

# Course Use Cases

- Add Course
  - prefix
  - number
  - credits
  - prereqs tree
  - coreqs tree
- Edit Course
  - edit/change any of the above
- Delete Course
  - delete everything about the course

# Constructing Use Case Based GUI Prototypes

- Constructing them, so that those who are not members of the group/team that constructed them can fully understand them, is not easy.
- Unfortunately, for this reason, many HW8 submissions were not completely understandable to those outside the groups that constructed them.
- So, we're going to adopt a highly stylized way of constructing use case based GUI prototypes and will do a second version of HW8 using it.
- The next slide begins a description of the way that (usually) universally understandable use case based prototypes can be constructed.

# Use Case Text

- The textual description of a use case consists of two parts, the part that describes what happens if everything goes as expected, and the part that describes what happens when it doesn't.
- The part of a use case description that relates what happens when everything goes right is called the “**basic course**” or the “**sunny day course**”.
- The part of a use case description that relates what happens when something goes wrong is called an “**alternate course**” or a “**rainy day course**.”
- The next slide shows the basic course of the generic Create Account use case in the style that we will be using.

# Create Account Use Case: Basic (Sunny Day) Course

User	System
	The system displays the Create Account Page.
The Customer types a userid, an e-mail address, and a password (twice), and then presses the Create Account button.	
	The system ensures that the Customer has provided valid data and then adds an Account to the Master Account Table using that data.
	Then the system returns the Customer to the Home Page.

# Create Account Use Case: Basic Course (cont.)

- As indicated on a previous slide, a use case's "basic course" is a description of how the relevant actor(s) and the application interact when / if everything goes right e.g., in this case if the customer, attempting to create a new account:
  - types a userid
  - types a syntactically valid email address
  - types a valid password
  - types exactly the same password for a second time

## Create Account Use Case: Alternate (Rainy Day) Courses

- The next slide shows what look to be five different alternate courses for the Open Account use case.
- It shows the style that we'll be using to document alternate courses.

# Create Account Use Case: Alternate Courses

user	system
If the Customer clicks the create account button without having entered a userid	
	the system displays an error message to that effect and prompts the Customer to type a name.
If the Customer clicks on the create account button without having entered an email address at all, or entered an invalid email address	
	the system displays an error message to that effect and prompts the Customer to type a different address.
If the Customer clicks on the create account button without having entered a first password at all, or entered an invalid password	
	the system displays an error message to that effect and prompts the Customer to type a longer password.
If the Customer clicks on the create account button without having entered a second password or entered a second password which is different from the first password	
	the system displays an error message to that effect and prompts the Customer to type the password correctly the second time.

# Create Account Use Case: Alternate Courses

- As indicated a few slides back, a use case's "alternative courses" are descriptions of how the application (system) reacts when the actor *doesn't* do "the right thing," e.g., in this case if the customer, attempting to create a new account:
  - doesn't type a userid; i.e., leaves the userid box empty
  - and / or types a syntactically invalid email address
  - and / or types an invalid password
  - and / or doesn't type exactly the same password for a second time



## Alternate Courses (cont.)

- A serious cause of software development failure is failure to identify and document all alternate courses of every use case
- An equally serious cause of software development failure is basic and alternate courses written in incomplete / unclear / ambiguous English – or whatever language is being used
- So, be very careful about these two issues and review your work multiple times before considering it to be done. (It would be best to have all group members check for the two issues listed above.)

But

- We've left out a very important alternate course.
- Can you think of it?

## Missing Alternate Course

- ***If the user has typed a name and an email address of an account that already exist.***
- It's sometimes very hard to come up with every last alternate course, but hopefully you understand how important it is to try

# Use Case Based GUI Prototypes

- Should have one screen shot per line of the use case text – one per line of the basic course's text and one per line of each alternate course's text
- Above the screen should be the line of the use case text
- The screen should show (an example of) the actor doing what the line of the use case text says.
- See the following slides for examples.

The system displays the Create Account Page.

Screen shot of the Create Account  
Page

The Customer types a userid, an e-mail address, and a password (twice), and then presses the Create Account button.

