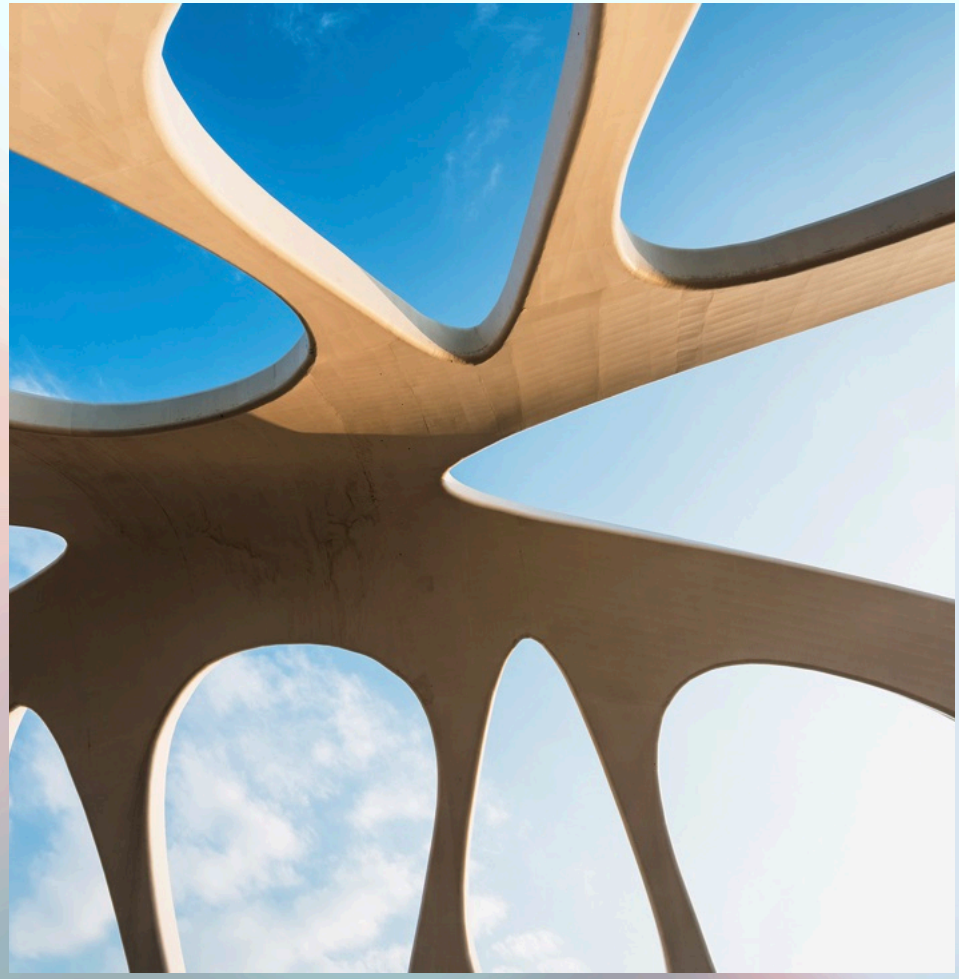
Week 6: Term Project Intro

CSC 667/867

John Roberts

Agenda
Week 7

- Group project details
- Games
- Technical Requirements
- Grading



Group Project Details

- 4 people, NO EXCEPTIONS
  - If you have group issues, going out on your own will earn you a zero

Group Work is Hard

Group Work is Hard

Clear Communication

- Be explicit about everything!
- Set a meeting day each week with your group, and don’t miss it
  - How or where will you meet?
  - Put it on calendars!
- Determine who owns what responsibility
- Clearly define your internal deadlines, and the expectation for what should be delivered

Group Work is Hard

Planning

- Don’t write code before you plan out what code you are writing (ever)
- Define the most discrete, smallest deliverables you can define (and document with words, pictures, drawings, etc.)
- Record your plan!
  - discrete tasks
  - who is responsible for that task
  - what work is in progress

Group Work is Hard

Use Tools

- Code - Pull requests on Github are a great way to socialize your code with the team, make sure everyone is on the same page and understands the code, and get feedback from your teammates
- Tasks - Github has task management built in; this semester we will trial using this as part of your weekly milestones
- Communication - Pick your tool; use Discord if you want to but remember that sometimes face to face communication (if possible) is faster and easier

