

# BOARDS

SCHOOL ORGANIZED

## TABLE OF CONTENTS

### Part

- 1 **Introduction**
- 2 **Overall Problem**
- 3 **Initial Paper Prototype**
- 4 **Testing Process and Results**
- 5 **Final Paper Prototype**
- 6 **Digital Mockup**
- 7 **Summary**
- 8 **QA**



BEING A STUDENT  
IS HARD

K E E P I N G  
T R A C K

F I N D I N G

H E L P

# INITIAL PAPER PROTOTYPE

OR HOW TO FAIL MISERABLY

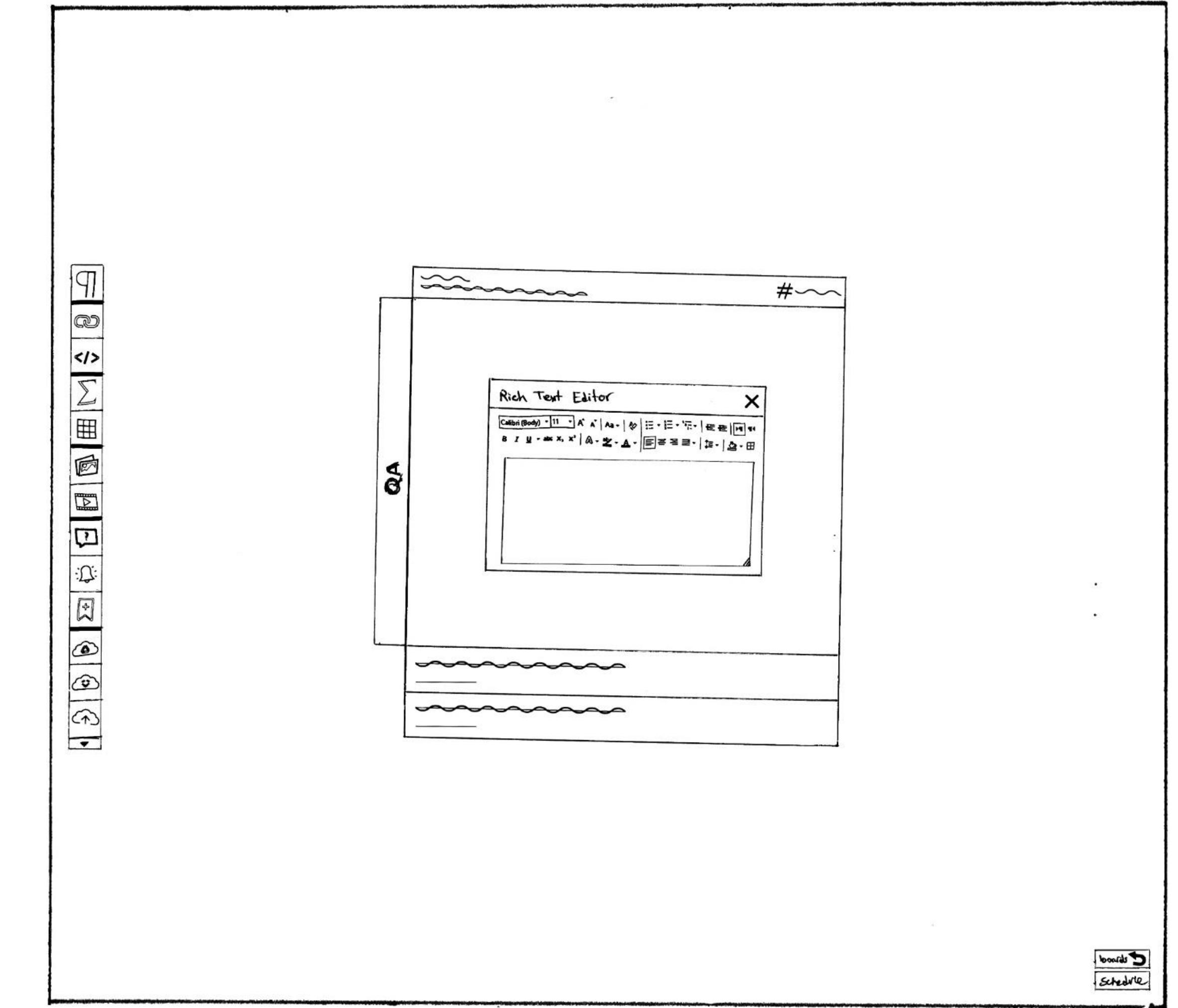
# M I S S I N G

UI UX  
PROMPTS  
FUNCTIONALITY  
CONFIRMATIONS

## A L O T

# OVERLY COMPLICATED

FAILED TO ADDRESS TASKS  
UNORGANIZED  
AMBITIOUS  
FORGETFUL



A D D

— ONE —

BY —

— ONE —

F E A T U R E S

# TESTING PROCESS PARTICIPANTS

## LEO

Leo, a tech savvy CS student was the first test to be done with our prototype.

## ALEX

A student who needs help with optimizing his current study habits. He is a senior computer science student with 15 credit hours

## CALIN

Communication student. This test was performed in the Marriott Library, due to it being a popular space that students go to study and finish assignments.