Screen shot of the Create Account Page with a userid, email address, and a password (typed twice) into the appropriate boxes

The system ensures that the Customer has provided valid data and then adds an Account to the Master Account Table using that data.

Then the system returns the Customer to the Home Page.



Screen shot of the home page

If the Customer clicks the create account button  
without having entered a userid

the system displays an error message to that effect  
and prompts the customer to type a name.

Screen shot of the Create Account  
Page with an empty userid box  
and with the error message shown



If the Customer clicks on the create account button  
without having entered an email address at all

Screen shot of the Create Account  
Page with an empty email box and  
with the error message shown

Etc., Etc., Etc.

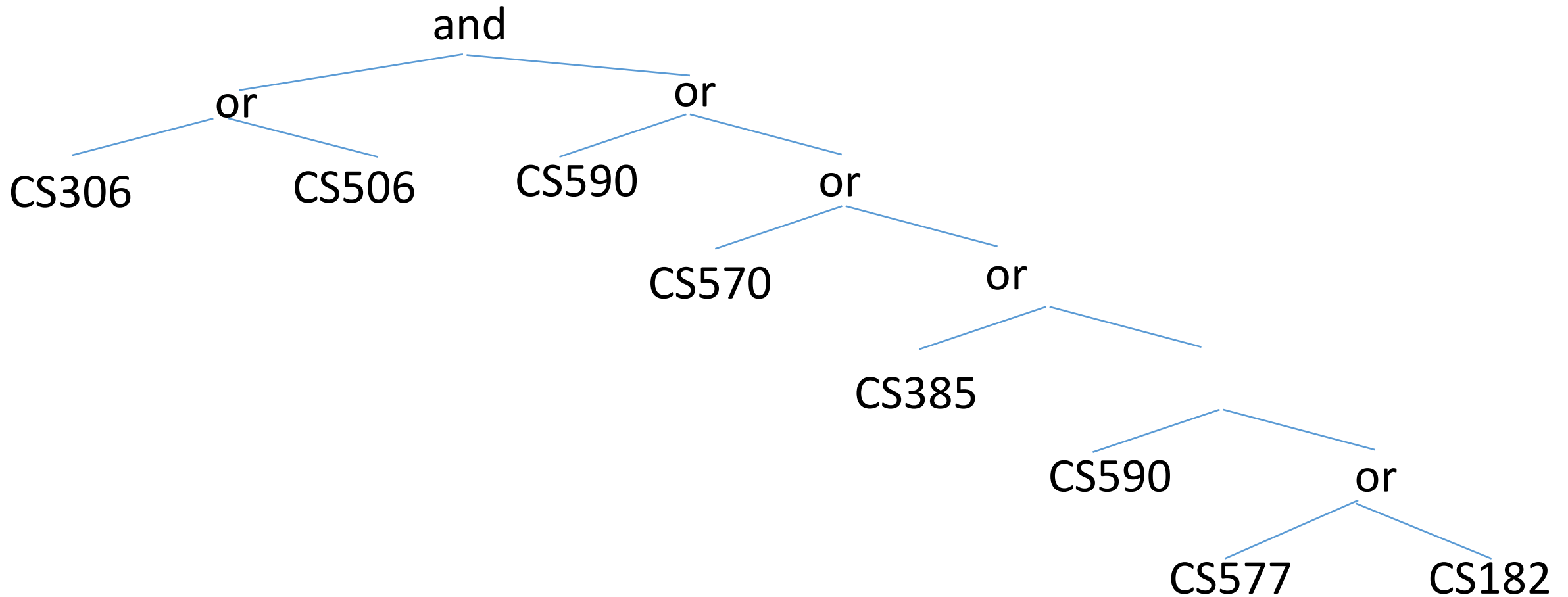
# Comments on Version 1 of the Course Use Case Prototype

- Some groups included a Log In use case in their submissions. No Log In use case was asked for in HW8, so there's no need to supply it in version 2.
- Some groups included details having to do with degree requirements, and/or degree programs in their submissions. This shouldn't be done in version 2, as Add Course, Edit Course, and Delete Course have to be done before courses can be put into degree requirements or degree programs. (Adding, Editing, and Deleting "course groups," that is groups of courses defined by just a prefix, a prefix and a range of numbers, or just a range of numbers must also be done first.)
- Most groups included either no alternate courses or very few of the required alternate courses.
- No groups used anything like the format for use case text and screen shots described above – and every group should use it in version 2.

