Individual Assignment 2

COMS W4170: User Interface Design (Fall 2025)

DUE: Friday, Sept. 26, 2025 at 11:59pm ET

(Last updated Sept. 4, 2024. Changes since first release highlighted in yellow)

Part 1: Affinity Diagramming	15 pts
Part 2: Styling and Dropdown Menu	25 pts
Part 3: To-Do List Functionality	30 pts
Part 4: Resetting and Saving	30 pts
TOTAL:	100 pts

OVERVIEW

In this assignment, we will practice affinity diagramming (Part 1) and build a fully functioning to-do list web application using HTML, CSS, and JavaScript (Parts 2–4). Starting with the static HTML/CSS layout from Individual Assignment 1, we will transform it into a complete, interactive application. By the end, we will have hands-on experience with event handling, DOM manipulation, and enhancing interactivity in web applications.

SUBMISSION INSTRUCTIONS:

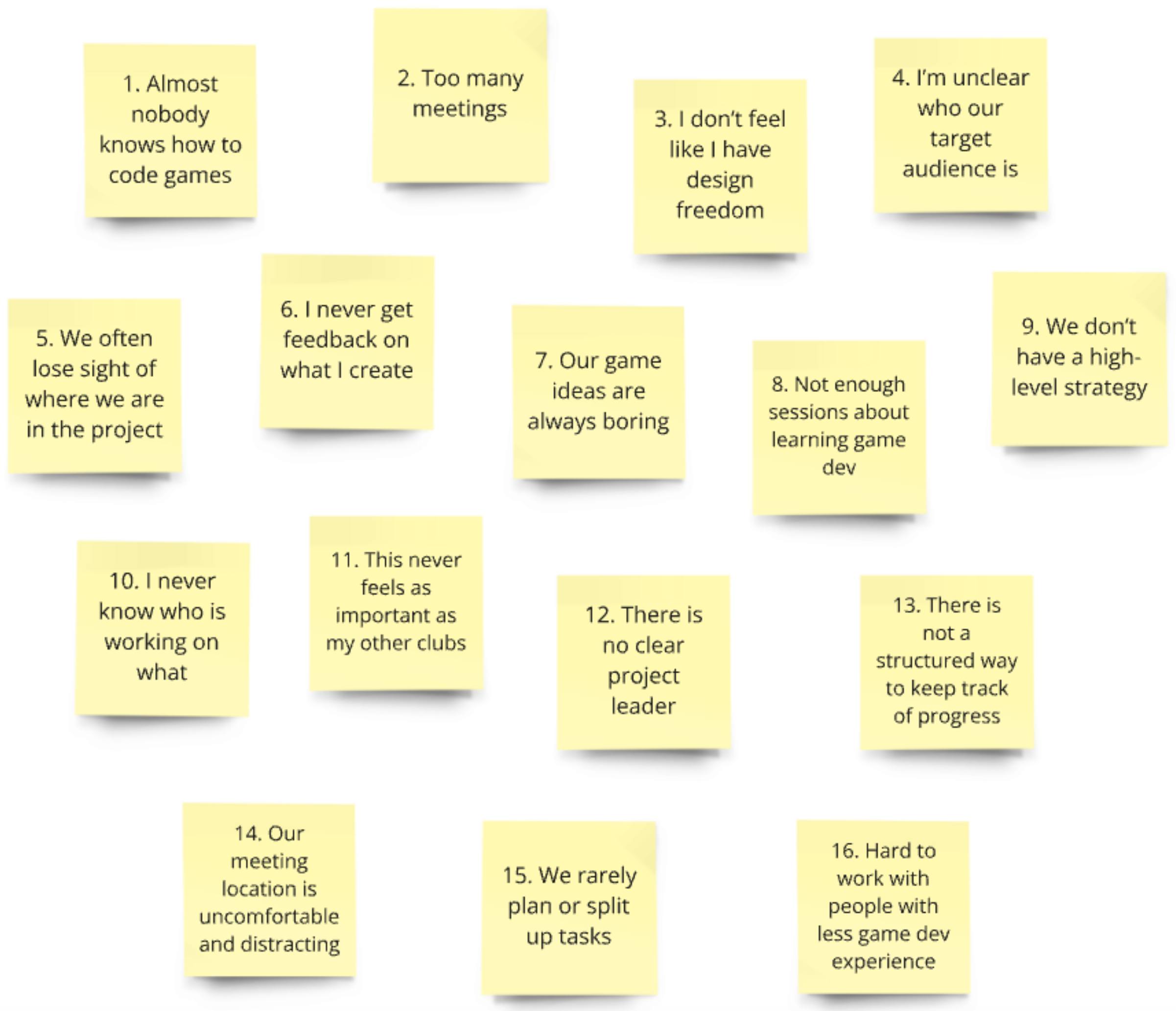
Please submit all of your files electronically via Gradescope. For ease of grading, we ask that you submit your files directly and that you do *not* submit them as a .zip file. If there are any special instructions for running your code, you should include them in a file named `README.txt`. If you have trouble submitting any of your files, please post a question to Ed Discussion and/or come to our office hours.