# TESTING PROCESS

# TASKS

**USING BOARDS TO**

CREATE A CLASS

ADD AN ALERT

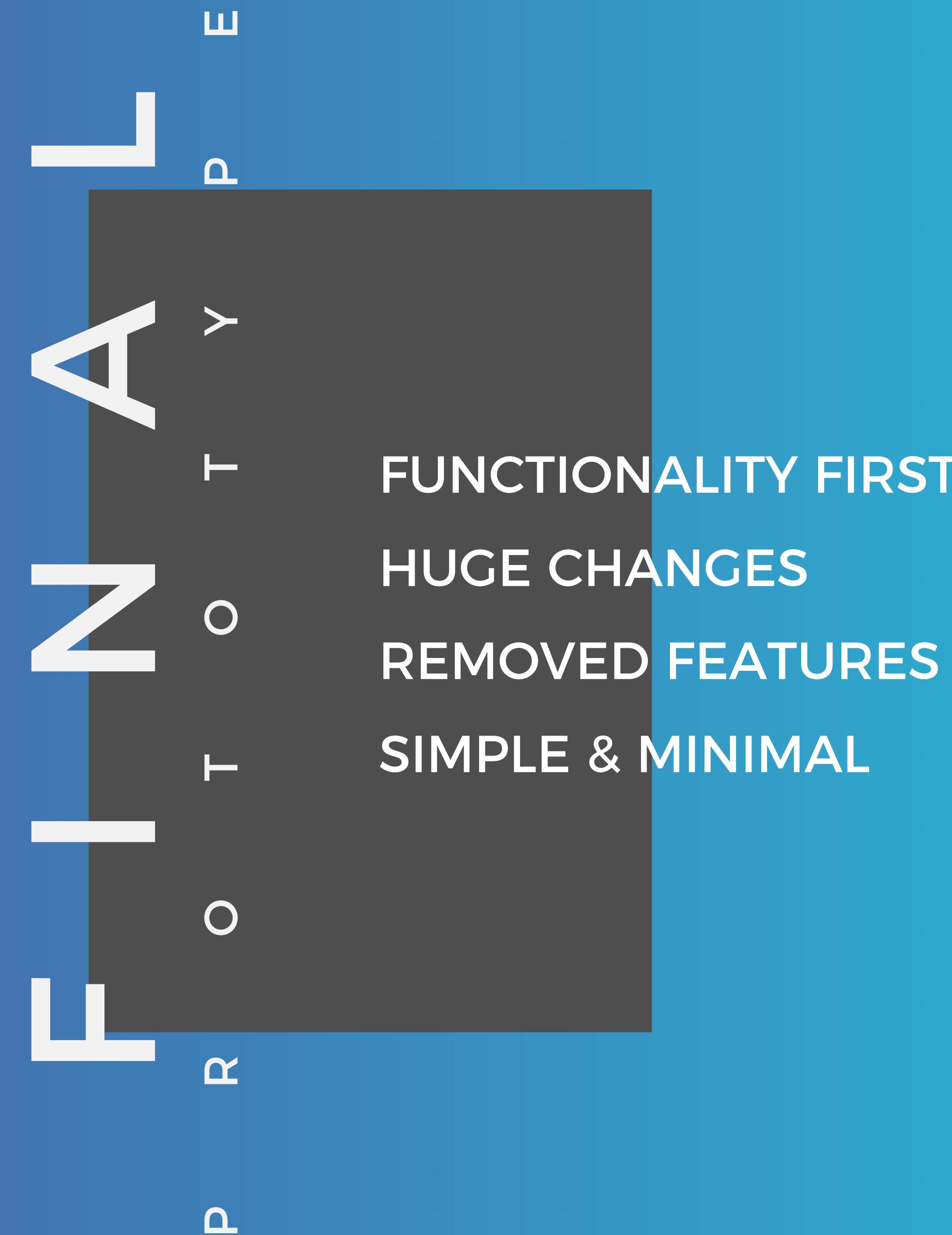
FIND DUE DATE

**USING CLASSES TO**

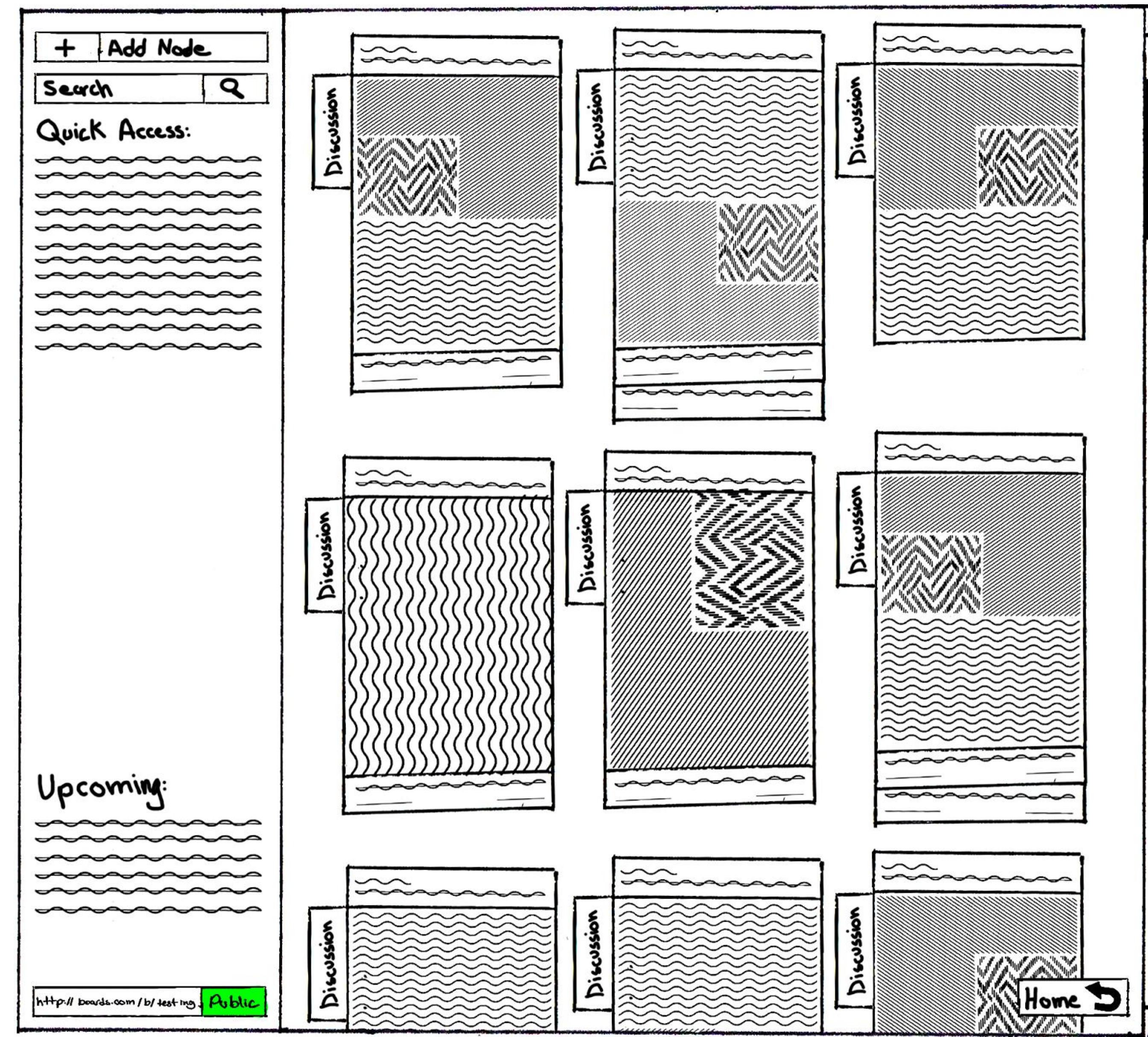
FIND HELP ON TOPIC

ADD A REMINDER

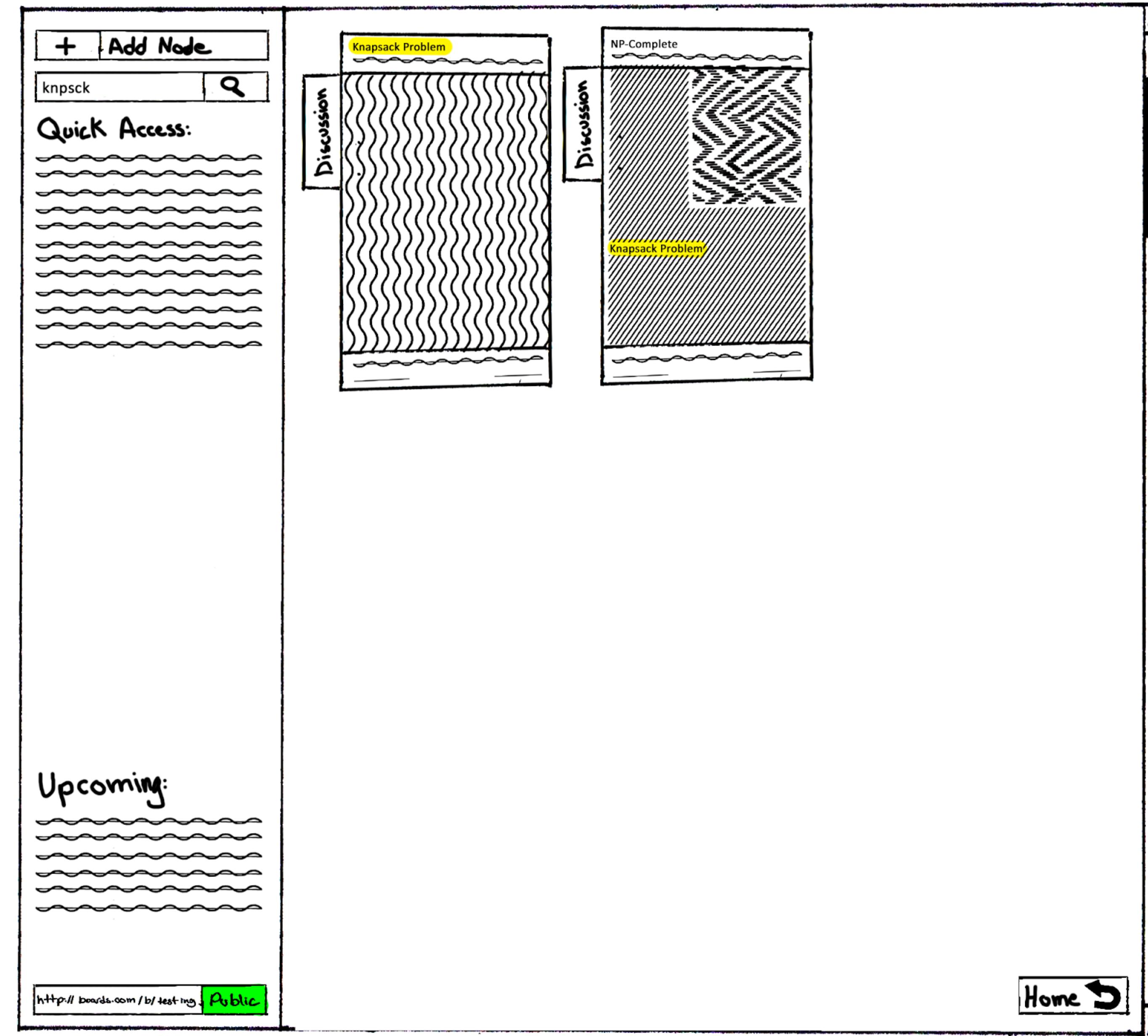
CHANGE DUE DATE



# TASK 1: GETTING HELP



# TASK 1: GETTING HELP



## knapsack problem / np-complete / big-o complexity



### Wikipedia:

#### Knapsack Problem

The **knapsack problem** or rucksack 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.

#### Continuous Knapsack Problem

In theoretical computer science, the continuous knapsack problem (also known as the fractional knapsack problem) is an algorithmic problem in combinatorial optimization in which the goal is to fill a container (the "knapsack") with fractional amounts of different materials chosen to maximize the value of the selected materials.[1][2] It resembles the classic knapsack problem, in which the items to be placed in the container are indivisible; however, the continuous knapsack problem may be solved in polynomial time whereas the classic knapsack problem is NP-hard.[1] It is a classic example of how a seemingly small change in the formulation of a problem can have a large impact on its computational complexity.

#### NP-Complete:

In computational complexity theory, a decision problem is **NP-complete** when it is both in **NP** and **NP-hard**. The set of **NP-complete** problems is often denoted by **NP-C** or **NPC**. The abbreviation **NP** refers to "nondeterministic polynomial time". Although any given solution to an **NP-complete** problem can be verified quickly (in polynomial time), there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of **NP-complete** problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows. As a consequence, determining whether or not it is possible to solve these problems quickly, called the **P** versus **NP** problem, is one of the principal unsolved problems ...

### Stackoverflow:

#### Python - Greedy Knapsack with Dictionary input

I am trying to implement a greedy **knapsack** algorithm in python given the data 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 value encoding scheme rather than a binary representation. The crossover function is as follows:

```
def cxSet(ind1, ind2):
    """Apply 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.
    """
    ...
```

#### 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 these items in a knapsack of capacity W to get the maximum total value in the 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 subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the **complete** item, or don't pick it (0-1 property).