# Comments on Version 1 of the Course Use Case Prototype

- ***Almost no group described a way of constructing prereq and coreq trees that the readers were able to understand.***
- In version 2, every group should use the highly stylized format described above, for all use cases, including, of course, the details of the use cases in which prereq trees and coreq trees are constructed.
- Recall that the example that we are using for prereqs is the prereqs for CS576. (See next slide.)

Prerequisite Tree for CS576 Secure Systems  
(CS306 or CS506)  
and  
(CS590 or CS570 or CS385 or CS577 or CS182)



# What We'd Like to See in the GUI Prototype Is

- A line from the use case text.
- The relevant screen shot
- And a separate screen shot that shows the current state of the prereqs tree – see next slide.

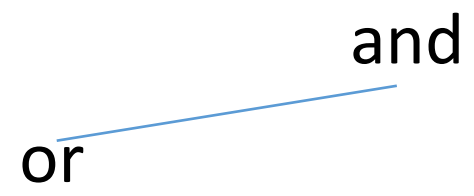
and

# What We'd Like to See Next in the GUI Prototype Is

- A line from the use case text.
- The relevant screen shot
- And a separate screen shot that shows the current state of the prereqs tree – see next slide.



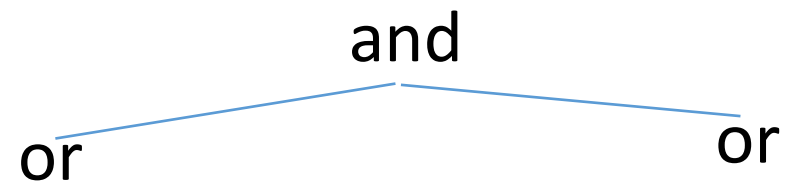
or and

A diagram consisting of the words "or" and "and" in a black sans-serif font. A thin blue line connects the top-right of the word "or" to the bottom-left of the word "and". The line is slightly curved, following the angle of the text.

# What We'd Like to See Next in the GUI Prototype Is

- A line from the use case text.
- The relevant screen shot
- And a separate screen shot that shows the current state of the prereqs tree – see next slide.

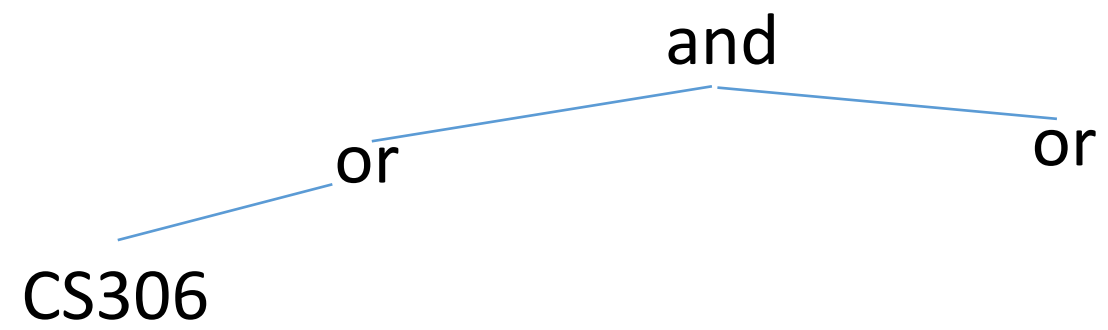
or and or



```
graph TD; A[and] --- B[or]; A --- C[or]
```

# What We'd Like to See Next in the GUI Prototype Is

- A line from the use case text.
- The relevant screen shot
- And a separate screen shot that shows the current state of the prereqs tree – see next slide.



# Etc., Etc., Etc.

PS When the user edits the tree, the user should be able to see the tree, to click on a node where the user wants to make a change, and to then be given choices as to what to change.

# HW9

- Do a second version of the GUI prototype for course, together with a first version of the GUI prototype for group of courses. (There should be tabs for both on the home screen.