



Edhub: Schoolwork Organized!

Stacey Kirby, Tanner Wilson, Hirad Sabaghian, Fahad Alothaimeen

11/14/2016

3F: Final Report

Team Members:

- **Fahad Alothaimeen:** writing, ideation
- **Stacey Kirby:** writing, web development, ideation
- **Hirad Sabaghian:** design ideation, writing
- **Tanner Wilson:** writing, management, ideation

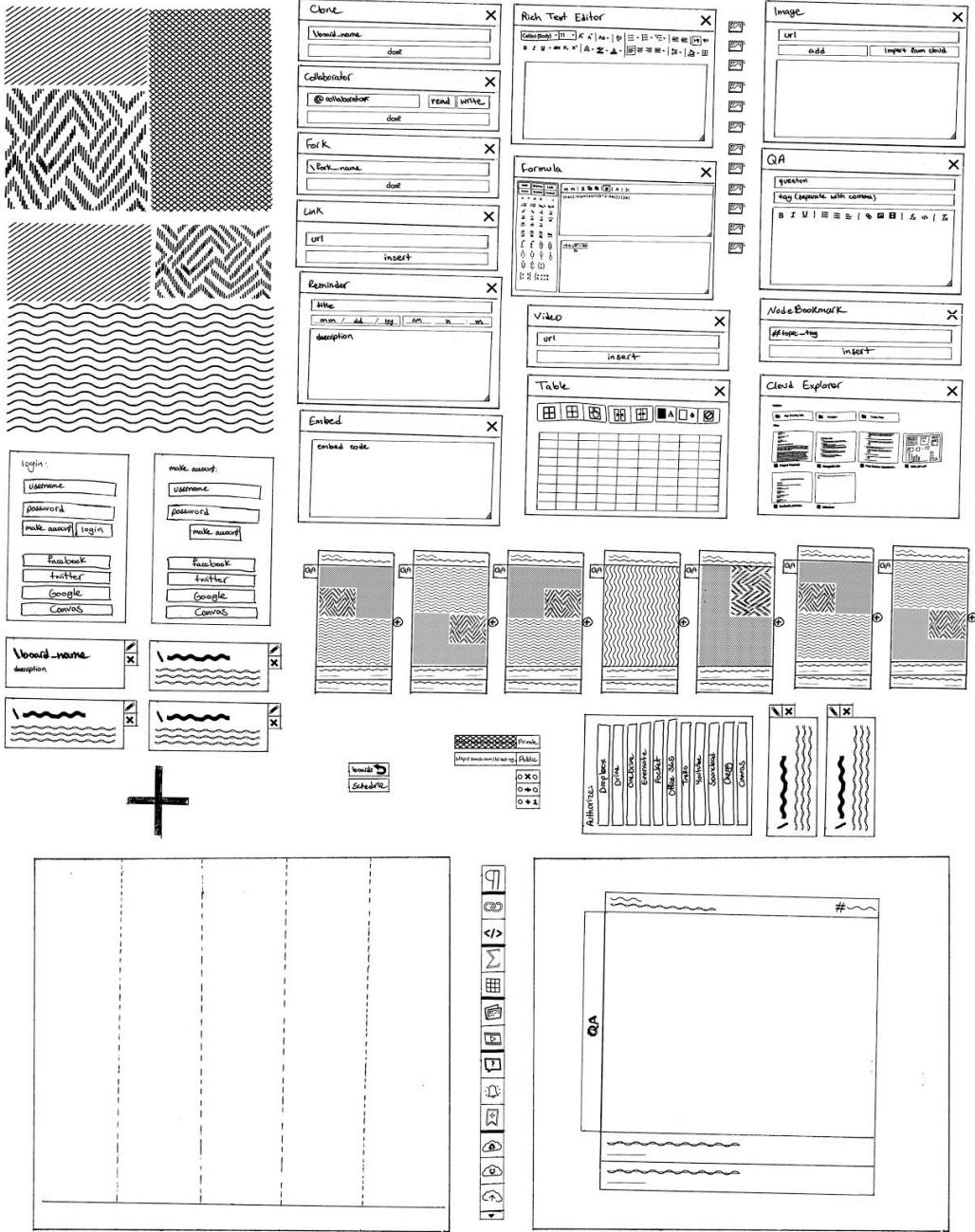
Problem and Solution Overview:

Being a student brings to mind images of stressful all-nighters studying for tests, doing homework, and researching dozens upon dozen of sites, which is not including the multiple sites that teachers use for each individual class.

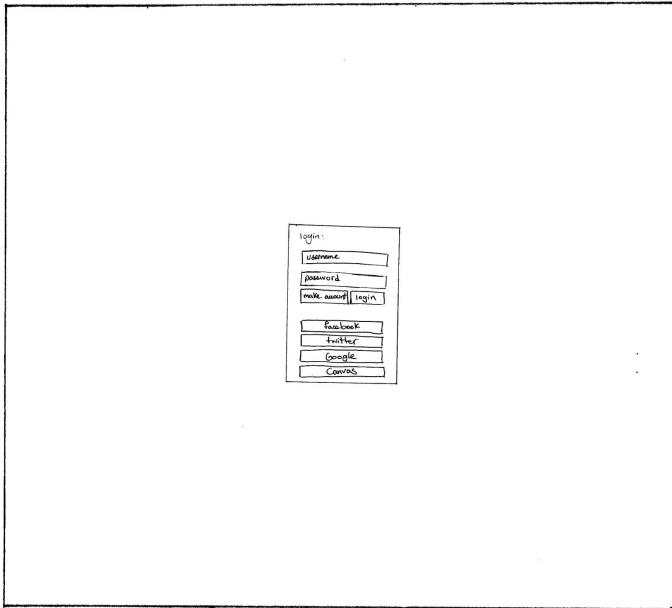
Initial Paper Prototype (Draft 1)

The goals for this design iteration were to be able to keep track of assignments and other activities and easily find help on different school concepts. Due to some confusion among the team we had failed to initially address our tasks for our first draft of paper prototype. However below we have included the recreated tasks. Unfortunately, due to naive expectations our initial design was overly complicated and unfeasible to fully execute during the course of the semester.

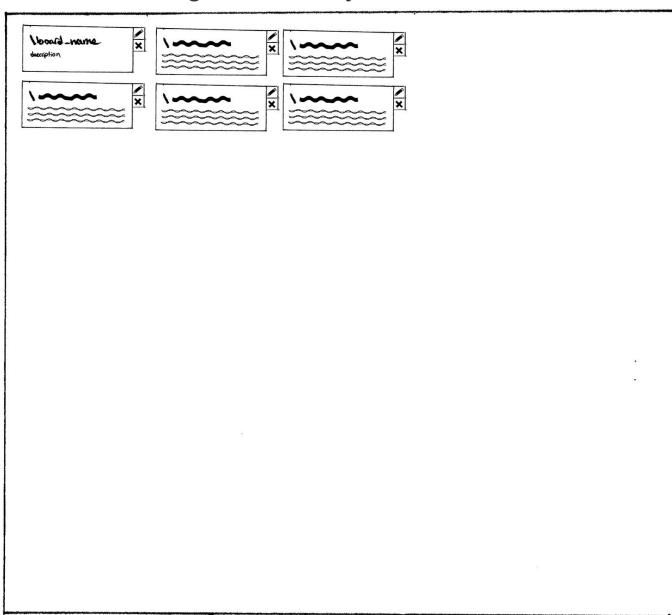
Overview:



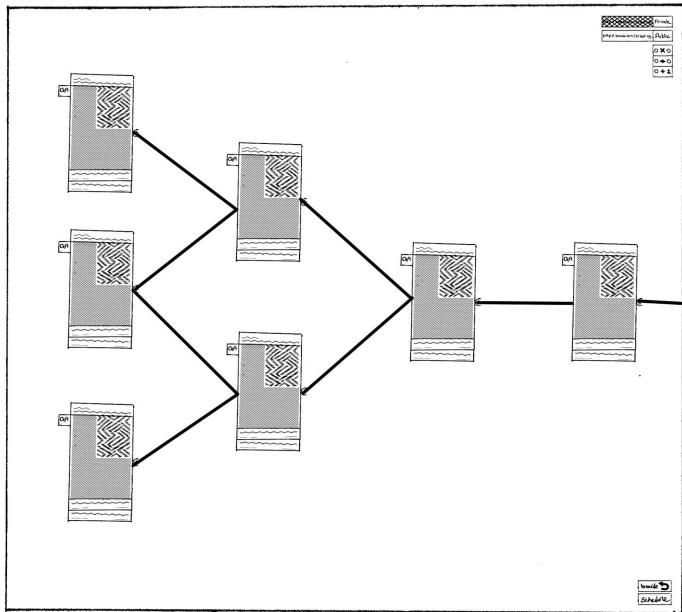
Draft 1 - Task 1: Find help on a given topic



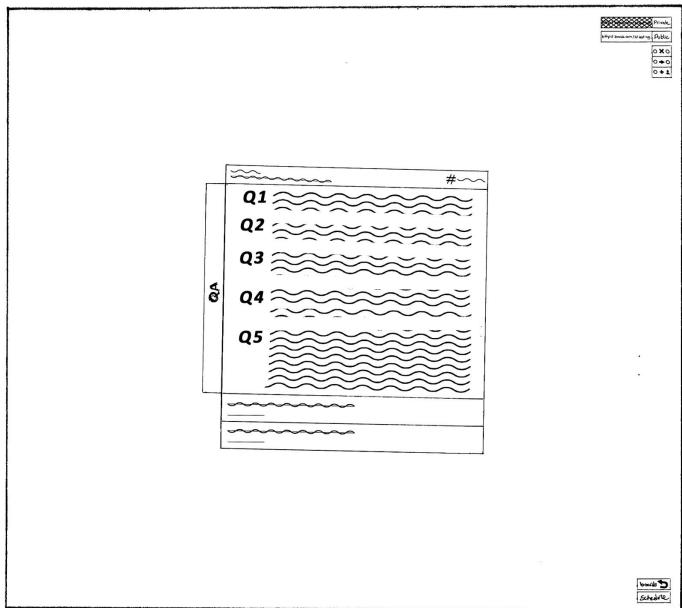
- User logs into the system



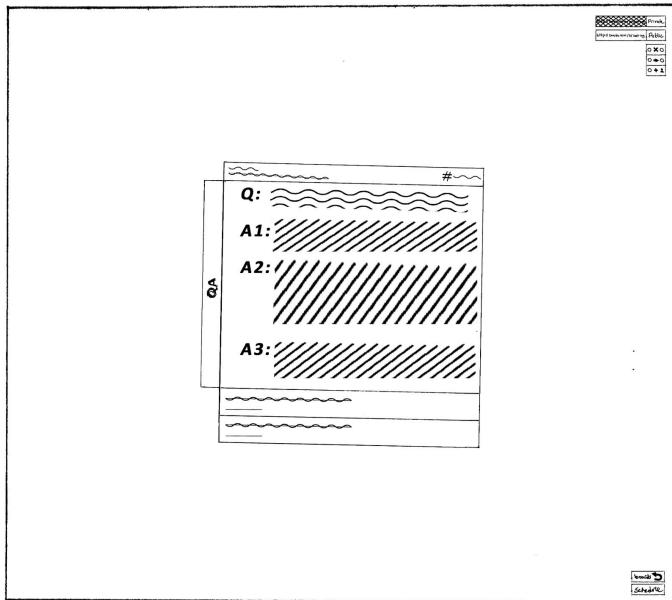
- User is presented with the classes they have created.
- They will select the class for which they need help with.



- After selecting a class, user is presented with topic within that course.
- They will click on the “QA” button on the top left of the topic for which they need help with.

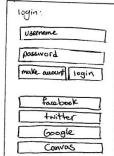


- Immediately they are presented with the QA board and presented with the questions for that topic asked by other users on the website.
- They will continue by selecting a question of their interest.



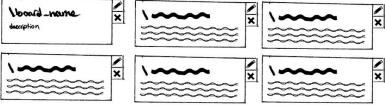
- Finally they are presented with the answers to that question.

Draft 1 - Task 2: Keep track of assignments and other activities



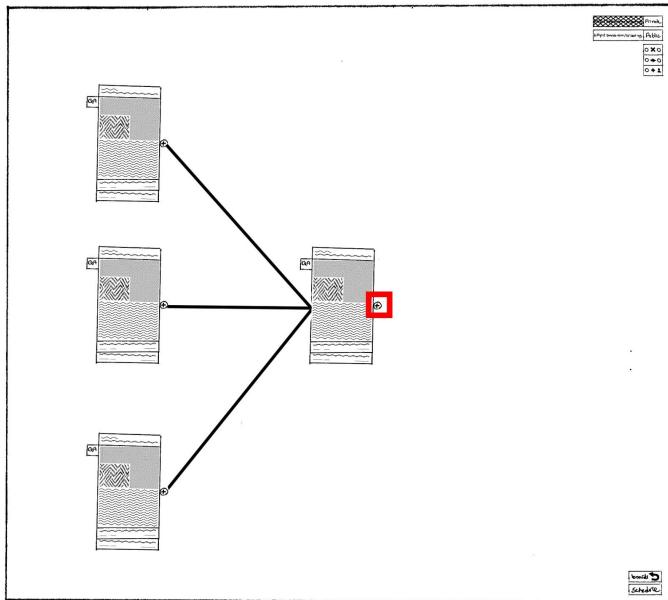
A wireframe diagram of a login interface. It features a title 'Login' at the top, followed by three input fields labeled 'Username', 'Password', and 'make account'. To the right of each input field is a small 'x' icon. Below these fields is a large rectangular area containing five social media logins: 'Facebook', 'Twitter', 'Google', 'Canvas', and another 'Facebook' option. At the bottom right of this area is a small 'x' icon.

- User logs into the system

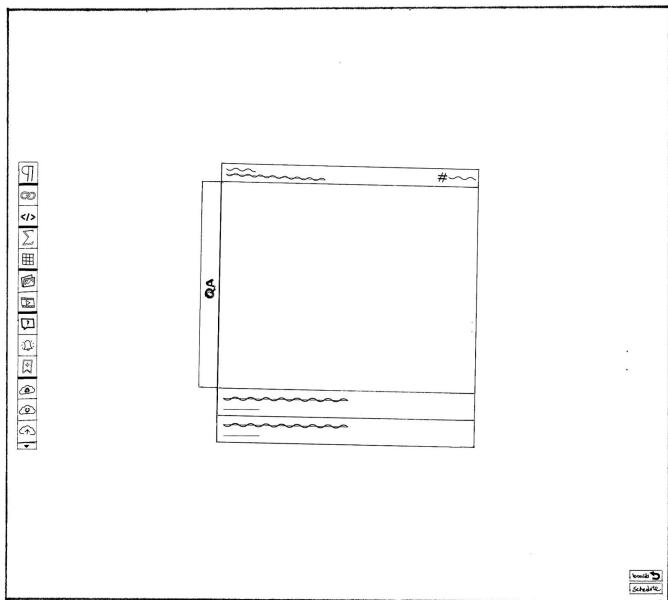


A wireframe diagram of a class selection interface. On the left, there is a search bar with the placeholder 'Search...' and a magnifying glass icon. Below the search bar is a table with two columns. The first column contains four rows, each with a wavy line icon and an 'x' icon. The second column also contains four rows, each with a wavy line icon and an 'x' icon. The entire table is enclosed in a light gray border.

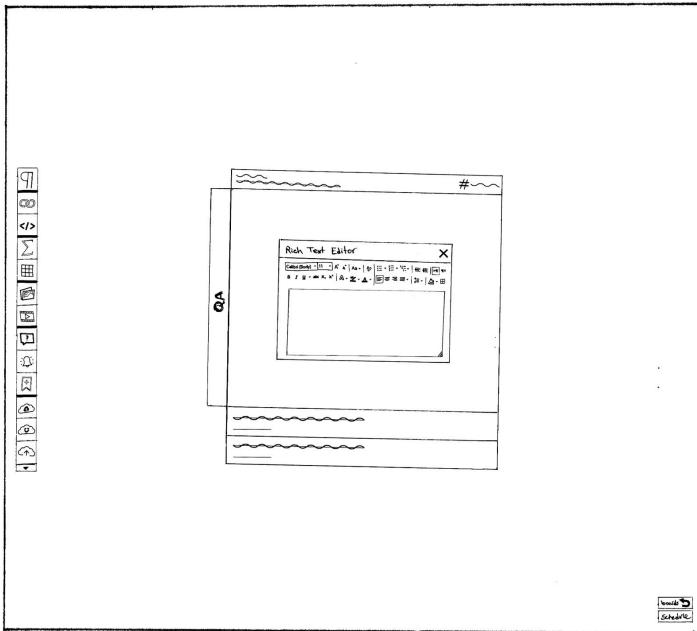
- User is presented with the classes they have created.
- They will select the class for which they want to add content.



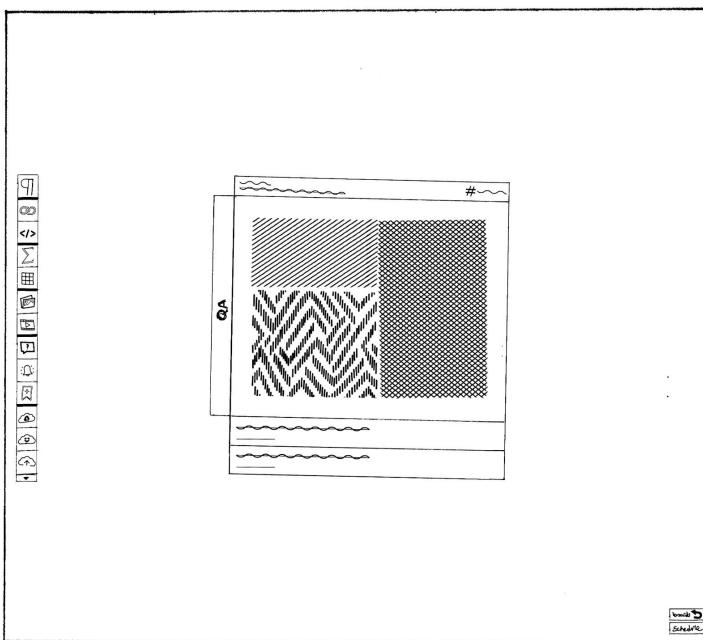
- They are presented with the current topics of the course.
- They will continue to add a new topic by clicking on the plus button of the parent topic.



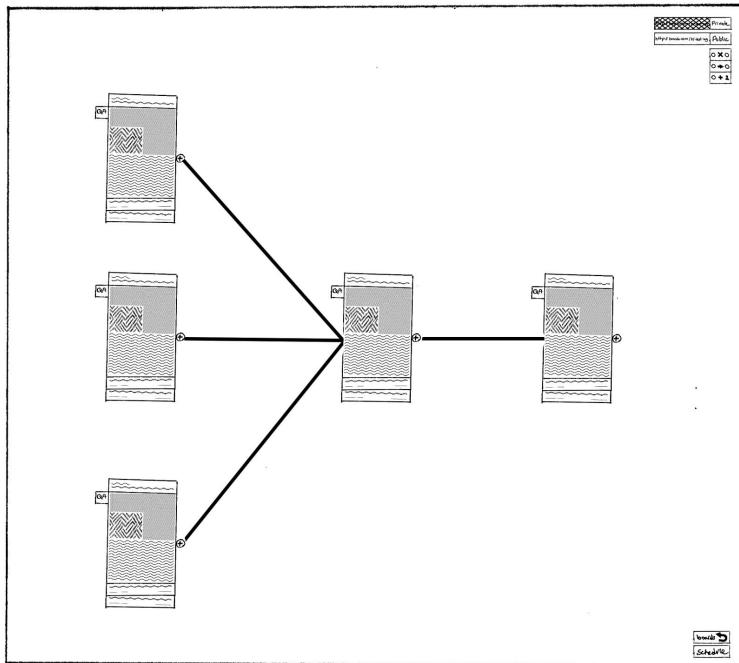
- User is taken to the edit window, where they have access to different module for adding content.
- They will select the Rich Text Editor from the modules.



- RTE prompt opens up and user continues to copy/paste their content in the editor.
- Eventually they will finalize their changes by pressing Ctrl+Enter



- The newly added content of the topic is shown in the viewer.
- User will click on the back button on bottom corner of the screen to return to their topics.



- The freshly created node is attached to the parent topic and added to the class.

Usability Testing

We performed 3 different testing process going from general to specific in order to better understand problems with our program.

