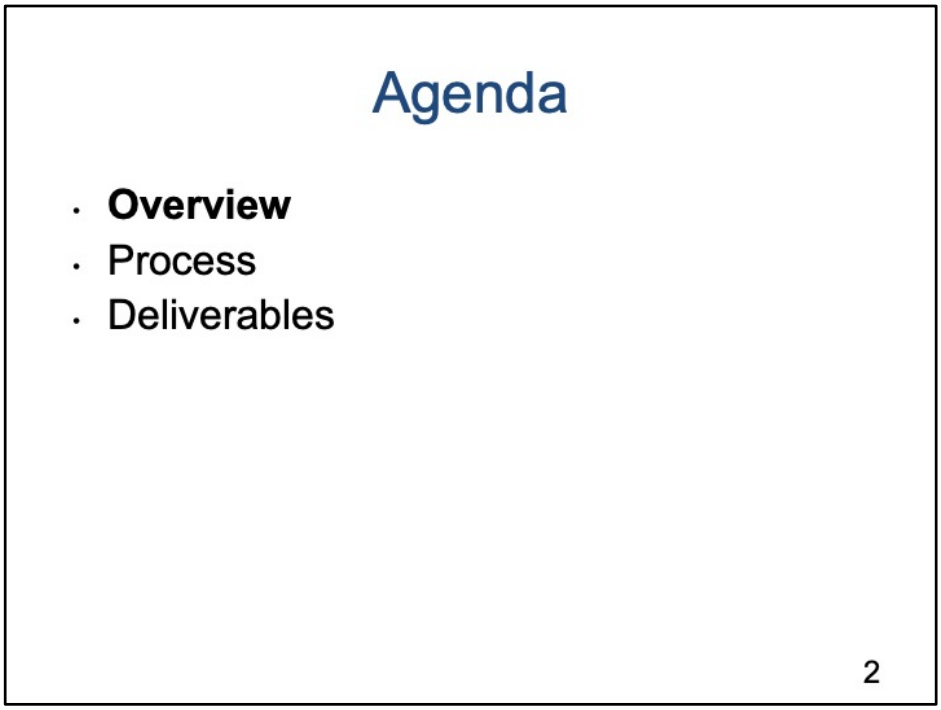
The COS 333 Project

Robert M. Dondero, Ph.D.
Princeton University



Agenda

First I'll give you an overview of the project

Then I'll suggest a process that you might use to develop your project

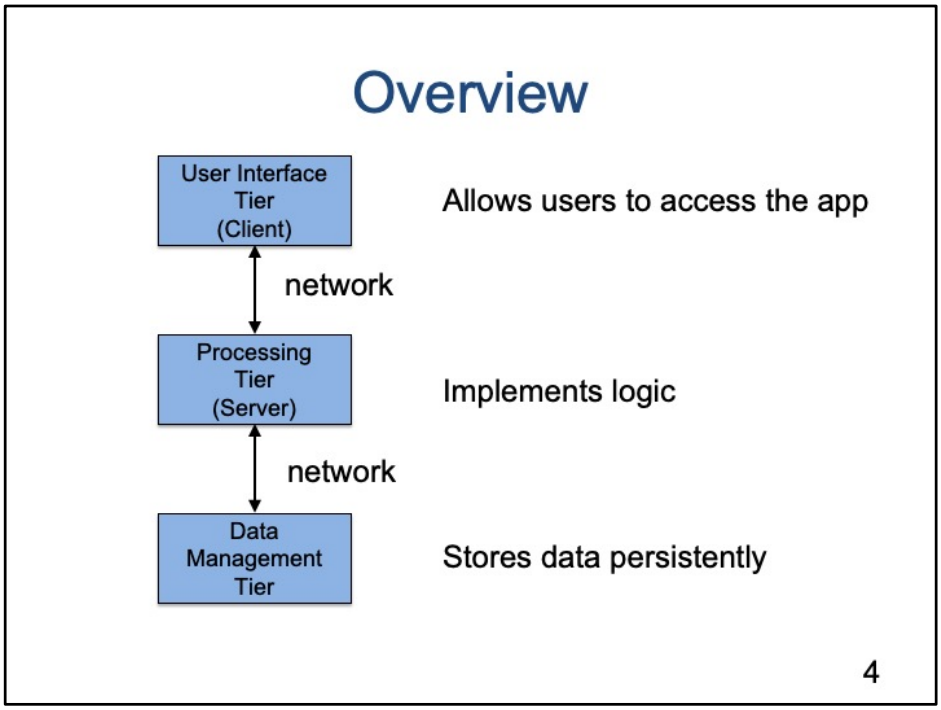
Finally I'll cover the project deliverables

Let's start with the overview…

Overview

- A simulation of reality
- In groups of 3-5 people...

- Build a substantial networked ***three tier*** application



Overview

What is meant by “three-tier”?

[see slide]

Data management tier

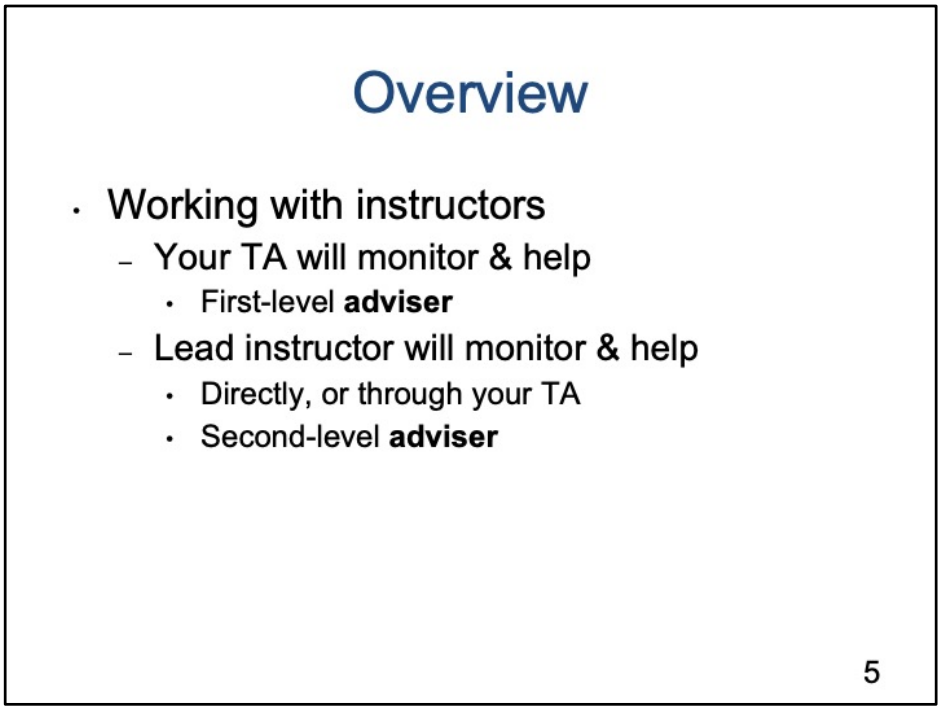
Stores data persistently
Communicates with the file system
Communicates with the processing tier over a network

Processing tier (the **server**)

Implements the application’s logic
Communicates with data management tier over a network
Communicates with user interface tier over a network

User interface tier (the **client)**

Allows users to access the app
Communicates with processing tier over a network
Communicates with user