PART 1: AFFINITY DIAGRAMMING

(15 points)

In this Part, we will cluster the following observation notes into logical clusters (i.e., cluster "high affinity" notes together) using the process we described in class. These observation notes are fictional, but suppose they were taken from a contextual inquiry that sought to answer the broader question of, "What is wrong with the game development club now?" Suppose the notes came from a contextual inquiry featuring several different participants. The affinity diagram that we will create should help inspire potential design solutions for improving the game development club.

Observation notes are usually not numbered, but we numbered them to make it easier for all of us to keep track of them. You can find a printable version of these on the CourseWorks assignment and a Miro board (electronic version) that you can copy here.



1. Almost nobody knows how to code games

2. Too many meetings

3. I don't feel like I have design freedom

4. I'm unclear who our target audience is

5. We often lose sight of where we are in the project

6. I never get feedback on what I create

7. Our game ideas are always boring

8. Not enough sessions about learning game dev

9. We don't have a high-level strategy

10. I never know who is working on what

11. This never feels as important as my other clubs

12. There is no clear project leader

13. There is not a structured way to keep track of progress

14. Our meeting location is uncomfortable and distracting

15. We rarely plan or split up tasks

16. Hard to work with people with less game dev experience

To get credit for the problems below, submit the affinity diagram as a photograph, PDF, or image. In all cases, it should be possible for us to clearly read all of the labels.

1. Group these notes into clusters, and write a summary label (traditionally a blue sticky note) for each cluster. The summary labels should be succinct and written in the first person as if the user's voice is speaking from the "wall."
   (7 pts.)

2. In at least a few sentences, describe why you grouped the notes how you did.
   (2 pts.)

3. Now, group your summary labels (blue sticky notes) into at least two broader themes, and write a summary label for each theme. You should use a new for the theme labels (we used pink in class). As with the blue summary labels, these themes should be written succinctly in the first person.
   (2 pts.)

4. In at least a few sentences, describe your reasoning behind how you grouped the clusters into broader themes.
   (2 pts.)