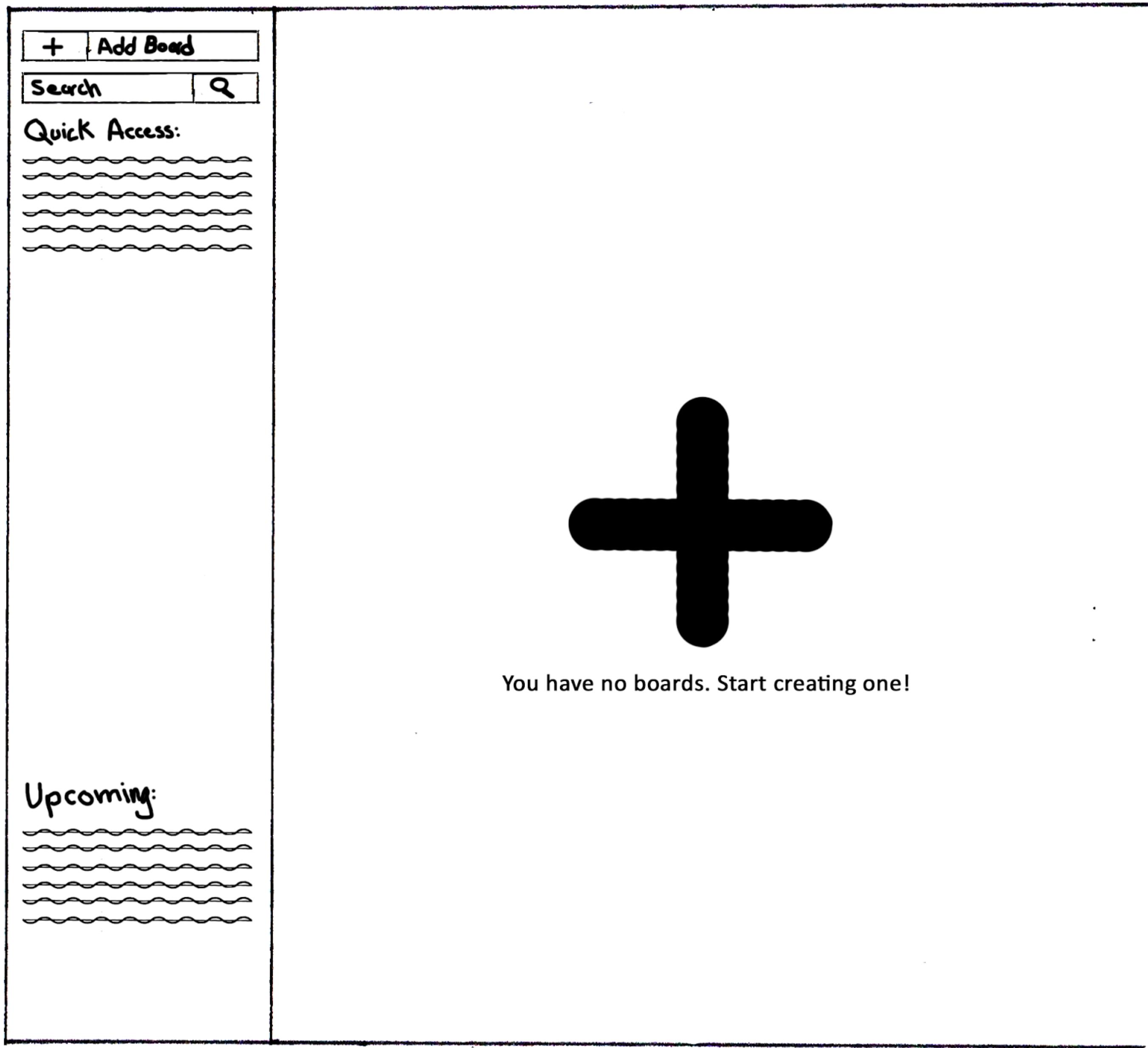
#### Lecture 13: The Knapsack Problem

Informal Description:  
We have computed  $\otimes$  data files that we want to store, and we have available bytes of storage.  
File A has size  $\otimes\otimes$  bytes and takes  $\otimes\otimes$  minutes to recompute.  
We want to avoid as much recomputing as possible, so we want to find a subset of files to store such that  
The files have combined size at most  $\otimes$   
The total computing time of the stored files is as large as possible.  
We can not store parts of files, it is the whole file or nothing.