Group Work is Hard

Peer Grading

- The final term project sum is scaled by a normalized average of your teammate's evaluations of your contribution
  - This does not need to be kept secret - absolutely collaborate and agree that everyone did the same amount of work (if that's what you want to do)
- Peer evaluation will be submitted individually on Canvas, and should contain:
  - FULL NAME of each team member
  - RELATIVE CONTRIBUTION of each team member (should sum to 100%)

Group Work is Hard

Defining Contribution

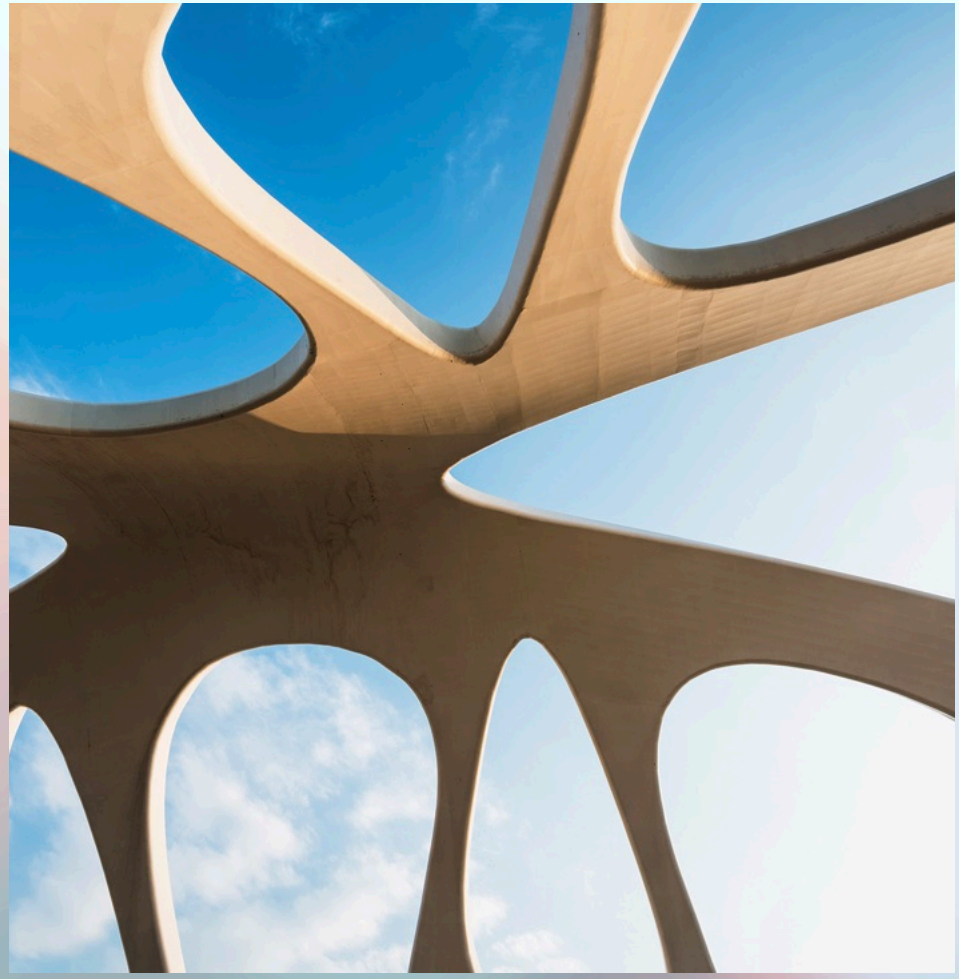
- This is different for different teams, and I don't care what you agree means "equal contribution", so long as you agree
- Don't guess about what your teammates think of your contribution, talk about it and provide feedback (Actionable, Specific, Kind)
- When someone is not contributing, tell them politely and MAKE AN EXPLICIT REQUEST that tells them how to fix their contribution
  - BAD: You need to do more
  - GOOD: You have not submitted any code to the team repository. The team would like you to own Feature X, and we would like to see a pull request daily so we can make sure we are on track to complete this. We think owning this responsibility would be a fair contribution