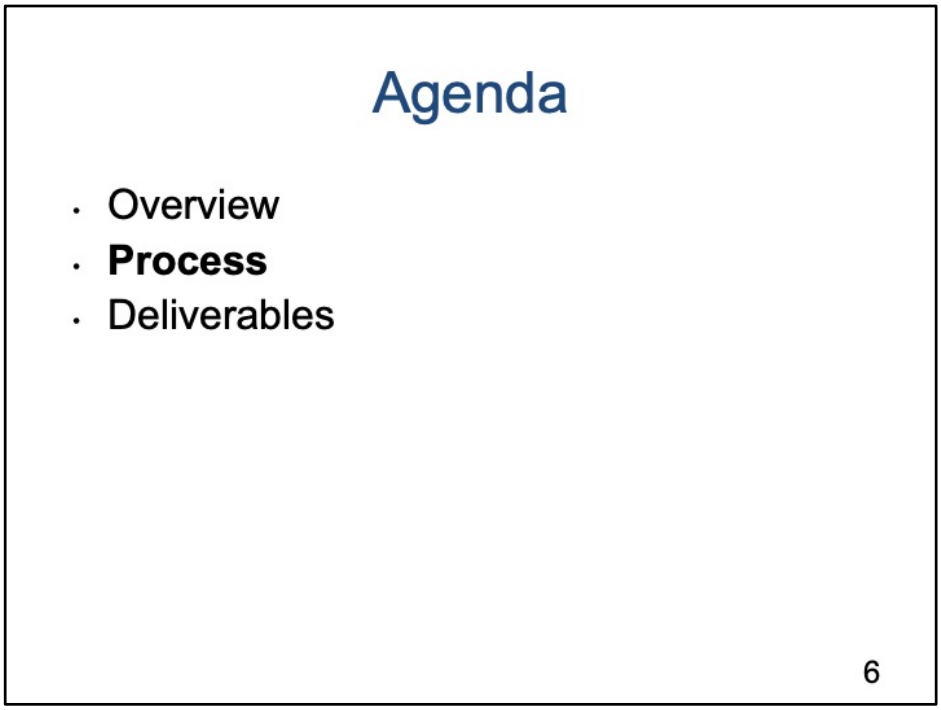


Overview

Working with instructors

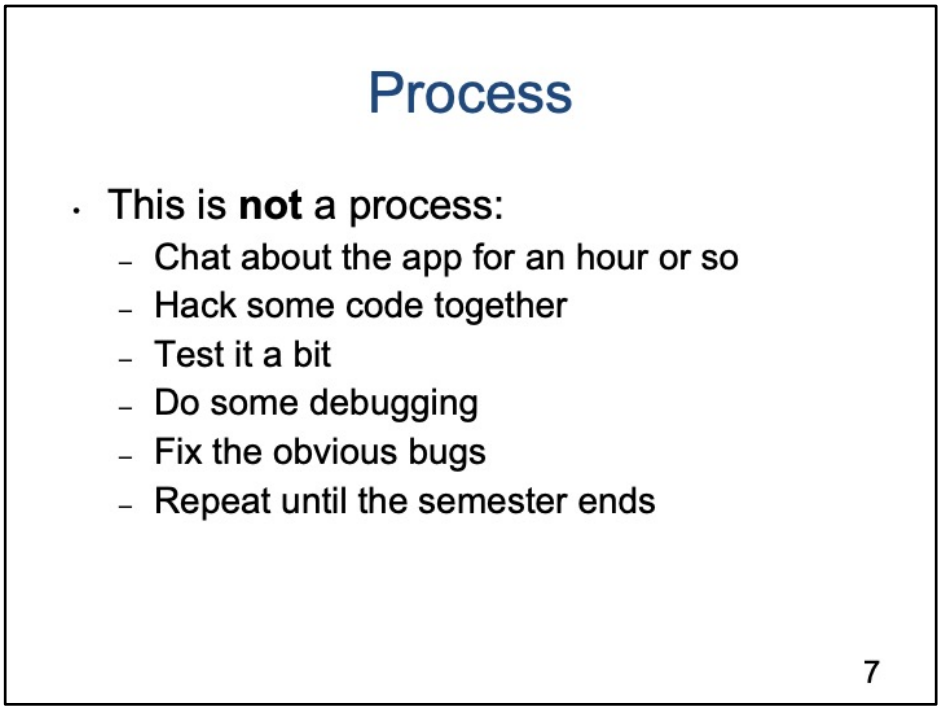
Your graduate student TA will monitor & help
First-level **adviser**
Not your **manager**
It's your project, not the TA's

Lead instructor will monitor & help
Directly, or through your TA
Second-level **adviser**
Again, not your **manager**



Process

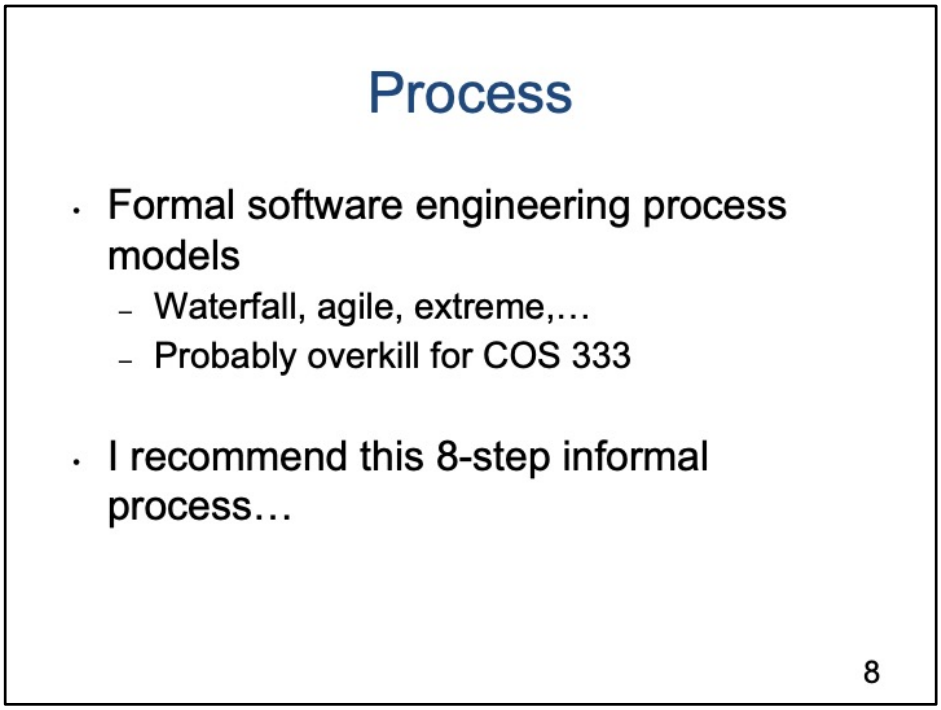
Now let's consider a process that you might use to develop your project…



Process

You must use an orderly process

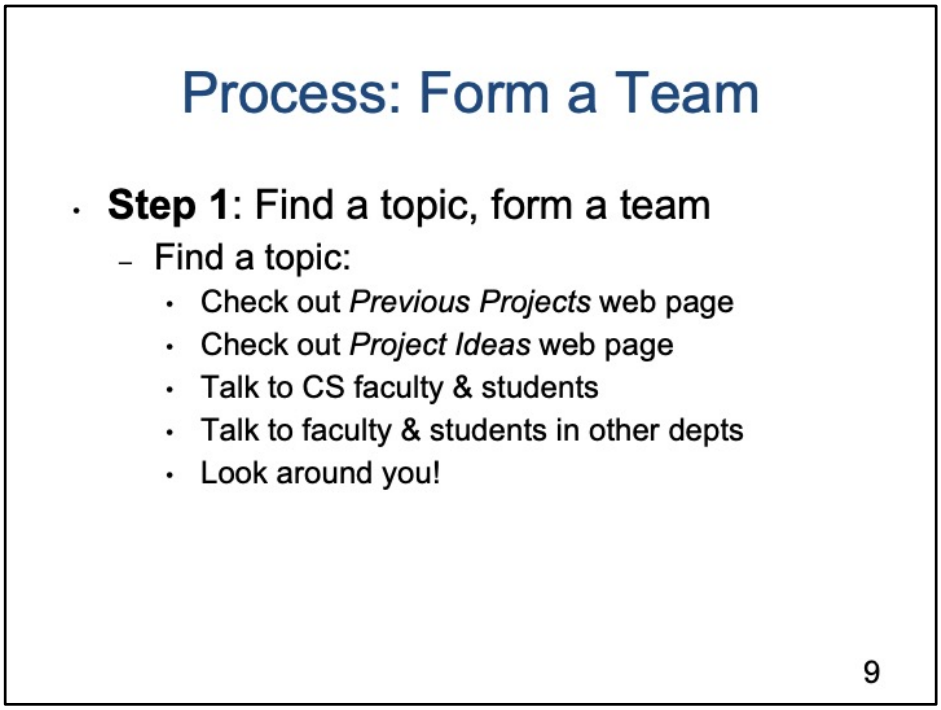
[see slide]