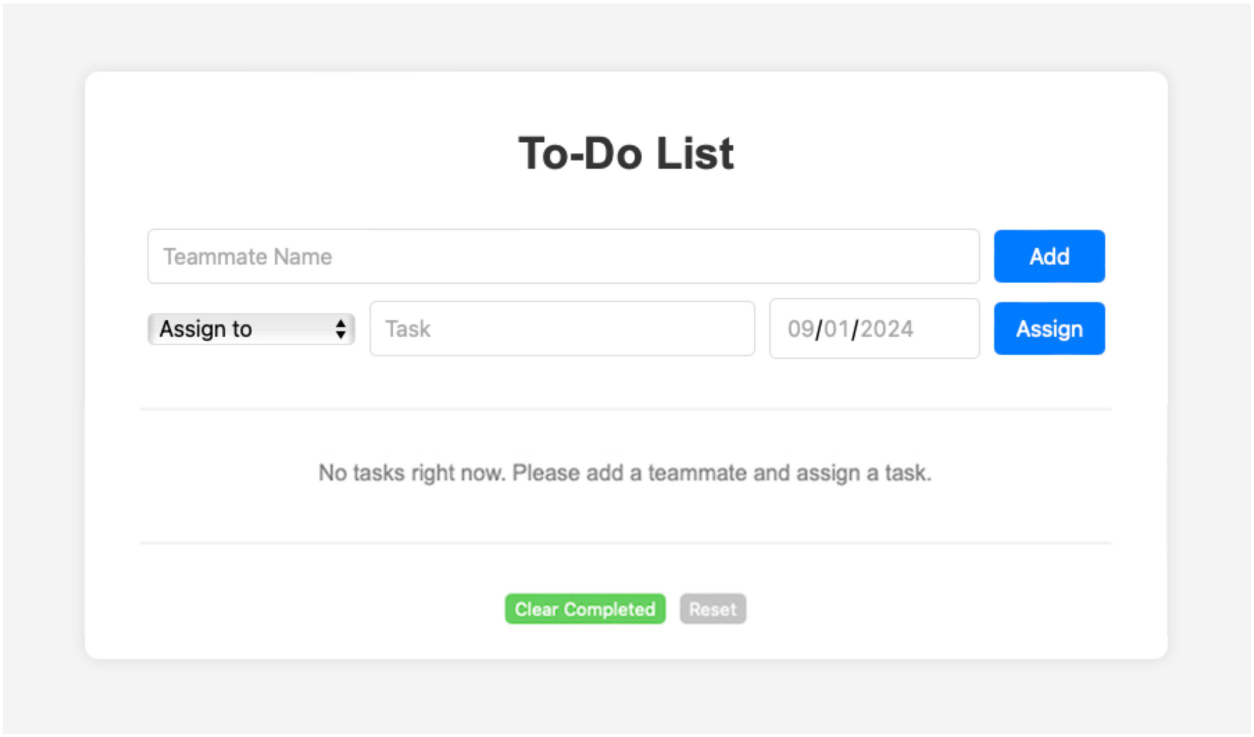
5. Name at least two ways to improve the game development club that follow from the affinity diagram.
   (2 pts.)

PART 2: STYLING AND DROPDOWN MENU

(25 points)

In this Part, we will create the layout for an empty to-do list and make the “Add” button functional. Once this is done, the user will be able to enter a teammate's name into the input field and click the "Add" button to make that name become a selectable option in the dropdown menu. This will allow the user to easily select team members for new to-do items. Please note that all of your JavaScript code should be in a separate `.js` file.

1. Create the following layout with HTML and CSS. You can copy over your code from Individual Assignment 1 and modify parts of it to match what is shown below. The code that you cut out will become helpful later on. The input field should have the text “Teammate Name” as placeholder text when it is empty.
   (3 pts.)



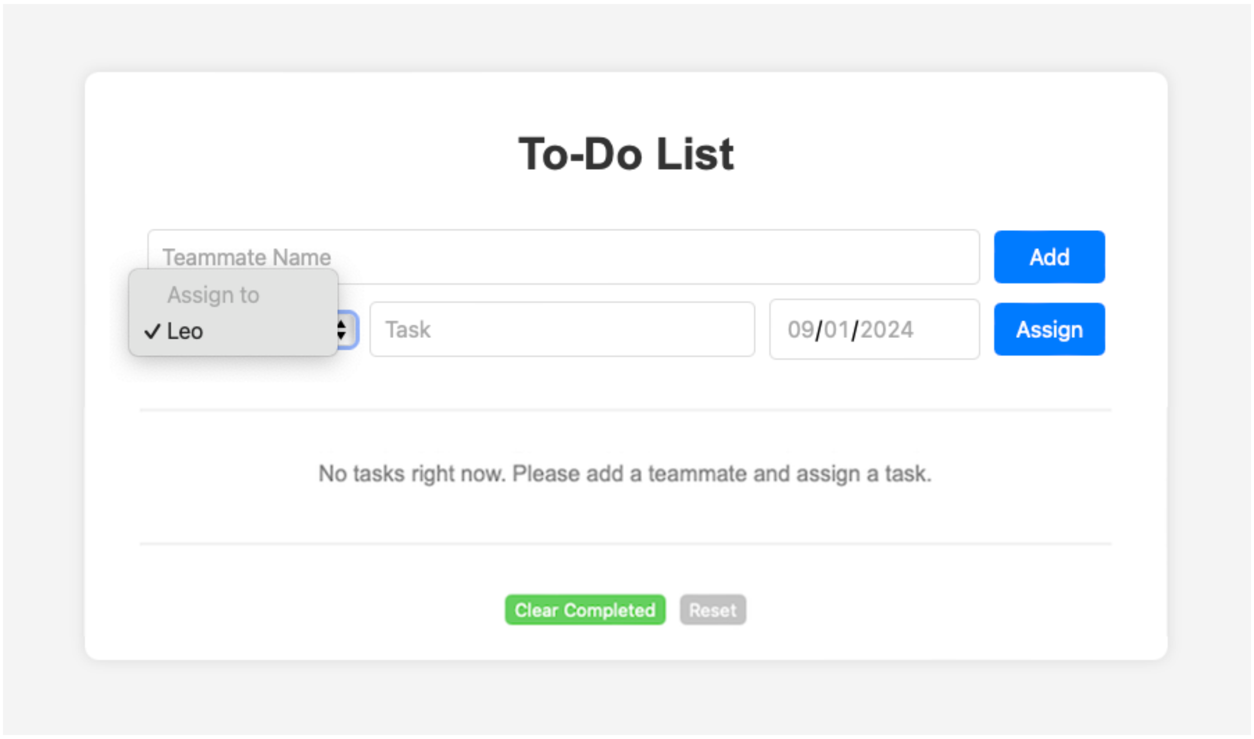
No tasks right now. Please add a teammate and assign a task.