Group Work is Hard

- Usually, the only reason students fail this course is if they have clearly not contributed to the term project

Agenda
Week 7

- Group project details
- Games
- Technical Requirements
- Grading



What You're Building

- A real time, multi player, online game that supports an arbitrary number of simultaneous games
- We will be building different pieces of the application together in class (instead of talking about specific topics like HTML, CSS, etc., we will just use those technologies together)

What You’re Building

- Many games are not of sufficient complexity to make this an interesting group project, so the set of games you may choose from is limited
- Many libraries exist that make creating some games trivial, so you are not allowed to use a javascript library that is not explicitly discussed in class, unless you ask for - and receive - permission to use it

Permitted Games

Card Games

- Texas Hold’em (or Poker variant)
- Uno
- Crazy Eights (Skip-Bo)
- Gin Rummy
- Sequence
- Golf

Permitted Games

Board Games

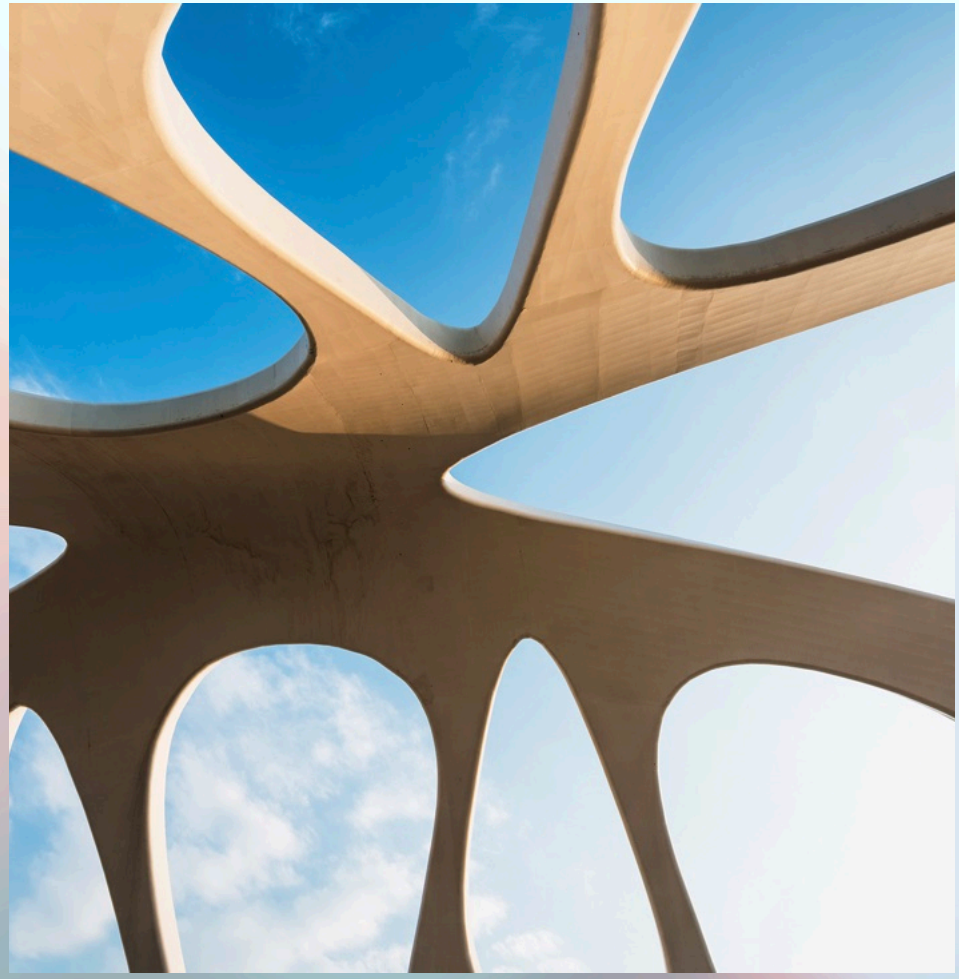
- Sorry
- Clue
- Monopoly
- Scrabble
- Bingo
- Code Names
- Dominoes

Permitted Games

- Is there another card or board game that you are interested in building? Now is the time to discuss if it will be permitted.