Process

Formal software engineering process models
- Waterfall, agile, extreme,…
- Probably overkill for COS 333
- But some process is essential

I recommend this 8-step informal process…



Process: Form a Team

Step 1: Find a topic, form a team

Find a topic:

Check out *Previous Projects* web page

Check out *Project Ideas* web page

Organizations from all over Princeton (and beyond) know about COS 333, have work to be done, and propose projects to COS 333 students

Such projects are a win-win situation:

The organization gets an application that it needs

The students get a meaningful learning experience, and often the satisfaction of developing an application that transcends the course and really helps people

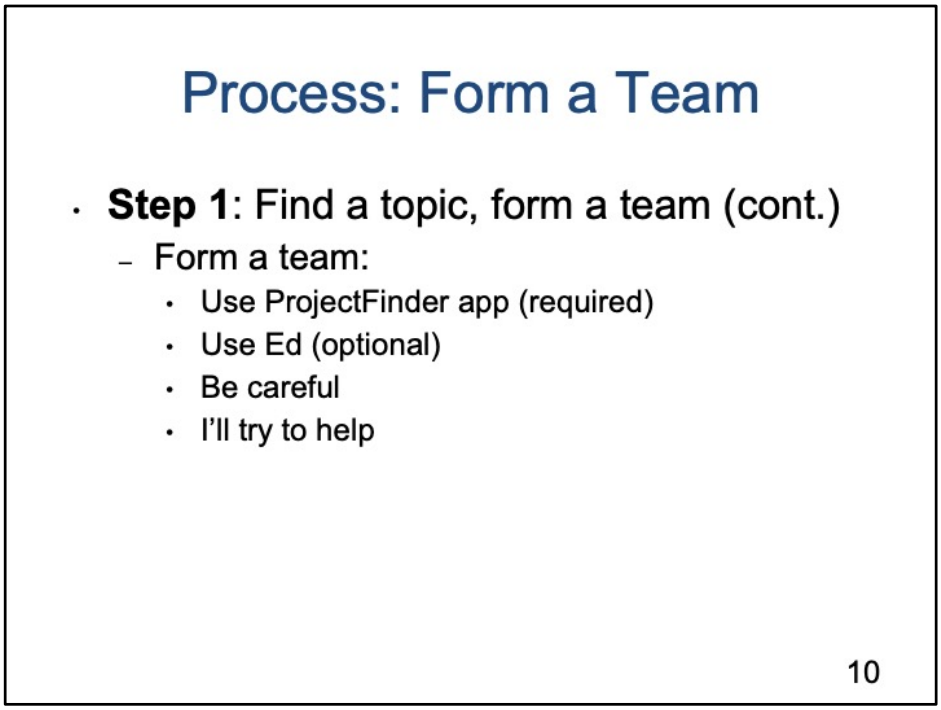
This semester two new ones are listed

Usually first-come-first-served

Talk to CS faculty & students

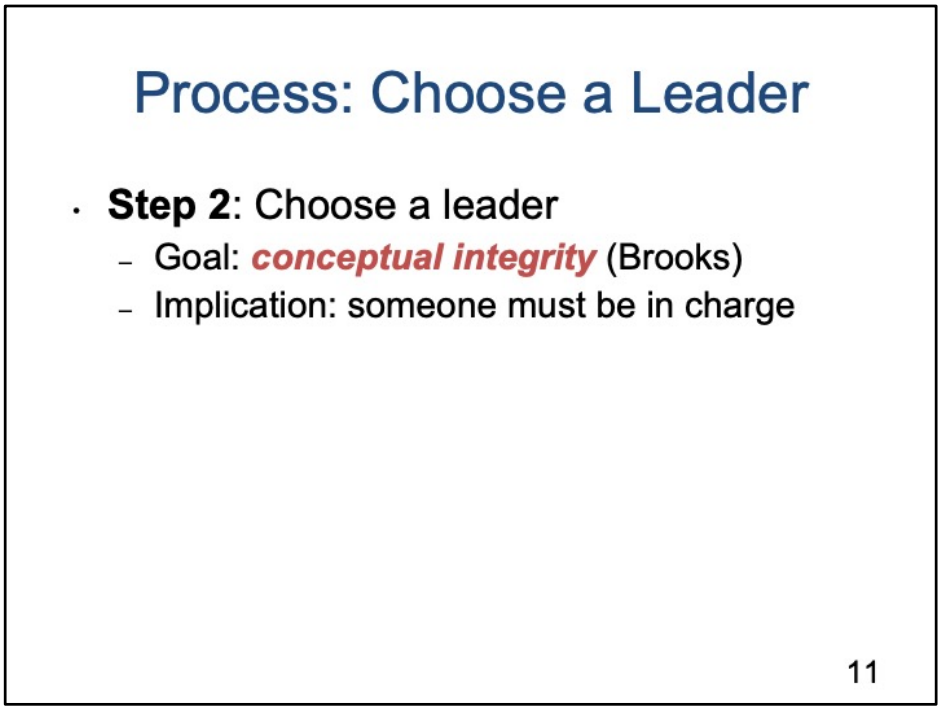
Talk to faculty & students in other depts

Look around you!



Process: Form a Team

Step 1: Find a topic, form a team (cont.)
- Form a team:
  - Use ProjectFinder app (required)
  - Use Ed (optional)
    - Advertise your project
    - Advertise your availability for a project
  - Be careful
    - **Make sure you're comfortable with your teammates**
    - "How many hours per week will you devote to the project?"
  - I'll try to help
    - Stay after this lecture!



Process: Choose a Leader

Step 2: Choose a leader

Goal: ***conceptual integrity*** (Brooks)

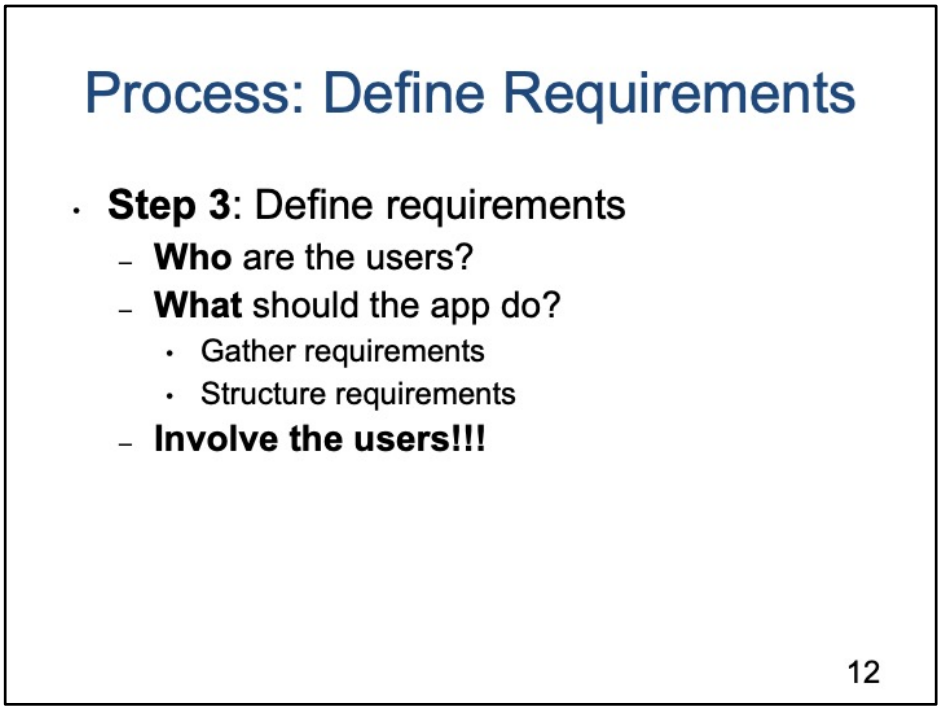
Make app so coherent that it appears to be the product of a single mind

Implications:

Everyone must pull together

Someone must be in charge

Successful software development projects are not democracies



Process: Define Requirements

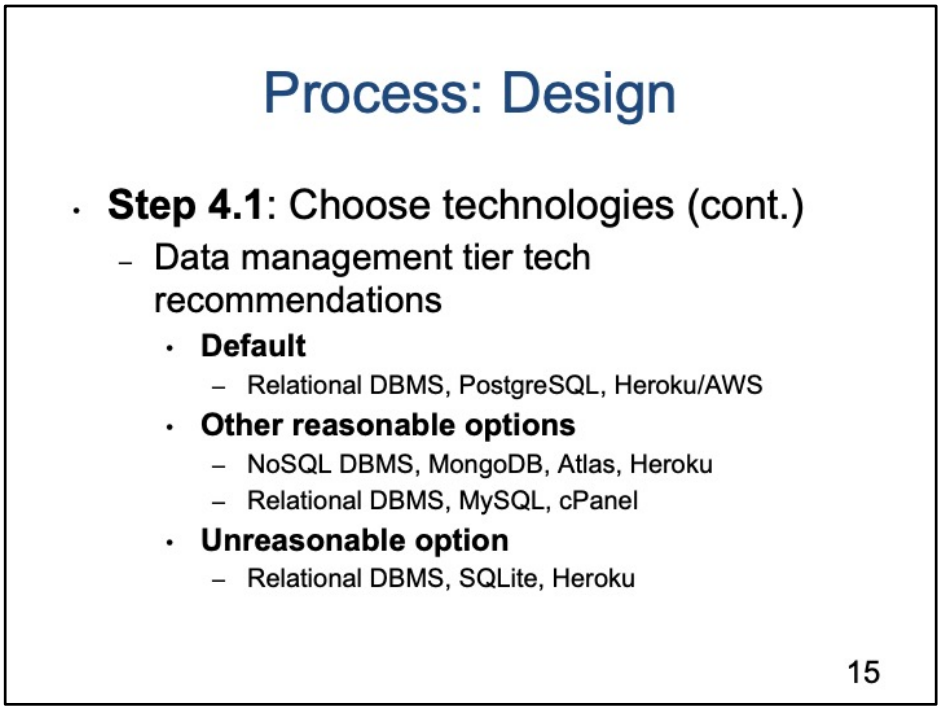
Step 3: Define requirements
- **Who** are the users?
- **What** should the app do?
  - Decide what user need(s) the app will fulfill
  - Gather requirements
    - Conduct interviews, watch users work, …
  - Structure requirements:
    - Compose models of user world, use cases, storyboards, …
- **Involve the users!!!**

Process: Design

- **Step 4**: Design
  - **How** will the app work?

Process: Design

- **Step 4.1**: Choose technologies
  - Data management tier technologies
    - **Data store**
      - Flat files
      - Spreadsheets
      - Relational DBMS: SQLite, MySQL, PostgreSQL, …
      - NoSQL DBMS: Redis, MongoDB, …
    - **Hosting service**: your computer, CS Dept computer, OIT computer, Heroku/AWS, …



Process: Design

[see slide]

The combination SQLite + Heroku is unreasonable
See me if you need to know more

Process: Design

- **Step 4.1**: Choose technologies (cont.)
  - Processing tier (server) technologies
    - **Language**: Python, Java, JavaScript, PHP, C, C++, C#, Perl, Ruby...
    - **Framework**:
      - For Python: Flask, Django, …
      - For Java: Spark, Spring, …
      - For JavaScript: Express, …
    - **Hosting service**: your computer, CS Dept computer, OIT computer, Heroku, AWS, …

Process: Design

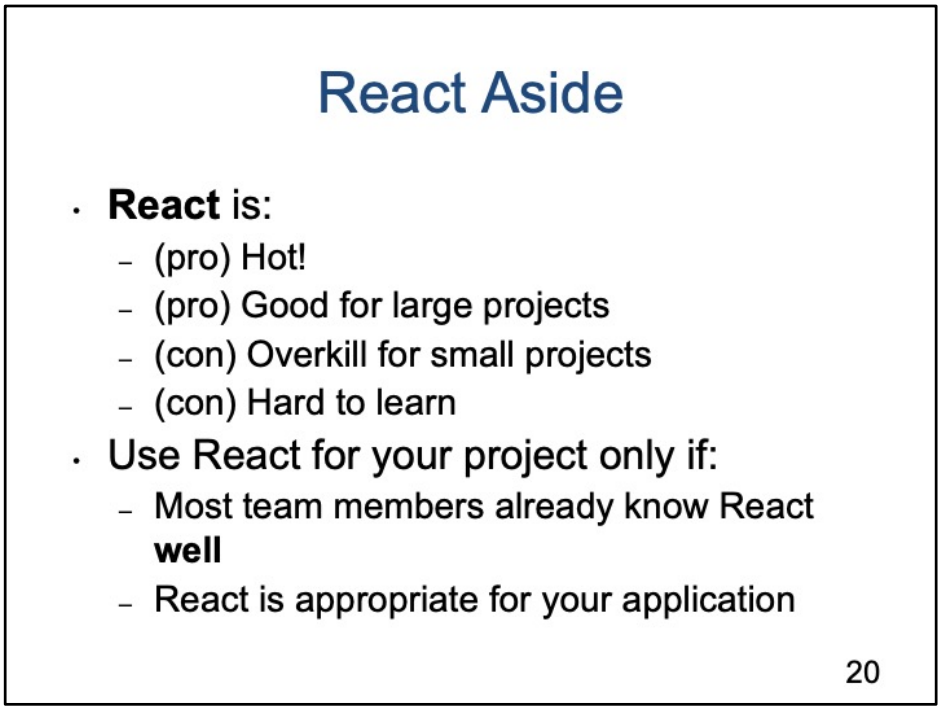
- **Step 4.1**: Choose technologies (cont.)
  - Processing tier tech recommendations
    - **Default**
      - Python, Flask, Heroku
    - **Other reasonable options**
      - Python, Django, Heroku
      - Python, Flask, cPanel
      - JavaScript, Express, Heroku
      - PHP, (no framework), Heroku
      - Java, Spark, Heroku
      - Java, Spring, Heroku

Process: Design

- **Step 4.1**: Choose technologies (cont.)
  - User interface tier (client) technologies
    - **Desktop app**: Python, PyQt5, Java, Swing, …
    - **Web app**: HTML, CSS, Bootstrap, JavaScript, AJAX, jQuery, React, …
    - **Mobile app**: Java, Kotlin, Objective-C, Swift, JavaScript, ReactNative, …

Process: Design

- **Step 4.1**: Choose technologies (cont.)
  - User interface tier tech recommendations
    - **Default**
      - (Responsive) web app, HTML, CSS, Bootstrap, JavaScript, AJAX, jQuery
    - **Other reasonable options**
      - Native mobile app, Java, Android
      - Native mobile app, Swift, iOS
      - Laptop/desktop app, Python, PyQt5
    - **Use with care**
      - (Responsive) web app, HTML, JavaScript, AJAX, **React**



React Aside

React is:

(pro) Hot!
(pro) Good for large projects
(con) Overkill for small projects
(con) Hard to learn

Use React for your COS 333 project only if:

Most team members already know React **well**
React is appropriate for your application

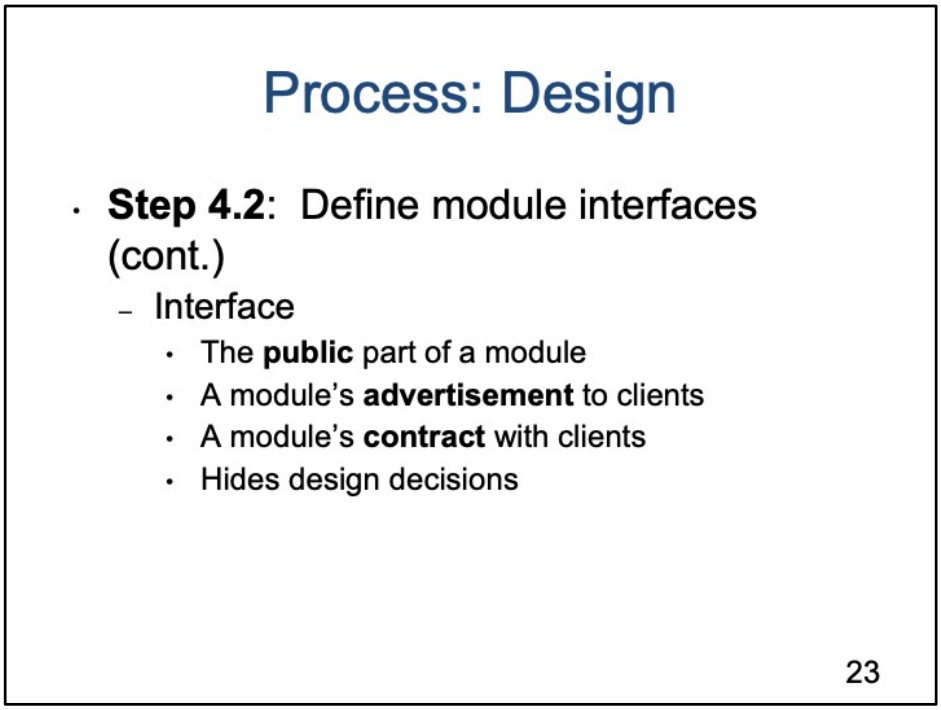
If you and your teammates tell us that you intend to use React, then…
For your sake, we will make you justify your choice