In Class Heuristic Tests

Early iterations of our design were far too complicated and not task oriented, so the in class heuristic tests did not yield a lot of meaningful results as they did not go very smoothly. This led to us stripping a lot of features from our design to help the rest of the tests flow well and to make it more task oriented.

Participant 1 -- Leo (CS Student)

Leo, a tech savvy CS student was the first test to be done with our prototype. The test was performed in the lobby of the WEB due to it being a commonplace that students use to work on homework and catch up on their studies. Due to Leo's expertise, we were able to see if our design would work for a power user.

The test protocol was to have Leo perform a walkthrough of the prototype to perform two tasks. The tasks were to create a board, add an alert to that board, and find a specific board's "due date" or time line from the home screen. During the test Hirad acted as the computer, Tanner was the facilitator and Fahad and Stacey acted as note takers.

Overall the test went smoothly after making the needed changes to our prototype, but it did lead to a few realizations by the team about the process. We realized that having longer and/or more specific tasks, allowed the user to openly explore the app to ensure the design flows well. A more open usage case will be very helpful to observe, especially if it is performed by a less tech savvy participant. We planned to target these users for the next tests.

Participant 2 -- Calin (Communications Student)

The second usability test was performed with Calin. This test was performed in the Marriott Library, due to it being a popular space that students go to study and finish assignments. Calin was chosen as a participant because we felt she represents the majority of the population who would use our design. Being a senior in college, she isn't a power user, but also isn't entirely unfamiliar with technology as she uses the internet and mobile applications on a daily basis and is familiar with standard UI tools.

The test protocol was a cognitive walkthrough of our application where we asked Calin to perform two tasks within the "Event Planning" board. First, to find help on a topic that was posted/dated Nov 7th and second, to add a reminder and notes to a node on Nov 10.

Calin was able to navigate our prototype with relative ease, but the majority of where she would get tripped up was when she didn't know where to click, or what she could click. This led to mostly cosmetic changes and the adding of a few buttons to ease the flow of the application.

Participant 3 -- Alex (CS Student)

The third usability test was performed with Alex, a student and needs help with optimizing his current study habits. He is a senior computer science student with 15 credit hours this semester. He is a 'A-' student and is always busy with homework and has a crowded desktop to keep track of all of his assignments and study materials. The test was performed at the Coffee Garden coffee shop where many students come to study due to its pleasant environment, and was chosen because it resamples the environment where students, like Alex, would prefer to be while working on their school work.

The test protocol was a cognitive walkthrough of our application where we asked Alex to perform two tasks around a simple look-up. First task, Alex was asked to find a specific existing board by name "e.g., CS-5540", and then look for a specific node by name "e.g. 3d:Usability Test". The second task was to change the due date on that node from: Nov,7 to last day of the semester Dec, 16 and then make sure that the node was now is scheduled for Dec, 16 from the schedule view.

Alex was eventually able to perform the two tasks but with a little bit of frustration mainly due to the absence of some buttons and features such as a search bar. As a result, we recorded those changes and applied them to our current design according to the following schedule.

Summary

We had some difficulties that we continued to uncover during the testing process which came primarily from our design being too difficult. We had attempted to fix this problem and changed the questions for our testing in order to better pinpoint where the users were having difficulties. We had them speak aloud which helped us understand some of the confusing aspects of our project so we would know where to simplify them. This helped a lot since we were able to get to where specifically all of the confusion arose from.

Testing Results

Heuristic Evaluation

Feedback : Our group discovered many problems mainly centering around the fact that our project had too many features which led to confusion among the users and the testers. In extent, our initial prototype was far to general and didn't really address our problem space, so stripping down features led to it being more task focused.

Resulting Action: We had decided to get rid of some features that we deemed unnecessary in order to simplify the design considerably. This led to some major renovations to our paper prototype.

Usability Test -- Leo CS Student

Feedback: The user had difficulty understanding what certain symbols and items were for in the application. They also had difficulty getting to, what would be, commonly used pages easy from the home screen.

Resulting Action: In order to fix these problems we had decided to simplify more and get rid of some of the features that cause confusion and were not vital to the design. By doing this we were able to address some of the issues that the user had with getting to important pages more easily and faster.

Usability Test -- Calin (Communications Student)

Feedback: The user had difficulty understanding which part were buttons and what were stagnant parts of the application. This led to the user touching locations on the application that did nothing and ignoring the buttons that were available by the user. Calin continued to have difficulty in understanding which buttons did what due to there being too much occurring in the application.

Resulting Action: In order to fix these problems we had decided to continue simplifying more and get rid of some of the features that cause confusion and were not vital to the design. We had also made the clickable portions of the site (buttons) more button-like so that the user will be able to know what is selectable.

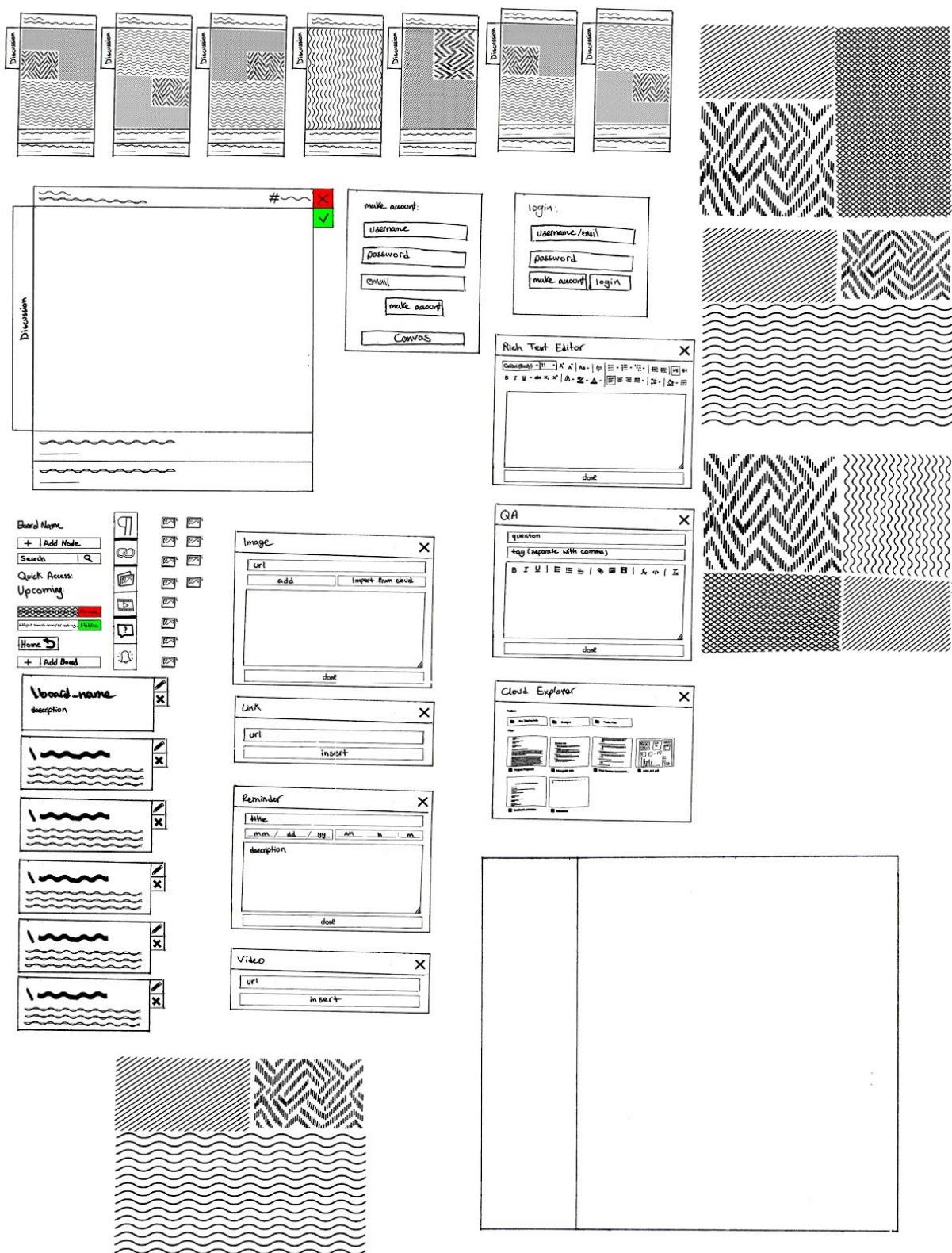
Usability Test -- Alex (CS Student)

Feedback: The user had difficulty being able to find specific items to do the different tasks that they needed to accomplish. They had make remarks of how they had wished that there was a search bar to allow him to find what he wanted. Alex had also complained about being confused on the vagueness of how to add a new boards.

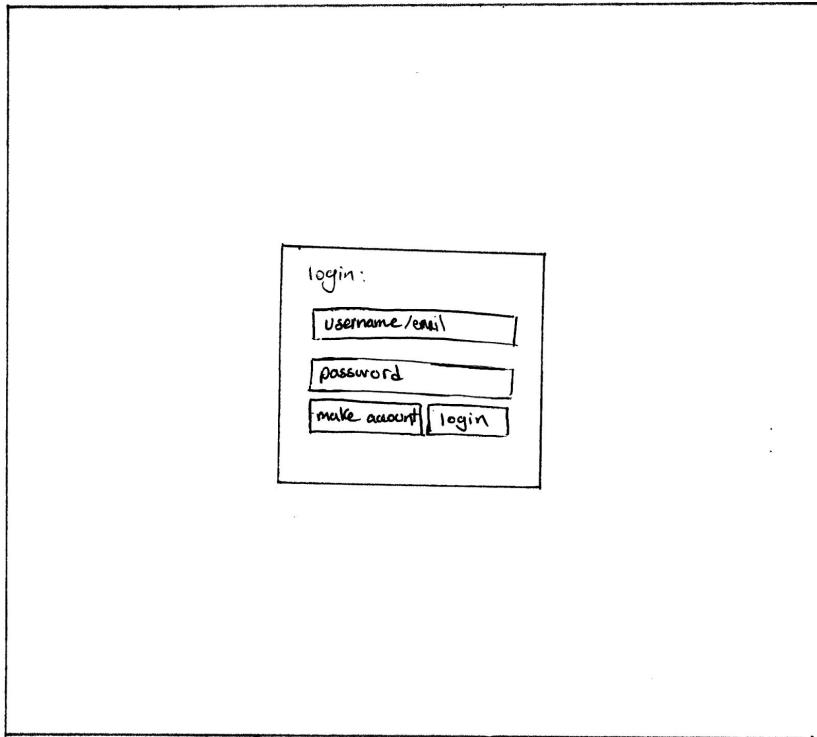
Resulting Action: In order to fix this problem we had decided to add a search bar so that the user will be able to quickly and easily find the different items that they needed, such as specific boards or nodes. We also included a clearly labeled button in order to signify where the user could add a new board.

Final Paper Prototype

Overview:



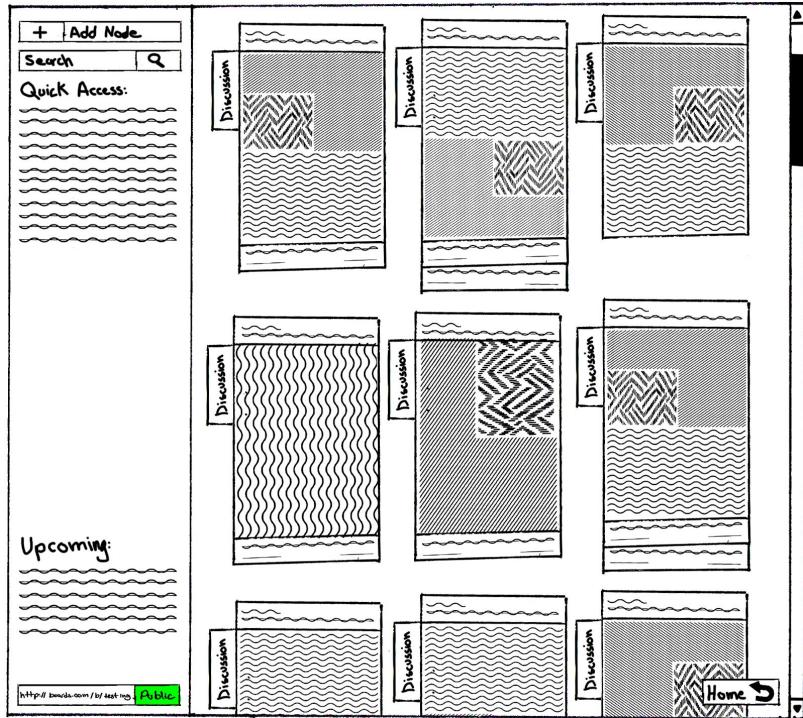
Final - Task 1: Find help on a given topic



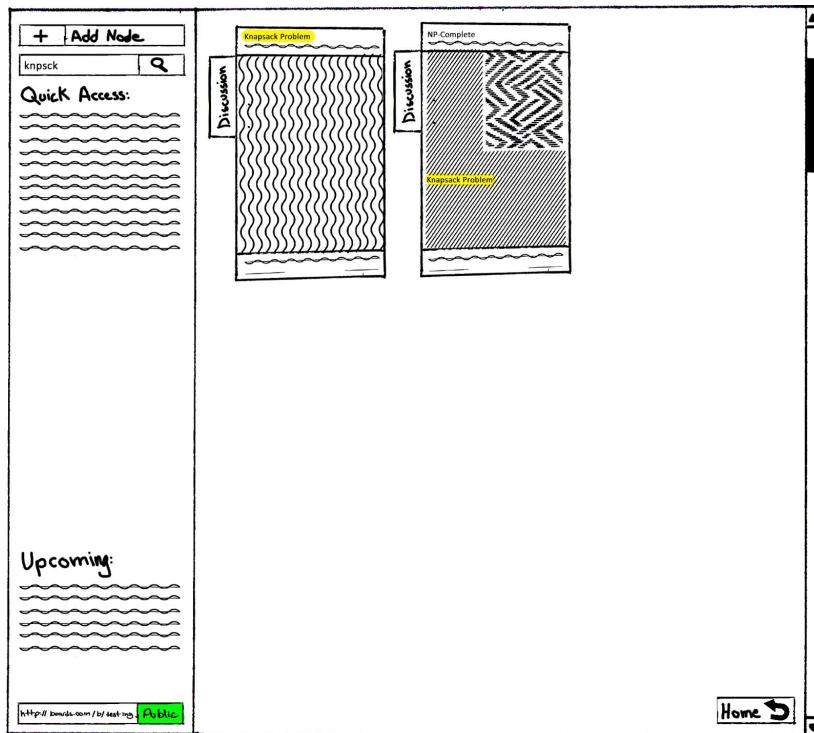
- User Logs In

A hand-drawn wireframe of a dashboard. On the left side, there is a sidebar with a search bar, a "Quick Access" section containing wavy lines, and an "Upcoming" section also containing wavy lines. The main area displays a grid of nine board cards. Each card has a header "Board-name", a "description" section below it, and a preview area showing several wavy lines. Each card also features edit and delete icons.

- User is presented with a view of their boards (classes)
- They will proceed by clicking on one of their boards



- After entering the board (class) the user is presented with all their topics.
- Notice the search bar on the top left.
- Notice how discussion is present on all topics.



- The user can enter their keyword (fuzzy search supported).
- The view will update with all the topics whose content or title matches the keyword.
- Notice the presence of discussion.

Wikipedia:

Knapsack Problem

The knapsack problem or knapsack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

The problem often arises in resource allocation where there are financial constraints and is studied in fields such as combinatorics, computer science, complexity theory, cryptography, applied mathematics, and daily fantasy sports.

Stackoverflow:

Python - Greedy Knapsack with Dictionary input

I am trying to implement a greedy knapsack algorithm in python given the set below. The output is supposed to be a list of lists, that observes the limit set. In example with the dataset below the output should be:

```
out = [[C, B, D, A], [Z, F, E]]
```

Crossover Operation for Binary Encoding

I'm following the genetic algorithm approach to solving the knapsack problem using DEAP. I understand that they used a direct encoding scheme rather than a binary representation. The crossover function is as follows:

```
def cxSetInd(ind1, ind2):
    """A crossover operation on input sets. The first child is the intersection of the two sets, the second child is the difference of the two sets.
    """
    ind1.set(ind1 & ind2)
    ind2.set(ind1 ^ ind2)
```

Multiple inner Knapsack and Fitness Calculation

Scratching my head on this one and I might be thinking it wrong. Basically it's the knapsack problem but modified. You have a set of items with various weights on them and you are to put it in three knapsacks with a capacity of 20 each.

I have codes to initialise all the items randomly in the sacks. That means I can have a sack with more than 20, less than 20, and equal to 20. The problem is that the items are all being added hence my total score is the same for all population making it impossible to mutate.

Google:

Dynamic Programming

Given weights and values of n items, put those items in a knapsack of capacity to get the maximum total value. This is knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value of sub-set [val] such that sum of weights of this subset is smaller than or equal to W. You can break an item, either pick the **single** item, or don't pick it (0-1 property).

Lecture 13: The Knapsack Problem

Informal Description

Suppose we are planning a hiking trip; and we are, therefore, interested in packing our knapsack with items that we want to take for the trip. There are different item types that are deemed desirable; these could include bottle of water, apple, orange, sandwich, and so on. We want to pack as many items as possible, but we also want to make sure that the total weight of the items is as small as possible. The files have combined size at most W . The total weight of the stored files is as large as possible.

We can not store parts of files, it is the whole file or nothing.

The Knapsack Problem

Suppose we are planning a hiking trip; and we are, therefore, interested in packing our knapsack with items that we want to take for the trip. There are different item types that are deemed desirable; these could include bottle of water, apple, orange, sandwich, and so on. We want to pack as many items as possible, but we also want to make sure that the total weight of the items is as small as possible. The files have combined size at most W . The total weight of the stored files is as large as possible.

- After clicking on the discussion button of a topic, the discussion page will appear. This page includes content aggregated from different reliable sources, such that they are related to the topic, title of the result topics, and keywords of the result topics. Notice how each result has an excerpt.
- Clicking on each result will take the user to that specific website.

Google:

How do I solve the 'classic' knapsack algorithm recursively?

This is my task

The knapsack problem is a classic in computer science. In its simplest form it involves trying to fit items of different weights into a knapsack with a maximum weight capacity and a total weight. You don't need to fit in all the items. For example, suppose you want your knapsack to weigh exactly 20 pounds, and you have a set of items with different weights and values. For small numbers of items, humans are pretty good at solving this problem by inspection. So you can probably figure out that one item with weight 10 and value 10 is the best choice. I really don't know where to begin writing this algorithm. I understand recursion when applied to factorials and triangle numbers. However I'm lost right now.

Trying to figure out the classic Knapsack recurrence

I am reading about the knapsack problem (unbounded) which I am I understand a classic dynamic programming problem. And I am trying to understand the solution as I read it, I am not clear how I can translate it to actual code. For example in the following recurrence formula:

$$M[i][j] = \max(M[i][j-1], M[i-1][j] + V[i])$$

I am not sure how I can translate this to code, since it is not clear to me if the inner MAX should be there or it should just be $M[i][j] = \max(M[i-1][j], M[i-1][j] + V[i])$ for $j > 1$

Any help to figure out the formula and to code it?

Knapsack problem with recursion

//Title of this code
#include<iostream>
#include<list>
using namespace std;

```
int values[3] = {2, 4, 4};  
int weights[3] = {8, 35, 34};  
int W = 18;  
int knapsack(int i, int W) {
```

Java stack to solve the "knapsack problem"

This package consists of 3.java. Each.java is a class
Class 1: Node.java
Packet package;
Node class {
Index int;
Value int;
Node public (AA int, BB int) {
Index = aa;
Value = bb;
}
}

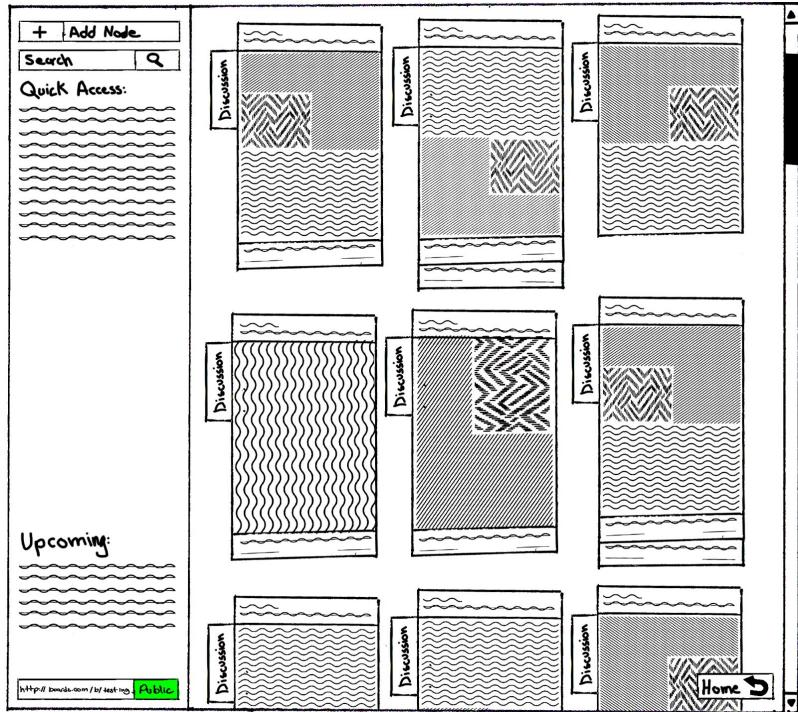
The 0-1 Knapsack Problem

The objectives of this laboratory are the mastery of the 0-1 knapsack problem and its algorithm as well as its derivation from the knapsack problem formulation. The laboratory is designed to enhance the development of understanding the use of dynamic programming to solve discrete optimization problems. To gain a better understanding of the knapsack problem and compare it with your textbook, read the [informal description](#) and compare it with your textbook, if available.