2. Configure the dropdown menu to read “Assign to” when there are no teammates added to it yet.
   (1 pt.)

3. Now, add JavaScript code so that, when the “Add” button is clicked, the text from the input field gets added to the dropdown menu instantly.
   (12 pts.)

4. Ensure that, once a teammate's name is added to the dropdown menu, the input field is automatically cleared (revealing the text "Teammate Name" again).
   (1 pt.)

5. Configure the dropdown menu to disable the "Assign to" option once at least one teammate is added to it. The dropdown menu should still show the option when it is clicked and expanded, however (albeit grayed out), so that the user understands what the purpose of the dropdown menu is. See the screenshot below.
   (2 pts.)



No tasks right now. Please add a teammate and assign a task.

6. Ensure that the "Add" button handles erroneous input. If the user attempts to add an empty name or a name that already exists in the dropdown menu, the system should respond with feedback appropriately (e.g., for the second case, "Leo already exists!").
   (2 pts.)

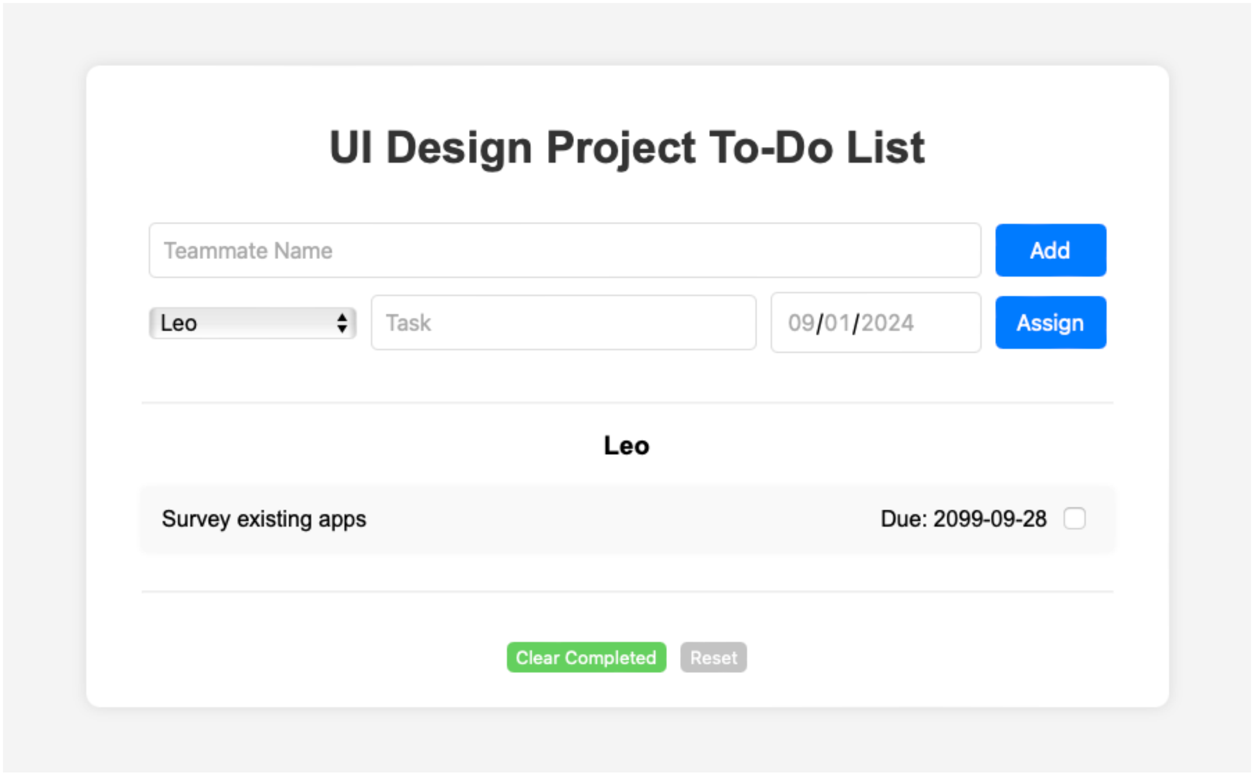
7. Ensure that the dropdown menu is always sorted in alphabetical order.
   (4 pts.)