Process: Design

- **Step 4.1**: Choose technologies (cont.)
  - Suggestions for choosing technologies:
    - Talk with course instructors
    - Do many simple tech experiments early
    - Consider:
      - (1) Suitability?
      - (2) Learning curve?
      - (3) Worth learning?

Process: Design

- **Step 4.2**: Define module interfaces
  - Module = interface + implementation



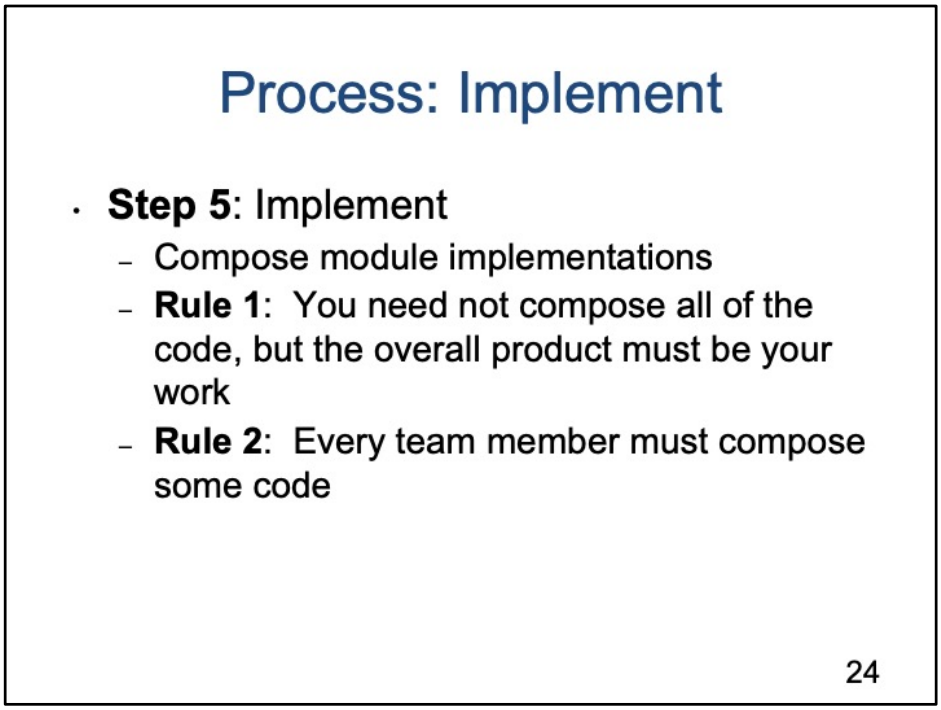
Process: Design

Step 4.2: Define module interfaces (cont.)

Interface
- The **public** part of a module
- A module's **advertisement** to clients
- A module's **contract** with clients
  - What are the inputs?
  - What are the outputs?
  - Who manages resources?
  - Who detects/reports errors?
- Hides design decisions

Common comment: "I wish we had done interfaces better"
Less common comment: "We thought hard about the interfaces so it was easy to make changes without breaking anything"

Try to stay friends!

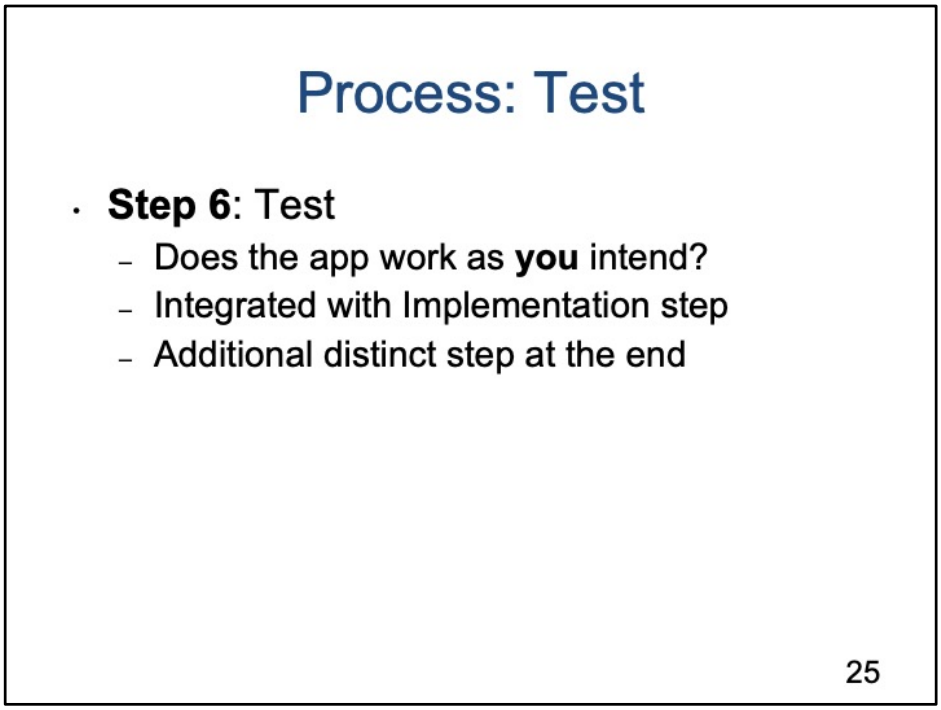


Process: Implement

Step 5: Implement
Compose module implementations

Rule 1: You need not compose all of the code
It's OK to use open-source code, but…
Must identify what you have used, and where it came from
Overall project design must be your work
Most of the code must be your work
(Your TA will help)

Rule 2: Every team member must compose some code
Every team member must compose a substantial amount of code
That includes the project leader



Process: Test

Step 6: Test

Does the app work as **you** intend?

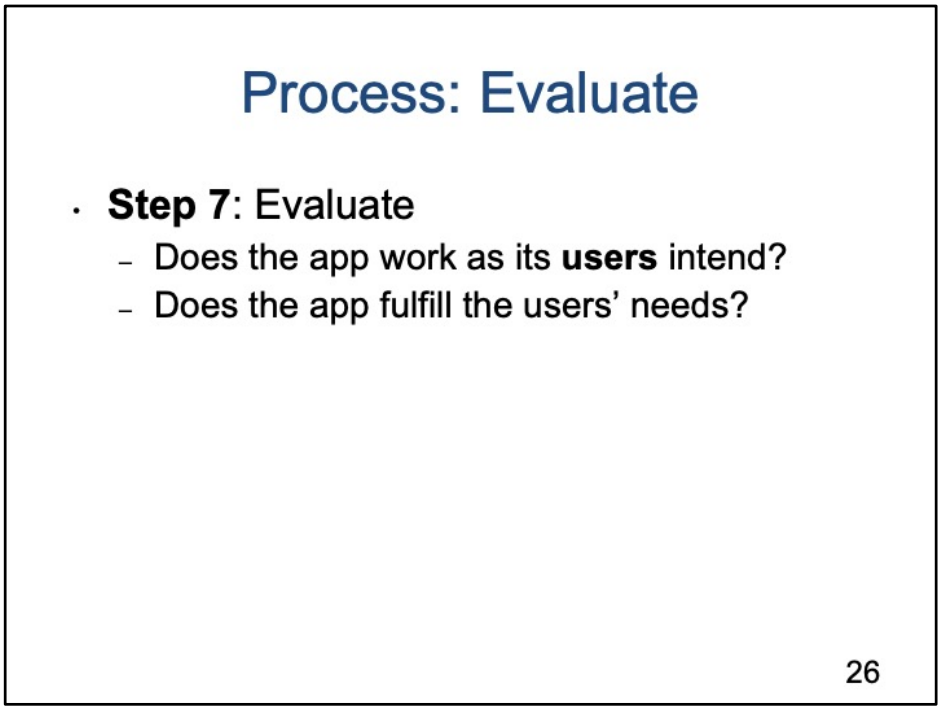
Integrated with Implementation step

Make sure it always works

Fix bugs before adding features

Additional distinct step at the end

Keep in mind the testing taxonomy referenced by the assignment specifications



Process: Evaluate

Step 7: Evaluate

Note that testing and evaluation are quite different!
Does the app work as its **users** intend?
Does the app fulfill the users' needs?
Approaches:

Evaluation by users
Evaluation by experts (you!)