The 0-1 Knapsack Algorithm

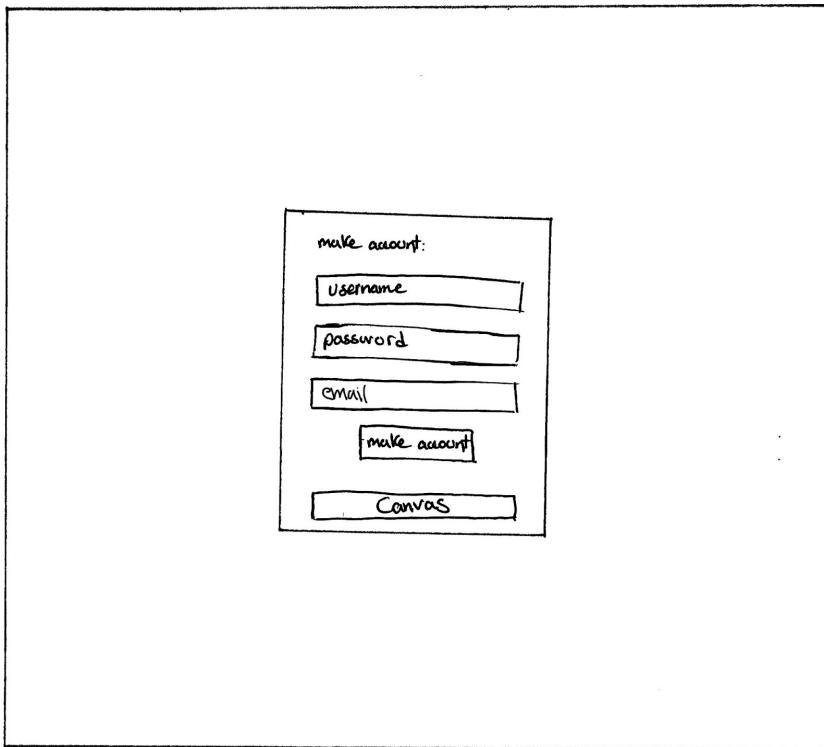
The knapsack problem is the classic integer linear programming problem with a single constraint. The 0-1 knapsack problem is for

- If the user cannot find the answer to their question, or their question is too specific, they can input it directly in the search box, and the top results of Google will show up with their summaries.

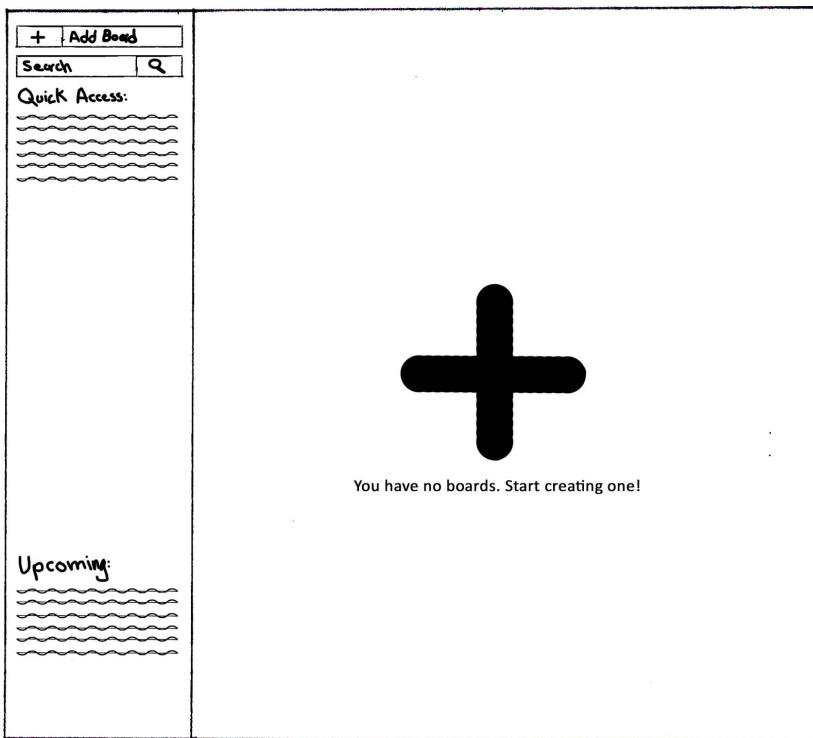


- When the user clicks on the back button on the bottom right corner of the discussion page they will be taken back to their board/class from which they started the discussion and shown the topics.

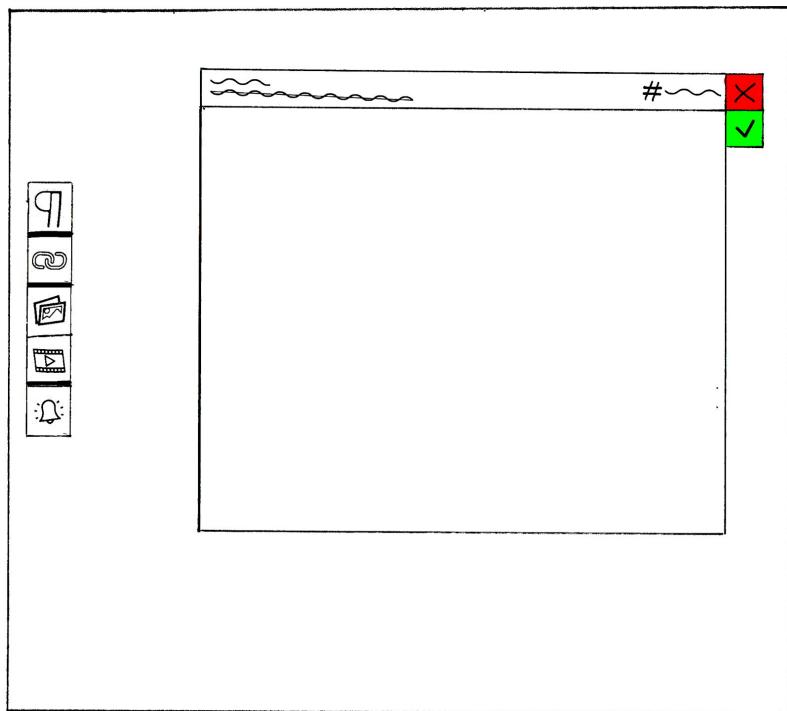
Final - Task 2: Keeping track of assignments and other activities



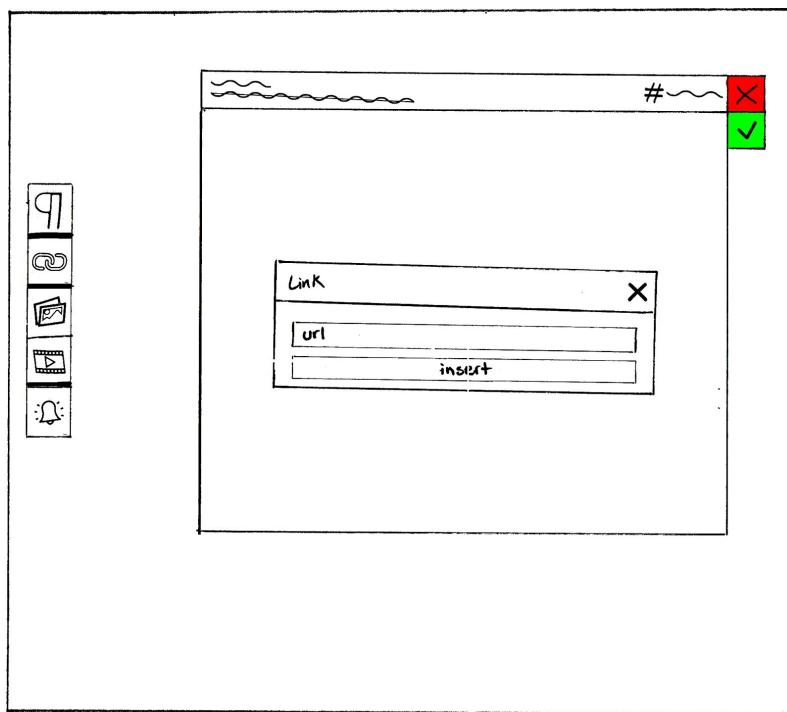
- User registers on the website either by email or Canvas.



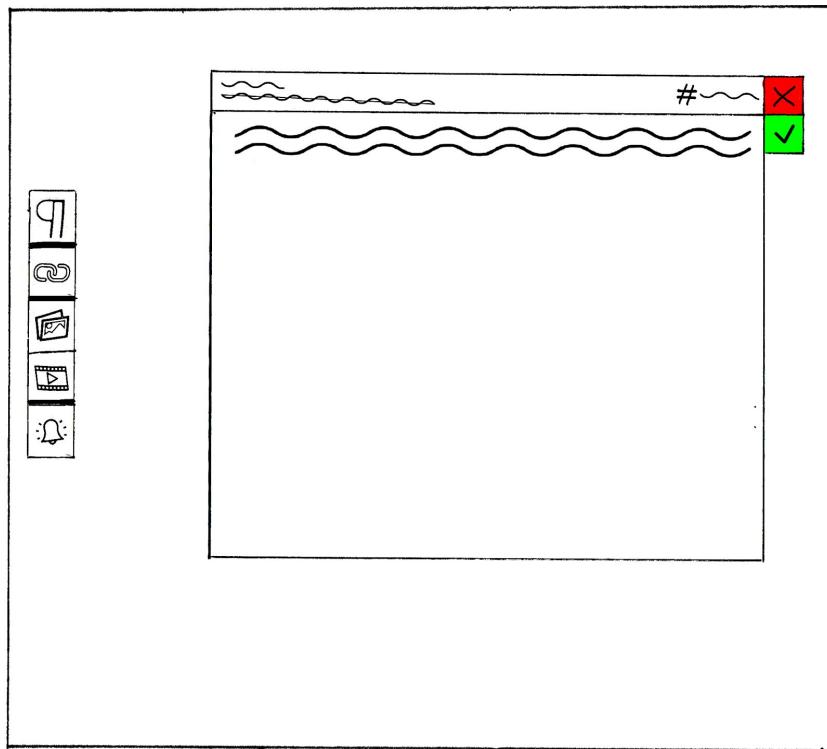
- They are presented with the empty homepage. Greeted and informed that there are no boards (classes) and they can add one.
- Notice the "Add Board" on top left.



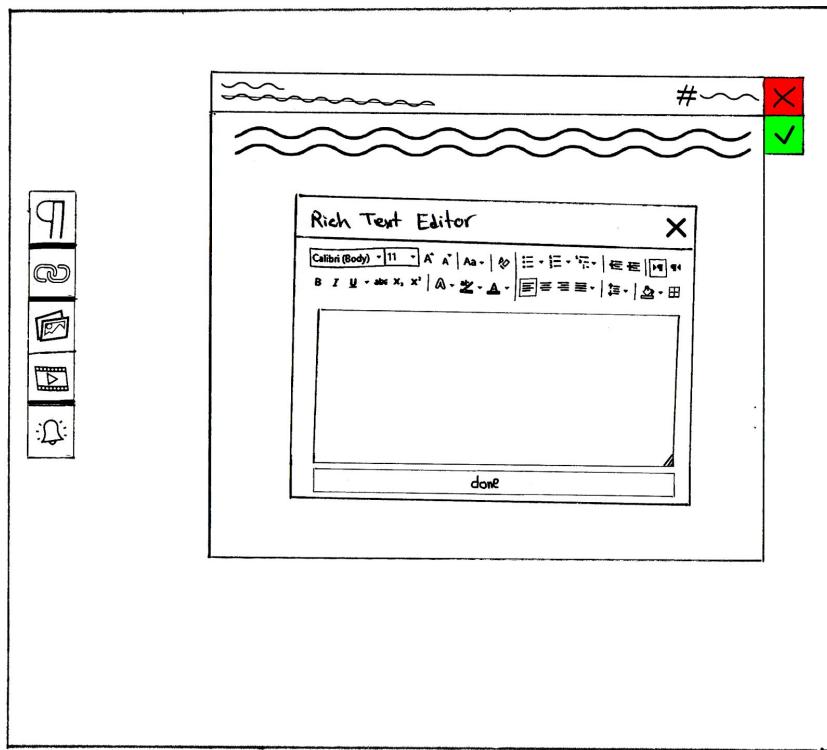
- After clicking on “Add board” they will be presented with the screen above. The content is initially empty. They can fill in the name, description, and a unique tag for the board on top of the edit module.



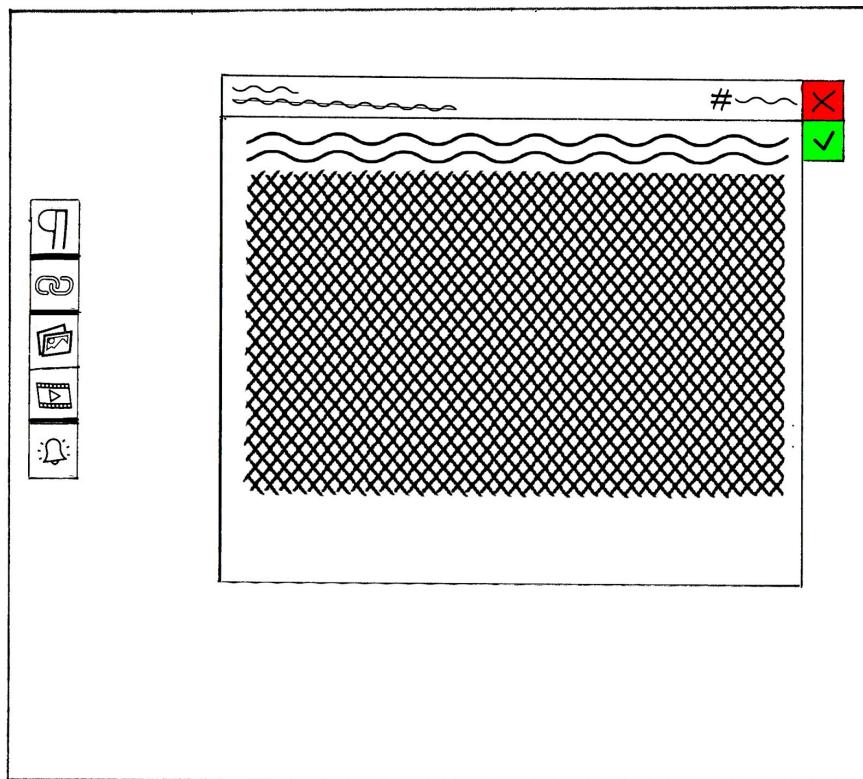
- The user has clicked on the second button (link) and presented with a prompt. They will fill in the information and click on “Insert”. At any point the user can close the prompt and discard it by clicking on X on top right of the prompt.



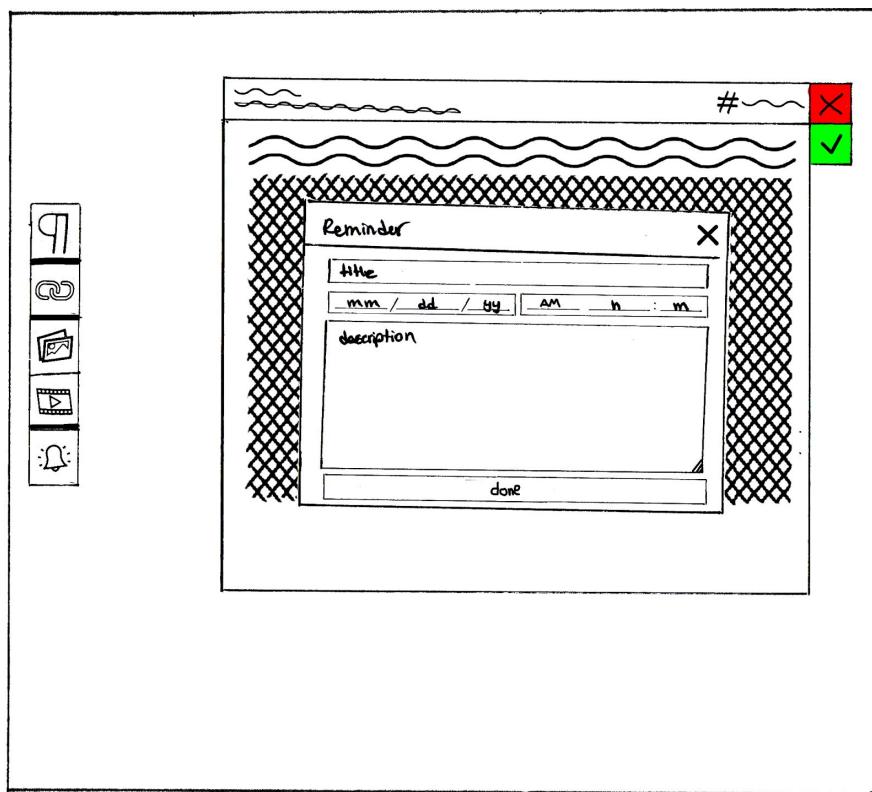
- Link has been added to the topic.



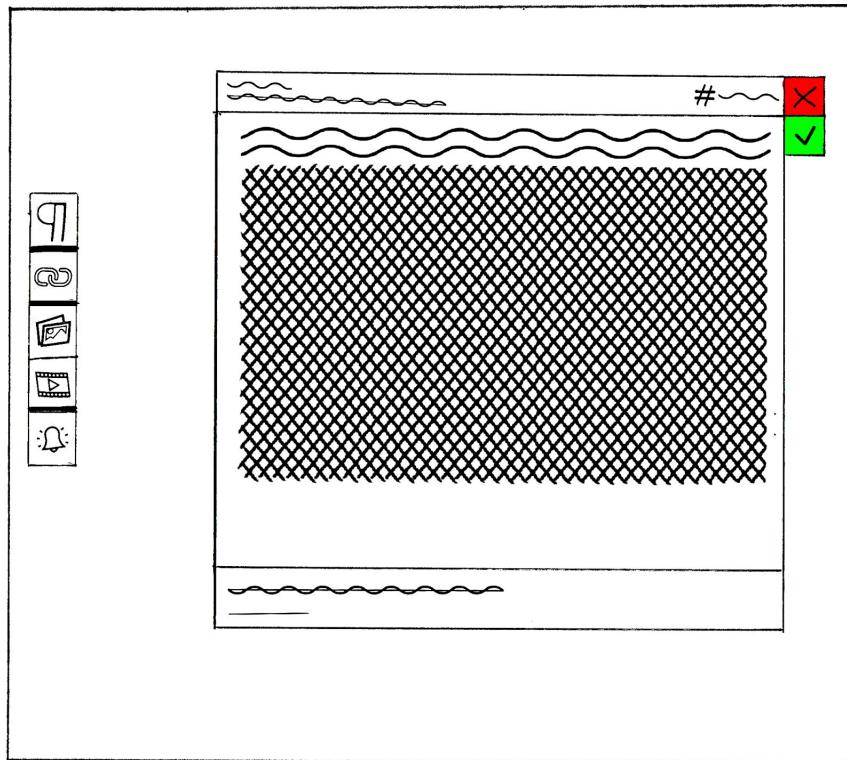
- The user clicks on the RTE (rich text editor, very first button) and the prompt will show up.



