# CSC468/568 Web and Undo/Redo

**DUE: As noted on D2L**

***The purpose of this assignment is to give you practice with HTML, CSS, Node.jS files, and the undo/redo pattern.***

***This must run on Firefox, Chrome, use Node.js/Express for files, and be uploaded to a Render server.***

***There is a checklist on this project, but the location to place the checks are different than standard.***

***WARNING: read the policy page on “making your submission difficult,” the “Site Requirements” and “Forbidden items” with a focus on elements in regard to sloppiness in the Web FAQs.***

# Overview

You will be coding a prototype custom cat tree creator form, nicknamed “If I fits, I sits.” You must support running on Firefox and Chrome.

To narrow the scope, the project will only build the tree; there will be no submission for manufacturing it. Physics rules, (aka, can it actually be built) are also not required. For the prototype, you will add the components in rows that can hold at most 3 components. The tree also can be changed. In addition, you will implement preconfigured sets and undo/redo. You must allow uploading, downloading, saving, and loading of the page. These tasks will give you only part of the required points. The remaining points will be extensions that you will choose.

Your program will have three pages: 1) the main application page, 2) a file management page, and 3) a help page which doubles as a grading aid page. Each page has a horizontal navigation bar and a content area below the bar. There are additional formatting requirements listed in the Formatting section.

# Online Server

You must get this project running on the specified online server (Render).

To uniquely identify your project, name your project “LastName FirstName GUI-Web 2023.”

Names your NodeJS starting file (the file that starts your server) *mainServer.js*.

When submitting, you must give your Render URL in the comment box so we can view it.

Give course staff access to your GitLab project for this assignment.

# Main Application Page

This page must provide the ability to

1. Start with 1 row of 3 “empty” components.
2. Change a component in the tree.
   1. The minimal component options are empty, pole, and house.
   2. Changing components must either cycle through component options on click OR use the side bar technique used in the earlier Java assignment. Alternatives are permitted as extensions.
3. Add a new row of 3 components (deleting a row is not required).
   1. You must support 5+ rows
4. Change the tree to a preconfigured option with one button click only.
   1. The first option must be “Reset,” which changes the main content to its initial state.
   2. The second option must be “All poles,” which changes the main content to the two rows of all “pole” components.
   3. The third option must be “Mixed,” which changes to three rows of components in the following configuration:

|  |  |  |
| --- | --- | --- |
| empty | empty | pole |
| house | empty | house |
| pole | house | pole |

1. Undo/redo.
2. Save the main content state to the server.
   1. Save must be done without losing the main contents or editing ability (I do not care about saving the undo/redo, only the main contents).
   2. The save button must be **on the main page**. You may either ask for a file name, or automatically name the file.

## Undo/Redo

Undo/redo functionality is only implemented here. You must be able to undo and redo for the tasks listed for the main application numbered 2-4.

Mark each of the undo/redo OOP pattern components of the following with the given comments. Specifically,

* The history manager class is marked with GRADING: MANAGE
* The command/memento class(es) are marked with GRADING: COMMAND
* The addition of the undo/redo classes to the history manager is marked with GRADING: ACTION

See the FAQ page for additional requirements on Undo/Redo.

# File Management Page

This page must allow you to see available files, upload a file, load a file (e.g., open a server file), and create a downloadable file. The goal is to be able to support the standard save 🡪 download 🡪 upload 🡪 open\load file processes. You may use a flat text file, json, xml, etc. to save the application content.

* This must be a separate page as I am testing if you can forward info between pages.
* You must permit a minimum of **one** file for **one** user to be saved on the server. Supporting more is an extension.
* For each available file on the server, you must be able to open the file, **automatically redirect** back to the main page, and replace the main area with the contents described in the file.
* A content file must be able to be downloaded, reuploaded, and opened.
* An uploaded file is added to the server list and is not immediately loaded.

# Help Page

This page should explain how to use your web application. You must divide the content with HTML headings. This **overrides** the standard checklist in the main file header requirement. See the formatting section for the complete list of requirements.

# Formatting

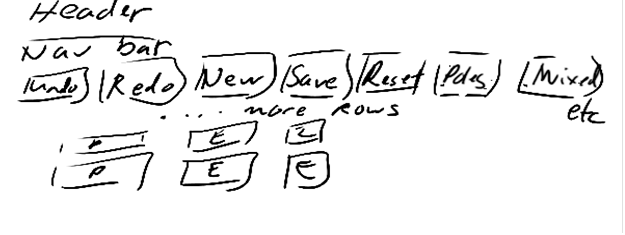
The navigation bar must have all your available pages listed on every page.

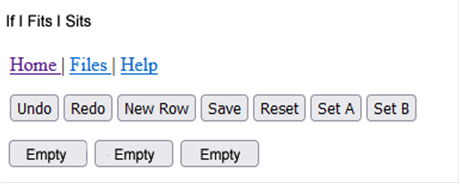
### Main Page

Format requirements:

* The vertical structure must contain the header, navigation bar, add button, component change button (if using that UI option) and save, and then the content area immediately below.
* The navigation bar must be primarily horizontal.
* Selecting an option must show the effect immediately (an immediate page refresh is still considered immediate).

The following is an intentionally rough outline and a screenshot without ANY styling (before I decided what the preconfigured options would be), using the cycle option for components, of this page.





