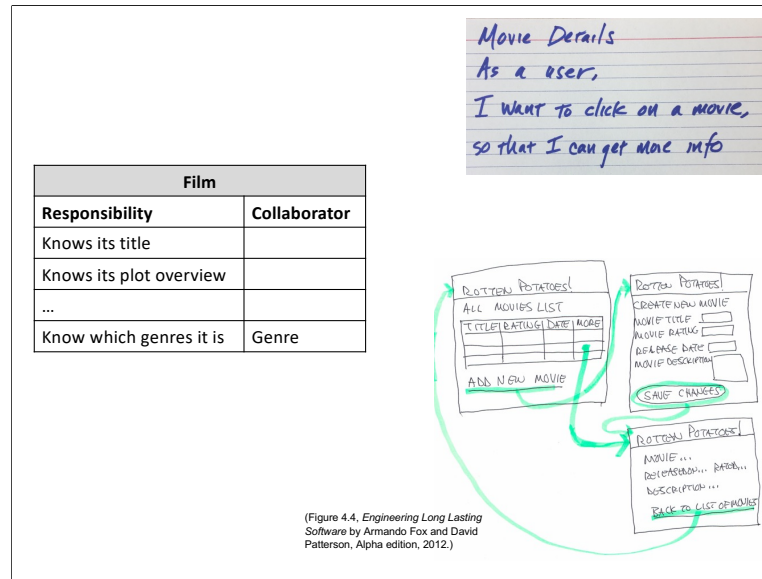
CI is part of a larger DevOps approach.

Recall the DevOps principles:

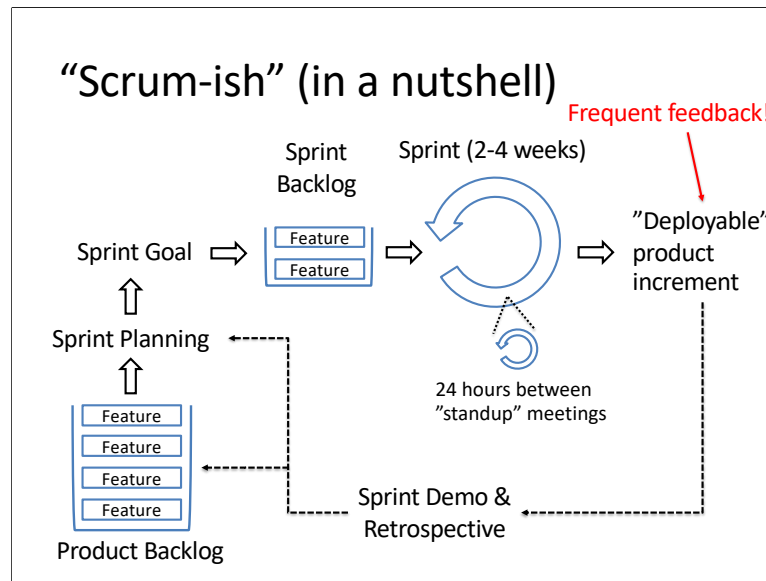
- Involve operations in each phase of a system’s design and development,
- Heavy reliance on automation versus human effort,
- The application of engineering practices and tools to operations tasks

This manifested for us in our use of GitHub actions to test our builds, and automated preparation (e.g. running migrations/seeding) and ultimately deployment with `csci312.dev`.

<https://www.atlassian.com/git/tutorials/using-branches>



You also learned about ways to approach design from user stories to CRC cards to lo-fi prototypes. Our goal is to be able to iterate on our design quickly and cheaply. These "lo-tech" tools are intended to facilitate conversations with our stakeholders, e.g., customers.



Agile development processes, Scrum in particular, played an important role for us. In Scrum the short sprints provide frequent opportunities to update our approach in response to what we have learned about the problem or the application. Recall the key idea of lower-case "a" agility:

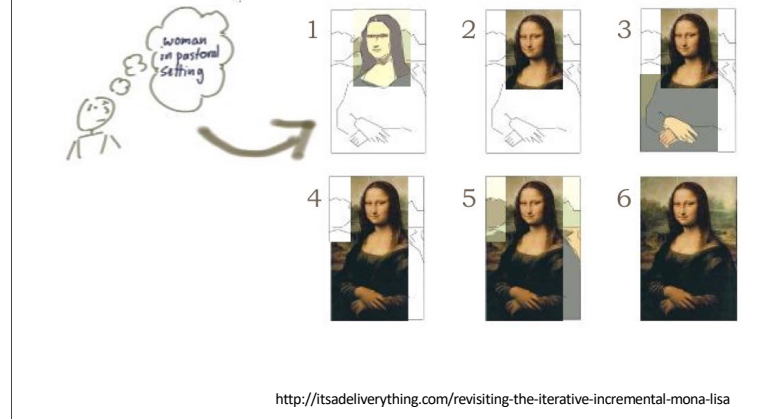
1. Find out where you are,
2. Take a small step towards your goal,
3. Adjust your understanding based on what you learned, and
4. Repeat

And when faced with two or more alternatives that deliver roughly the same value, choose the path that makes future change easier