PART 3: TO-DO LIST FUNCTIONALITY

(30 points)

Now that it is possible to add team members' names, our next goal is to make it possible to create to-do items. Each to-do item will be associated with one of the team members from the dropdown list.

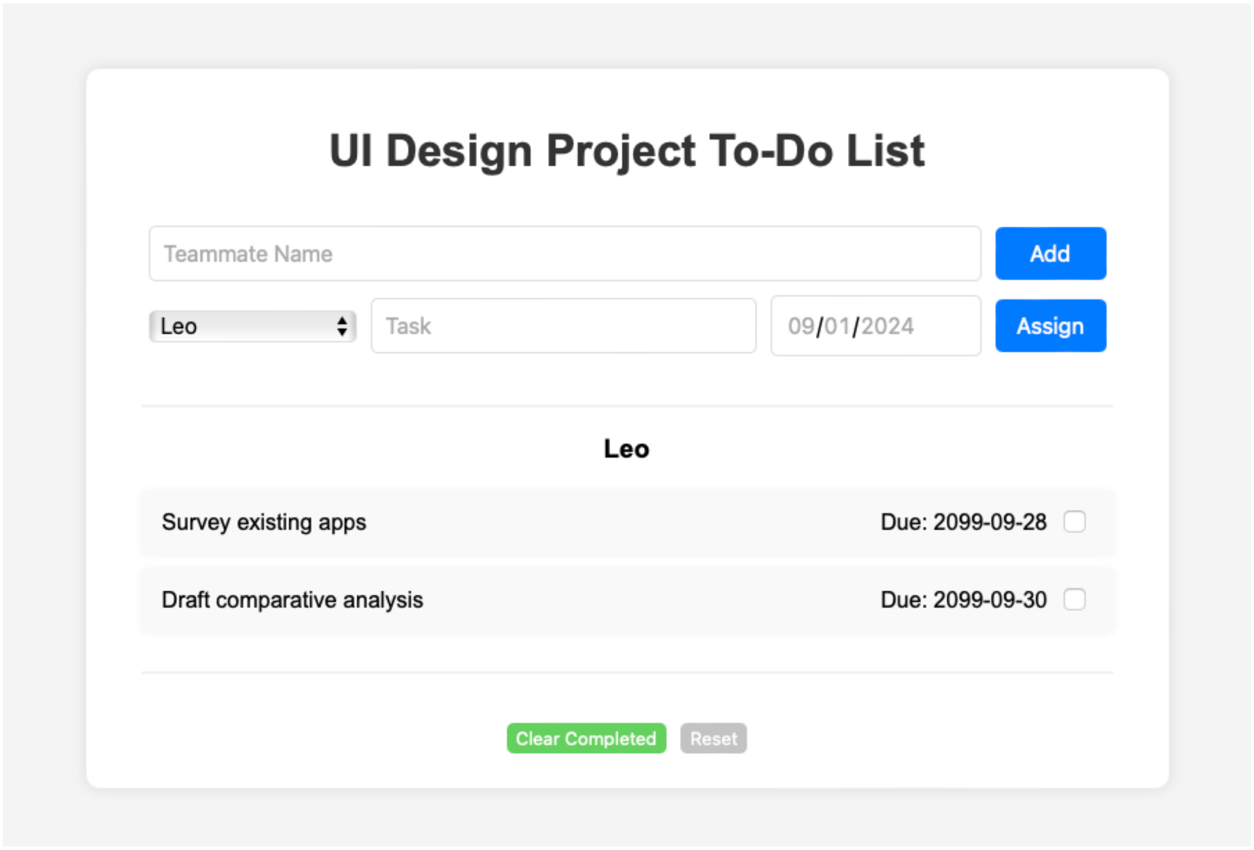
1. Add code so that, when the user clicks the "Assign" button, a new to-do item is rendered for that teammate, with the due date shown as part of the to-do item. A heading with the teammate's name should also appear above the task. As an example, if the user were to add a task with the text "Survey existing apps" for a teammate named Leo, the to-do list should appear as in the following screenshot. The HTML/CSS code from Individual Assignment 1 will be useful to reference here.
   (10 pts.)