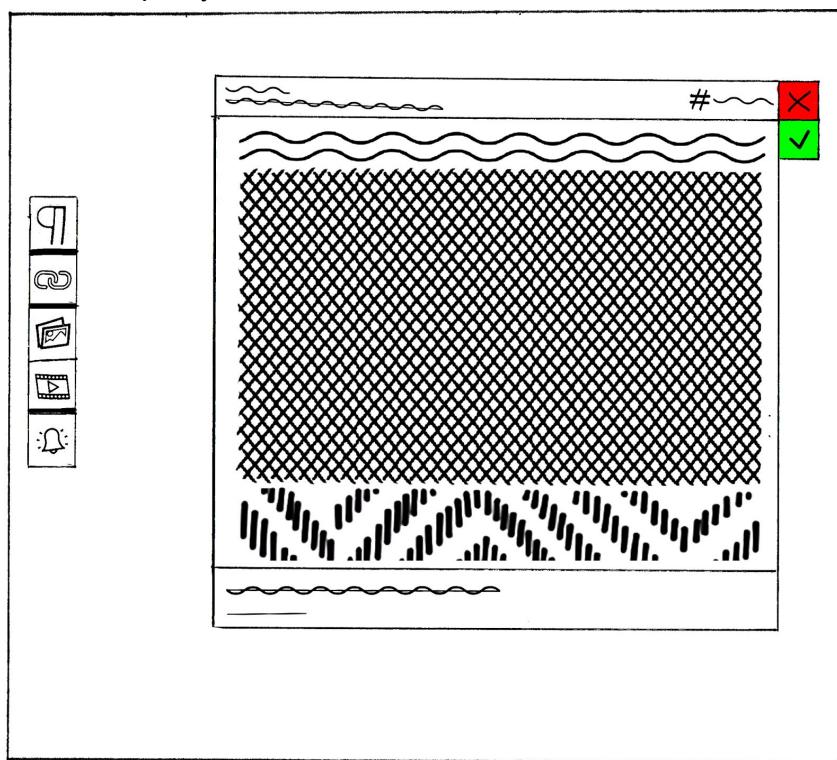
- The text/notes content has been added to the topic.



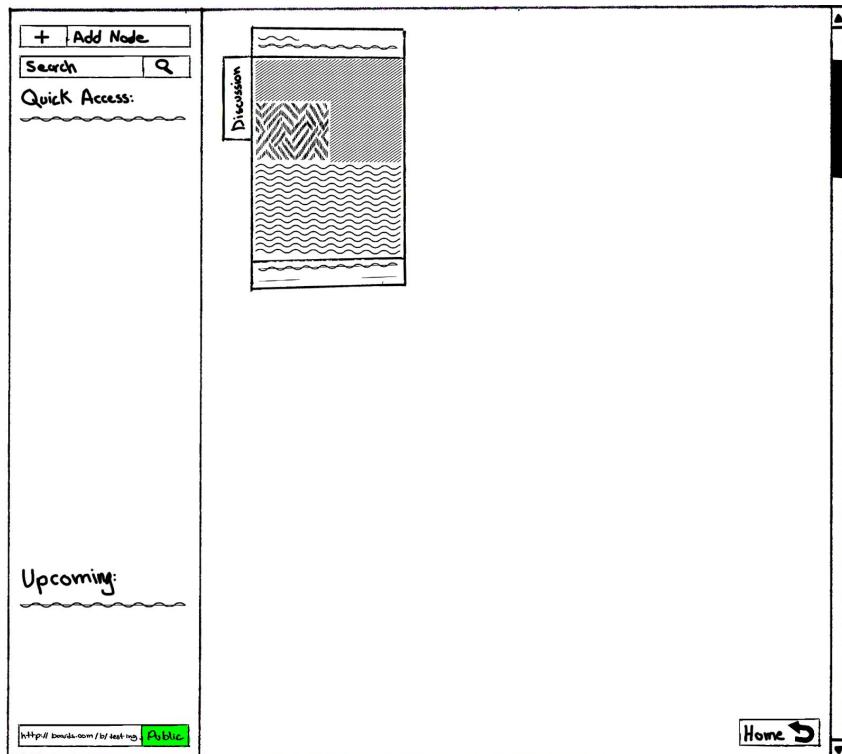
- The user clicks on the reminder button (last button).



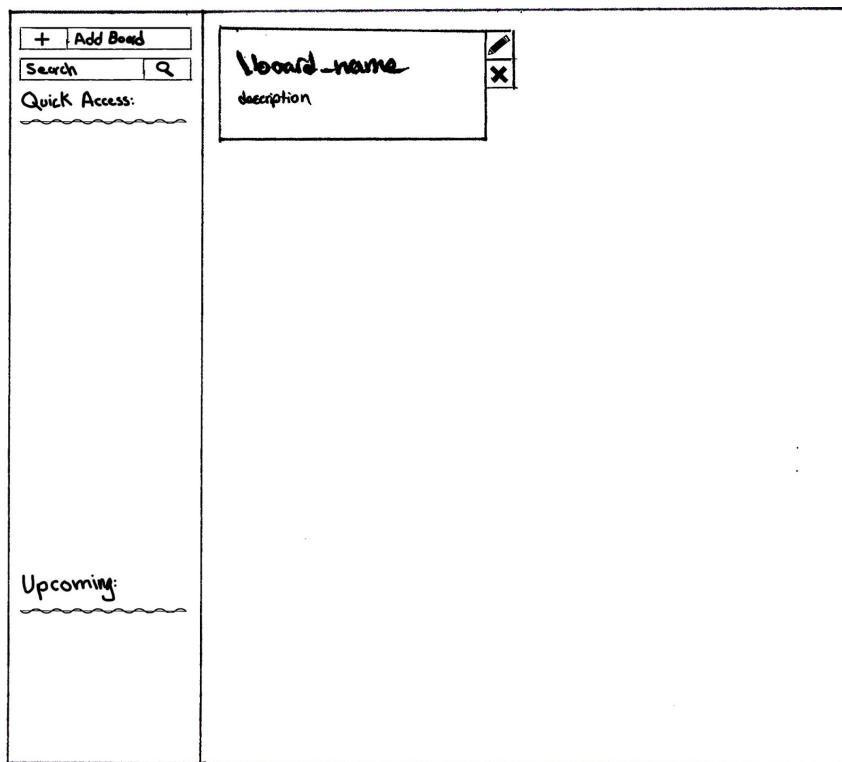
- Reminder has been added. Notice that reminders are added to the bottom of topics for simplicity and visual affordance.



- User has repeated the previous procedure and has populated the topic. Now they will click on the green checkmark to confirm their changes.



- User is taken back to their board (class) view and they can see the recently added topic. They will click on the “back to home” in the bottom corner.



- User is taken to their home, and shows the class.

Final Paper Prototype Important Revisions

The two most important revisions that came from the usability tests were decreasing the number of unnecessary pages and making a universal toolbar. In particular we removed the schedules page altogether and added a new toolbar on the left side of the screen that will persist as the user navigates through the screens.

First, we removed the schedule screen. We realized that this change was necessary after the second usability test when the user ran into the majority of their issues while using or viewing the schedule screen and this was affirmed by the third test as well. We felt that while it was a good idea, the schedule screen took away from the flow of the design and added a layer of complexity pertaining to when it would be updated and how it would be related to each board in general. This page didn't have much of a place in the issues and needs that we found from users in our contextual inquiries and research on the problem domain. For these reasons we felt that the design would be easier to use without the schedules page. The user is still able to keep updated with their upcoming activities through the "Upcoming" section of the toolbar for all their boards (classes) and all their nodes (topics) within each board.

Second, we added the universal toolbar to the left side of the screen that would be accessible throughout the application. This served to fix multiple issues we found during the usability tests around finding specific tools for certain reasons. Primarily it addressed the issues of buttons existing on some pages and not existing on others. This issue resulted with the users sometimes getting stuck on a certain page or taking a long time to figure out how to get to the screen they wanted to see. Having this toolbar allows quick access to recent page locations and offers quicker navigation between the application's screens. Furthermore it addresses the primary concerns raised by the third test where Alex felt that a search function would help users find and use boards much easier and faster by giving a consistent multifunctional tool. Lastly this toolbar will give the user more information while they are browsing the app by showing what can be done on that specific page. This wasn't a major issue raised in any of the tests, but we felt that it would also be a good place to post reminders about upcoming tasks so the user will always have important time sensitive information nearby.

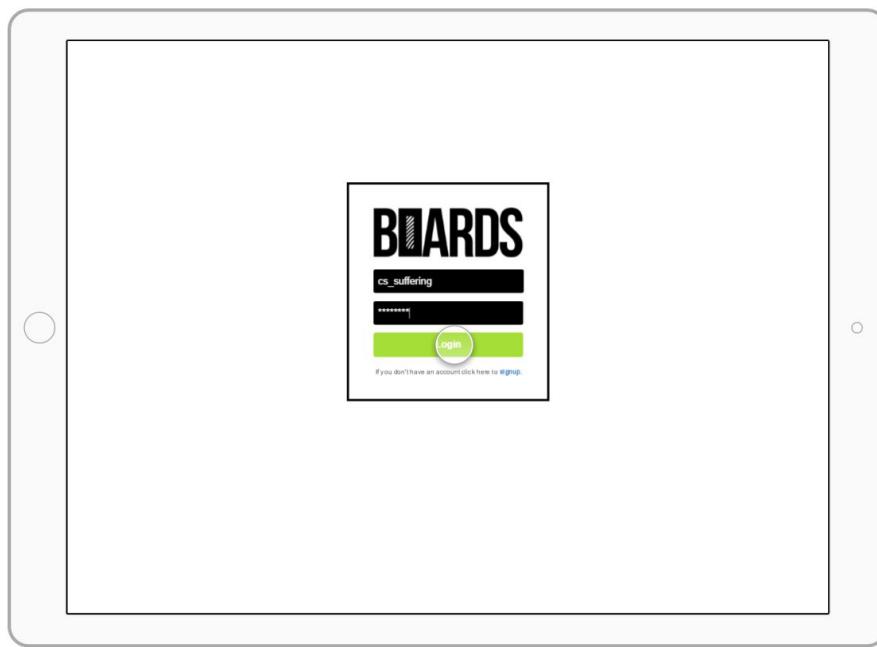
Ultimately, these changes helped to simplify our design a great deal while making it easier for users to navigate and use our application. We believe that these changes will help our two tasks by allowing students to have an easier time accessing helpful information, and keeping track of their schedules and to-dos.

Digital Mockup

Overview:

See appendix

Digital - Task 1: Find help on a given topic



- User enters credentials and logs in the system.

A digital mockup of a mobile device displaying the BOARDS home screen. On the left, there's a sidebar titled "Add Board" with a search bar and a section for "Access your frequently used boards quickly" showing "Calculus III", "CS 4150", "CS 3500", and "CS 3810". Below this is a list of "Upcoming events". The main area shows a grid of course boards. Each board has a title, a brief description, and two small icons (green checkmark and red X). The courses listed are: Calculus I, Calculus II, Calculus III, CS 1410, CS 2420, CS 2100, CS 3100, CS 3200, CS 3500, CS 3505, CS 4150, CS 4490, CS 3810, CS 5100, and CS 5470. The device has a light gray border and circular corner marks.

- Initially they are presented with their home screen -- an overview of all of the classes they have created so far.

- This user decides clicks on **CS 4150** tile.

CS 4150

Add Topic

Search

Access frequently used boards quickly! Please fill out this field.

Knapsack Prob

Dynamic Prog 1

Dynamic Prog 2

Mod Arithmetic

Here are the next upcoming events based on your reminders.

PS9-1	11/19/16
PS9-2	11/19/16
PS9-3	11/19/16
PS9-4	11/19/16
PS9-5	11/19/16

Private

Knapsack Problem
Dynamic programming and memorization

Dynamic Programming 1
Dynamic programming and memorization

Dynamic Programming 2
Dynamic programming and memorization

Test Title
Dynamic programming and memorization

Another Test Title
Dynamic programming and memorization

Yet Another
Dynamic programming and memorization

We Meet Again
Dynamic programming and memorization

Why not?
Dynamic programming and memorization

Yes!
Dynamic programming and memorization

Home

- User enters CS 4150 class and is presented with all of their topics.
- They decide to search for a keyword

CS 4150

Add Topic

Search

Access frequently used boards quickly!

Knapsack

Knapsack Prob

Dynamic Prog 1

Dynamic Prog 2

Mod Arithmetic

Here are the next upcoming events based on your reminders.

PS9-1	11/19/16
PS9-2	11/19/16
PS9-3	11/19/16
PS9-4	11/19/16
PS9-5	11/19/16

Private

Knapsack Problem
Dynamic programming and memorization

PS9-1
11/19/16

PS9-2
11/19/16

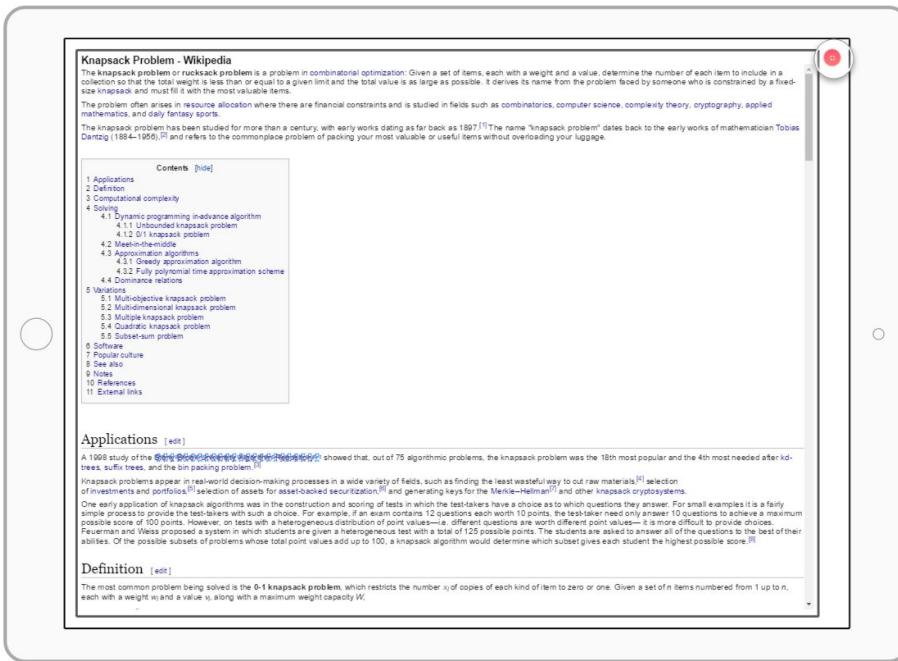
Home

- User enters “knapsack” and the view is dynamically updated to include only the related topics, based on relevancy of the title and content.

- User clicks on the discussion tab for the “Knapsack Problem” topic.

- The “Discussions” view is automatically populated, with content aggregated from the internet. Discussions view automatically determines the relevant keywords based on title and content of the topic.
- You can see that the user clicked on the discussion board of “Knapsack Problem” however the search bar contains “knapsack problem” as well as two more terms closely related to the topic. This is due to the fact that “Discussions” view is content aware.

- Furthermore the user clicks on the green "Fullscreen" button next to a result.



- The result takes up the whole screen. This is an efficient way to quickly find related content and get an overview of their info based on the summary, and later quickly expand the result for more information and in-depth analysis.
- The user will click on the red "Close Fullscreen"

The screenshot shows a search results page with a header stating "I can search for a custom term and results will show up here". Below this, there are five search results, each showing a snippet of the Wikipedia page for the Knapsack Problem. Each snippet includes a green 'Fullscreen' button and a small green circular icon with a plus sign. A red circle highlights the search bar at the top of the page, indicating where the user entered their search term.

- User is taken back to the results page, where they can furthermore explore the topic.
- If the user is not satisfied with the automatic results, they can refine it by entering their own search term in the search bar similar to search engines like Google.

- After exploring the content and information on “Knapsack Problem” and other closely related topics, user decides to return back to the class view.

- User is taken to their topic search view. Even though in the image above there is only a single topic as the search result, there can be many more depending on the search term. Thus bringing the user back to their topic search view, allows them start exploring the “Discussions” of other topics that can be the result of their search.
- User decides to clear the search bar, and upon doing so the dynamic search view will disappear.

The screenshot shows the EdHub interface for the CS 4150 class. On the left, there's a sidebar with a search bar and links to 'Knapsack Prob', 'Dynamic Prog 1', 'Dynamic Prog 2', and 'Mod Arithmetic'. Below this is a section for 'Upcoming events based on your reminders' listing assignments PS9-1 through PS9-5. At the bottom of the sidebar is a 'Private' button.

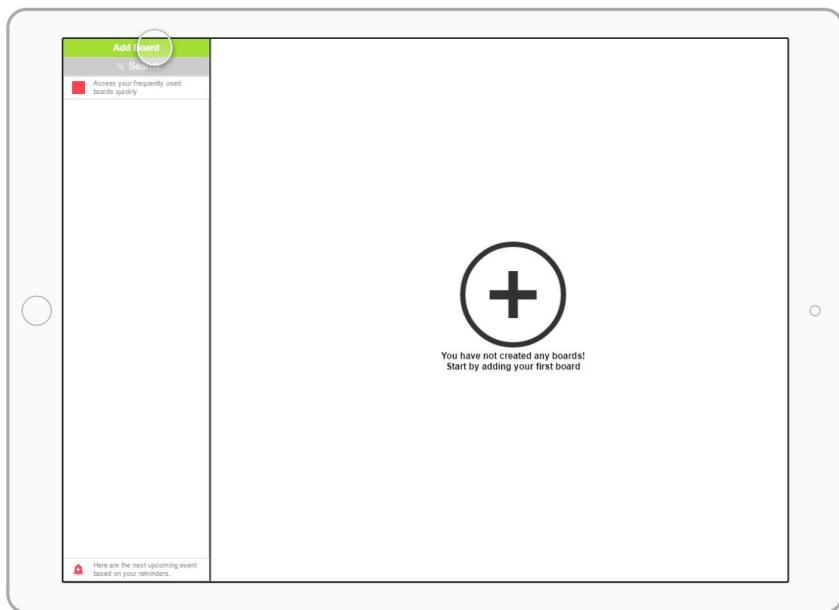
The main area contains several cards:

- Knapsack Problem**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.
- Dynamic Programming 1**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.
- Dynamic Programming 2**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a red error icon.
- Test Title**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.
- Another Test Title**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.
- Yet Another**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a red error icon.
- We Meet Again**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.
- Why not?**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a red error icon.
- Yes!**: Dynamic programming and memoization. Includes a discussion section with two entries, each with a green checkmark icon.

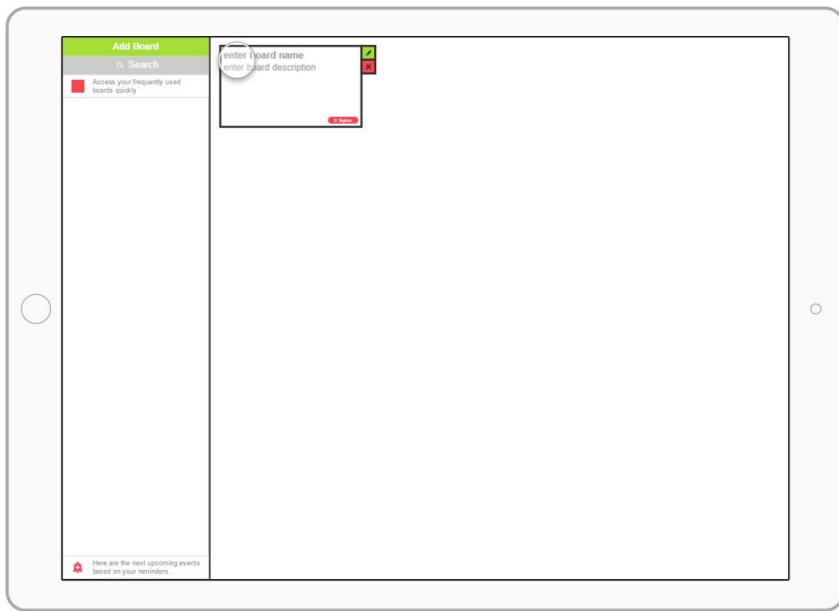
At the bottom right of the main area is a 'Home' button.

- Dynamic search view has disappeared and once again user has full access to view all of their topics within the CS 4150 class.