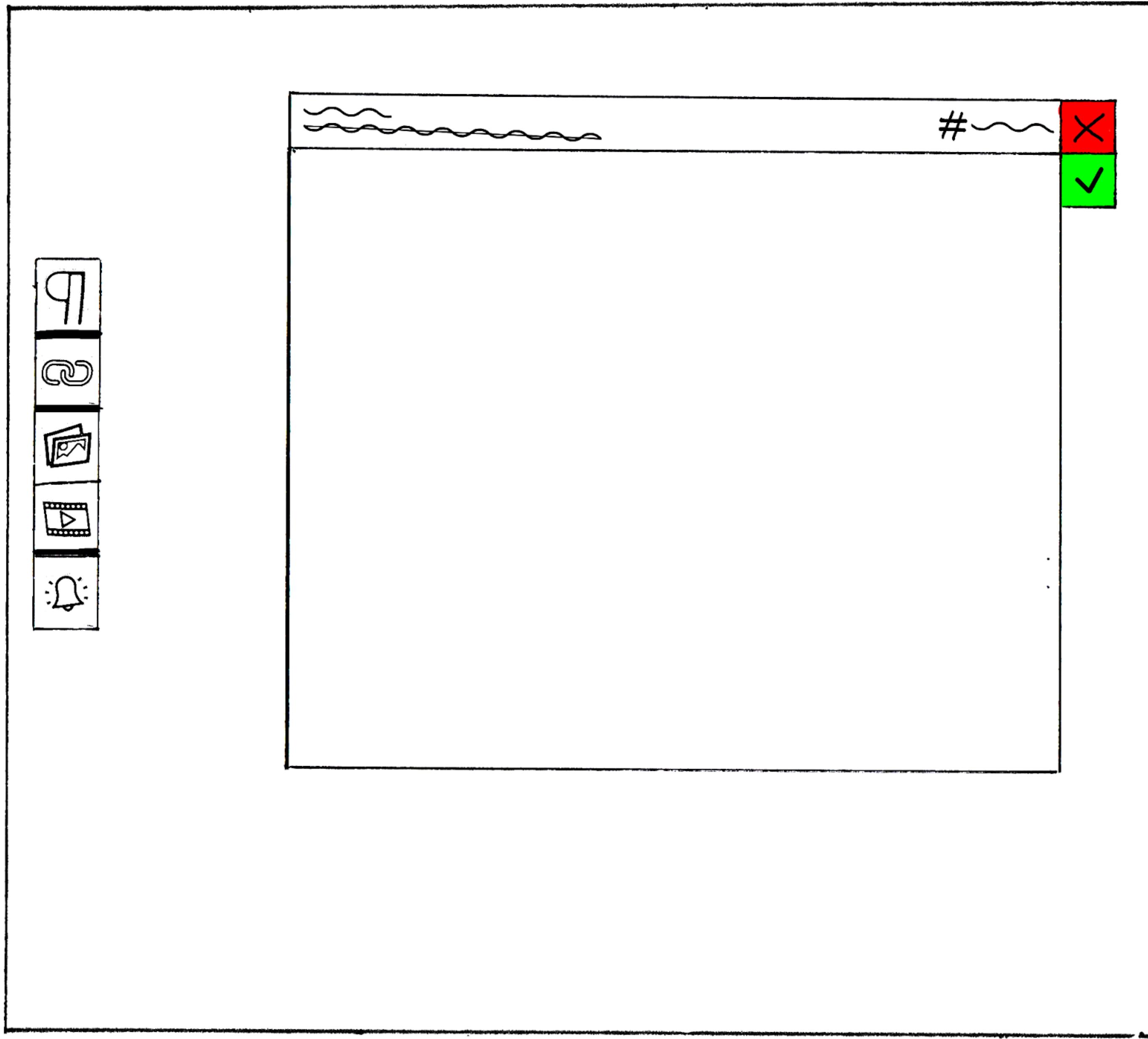
#### The Knapsack Problem

Suppose we are planning a hiking trip; and we are, therefore, interested in filling a **knapsack** with items that are considered necessary for the trip. There are N different item types that are deemed desirable; these could include bottle of water, apple, orange, sandwich, and so forth. Each item type has a given set of two attributes, namely a weight (or volume) and a value that quantifies the level of importance associated with each unit of that type of item. Since the knapsack has a limited weight (or volume) capacity, the problem of interest is to figure out how to load the knapsack with a combination of units of the specified types of items that yields the greatest total value. What we have just described is called the knapsack problem.