Agenda
Week 7

- Group project details
- Games
- Technical Requirements
- Grading



Requirements

User Authentication

- Users must be able to
  - create accounts
  - log in
  - log out
- Most pages on the site must prevent access to users who have not logged in
- Some pages on the site may need to prevent further access to specific users (i.e. users not in a game should not have access to the game unless observers are permitted)

Requirements

Real Time Chat

- Chat must be enabled on the game creation page (the home page after users have authenticated)
- Chat must be enabled in each game room, for those users participating in a game

Requirements

Game State

- All game state must be persisted in a database on the server (not in the client)
  - Local storage is not allowed - the server must be the source of truth for the state of a game at any point in time
- If a user closes a game tab, and then reconnects to the game, the game must be able to be reloaded in the current state for that user
- Only relevant game state should be sent to each user (i.e. don’t send all player’s card hands to all other players)
- Game state must be updated in real time in response to user events and interaction with the game

Requirements

Arbitrary Number of Games

- The application must support an arbitrary (infinite) number of games
- Any user must be able to participate in any number of games
  - For example, I should be able to join multiple games, and play each game in a separate browser tab

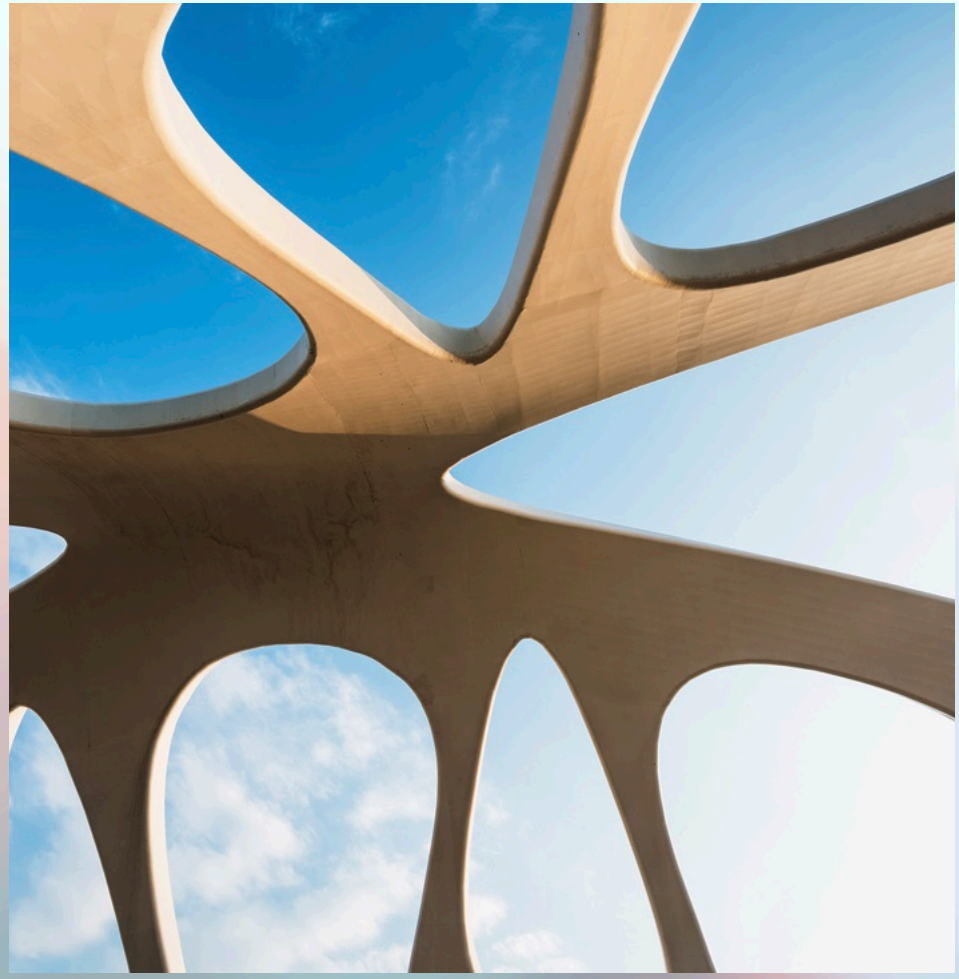
Requirements

Appearance

- We are not graphic designers, so lets just aim at not having our application look terrible