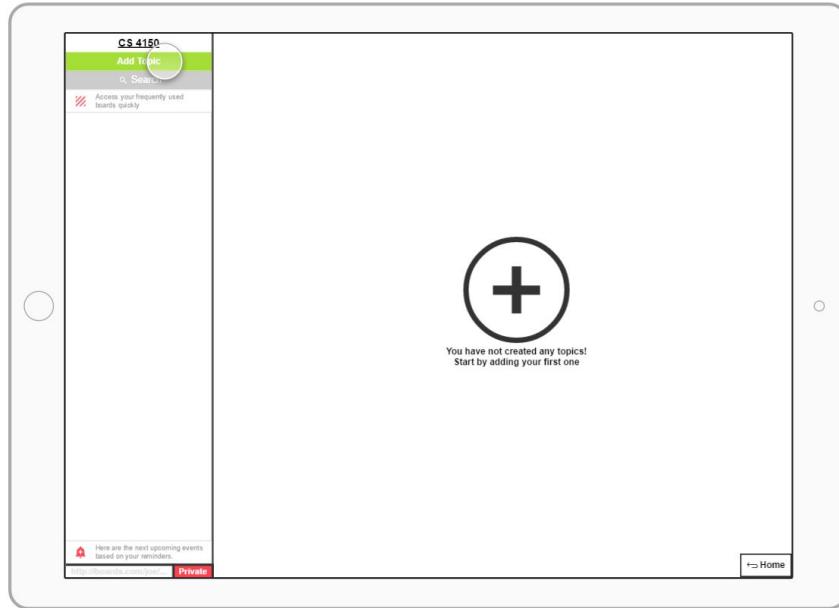
Digital - Task 2: Keep track of assignments and other activities



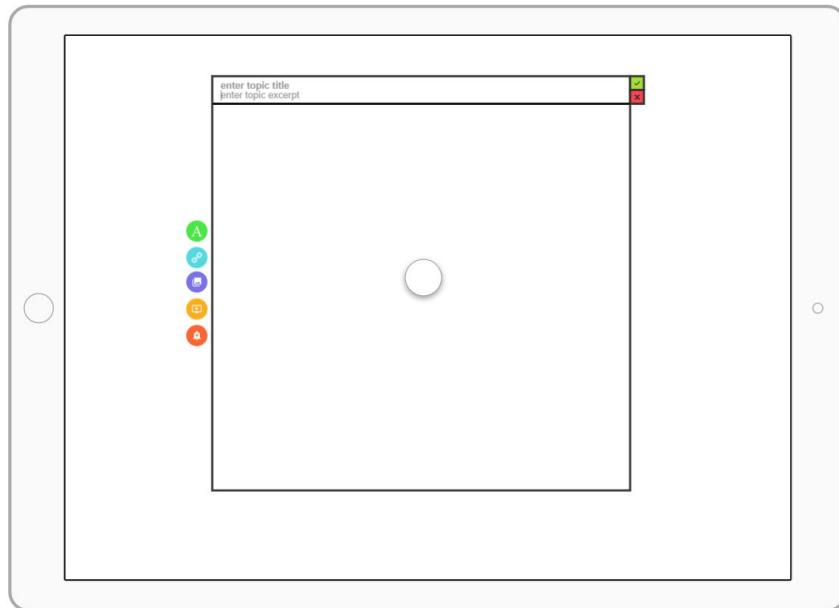
- Initially they are presented with an empty home page and informed that they not have classes (boards).
- User will proceed by clicking on the "Add Board" button



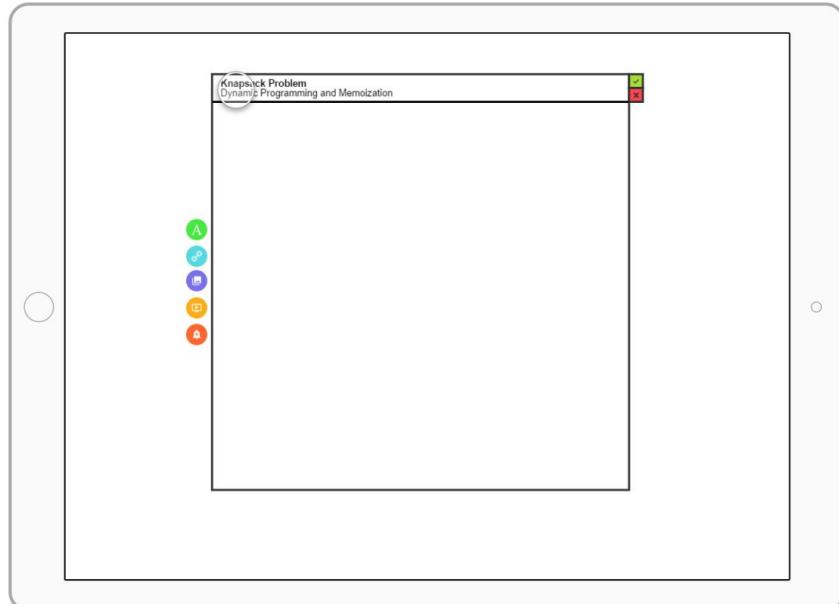
- An empty board is created with blank title and summary.
- User will proceed by filling the title input field and summary area field and click on the board tile.



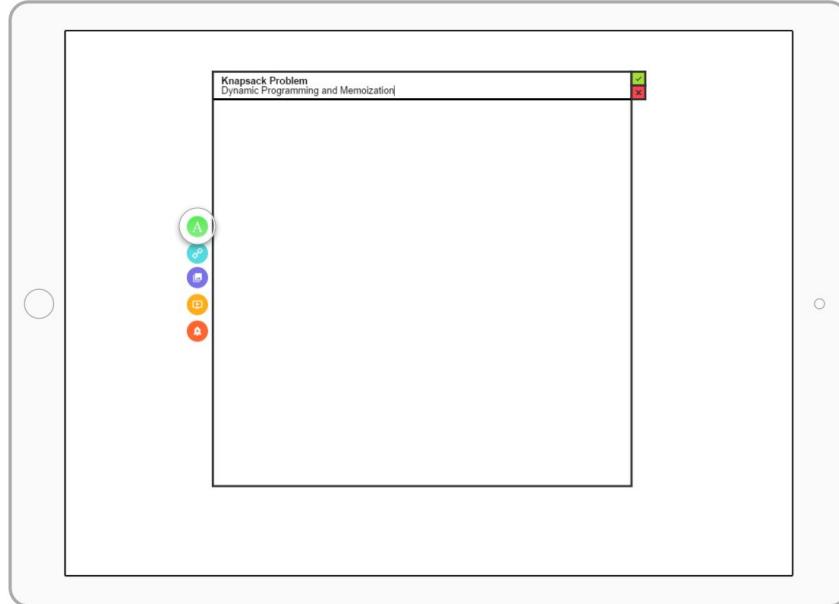
- User has entered their newly created CS 4150 class. They are informed that they have not yet created any topics for this class.
- User will proceed by clicking on the "Add Topic" button.



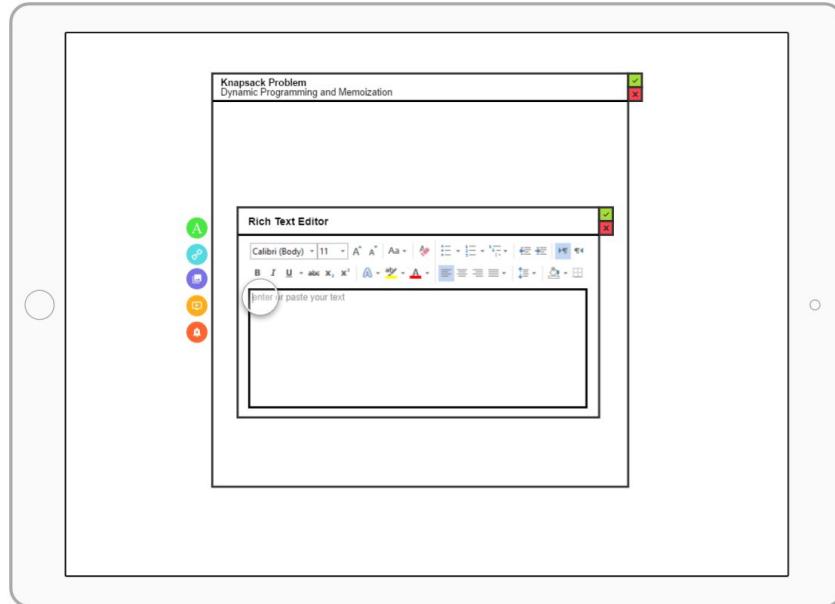
- Next they enter the edit/view screen, where they can edit and create a topic, read a topic or modify a previously created one.
- In this scenario the topic was just created and thus is empty.



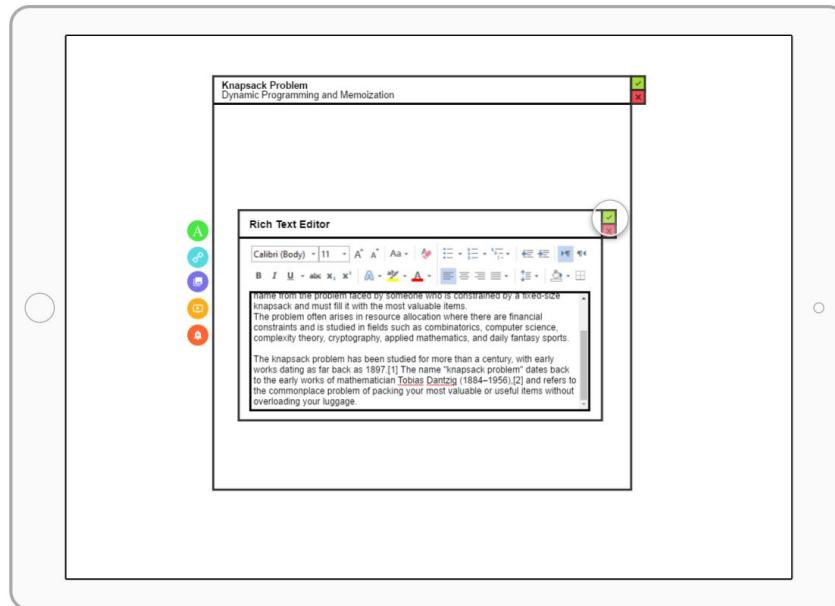
- User will fill in the title and short summary of the topic



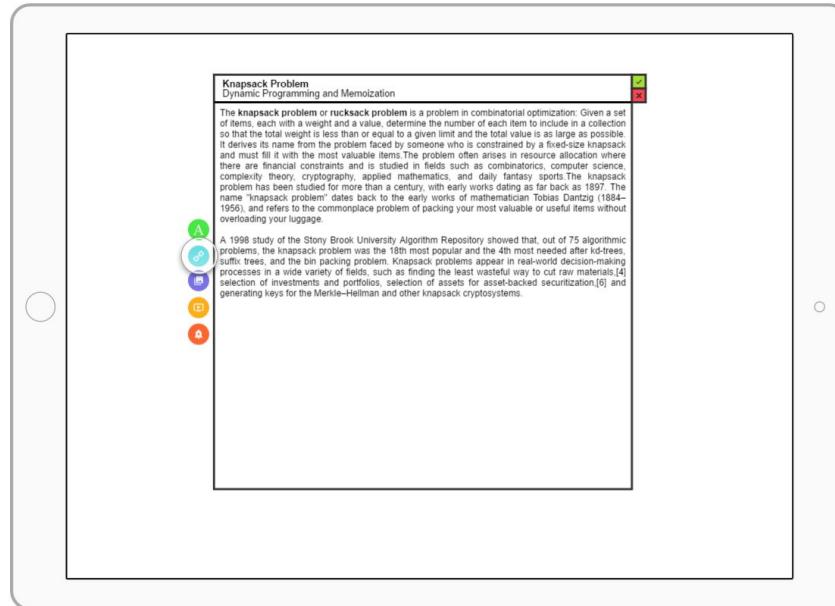
- Next they will click on the "Text" button where they can create rich text components for a node.



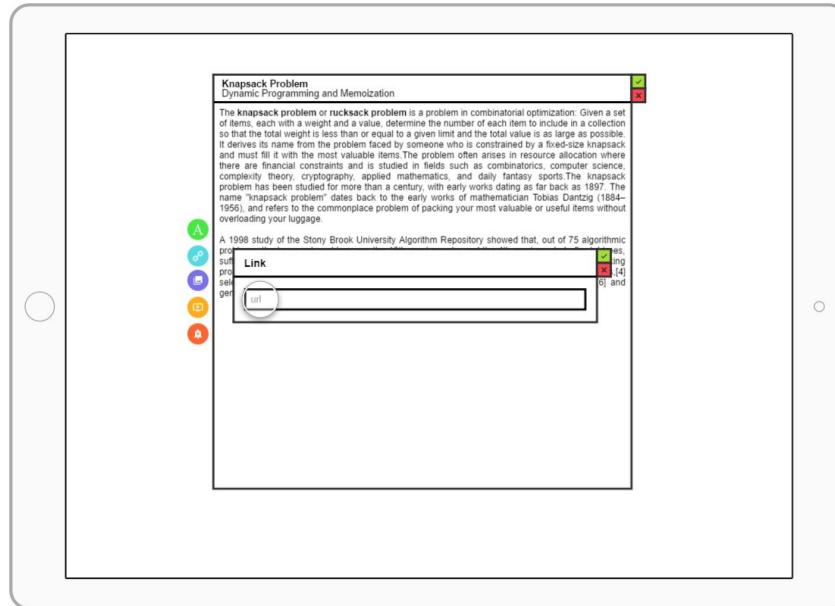
- The “Text” prompt opens.



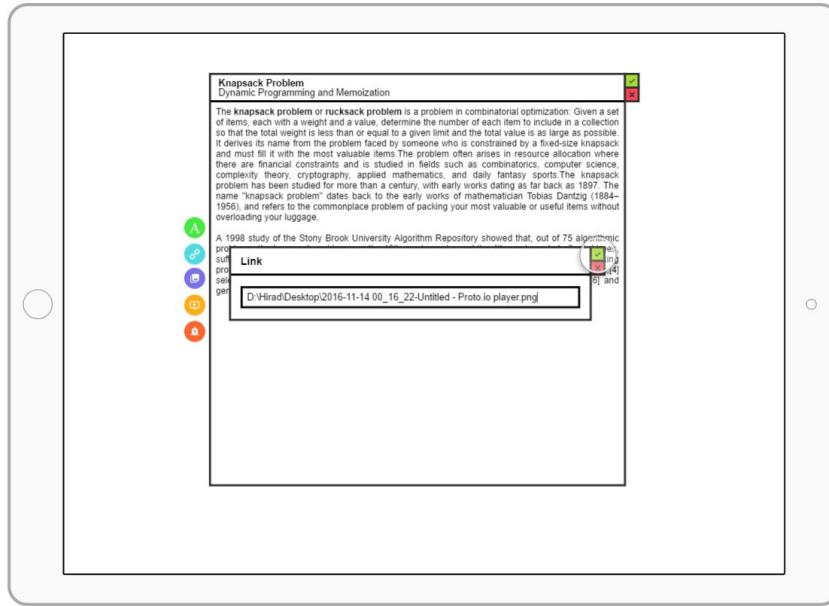
- User will paste/type the text of their choice and click on the checkmark to approve their additions.



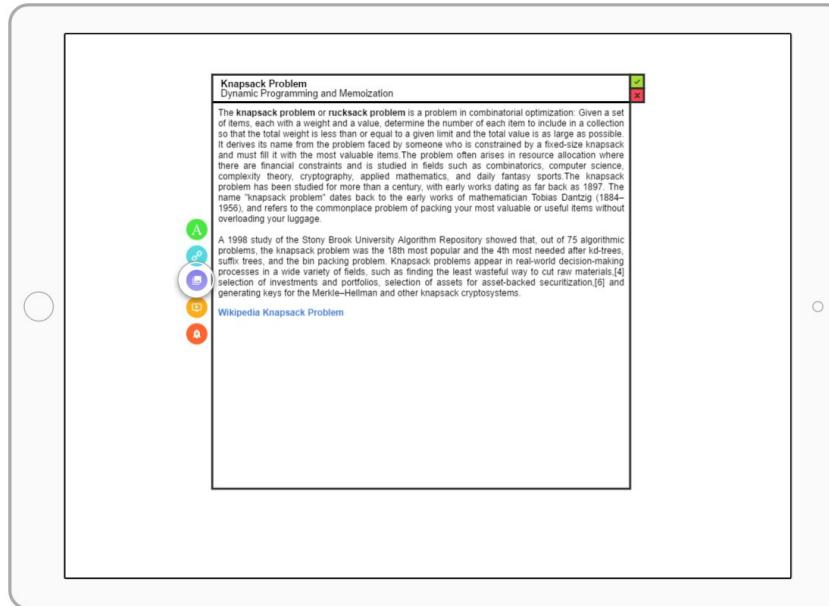
- User's text is added to the topic and displayed.
- Next they will click on the "Link" button to add a link.



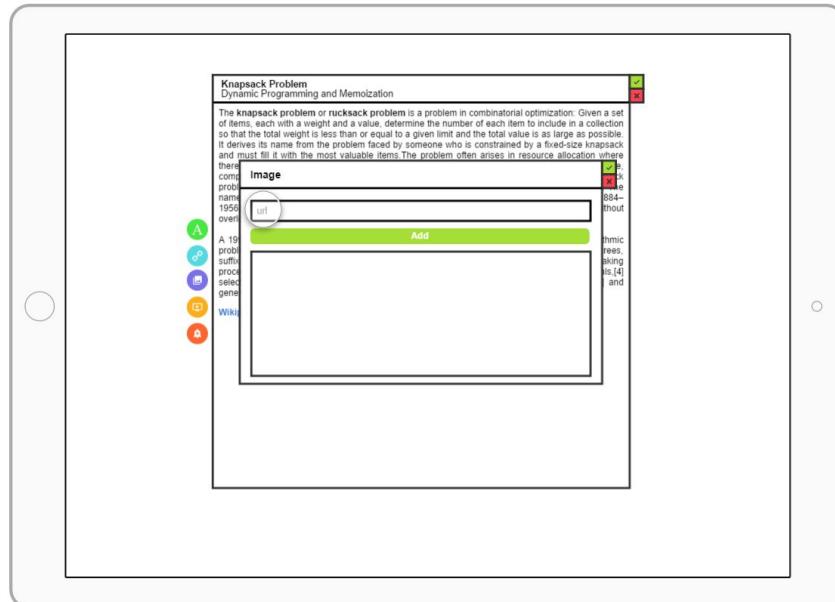
- The "Link" prompt opens up.



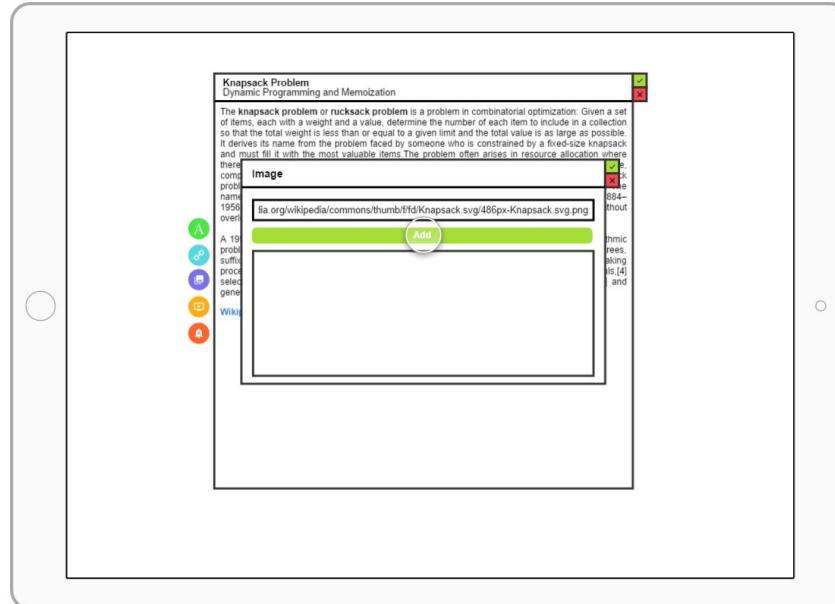
- User enters their link and clicks on the checkmark to approve their additions.