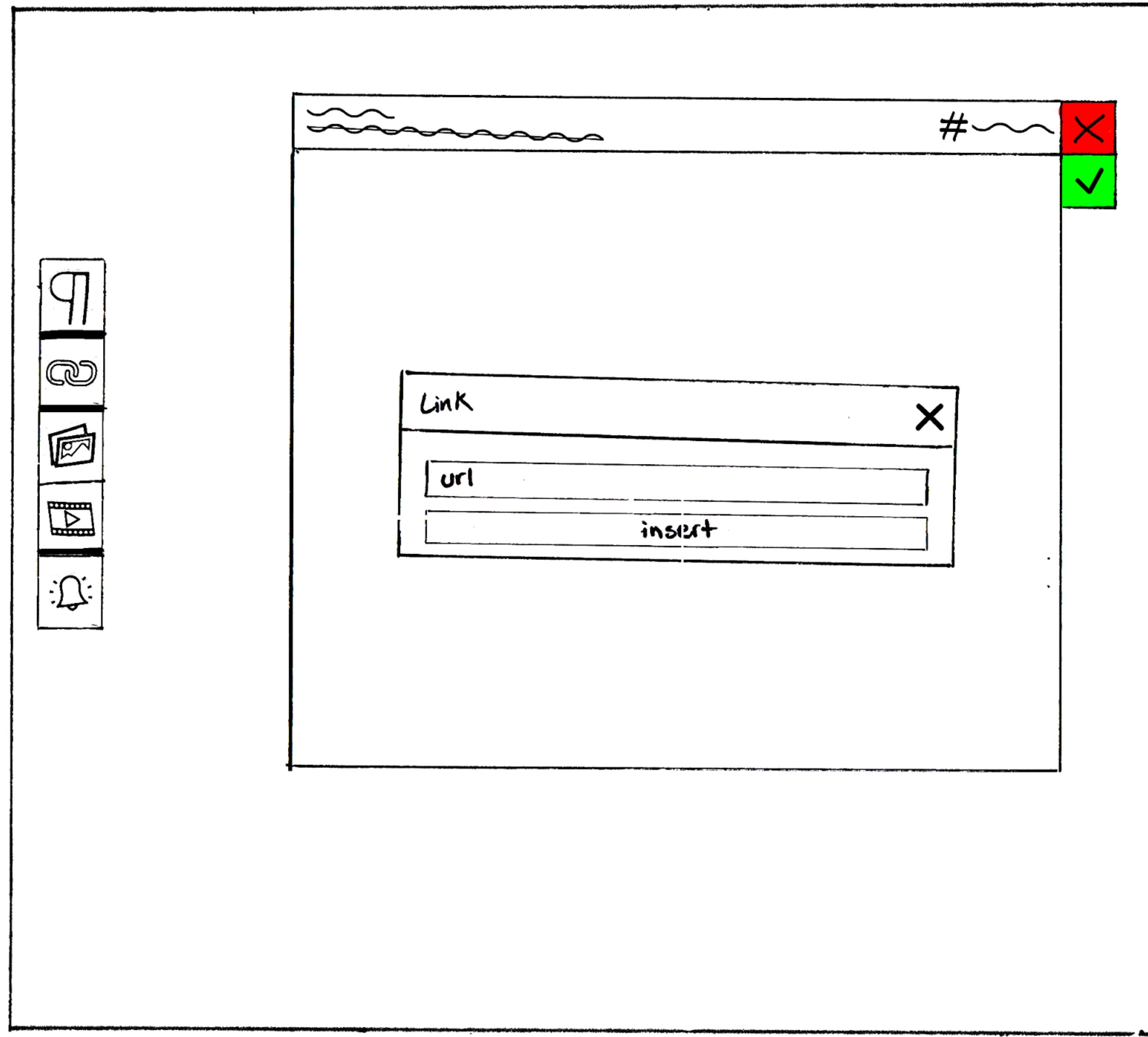
# TASK 2: ORGANIZE COURSE MATERIAL



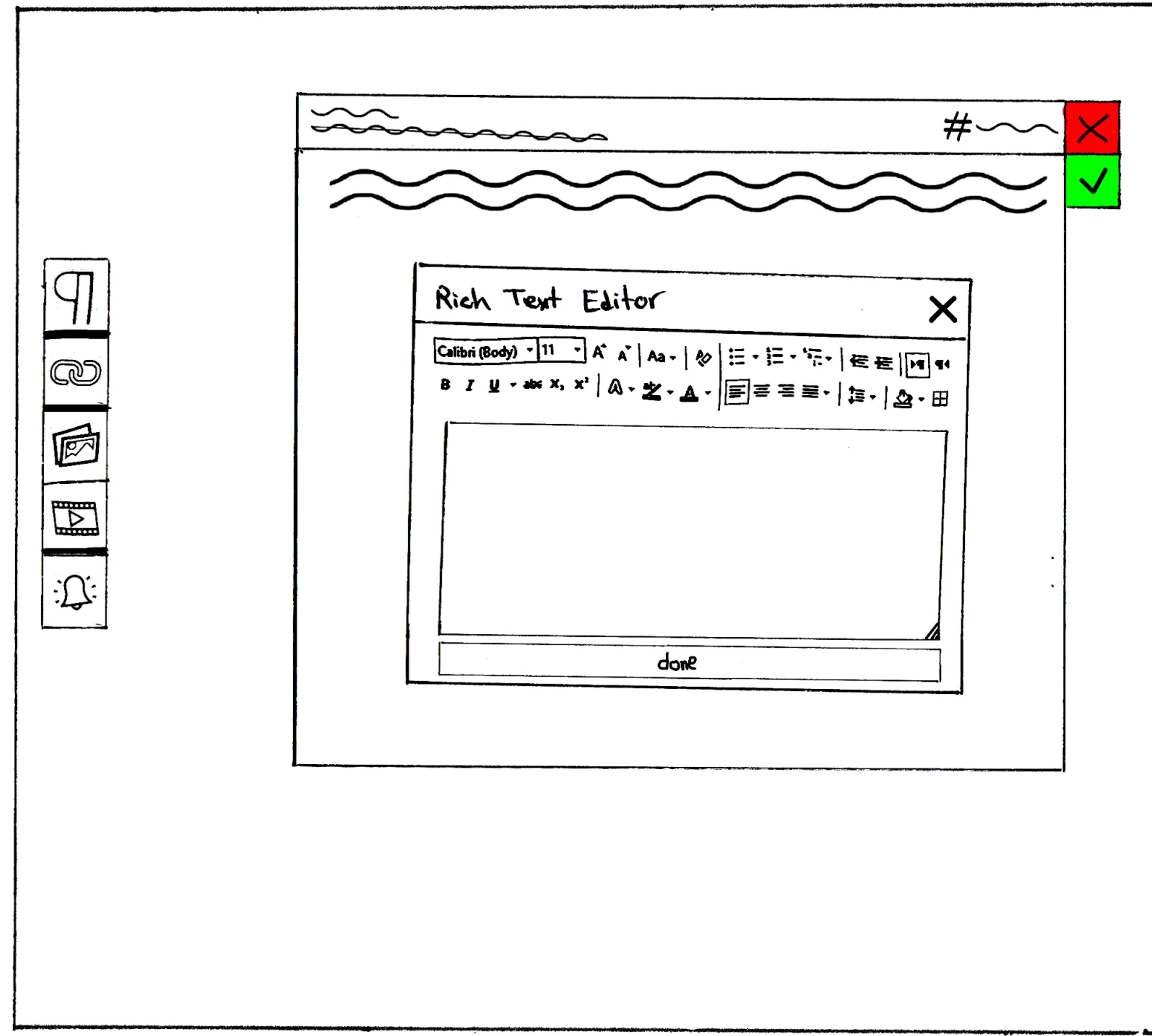
# TASK 2: ORGANIZE COURSE MATERIAL



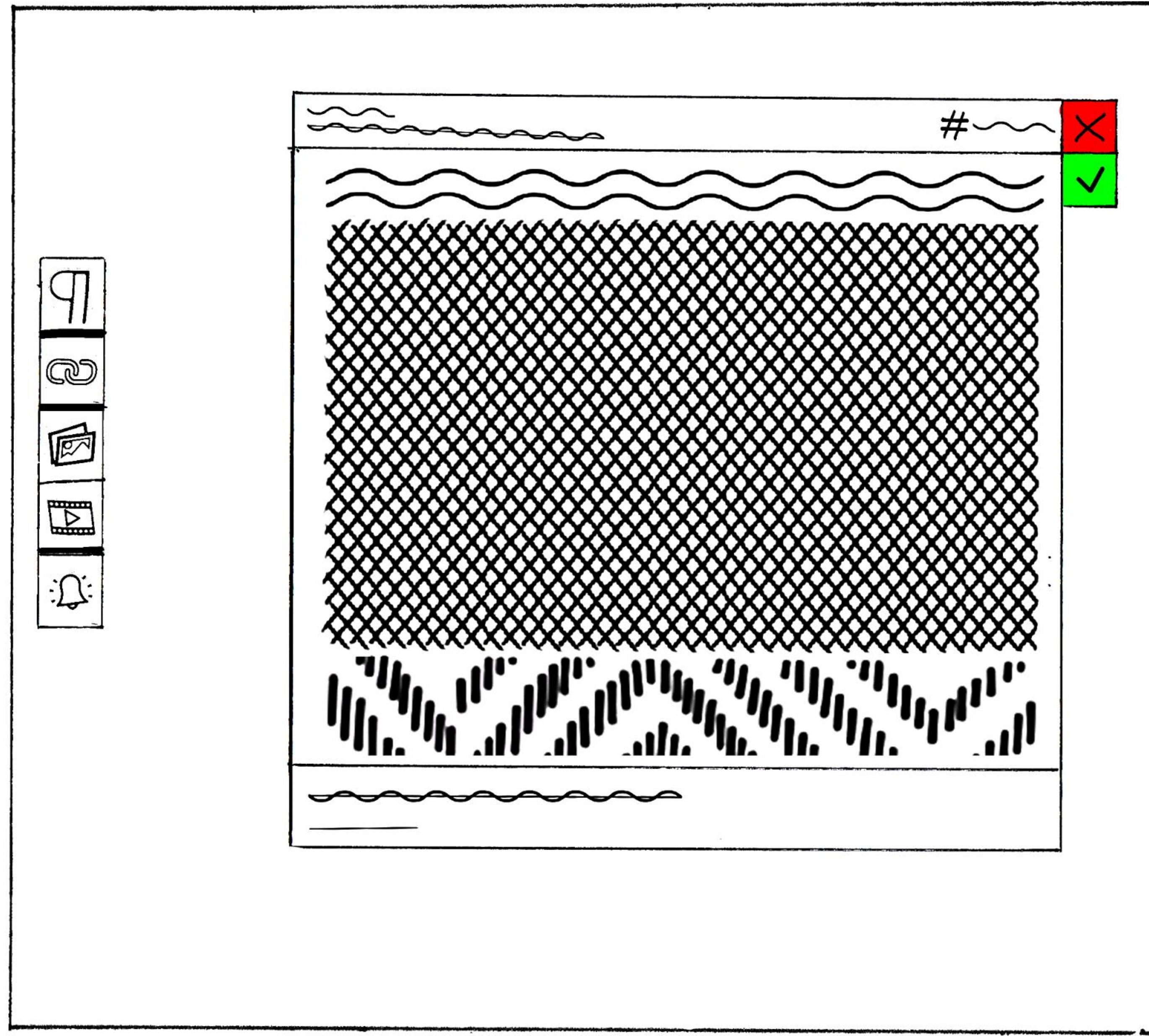
# TASK 2: ORGANIZE COURSE MATERIAL



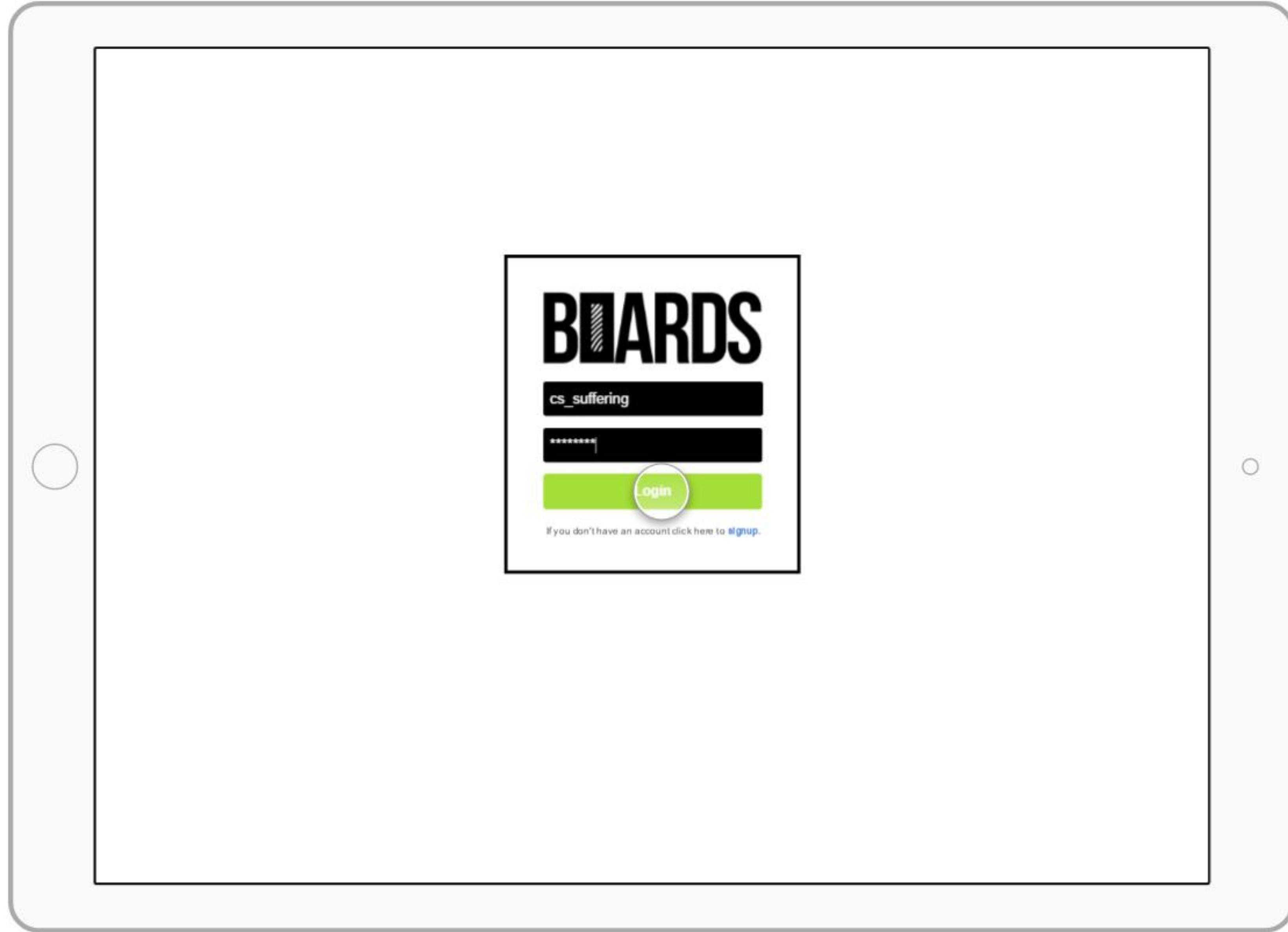
# TASK 2: ORGANIZE COURSE MATERIAL



# TASK 2: ORGANIZE COURSE MATERIAL



**LETS GET DIGITAL**



**TASK 1: GETTING HELP**

**Add Board**

Search

Access your frequently used boards quickly

- Calculus III 2 Reminders
- CS 4150 2 Reminders
- CS 3500 1 Reminder
- CS 3810 2 Reminders

Here are the next upcoming events based on your reminders.

- HW 3 CS 3810 11/19/16
- Project Draft Calculus III 11/21/16
- PS9-2 CS 4150 11/24/16
- Reading 6 CS 3500 11/24/16
- PS9-3 CS 4150 11/26/16

**Calculus I**  
Functions and their graphs, differentiation of polynomial, rational and trigonometric functions. Velocity...

**Calculus II**  
Geometric applications of the integral, logarithmic, and exponential functions, techniques of integration...

**Calculus III**  
Vectors in the plane and in 3-space, differential calculus in several variables, integration and its applications...

**CS 1410**  
Introduction to the engineering and mathematical skills required to effectively program computers

**CS 2420**  
This course provides an introduction to the problem of engineering computational efficiency into programs.

**CS 2100**  
Introduction to propositional logic, predicate logic, formal logical arguments, finite sets, functions, relations...

**CS 3100**  
This course covers different models of computation and how they relate to the understanding and better...