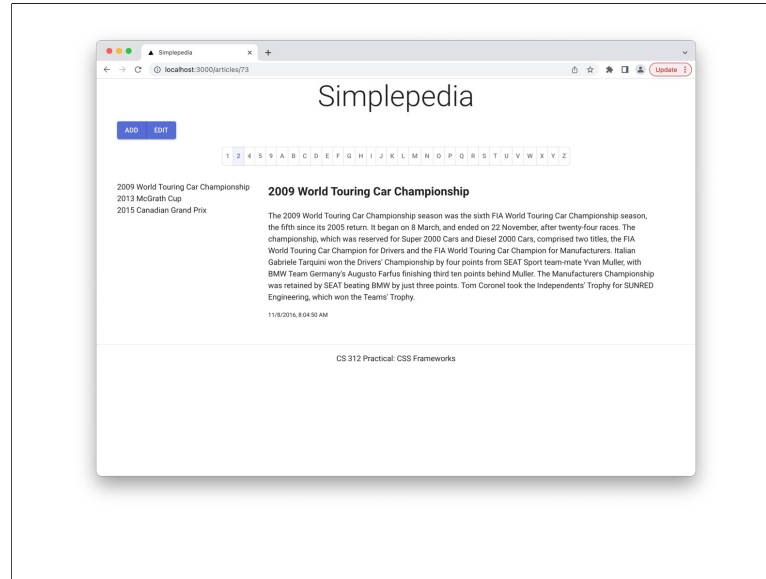
Adapted from Mountain Goat Software  
<https://www.mountaingoatsoftware.com/uploads/presentations/Getting-Agile-With-Scrum-Norwegian-Developers-Conference-2014.pdf>

Adapted from Dave Thomas (<https://www.youtube.com/watch?v=a-BOSpxYJ9M>)

## Iterative Incremental



And we thought about how to approach building our own "Mona Lisa"s. Recall that an incremental approach calls for building a fully formed idea a bit at a time, and thus requires having a fully formed idea. In contrast, iterating allows you to move from vague idea to realization. The catch is we have to address the entire scope at one time, i.e., we are working on the entire image (a tricky and risky approach). Thus, we sought the of the best of both where in each sprint we both add new features (incremental) and refine existing functionality (iterative) with strong focus the highest priority features (the "face").



Along the way, you have completed four assignments and 10 practical exercises yielding a Wikipedia-like platform with database persistence and user authentication.

Between the assignments, practical and the project, you/we have created 466 repositories in our GitHub organization. On the just the main branches of your projects (as of yesterday) you have made 481 commits (and I suspect there are many more in total) and many, many, branches (I didn't even want to try to count!)

63+216+131+11+60



### Commandments for being a bad SW team player (and some alternatives)

Those fails don't matter	Never push failing tests
My branches, my sanctuary	Have short-lived branches by integrating frequently
It's just a simple change	Test everything
I am a special snowflake	One coding style
Cleverness is impressive	Transparency is humble
Just change it quickly on the production server	Make every change automatable
Time spent looking stuff up is wasted time (not coding)	Spend 5 minutes searching for less or better code
"Green fever": Catch it!	More tests ≠ higher quality
Weeks of coding can save hours of planning & thought	Work through your design
When blocked, I am stuck	I unblock myself, or move on to the next task

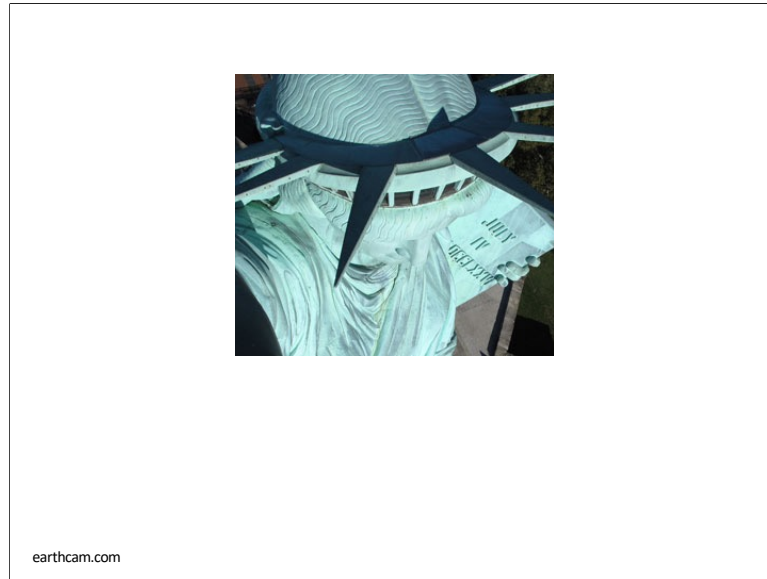
Anti-patterns
Alternatives

I suspect these "anti-patterns" resonate now as we are in the thick of the project and than they did before!

Adapted from Armando Fox and David Patterson (Berkeley cs169) under CC-BY-SA-NC license.

## Take-aways

- Behind every design decision there should be a user story (a stakeholder and a motivation!)
- Testing, not just a class requirement, it's a good idea
- Develop iteratively *and* incrementally
- There should be one source of truth
- Don't repeat yourself
- Don't mutate props or state
- Do really read error messages (and the docs!)
- Automate all the things
- Don't break the "Build"
- Program strategically, not tactically



Write beautiful code, do the Right Thing, give your Statue of Liberty hair!

`go/crf`  
`go/cshelp-evals`

## Wrapping Up

1. Course Response Forms: [go/crf](#)
2. CS Help Feedback: [go/cshelp-evals](#)
3. Honor Code Vote: [go/middpresence](#)
  - Info: [go/hcvote](#)