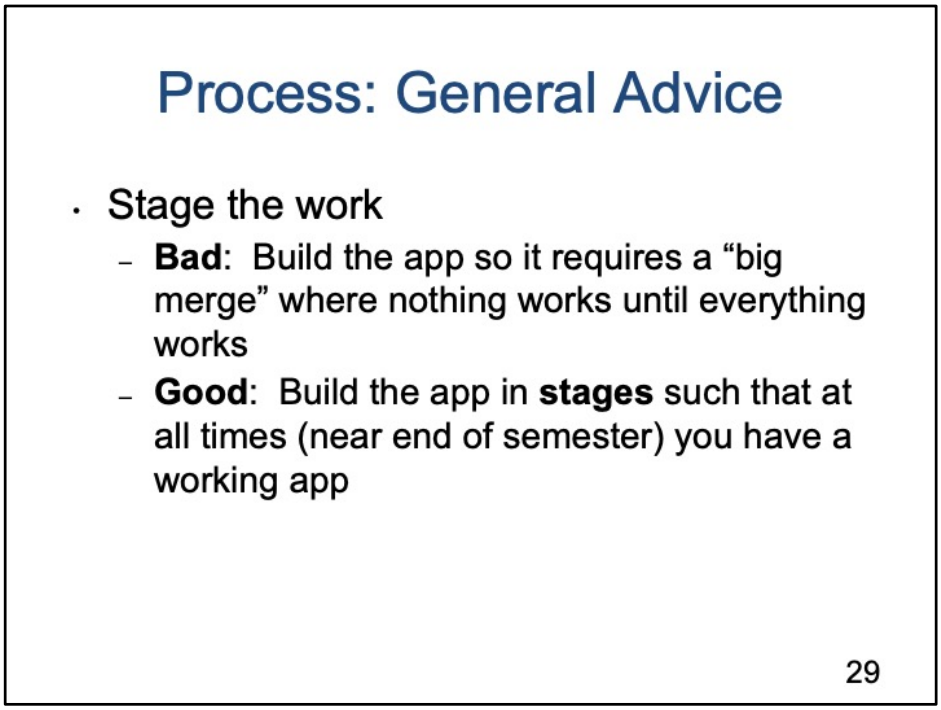
We'll cover evaluation techniques near the end of the course

Process: Document

- **Step 8**: Document
  - Integrated with previous steps
  - Additional distinct step at the end
    - User's guide, programmer's guide, …

Process: General Advice

- Iterate
  - Iterate between **Implement** and **Test** frequently
  - Revisit **Define Requirements** and **Design** less frequently



Process: General Advice

Stage the work

Bad: Build the app so it requires a "big merge" where nothing works until everything works

Good: Build the app in **stages** such that at all times (near end of semester) you have a working app

Bad approach (no staging)

Define the entire app; implement the entire app

What if the semester ends before you're finished???!!!

Good approach (staging)

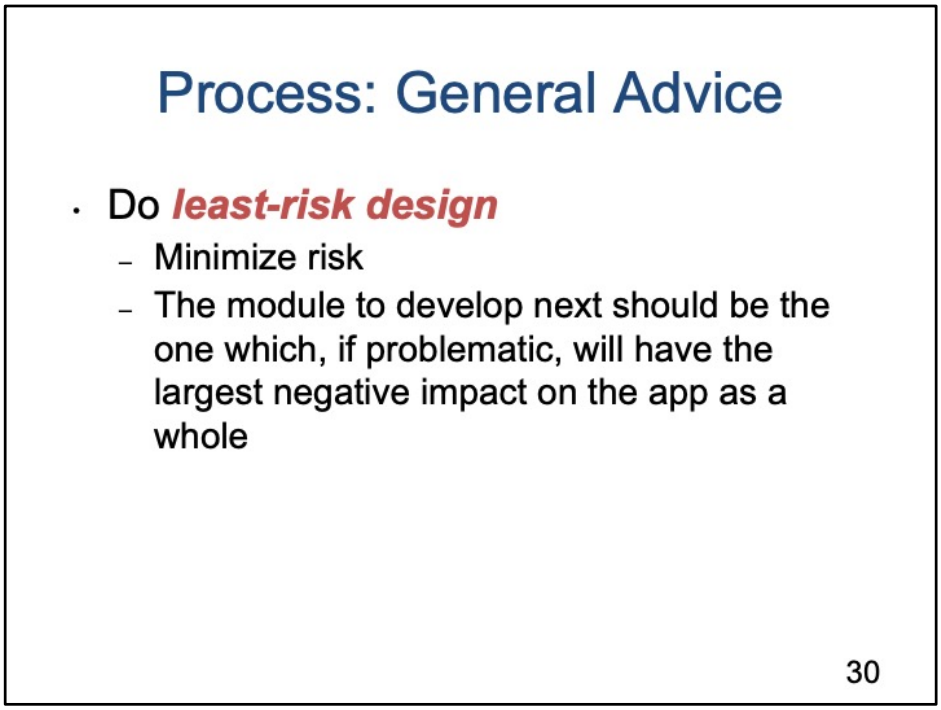
Define the ***minimum viable product (MVP)***; implement the MVP

Define stretch goal 1; enhance the app to implement stretch goal 1

Define stretch goal 2: enhance the app to implement stretch goal 2

…

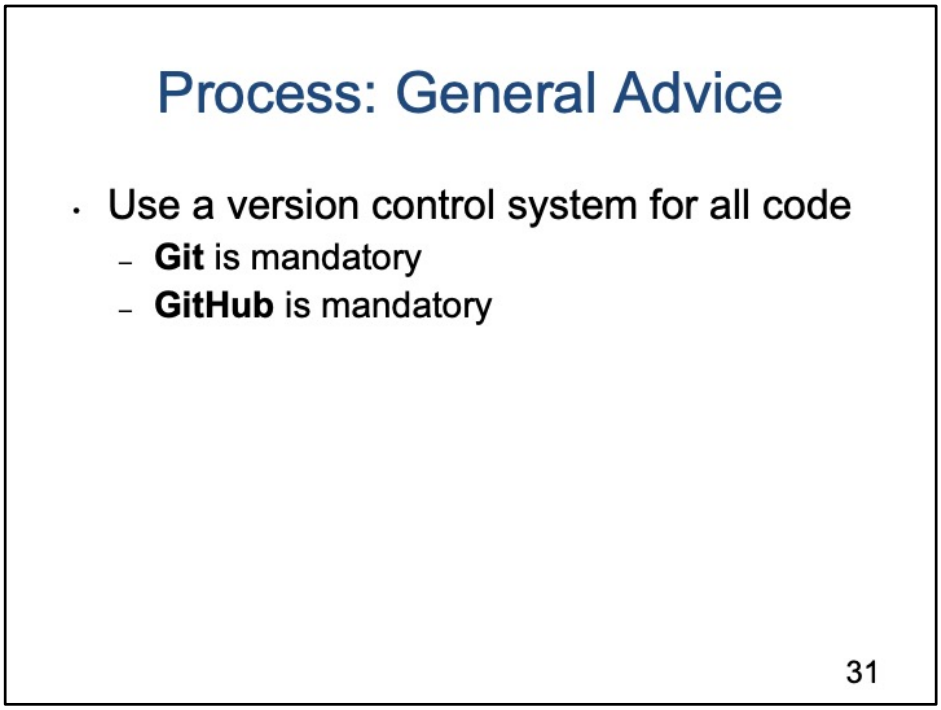
Always be able to stop and declare success