**CS 3200**  
Scientific computation relevant to computational science and engineering, with emphasis on the process of modeling...

**CS 3500**  
Practical exposure to the process of creating large software systems, including requirements specifications, design...

**CS 3505**  
An in-depth study of traditional software development (using UML) from inception through implementation.

**CS 4150**  
Study of algorithms, data structures, and complexity analysis beyond the introductory treatment from CS 2420.

**CS 4400**  
Introduction to computer systems from a programmer's point of view. Machine level representations of programs...

**CS 3810**  
An in-depth study of computer architecture and design, including topics such as RISC and CISC instruction set architectures...

**CS 5100**  
A survey of topics in theoretical computer science, focusing on computability and complexity. Turing machine ...

**CS 5470**  
Lexical analysis, top-down and bottom-up parsing, symbol tables, internal forms and intermediate languages...

## TASK 1: GETTING HELP

**CS 4150**

**Add Topic**

**Search**

Access your frequently used boards quickly.

Please fill out this field.

**Knapsack Problem**  
Dynamic programming and memoization

Discussions

**Dynamic Programming 1**  
Dynamic programming and memoization

Discussions

**Dynamic Programming 2**  
Dynamic programming and memoization

Discussions

**Test Title**  
Dynamic programming and memoization

Discussions

**Another Test Title**  
Dynamic programming and memoization

Discussions

**Yet Another**  
Dynamic programming and memoization

Discussions

Here are the next upcoming events based on your reminders.