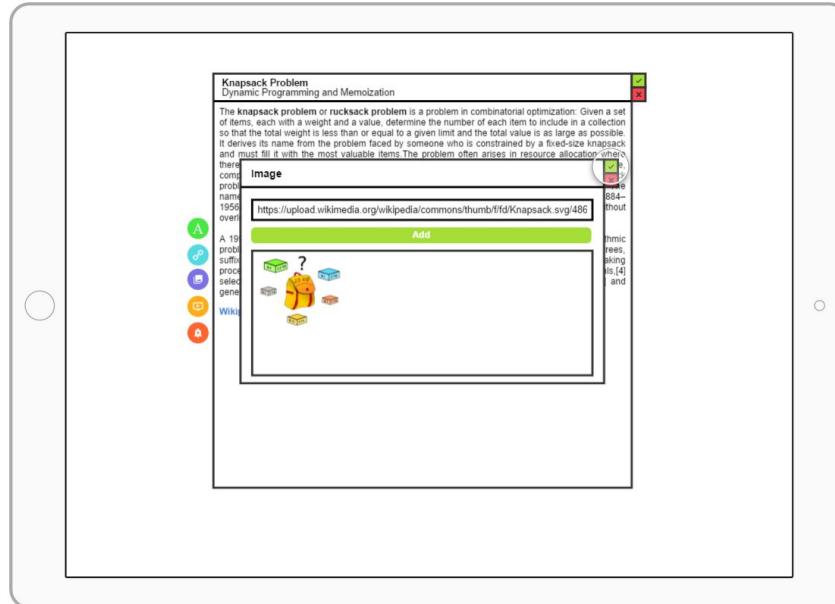
- The link is added to the topic.
- User will next click on the "Image" button.



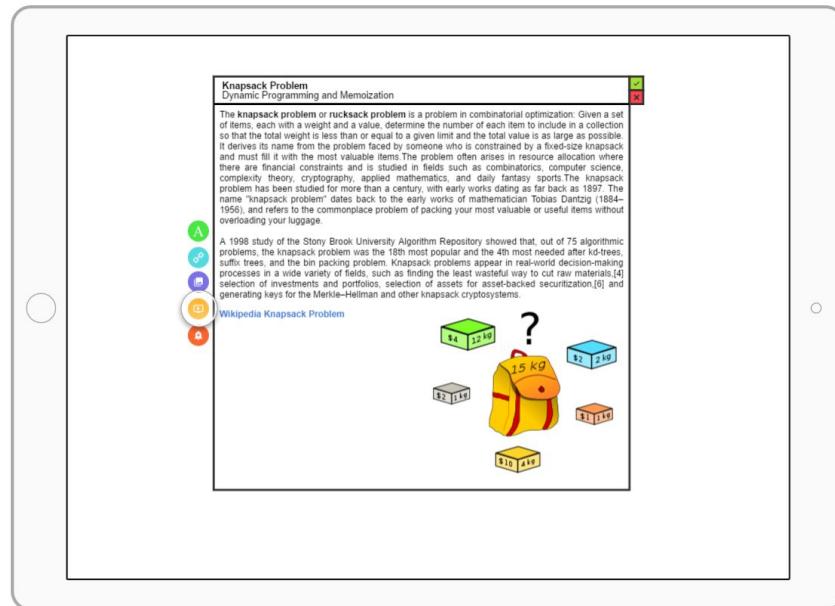
- Image prompt opens up, and user will paste the link to their image.



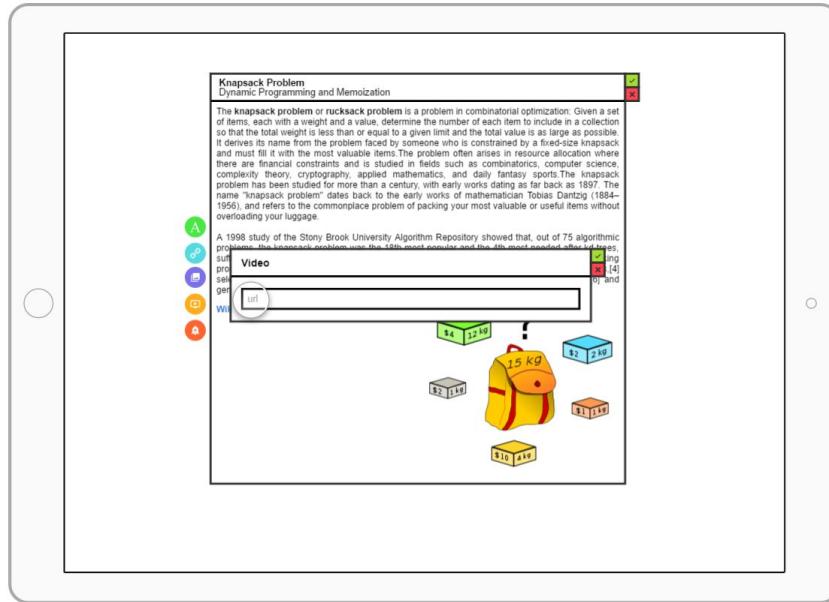
- After copying and pasting the text in the URL field, user will click on add.



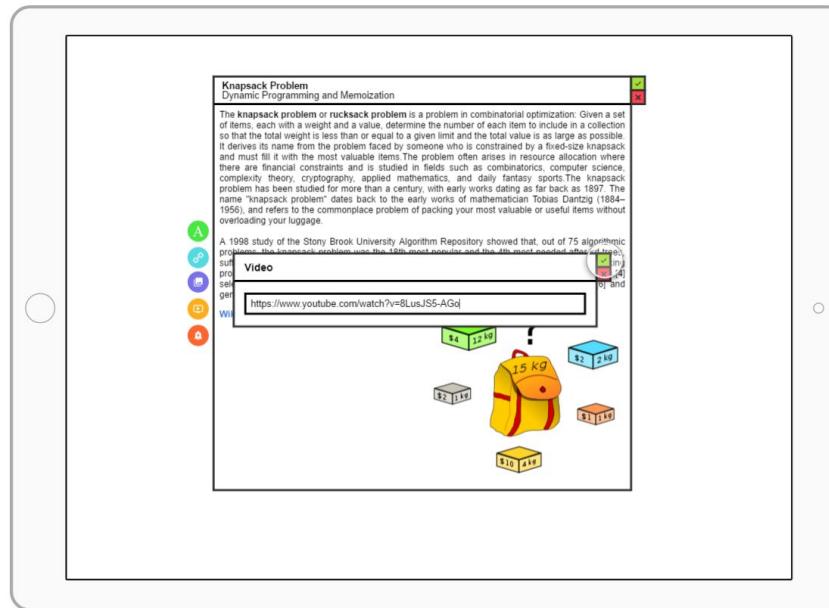
- Their image is now added to the gallery. The image prompt allows the user to add a single or multiple images in one go. As images are added, their thumbnails shows up below the add button.
- In this case user was interested in adding only a single image. They will click on the checkmark and approve their additions.



- The image is now added to the topic.
- User will then click on the "Video" button.



- The “Video” prompt opens up, allowing the user to add videos from popular video hosting websites.



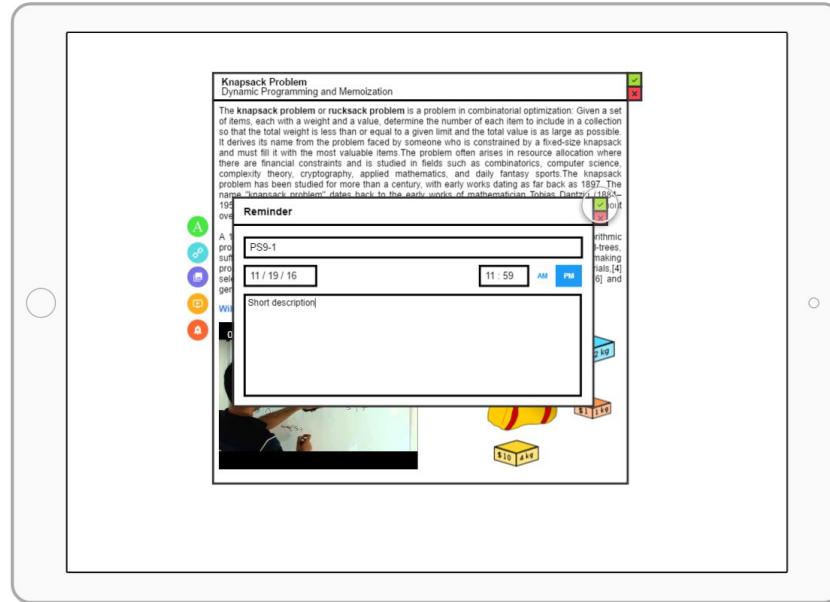
- User has pasted a Youtube link for a video related to the Knapsack Problem. They will click on the checkmark to approve their additions.

The screenshot shows a digital whiteboard interface. At the top, there's a header bar with the title "Knapsack Problem" and "Dynamic Programming and Memoization". Below the header is a detailed text description of the Knapsack Problem, mentioning its history and applications. To the left of the text is a vertical sidebar with icons for different sections: A (Algorithm), D (Dynamic Programming), S (Suffix Trees), B (Bin Packing), W (Wiki), and R (Reminder). In the center of the screen, there's a video player showing a person writing on a whiteboard with a diagram of a knapsack and several items labeled with weight and value. To the right of the video player is a small illustration of a yellow backpack labeled "15 kg" surrounded by various items.

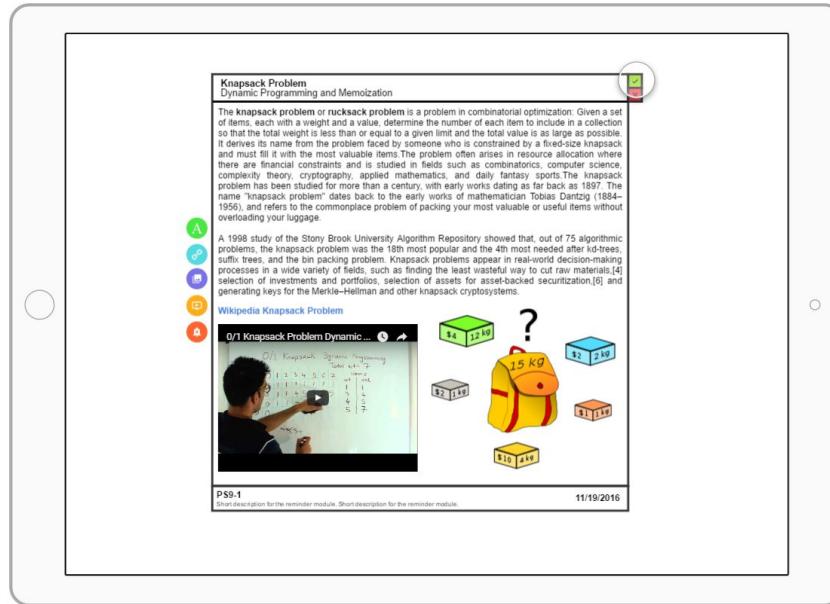
- The video is added to the topic.
- User will then proceed by adding a reminder. They will click on “Reminder” button.

This screenshot shows the same digital whiteboard interface as the previous one, but with a "Reminder" dialog box overlaid. The dialog box has fields for "Title" (with "Knap" highlighted), "Date / Time" (set to "MM / DD / YY" and "11:11 AM PM"), and "Reminder description" (empty). The background content remains the same, including the video player and the knapsack diagram.

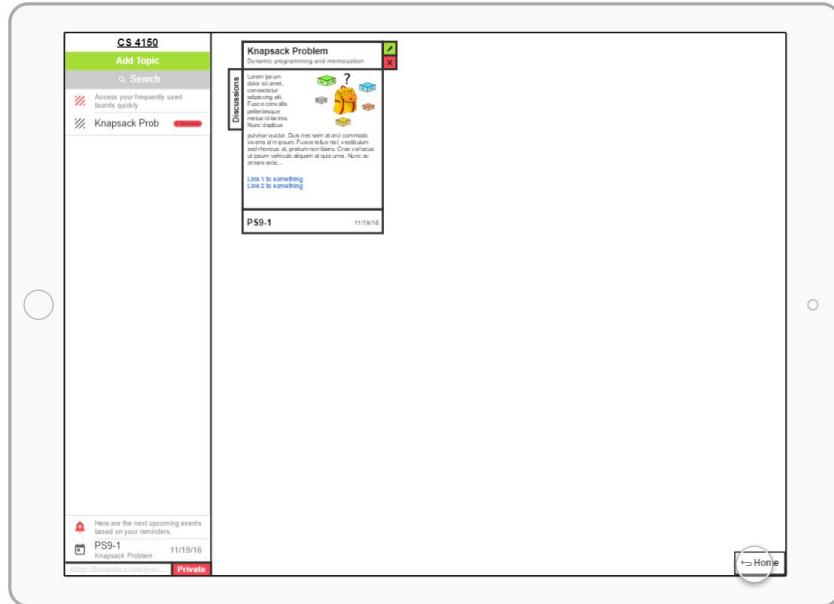
- The “Reminder” prompt opens up.



- User has entered the information for their reminder, including the title, date, time, and description. They will continue by clicking on the prompt checkmark.



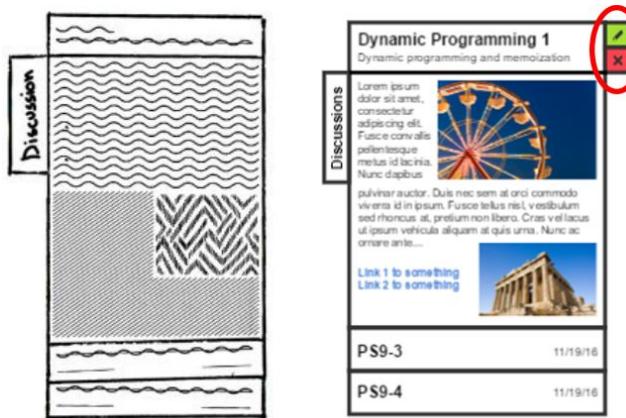
- Reminder is added to the bottom of the topic. For visual simplicity and quick feedback, reminders are always added to the bottom of the topics.
- User will click on the topic checkmark button to approve the completion of their modifications.



- The “Knapsack Problem” topic has been successfully added to the “CS 4150” class. And a thumbnail of the topic is visible to the user for quick visual feedback.
- Also the “Quick Access” section on the left now shows the topic “Knapsack Problem” since this was the most frequently interacted topic in this class.
- The “Upcoming” section on bottom left also displays the one and only reminder the user added to the topic, which is the only one present in the entire class as well.

Moving from Paper to Digital / Critiques & Changes:

Generally there were no major changes to our design. However below we have discussed the minor changes and additions which have occurred in the transition from paper to digital.



- We have added an edit/delete combo button to add topic previews. So the user can quickly remove a topic, or continue to modify its content.

Even though the user can directly modify a topic while viewing/reading it, the green edit button defined a concrete visual element for the edit task.

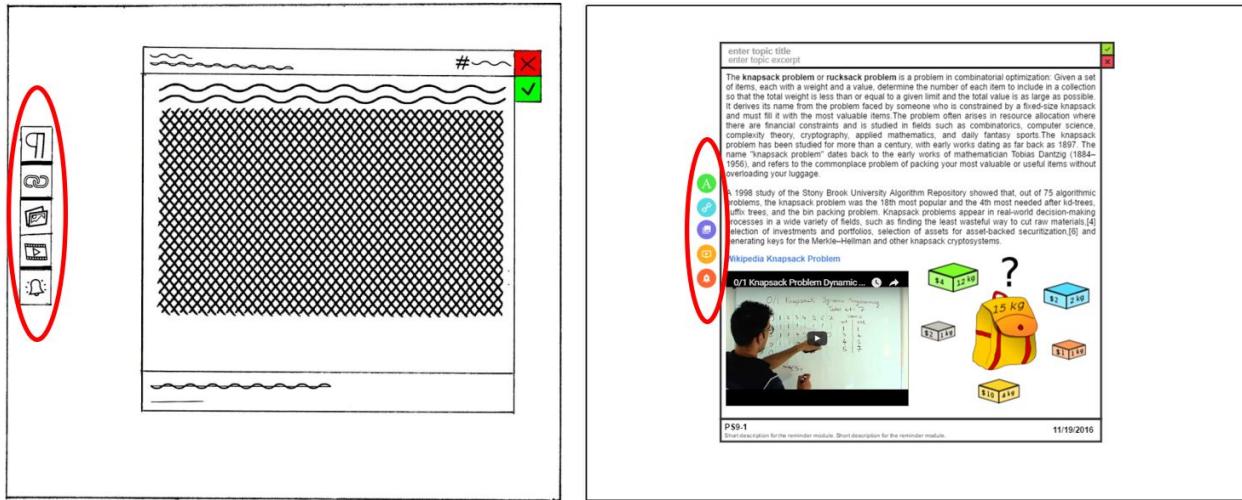
2. We have removed the **Source** title of the results in the discussion page. This is due to the fact that this info can be easily combined with the result title to reduce distraction and provide a cleaner visual feedback.

3. Also we have added a fullscreen button to all discussion results so that the user can quickly expand and consume the content. The fullscreen view removes all distractions from the webpages, and behaves very similar to the **Reading Mode** available on modern browsers. Users can quickly collapse the fullscreen view to return to the discussion results.



4. For module prompts, which are accessible from the topic edit view, we have removed the confirm button (**insert** in above image, **done** in other prompts) and combined it with the **close** button as a combo approve/discard functionality in the upper right corner of prompt in our digital mockup.

This decision was made to reduce the redundancy of button throughout edit prompts and unify the experience with the topics, and class tiles.



5. We have placed the module buttons for editing topics in close proximity of the edit window itself. This decision was made during our digital testing, when we realized that the distance the mouse needs to travel from the center of the edit window to the click on the modules on far left can be tiresome and irritating if the user needs to add a large number of modules to the topic.

Discussion

Throughout the process of making the right design for our project we gradually grew more aware of the scope of the project and refined our original ideas. Thanks to the process of iterative design we were able to start off really small and work our way up using incremental building blocks that weren't necessarily challenging on their own. We think that if we had started putting our original ideas into the final design right away we wouldn't be able to come up with such a very professional and polished results similar to what we have right now. Furthermore, the process of iterative design made revising and editing our designs much easier due to the size of every specific iteration. For example, when we were working on our storyboard, it was easy to fix any flaws with that part of the project without worrying about other parts or even future parts that haven't been integrated yet. This process helped us see the importance of focusing on the smallest building block first and construct our design from that block. It ensures the best possible result due to the fact that every building block has been polished and revised before moving to the next one. As a result, all of the blocks will eventually yield a collection of almost perfect blocks built on top of each other to create the perfect design. This is exactly how this process helped us in coming up with our current final design.

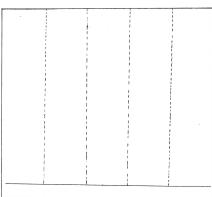
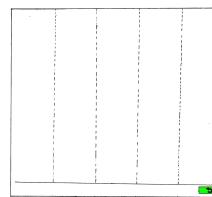
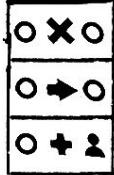
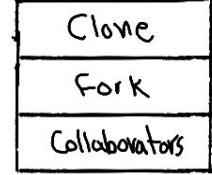
Our original tasks were general and then changed to become more specific due to our usability tests and the fact that our initial iterations were too difficult to be task oriented. During our usability tests, we noted that the participants were not easily able to complete the tasks assigned to them because they involved a lot of smaller steps that weren't accounted for in our initial iterations. This pushed us toward two main changes to be made to our design. First, we had to get rid of many features to bring the size of the design small enough to be task oriented, intuitive and easy to understand. Second, we had to change our initial broad tasks to more specific ones that targeted specific desired functionality within our design.

Although the process of iterative design was very helpful to us, we would argue that the number of iterations given couldn't be fewer. The reason for making that statement is that until the last iteration of our design, we still had to make a lot of changes to get it right. As for having more iterations, we would argue that there is no need for more iterations due to the fact that we are confident that the design we have right now is sufficiently good enough, for the scope of this course, to meet its objectives.

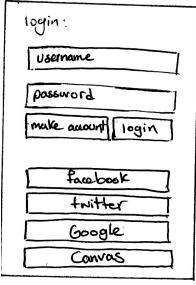
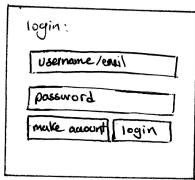
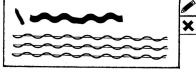
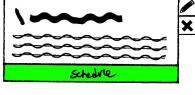
Appendix

Heuristic Test

Prototype Image	Identified Issue	Heuristic & Severity	Revised Image	Revised Explanation
	No confirmation button to save or apply the changes to a board.	H: 3 S: 6 A minor visual fix		The Check button was added so that the user can save a board change.
	No cancel button to save or apply the changes to a board.	H: 3 S: 5 minor visual fix		Added a "Cancel" Button to allow the user to exit the edit and to not save their changes
	No confirmation button on editing of fields.	H: 3 S: 5 Same as above issue		Added a "Done" button to allow the user to exit the edit
	The editing menu selections on the left side of the board edit screen has too many options	H: 7 and 8 S: 2 A smaller issue, but takes away from the ease of use of the design		A few of the options were taken out to make the design seem simple and efficient to make it cater to our chosen tasks.