2. Add error-checking code to ensure that it is not possible to add tasks with empty text, no assigned teammate, no assigned due date, or a due date earlier than the current date.
   (4 pts.)

3. Add code so that, if more than one to-do item is added for a given teammate, the heading with the teammate's name only renders once. As an example, if a second to-do item reading "Draft comparative analysis" were added for Leo, the

to-do list should appear like the below screenshot, without a second "Leo" heading.
(4 pts.)



Survey existing apps Due: 2099-09-28

Draft comparative analysis Due: 2099-09-30

4. Ensure that the to-do items for each teammate are sorted by due date, with to-do items due sooner shown above to-do items due later.
   (4 pts.)

5. Add the ability to mark a to-do list item as completed by checking the item's checkbox. The user interface should clearly reflect which tasks are completed. One way is to render the to-do item's text in a ~~strikethrough~~ style, but you are free to use another method as long as the indication is clear.
   (2 pts.)

6. Ensure that the teammates' lists of to-do items are shown in alphabetical order by the teammates' names. As an example, for two teammates Arnavi and Leo, Arnavi's to-do items should always be shown before Leo's to-do items, regardless of the order they were added.
   (6 pts.)

PART 4: RESETTING AND SAVING

(30 points)

In this last Part, we will add the ability to reset the to-do list and the ability to save the to-do list to local storage (i.e., save the data to a special place on the user's hard drive that the browser will load even if it is closed and reopened again).

1. Add code to make the "Clear Completed" button functional. When clicked, the button should remove all to-do items that are currently marked as completed. Those to-do items should disappear from the rendered list of to-do items.
   (6 pts.)

2. Ensure that the "Clear Completed" button does not do anything if there are not any to-do items present or marked as completed.
   (4 pts.)

3. Ensure that, whenever completed to-do items are cleared and a teammate no longer has any to-do items remaining, their name (i.e., that teammate's heading) gets removed from the view as well.
   (3 pts.)

4. Ensure that, whenever the very last to-do item is cleared from the view, the list area reverts to a single label reading "No tasks right now." or something similar. The label should be vertically centered between the layout's two horizontal rules.
   (4 pts.)

5. Add code to make the "Reset" button functional. When it is clicked, all to-do items and teammates get deleted, and the form resets to its initial appearance, like the image for Part 1, #1 of this assignment.
   (4 pts.)

6. Add some form of confirmation prompt or dialog box to the "Reset" button to help protect users from accidentally deleting all of the to-do items. The confirmation prompt might read something like, "Are you sure you want to reset all teammates and to-do items?"
   (2 pts.)

7. Last, add code to automatically save the current state of the app to local storage whenever the to-do list or set of teammates is updated in any way. This freeCodeCamp article and this GeeksforGeeks article describe how to read and write from local storage. The app should load from local storage on page load and save to local storage (overwriting the data already saved there) whenever a teammate is added, a to-do item is added, the "Clear Completed" button is clicked, or the "Reset" button is clicked.
   (7 pts.)