Agenda
Week 7

- Group project details
- Games
- Technical Requirements
- Grading



Grading

- Weekly milestones will be required that will be a single submission from the group including:
  - What was accomplished that week
  - A link to the issues and planning board on your GitHub project (you must use this for planning)
- Code Quality (I simply sample the files in your repository, so be consistent)
- Functionality
- Presentation

Grading

Graded Item	% Score	
Milestone submission	20%	Either you submit or you don't
Code Quality	10%	Your code should be readable - well formatted, no tabs, reasonable white space, minimal comments, etc.
Functionality	40%	How many of the requirements did you meet?
Presentation	30%	

Rubric

Milestone Submission

Term Project Rubric

Category	Description			Notes
Milestone Submission		**20**		(-varies) Depends on number of milestones we get to and how many were submitted on time
POINTS		**20**		

Rubric

Code Quality

Term Project Rubric

Category	Description			Notes
Code Quality	Code is clean, well formatted (appropriate white space and indentation)	**5**		(-1/per) Minor whitespace issues (missing whitespace between logical sections, excessive whitespace, functions, occasional misalignment, run-together code) (-3) Major whitespace issues (inconsistent formatting, major misalignment, little use of whitespace, debug statements, inlined control flow) (-1) Tabs used anywhere in the codebase (-1) Failed to use explicit blocks (-1) Compile warning when compiling on command line (-1) if(expression) return true; else return false should just return expression (-1) Debug output (-1) unused code paths (i.e. empty else block) WRITE EXPLICIT CODE! Things like `input && input.addEventListener()` requires implicit knowledge about the possible states of input (and javascript)… increases readability dramatically to replace that with an if with an explicit comparison: `if(input !== null) { input.addEventListener() }`
	Modules, methods, and variables are meaningfully named (no comments exist to explain functionality - the identifiers serve that purpose)	**2**		(-1) Unnecessary comments (-2) Poorly selected identifiers (variable names that are not intention revealing)
	Methods/Modules/Classes are small and serve a single purpose	**2**		(-1) Method/module/class is run on (contains multiple responsibilities that should have been refactored into separate functions) (-2) Poor class design (Single responsibility principle [SRP] violation)
	Code is well organized into a meaningful file structure	**1**		(-1) Should use modules to organize related code (i.e. config, response, etc.)
POINTS		**10**		

Rubric

Functionality

Term Project Rubric

Category	Description			Notes
Functionality	Authentication & Sessions	**5**		Users can create account, login, logout. Access is prevented to pages specific users should not have access to
	Chat - Lobby	**1**		Chat on the authenticated home page works
	Chat - Game	**1**		Chat in each of the game rooms works
	Game state persistence	**10**		The state of the game is persisted on the server, and can be used to recreate a game when a user returns
	Game data flow	**8**		- Server sends HTML pages that contain page structure, CSS, JS, etc. - State changes are submitted asynchronously to the server using HTTP requests - Game updates in the client are made when the client receives a push notification from the server (the server is the source of truth for game state; the client should not hold any state information)
	Game functionality	**15**		The game works (with a reasonable set of functionality)
POINTS		**40**		

Rubric

Presentation

Term Project Rubric

Category	Description			Notes
Presentation	Slides are legible	**1**		
	Slides have no spelling or grammar errors	**1**		
	Voice level of presenter is appropriate	**1**		
	The presenter can be understood	**1**		
	The presentation begins with an overview (agenda)	**4**		
	The presentation is organized (using the agenda as a guide)	**5**		
	Sufficient technical details are provided	**4**		
	Discussion of any difficulties encountered	**2**		
	Discussion of learnings	**2**		
	An organized demo of the game is presented (~3 minutes)	**7**		
	The presentations is between 10 and 12 minutes	**2**		
POINTS		**30**		