	No back button on the schedule page to return to the home/boards screens	H: 3 S: 8 Makes schedule screen virtually unusable as the user cannot navigate anywhere from there		Add back buttons to return to both the board and home screens
	Clone and fork buttons are small and not intuitive as to what they do	H: 2 S: 3 Doesn't afford usability leading to them not being used.		These buttons were changed to text to better afford their usage. The terminology was also changed to more relatable terms

Usability Test 1 -- Leo

Prototype Image	Incident	Severity (Negative)	Revised Image	Explanation of change
	User was confused as to what the "Facebook" "Google" & etc. buttons were meant for on the home screen	4 Takes away from the simplicity of the application at the risk of over complicating it		We decided to take these out to cut down on the clutter on the home screen and to make the app easier to use
	User expressed frustration with not being able to view the schedule of a board from the home screen	2 A minor complaint but takes away from the ease of use of the design		We added a schedule button to each of the saved boards on the home screen

Usability Test 2 -- Calin

Prototype Image	Incident	Severity	Revised Image	Explanation of change
	In the schedule page, the user clicked on the date at the bottom of the page instead of the node above. Nodes don't afford being clickable.	S: 1 Mostly an artifact of the paper prototype and will be better addressed in the digital mock-up.		Schedule page has been fully replaced to remove the complexity related to this issue and other issues.
	User thought that the QA tab on the left of the board was a label so didn't click it to find the help.	S:4 We want it to be easy to find help on issues, this makes the process harder.		Text has been changed to "Discussion" to promote a different level of clarity and usability.
	Related to above issue, the user didn't know that they could click on the QA tab and "QA" didn't afford itself to finding help.	S: 4 Button needs to be easy to tell it's clickable, and also descriptive about its purpose		The "clickability" of the button is an artifact of the paper prototype. QA was changed to "Discussions"
	User was unable to get back to schedule page from the board edit/summary page	S: 3 Minor fix, but a vital one as the user needs to be able to easily and quickly get to the schedule		Removed the confusing back to boards button. Now clicking on the X or checkmark will take the user back to their topics/nodes view.

	User noticed the "home" button on the schedule page is different from the "boards" button on other screens	S: 1 Minor fix to maintain consistency		Changed all "boards" buttons to say "Home"
--	--	---	--	--

Usability Test 3 -- Alex

Prototype Image	Incident	Severity	Revised Image	Explanation of change
N/A Missing UI element	In boards view, Alex was frustrated because he couldn't look boards up with a search bar.	S: 1 Users must be able to find any board easily.		We added a search bar at the top of the boards view that limits the boards to matching search criteria.
N/A Missing UI element	In nodes view, Alex was frustrated because he couldn't look nodes up with a search bar!	S: 2 Users must be able to find any node easily within the linked nodes view. It can be confusing without a simple finder.		We added a "search" field to each board so the user can quickly fuzzy search nodes.
N/A Missing UI element	Alex noted that he couldn't add a board to an existing list of boards.	S: 4 Users Should be able to add boards to an existing list of boards.		We added a "+" sign at the end of the boards list to allow users to add new boards.

Overview of Digital Design



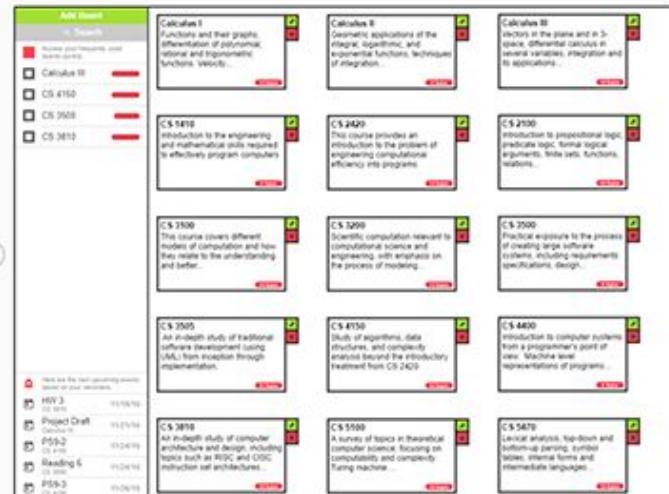
Screen 1: Login Page



Screen 2: Signup page



Screen 3: Empty home screen



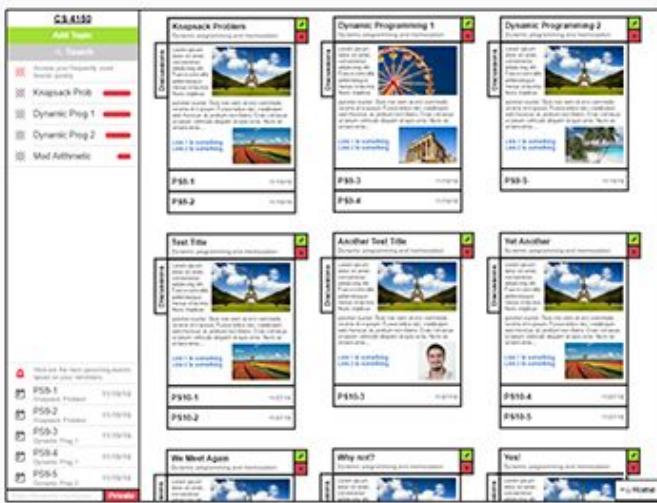
Screen 4: Populate home screen with classes



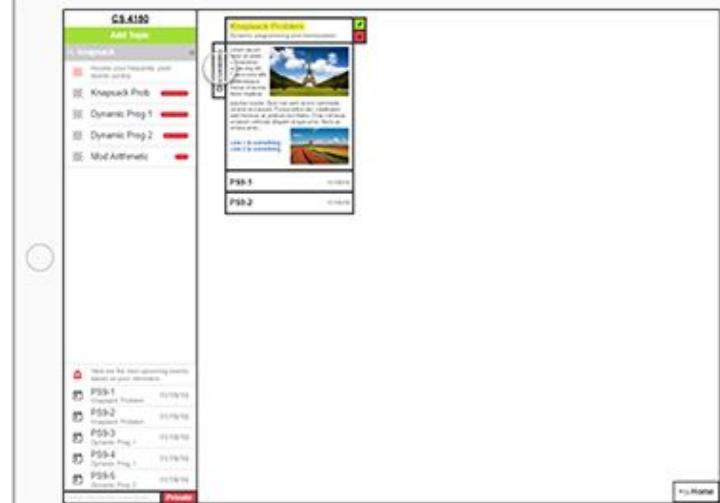
Screen 5: Empty class screen



Screen 6: Class screen with a single newly created topic



Screen 7: Class screen populated with different topics



Screen 8: Search view of a class for a specific keyword



Screen 9: Edit/Read view for editing and modifying topics



Screen 10: Blank edit/read view for a newly created topic



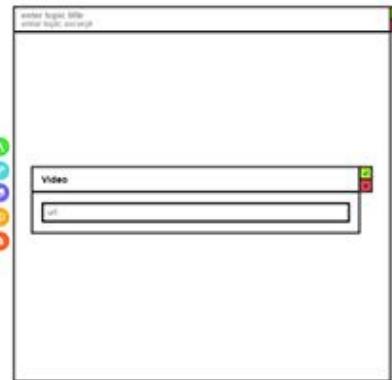
Screen 11: Rich Text Editor prompt within edit/read view



Screen 12: Link prompt within edit/read view



Screen 13: Image prompt within edit/read view



Screen 14: Video prompt within edit/read view



Screen 15: Reminder prompt within edit/read view



Screen 16: Edit/Read view populated with content

knapsack problem / dynamic programming / np-complete

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The problem often arises in resource allocation where there are financial constraints and it is called in finance as knapsack problem.

← Back

Screen 17: Discussion view for a specific topic

Knapsack Problem - Wikipedia

The knapsack problem or knapsack problem is a combinatorial optimization problem where one is given a set of items, each with a weight and a value. The objective is to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a limited capacity bag (such as a knapsack) and must select items to maximize their value.

The knapsack problem has been studied for more than a century, with many works citing as far back as 1801.¹⁷ The name "knapsack problem" dates back to the early work of mathematics Nobel laureates Carl Friedrich Gauss and Karl Drichmanns in the 1800s. Carl Friedrich Gauss's work was presented at the University of Göttingen in 1801, while Karl Drichmanns' work was presented at the University of Copenhagen in 1850.¹⁸ Both of these studies concerned themselves with financial constraints and its relation to fields such as combinatorics, number theory, complexity theory, asymptotic analysis, mathematics, and data fitting.

Contents (2/4)

- 1 Applications
- 2 Computational complexity
- 3 Related problems
- 4 Solutions
- 5 References
- 6 External links

Applications (1/1)

A 1981 study of 1,900 business managers¹⁹ showed that, out of 70 optimization problems, the knapsack problem was the 10th most popular and the 8th most needed after the Head, Audit, Risk, and Job-scheduling problems.²⁰

Knapsack problem solver A method of decision-making processes in a wide variety of fields, such as finding the least costly way to get from one place to another,²¹ solving the knapsack problem, solving the traveling salesman problem,²² and generating keys for the Data Encryption Standard.²³

One early application of knapsack algorithms was in the construction and storing of ideals in which the test elements have a chance to be which qualifies them as prime. For ideal elements I , a fully unique element α is chosen from the elements. If α is not unique, then a new element β is chosen until $\alpha \beta^{-1}$ is a unit. This is done to ensure that I is a maximum divisor of α .²⁴ In 1979, the National Bureau of Standards proposed a system in which students would receive a knapsack problem and a list of 102 possible choices. The students would be asked if one of the answers is the best of their choices. Of the possible answers of choices within their problem selected up to 100, a knapsack algorithm would determine which solution was the highest possible score.²⁵

Definition (1/1)

The most common problem setting is the 0-1 knapsack problem, which restricts the number of copies of each kind of item to one. Given a set of n items numbered from 1 to n ,

each with a weight w_i and a value v_i , along with a maximum weight capacity M ,

Screen 18: Fullscreen view of a discussion result.

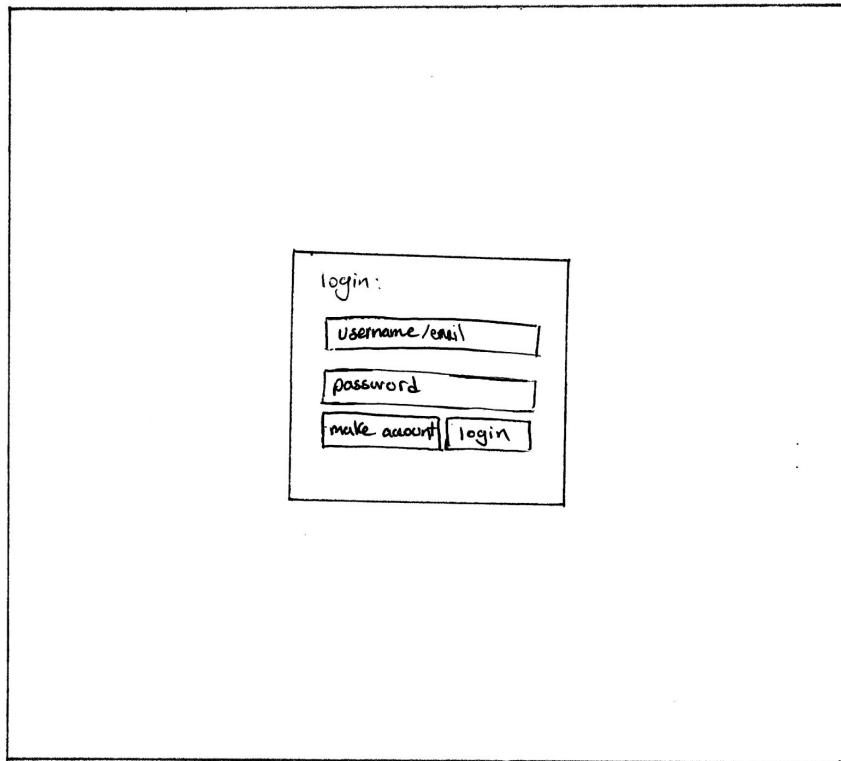
Unsubmitted Revision

The rest of this document revolves around a revision that occurred between our initial and final prototyping phase. This serves to clarify some of the transitions from initial paper prototype to final prototype that might seem ambiguous without this information.

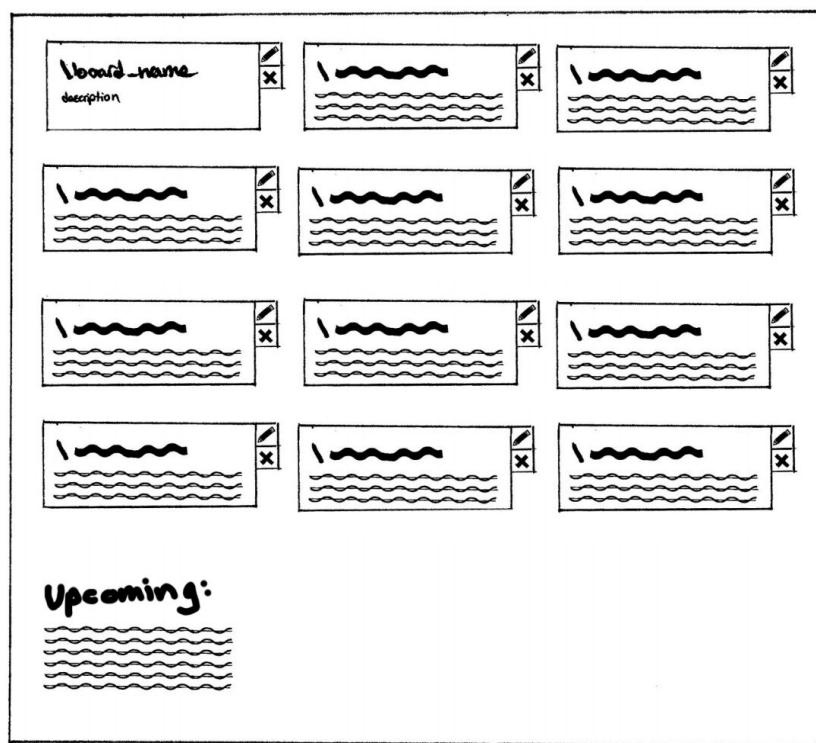
Before proceeding with our remaining usability tests, we made some modifications to our design based on Asmaa's input that have not been reflected in any of our submissions. This occurred between 3C and 3D. Major changes include:

- **Removing the “Relation View”**
 - We were informed that this relational/graph view of the nodes is not addressing any of our tasks and goes beyond the scope of this project. Thus this view was removed, and consequently the “Schedule View” became the primary mode of interaction with the nodes (topics of a board).
- **Added “Upcoming” to the Home Page**
 - There was no apparent way for the user to be informed of their upcoming reminders.
- **Removed the “Schedule” button from the boards’ tiles on the home page.**
 - Since the primary and only way of viewing the nodes was the Schedule view there was no point in having redundant buttons.
- **Removed the collaborative aspects of the boards (Clone, Fork, Collabs)**
 - This added to the complexity, and again was not addressing any of our issues. However since this was the primary way of populating the QA boards, we decided to have QA's aggregate content automagically from the internet based on the keywords of the topic.

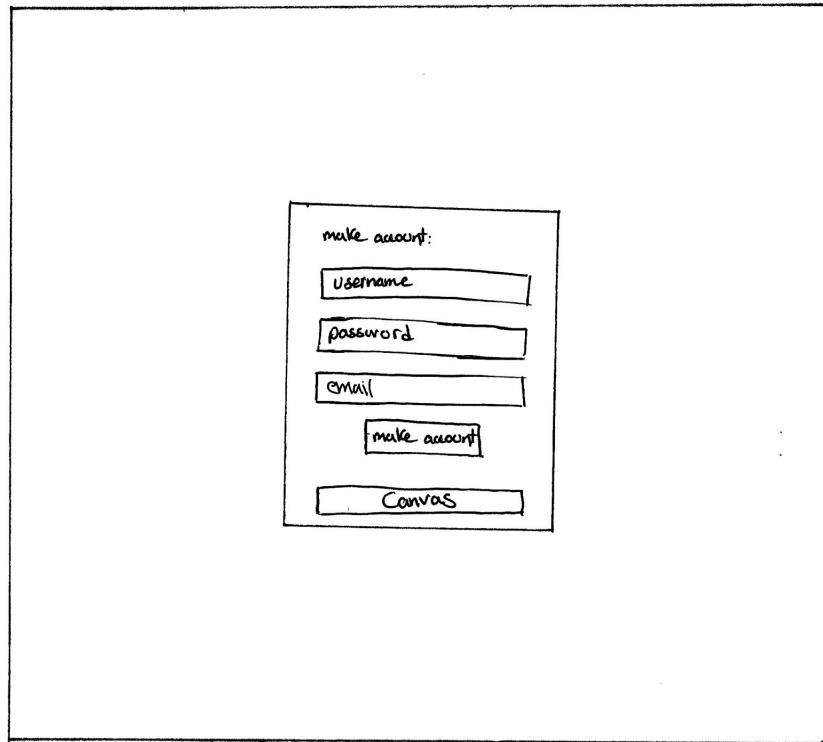
Below are some of the images of these revisions.