Process

[see slide]

To minimize the risk to your project as a whole, the next module that you develop should be the riskiest one



Process

Use version control system for all code

Git (or, by permission, some other VCS) **is mandatory**

GitHub (or, by permission, some other Internet repository service) **is mandatory**

Process: General Advice

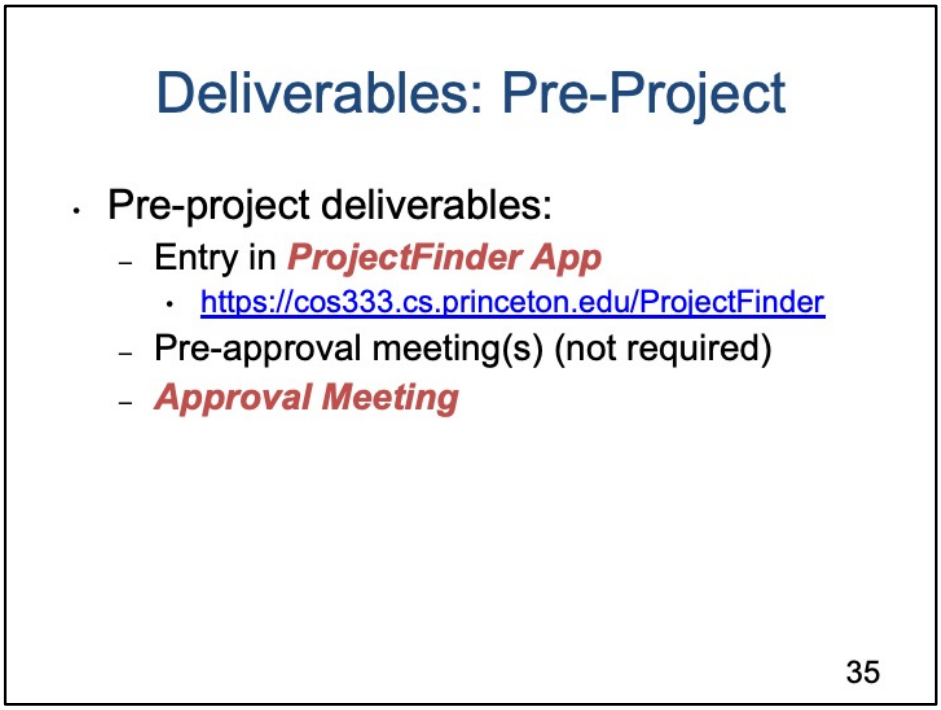
- Allocate time for "overhead" activities
  - **Changing your mind**: You will reverse decisions; you will redo work
  - **Disaster**: Deleted files, broken hardware, ...
  - **Sickness**: You will lose time for unavoidable reasons
  - **Health**: There is more to life than this project!
  - **Deliverables**: You must package your system for delivery…

Agenda

- Overview
- Process
- **Deliverables**

Deliverables

- See *Project* web page for details
- See *Schedule* web page for due dates
- All deliverables are graded



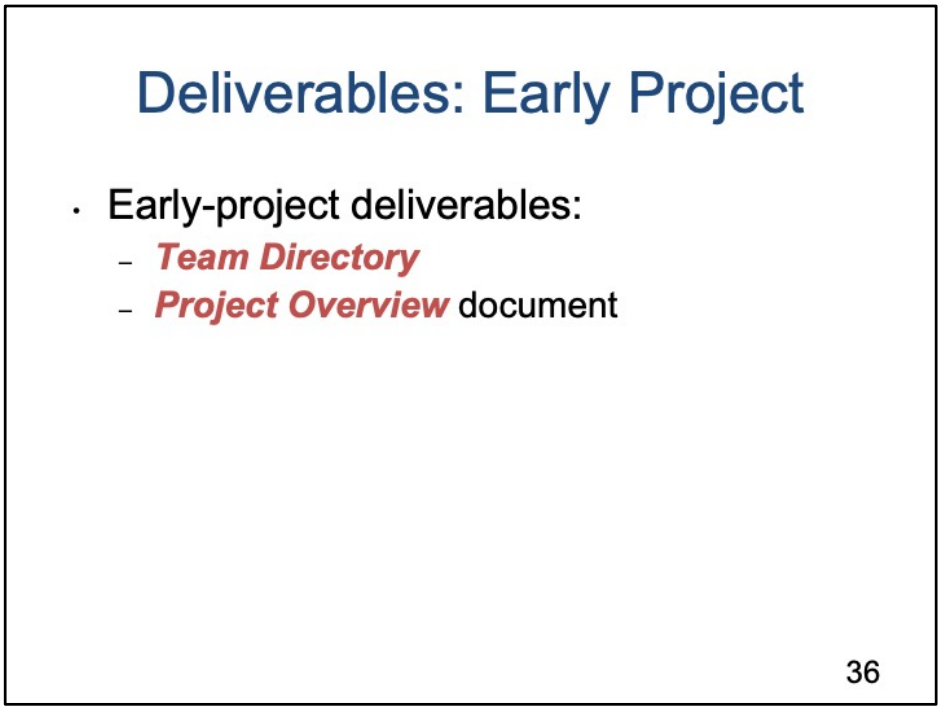
Deliverables

Pre-project deliverables:

Entry in ***ProjectFinder Application***
Goal: Help you form a project team
Goal: Inform others what you're doing
Add row and repeatedly update it to indicate your:
Team status, project topic status, technical interests, project interests,
And eventually…
Project name, project description
https://cos333.cs.princeton.edu/ProjectFinder

Pre-approval meeting(s) (not required)
Meeting with lead instructor to discuss project idea

Approval Meeting
Team meets with lead instructor and TAs, to be sure your project idea is OK
Team presents one reasonably firm consensus idea, not several vague ones



Deliverables

Early-project deliverables:

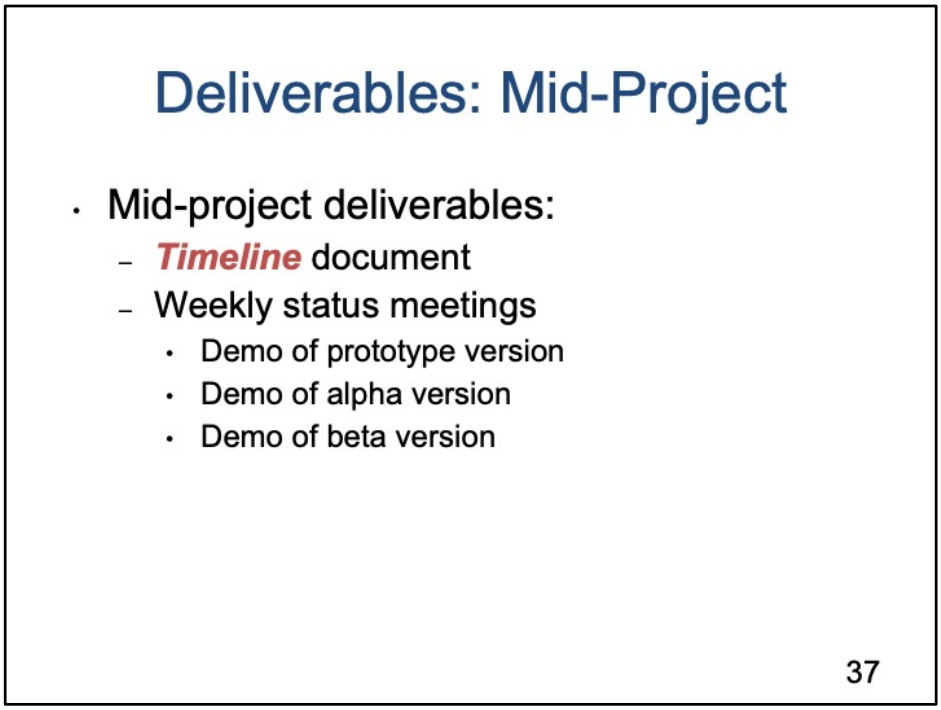
Team Directory

On Google drive of team leader

Project Overview document

In Team Directory

Elevator speech, overview, requirements, functionality, design, milestones, risks