### File Management

Layout requirements:

* The vertical structure must contain the header, navigation bar, upload, and then the server-file area below.
* File downloading must be either 1) add a text box to type in the file you want (must be under upload), and\or 2) add a download button/link next to each file.
* There must be a load button next to each file.

The following is an intentionally rough outline of this page.

Text, letter

Description automatically generated

### CSS

The required CSS rules you must personally create are

1. The background of a page must include a color other than white or a background image.
2. There must be a custom CSS header included in all pages (note, not heading).
3. The CSS on the navigation bar must include a background color/image rule.
4. There must be a custom link (specifically on the <a> tag) rule.
5. The main application content area must include a border of some sort.
6. The help page headings must have a font style/font change.

See the FAQ page for *additional requirements* on CSS.

### Help



The help page is what would normally be put in your main file header. That is harder to define in the web, so this is the substitute.

Layout Requirements:

* The vertical structure must contain the header, navigation bar, within page links to each major heading (e.g., long articles in Wikipedia, where these links are marked in blue), description, checklist, CSS modifications, and then the program usage below.
  + Any additional main headings must be added below the required headings.
* The **heading tags** must be used in this page to denote sections.
* The description is your standard project description with your name, course, description, etc.
* The CSS heading is what you did for the CSS requirements and includes a subpage link directed to where the **effect** **is** ***applied*** (not code) on your webpages (e.g., the custom link is used on the navigation bar). ***Missing either of these is equivalent to a missing grading tag (so, 0 points)***.
  + If a direct link is not applicable (e.g., the background is green), instead explain the location the effect is applied.
* The Usage heading is for how to use your project. Are there limitations? Do I have to initialize something? This is essentially the “bug report” section in a normal checklist.

The following is an intentionally rough outline of this page.

Text

Description automatically generated with medium confidence

# Extensions:

Extensions with their point values are listed in a separate document.

# Submission instructions

1. Do a search for GRADING in the writeup to make sure you are not missing any required elements.
2. Make a fresh upload to the Render server, and ensure everything still work.
3. Copy the posted checklist into the help page. Check off what you have done.
4. Check the coding conventions before submission.
5. Zip your **ENTIRE PROJECT** into ***ONE*** zip folder. The file hierarchy is **critical** this time. The starting files should be at the root.
6. Submit to D2L. The dropbox will be under the associated topic's content page.
   * **LIST YOUR *HOMEPAGE URL* AND *RENDER URL* FOR THIS APP IN THE COMMENT BOX. Failing to do so is a letter grade deduction. I know those URLs could be anywhere.**
7. *Check* that your submission uploaded properly. No receipt, no submission.

You may upload as many times as you please, but only the last submission will be graded and stored.

If you have any trouble with D2L, please email me (with your submission if necessary).

# Grading Tiers

These tiers start with the simplest tasks, and go to the most involved. Please refer to the rubric for the tiers. You must “reasonably” complete the lower tiers before gaining points for the later tiers. By “reasonably,” I can reasonably assume you honestly thought it was complete. **Treat the rubric as a checklist!**

***Special reminder for this assignment: the extension points will not be given until the prior tiers are completed. If your app doesn’t work, it doesn’t matter how nice it looks! Get the minimum working first, and use the MVC pattern to your benefit.***

# Rubric

|  |  |
| --- | --- |
| **Item** | **Points** |
| **Undo/redo pattern (tier-less) \*** | 30 |
| a. Undo/redo OOP followed (-50% for 1 moderate error) | 6 |
| b. Able to do at least 1 undo/redo for 1 change (50% each) | 6 |
| c. Able to do at least 1 undo/redo for 1 preconfigured change (50% each) | 6 |
| d. Able to do at least 5 undo/redo for 1 change (50% each) | 6 |
| f. Able to do at least 5 undo/redo for 1 preconfigured change (50% each) | 6 |
| 1. **Basic content** | 36 |
| a. All 3 pages present | 3 |
| b. Header in all pages in correct spot | 4 |
| c. Nav bar in all pages in correct spot | 4 |
| d. All interaction buttons added to main page in correct spot | 4 |
| e. Properly display the default content in main page in correct spot | 4 |
| f. At minimum, a place holder for all items in the file management page | 2 |
| h. All headings added to Help page in correct order | 3 |
| i. Help page content filled in | 6 |
| j. Within page link help page working | 6 |
| **2a. Basic application tasks** | 28 |
| a. Able to change a component in first row (-33% for each error) | 12 |
| b. New row adds a row | 7 |
| c. Able to change components in new rows (-33% for each error) | 9 |
| **2b. CSS rules\*** | 18 |
| a. Background color/image | 3 |
| b. CSS header | 3 |
| c. CSS nav bar color/image | 3 |
| d. CSS <a> rule | 3 |
| f. Content border | 3 |
| g. Help heading font change | 3 |
| **3a More advanced main page options \*** | 18 |
| a. Able to change to preconfigured options (-33% for each error) | 12 |
| b. Main content area unaffected by save, and save stays on page | 6 |
| **3b Tier: Files** | 40 |
| a. Able to save the current content (must show in file page) | 5 |
| b. Able to download at least one file (default is ok) | 5 |
| c. Able to download an application generated file | 5 |
| d. Able to upload a file | 5 |
| e. Loading a file redirects back to the main page | 5 |
| f. Loading properly displays with app generated file (-50% is some error) | 10 |
| g. Able to save, download, upload, and load that same file | 5 |
| **4 Tier: Extensions** | 30 |
| **Total** | **200** |

\* There must be no styling/script inside the HTML/PHP files.