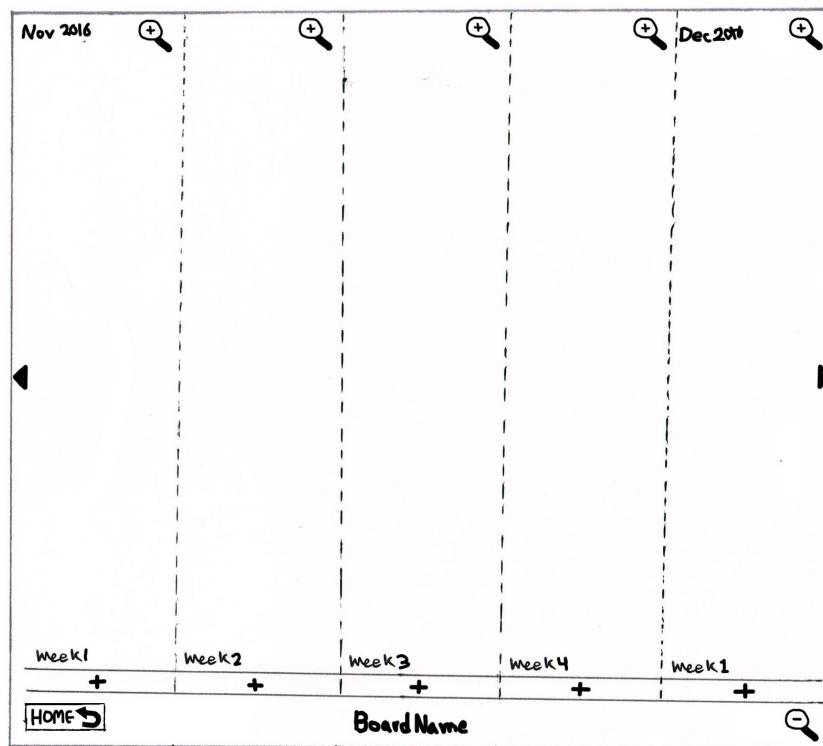
Login



Home View
Upcoming added

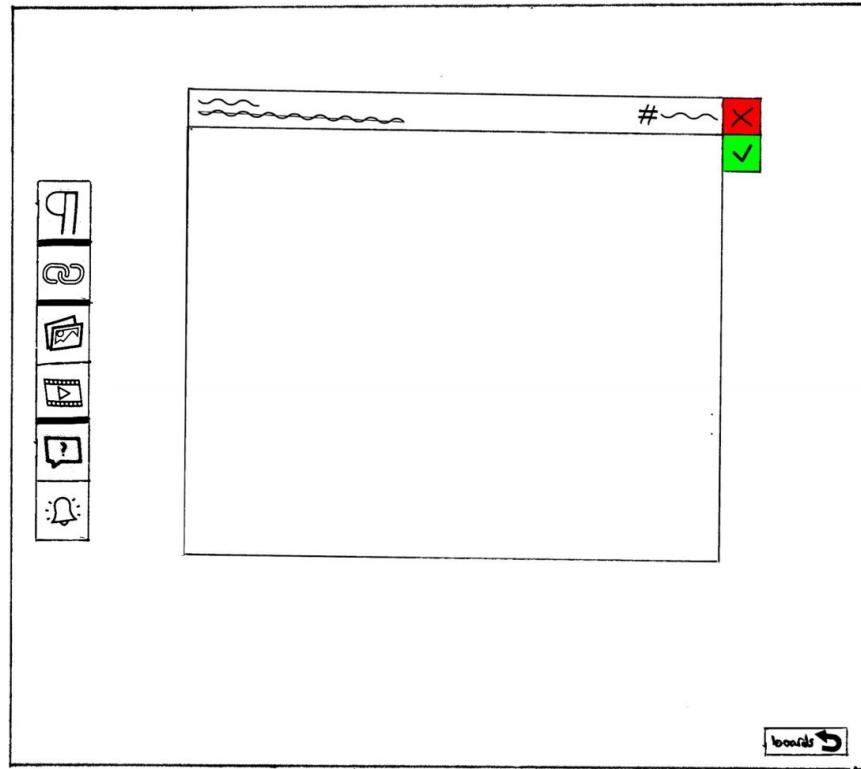


Create Account

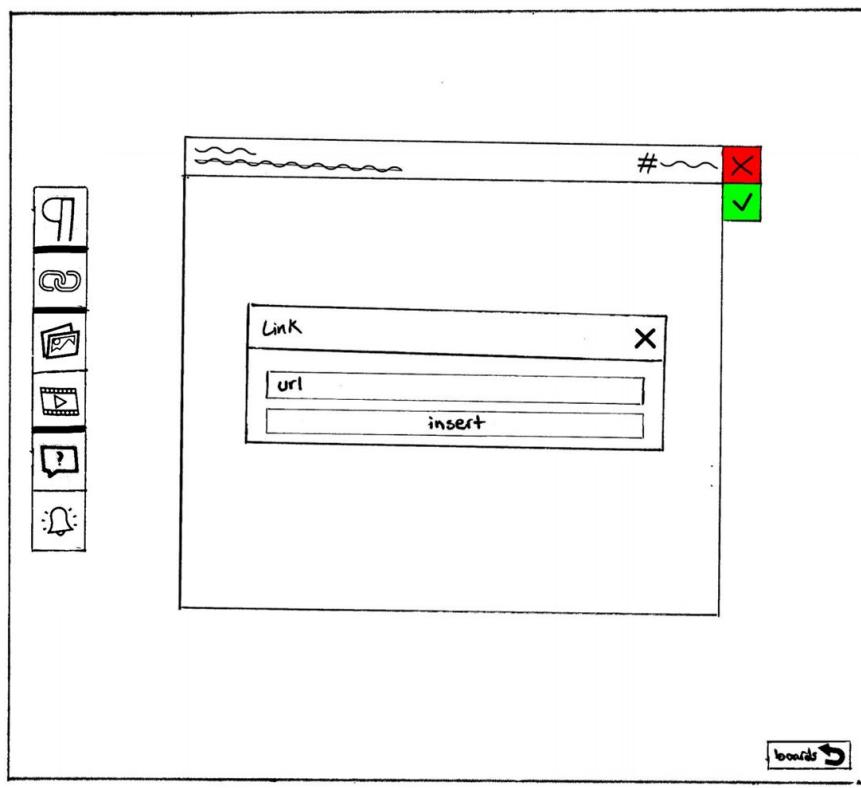


Empty Board / User clicks on + to create node

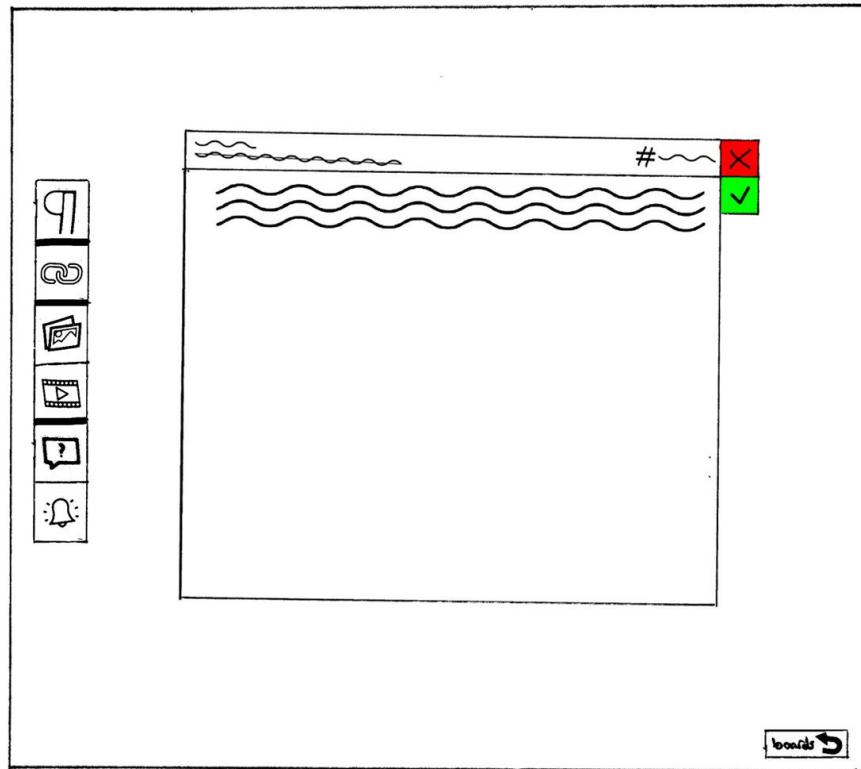
Notice the relation view does not exist anymore. It's only schedule now.



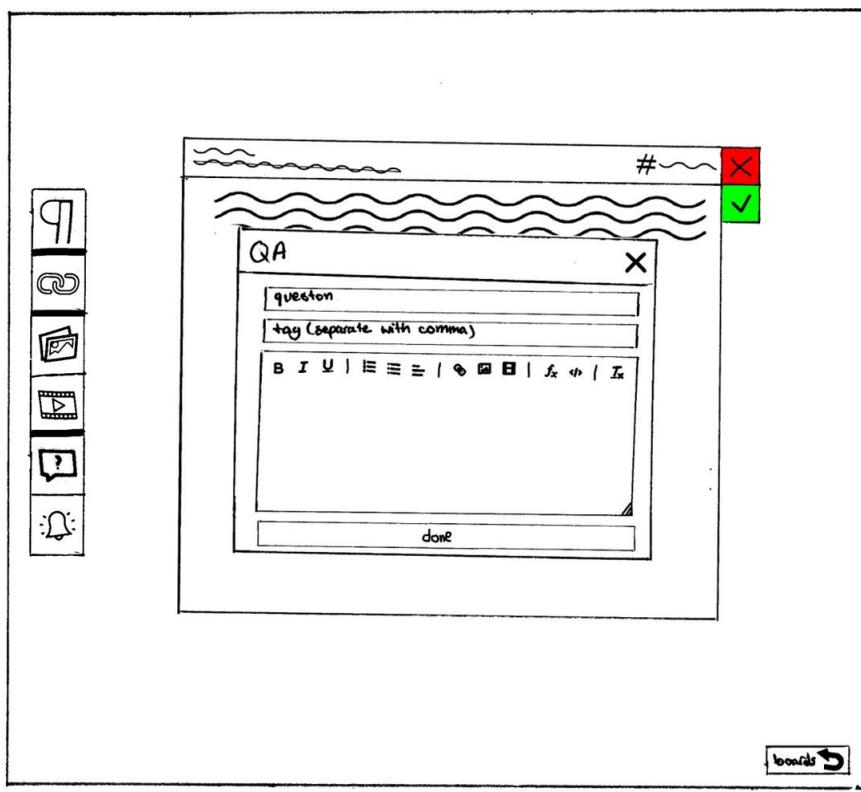
Node Editor
Notice the removal of the embed button.



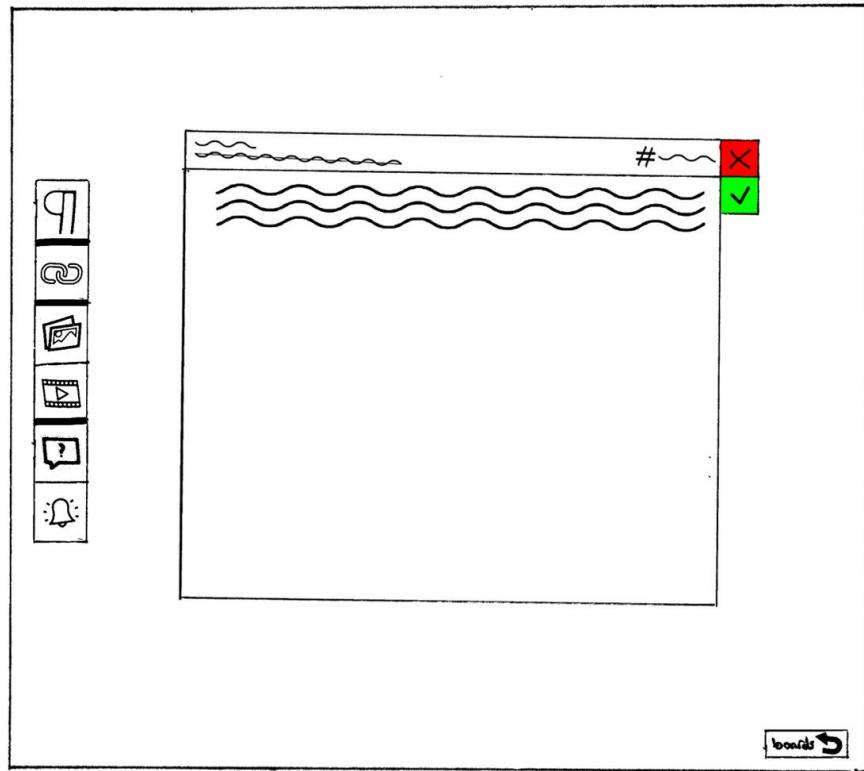
Add Link



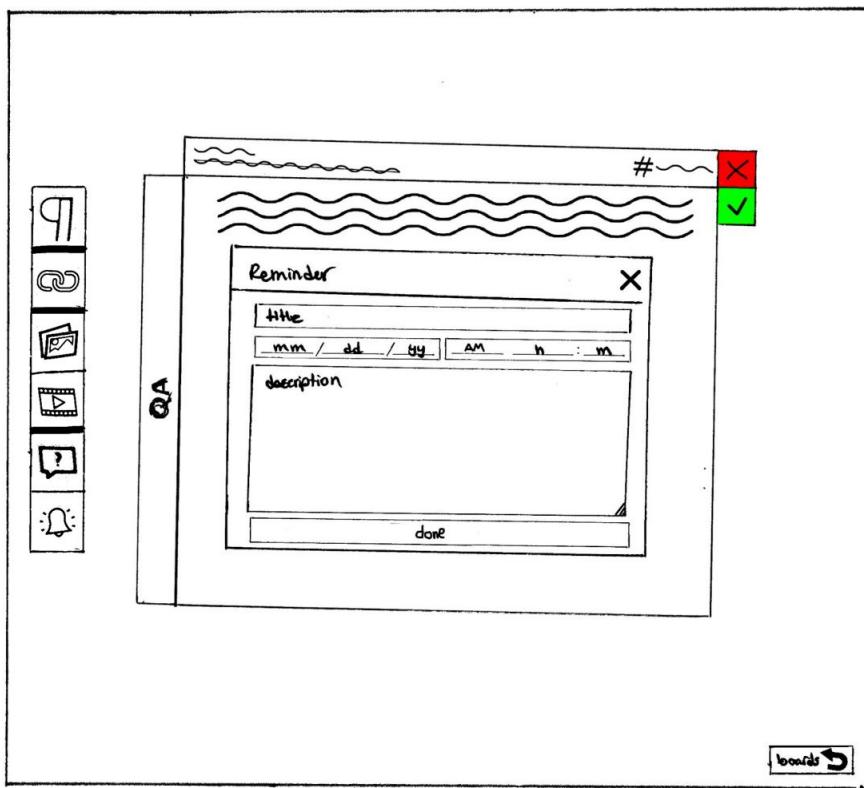
Link Added



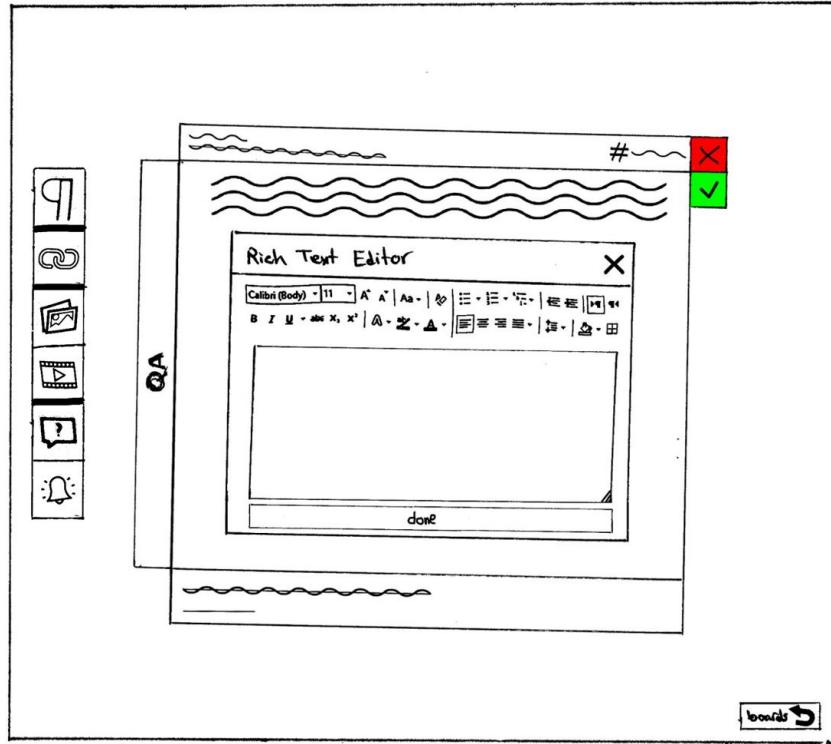
Add QA



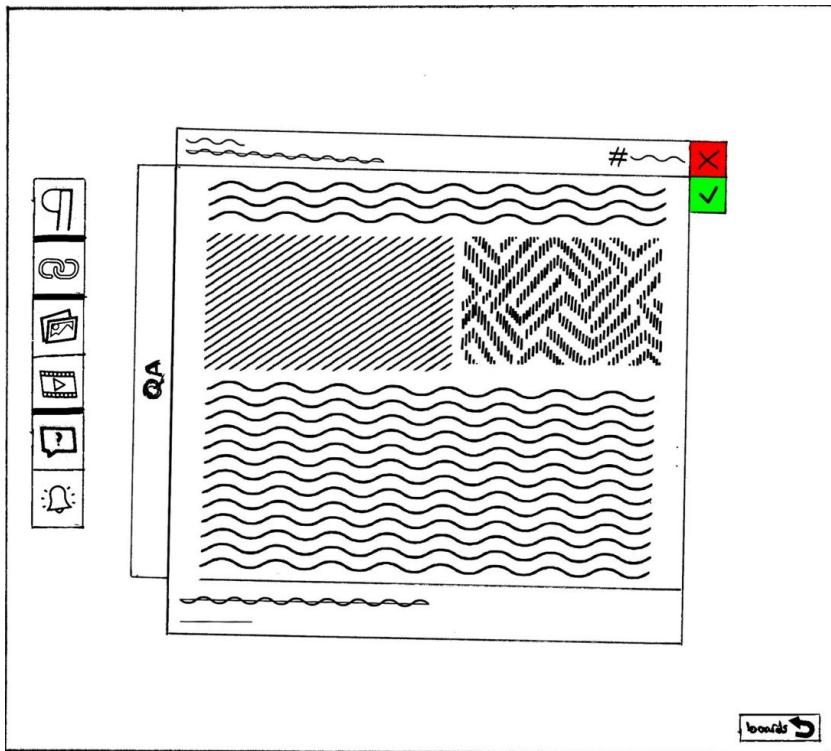
QA Added



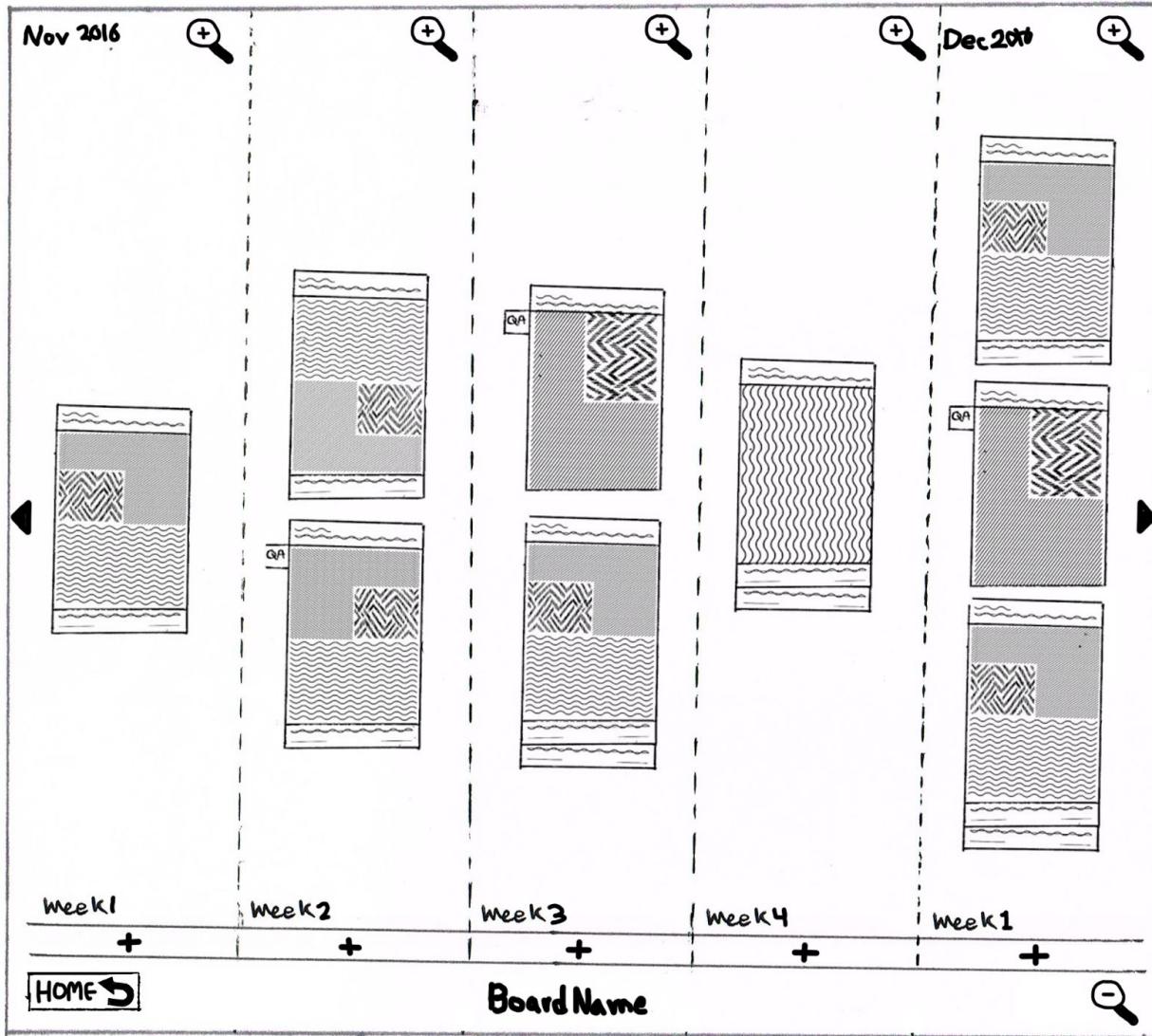
Add Reminder



Reminder Added / Add Rich Text



Multiple modules added Eg. Link, Text, Image, Video, QA, Reminder



Schedule View

Notice that the relations view is gone.

Schedule is the only view for looking at the topics of a board.

UI Elements of Schedule View:

- **Left & Right Arrows:** These guys are for horizontal scrolling
- **Magnifier Plus:** This will zoom in on the time. If it is showing weeks (example above) it will start showing days. If it is months it will switch to weeks. Note that if we are on days view, then it disappears.
 - Months → Weeks → Days (Disappears)
- **Magnifier Minus:** This is the opposite of the above. Except that it disappears on months.
 - Days → Weeks → Months (Disappears)