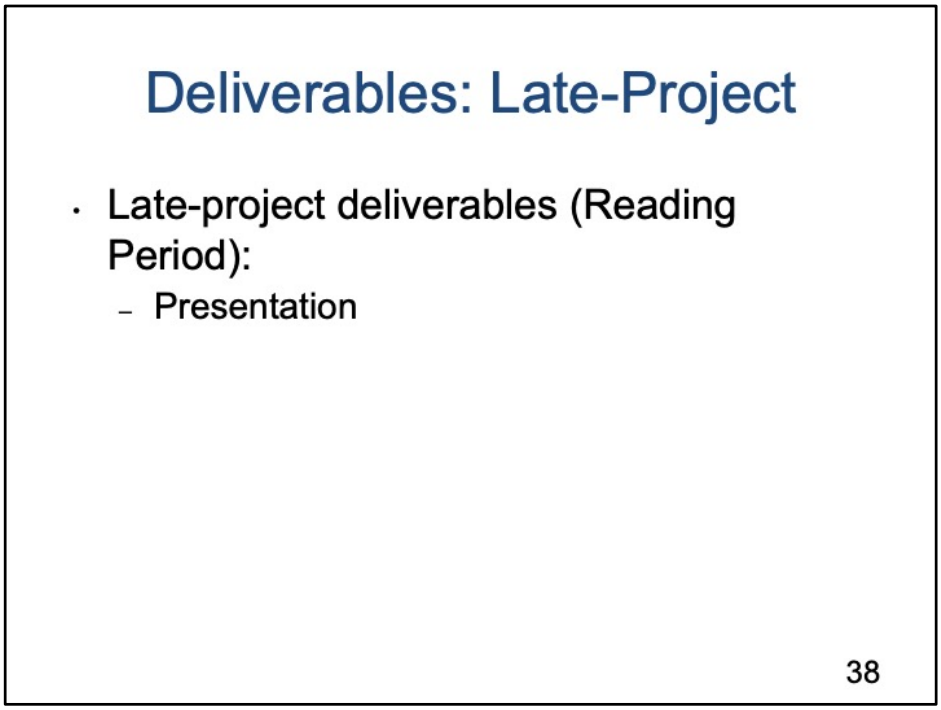
Deliverables

Mid-project deliverables:

Timeline document
In Team Directory
One paragraph for each team member
Updated weekly

Weekly status meetings
Attendance is mandatory
Be prepared to describe what you did, what you plan to do, anticipated risks

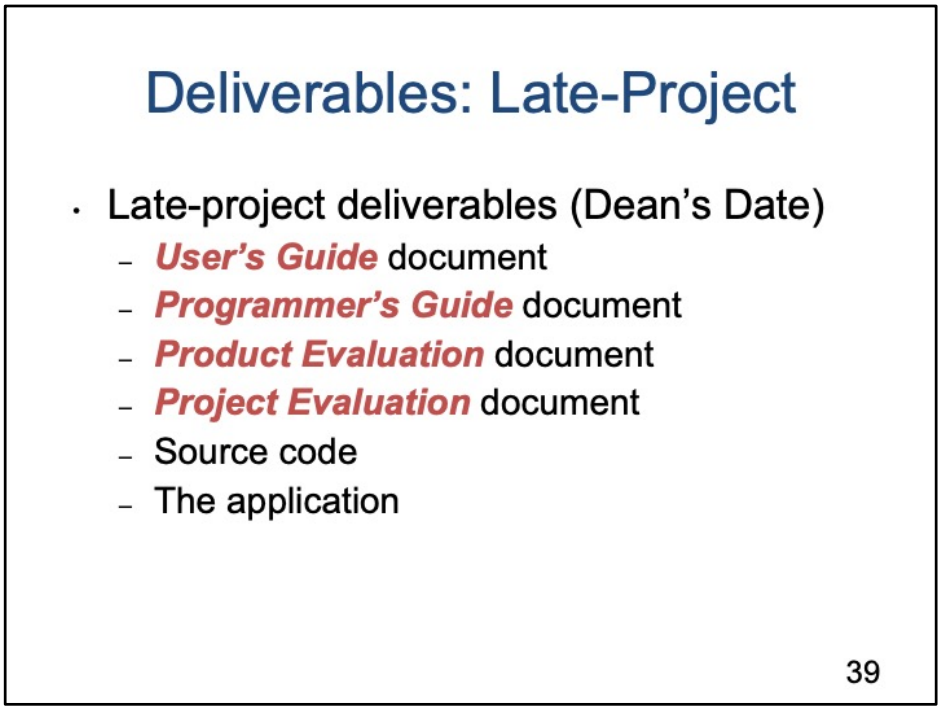
Demo of prototype version
Demo of alpha version
Demo of beta version



Deliverables

Late-project deliverables (during Reading Period):

Presentation
Slides in Team Directory



Deliverables

Late-project deliverables (Dean's Date)

User's Guide document
In Team Directory
Audience: (non-technical) end user
Description of **what your app does**
Probably the most important document

Programmer's Guide document
In Team Directory
Audience: maintenance programmer
Description of **how your app works**

Product Evaluation document
In Team Directory
Results of testing
Does the app work as you intend?
Results of formal evaluation
Does the app fulfill its users' needs?

Project Evaluation document
In Team Directory
What went well? What did not go well? Surprises? Lessons learned? Advice to

future teams?

Source code

In Team Directory

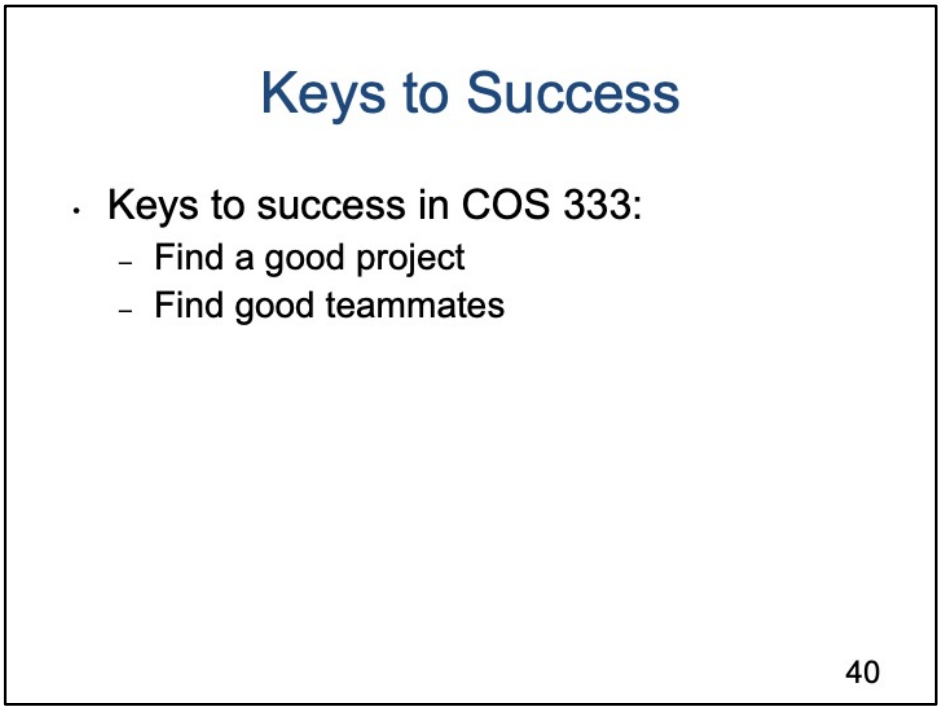
The application

Web app deployed to cloud: give instructors a URL

Web app on your local computer: give instructors a URL and make sure your computer is available

Native Android app: Give instructors installation instructions (and an Android phone???)

Native iOS app: Give instructors installation instructions (and an iPhone???)



Keys to Success

The key to success in COS 333 is your use of the first 3 weeks

During the first 3 weeks you must find (1) a project that interests you, and (2) good teammates

No matter what, I believe the course will be educational for you.
If you find a project that interests you and teammates who are dedicated to the project, then the course also can be very fulfilling, and a lot of fun.

And there's a chance that your work could transcend the course and truly help people.