**PS9-1** 11/19/16 Knapsack Problem

**PS9-2** 11/19/16 Knapsack Problem

**PS9-3** 11/19/16 Dynamic Prog 1

**PS9-4** 11/19/16 Dynamic Prog 1

**PS9-5** 11/19/16 Dynamic Prog 2

<http://boards.com/joef...> Private

**We Meet Again**  
Dynamic programming and memoization

Discussions

**Why not?**  
Dynamic programming and memoization

Discussions

**Yes!**  
Dynamic programming and memoization

Discussions

Home

# TASK 1: GETTING HELP

CS 4150

Add Topic

knapsack

Access your frequently used boards quickly

Knapsack Prob 0 reminders

Dynamic Prog 1 0 reminders

Dynamic Prog 2 1 reminder

Mod Arithmetic 0 reminders

Discussions

Knapsack Problem Dynamic programming and memoization

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. FUSCE CONVallis PELLentesque METUS ID LACINIa. Nunc dapibus pulvinar auctor. Quis nec sem at orci commodo viverra id in ipsum. Fusce tellus nisl, vestibulum sed rhoncus at, pretium non libero. Cras vel lacus ut ipsum vehicula aliquam at quis urna. Nunc ac ornare ante...

Link 1 to something  
Link 2 to something

PS9-1 11/19/16

PS9-2 11/19/16

Here are the next upcoming events based on your reminders.

PS9-1 Knapsack Problem 11/19/16

PS9-2 Knapsack Problem 11/19/16

PS9-3 Dynamic Prog 1 11/19/16

PS9-4 Dynamic Prog 1 11/19/16

PS9-5 Dynamic Prog 2 11/19/16

<http://boards.com/joe/> Private

Home

## TASK 1: GETTING HELP

CS 4150

Add Topic

Search: knapsack

- Access your frequently used boards quickly
- Knapsack Prob (2 Reminders)
- Dynamic Prog 1 (2 Reminders)
- Dynamic Prog 2 (1 Reminder)
- Mod Arithmetic (None)

Discussion (1)

Knapsack Problem  
Dynamic programming and memoization

Discussion (1)

PS9-1 11/19/16

PS9-2 11/19/16

Here are the next upcoming events based on your reminders.

PS9-1 11/19/16  
Knapsack Problem

PS9-2 11/19/16  
Knapsack Problem

PS9-3 11/19/16  
Dynamic Prog 1

PS9-4 11/19/16  
Dynamic Prog 1

PS9-5 11/19/16  
Dynamic Prog 2

<http://boards.com/joel...> Private

Home

## TASK 1: GETTING HELP

knapsack problem / dynamic programming / np-complete

Search terms are automatically detected based on selected topic. You can use the search bar to perform a generic search on google.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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 dynamic programming and NP-completeness.

**Back**

## TASK 1: GETTING HELP

knapsack problem / dynamic programming / np-complete

Search terms are automatically detected based on selected topic. You can use the search bar to perform a generic search on google.

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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

Back

## TASK 1: GETTING HELP

**Knapsack Problem - Wikipedia**

The knapsack problem or rucksack 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](#).

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897.<sup>[1]</sup> The name "knapsack problem" dates back to the early works of mathematician [Tobias Dantzig](#) (1884–1956),<sup>[2]</sup> and refers to the commonplace problem of packing your most valuable or useful items without overloading your luggage.

**Contents** [hide]

- 1 Applications
- 2 Definition
- 3 Computational complexity
- 4 Solving
  - 4.1 Dynamic programming in-advance algorithm
    - 4.1.1 Unbounded knapsack problem
    - 4.1.2 0/1 knapsack problem
  - 4.2 Meet-in-the-middle
  - 4.3 Approximation algorithms
    - 4.3.1 Greedy approximation algorithm
    - 4.3.2 Fully polynomial time approximation scheme
  - 4.4 Dominance relations
- 5 Variations
  - 5.1 Multi-objective knapsack problem
  - 5.2 Multi-dimensional knapsack problem
  - 5.3 Multiple knapsack problem
  - 5.4 Quadratic knapsack problem
  - 5.5 Subset-sum problem
- 6 Software
- 7 Popular culture
- 8 See also
- 9 Notes
- 10 References
- 11 External links

## Applications [edit]

A 1998 study of the [Stanford University Algorithm Repository](#) showed that, out of 75 algorithmic problems, the knapsack problem was the 18th most popular and the 4th most needed after [kd-trees](#), [suffix trees](#), and the [bin packing problem](#).<sup>[3]</sup>

Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials,<sup>[4]</sup> selection of investments and portfolios,<sup>[5]</sup> selection of assets for asset-backed securitization,<sup>[6]</sup> and generating keys for the [Merkle–Hellman](#)<sup>[7]</sup> and other knapsack cryptosystems.

One early application of knapsack algorithms was in the construction and scoring of tests in which the test-takers have a choice as to which questions they answer. For small examples it is a fairly simple process to provide the test-takers with such a choice. For example, if an exam contains 12 questions each worth 10 points, the test-taker need only answer 10 questions to achieve a maximum possible score of 100 points. However, on tests with a heterogeneous distribution of point values—i.e. different questions are worth different point values—it is more difficult to provide choices. Feuerman and Weiss proposed a system in which students are given a heterogeneous test with a total of 125 possible points. The students are asked to answer all of the questions to the best of their abilities. Of the possible subsets of problems whose total point values add up to 100, a knapsack algorithm would determine which subset gives each student the highest possible score.<sup>[8]</sup>

## Definition [edit]

The most common problem being solved is the 0-1 knapsack problem, which restricts the number  $x_i$  of copies of each kind of item to zero or one. Given a set of  $n$  items numbered from 1 up to  $n$ , each with a weight  $w_i$  and a value  $v_i$ , along with a maximum weight capacity  $W$ ,

# TASK 1: GETTING HELP

CS 4150

Add Topic

knapsack

Access your frequently used boards quickly

Knapsack Prob 2 reminders

Dynamic Prog 1 0 reminders

Dynamic Prog 2 0 reminders

Mod Arithmetic 0 reminders

Discussions

Knapsack Problem Dynamic programming and memoization

Knapsack Problem

Dynamic programming and memoization

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. FUSCE CONVallis PELLentesque METUS ID LACINIa. Nunc dapibus pulvinar auctor. Duis nec sem at orci commodo viverra id in ipsum. Fusce tellus nisl, vestibulum sed rhoncus at, preifum non libero. Cras vel lacus ut ipsum vehicula aliquam at quis urna. Nunc ac ornare ante....

Link 1 to something  
Link 2 to something

PS9-1 11/19/16

PS9-2 11/19/16

Here are the next upcoming events based on your reminders.

PS9-1 Knapsack Problem 11/19/16

PS9-2 Knapsack Problem 11/19/16

PS9-3 Dynamic Prog 1 11/19/16

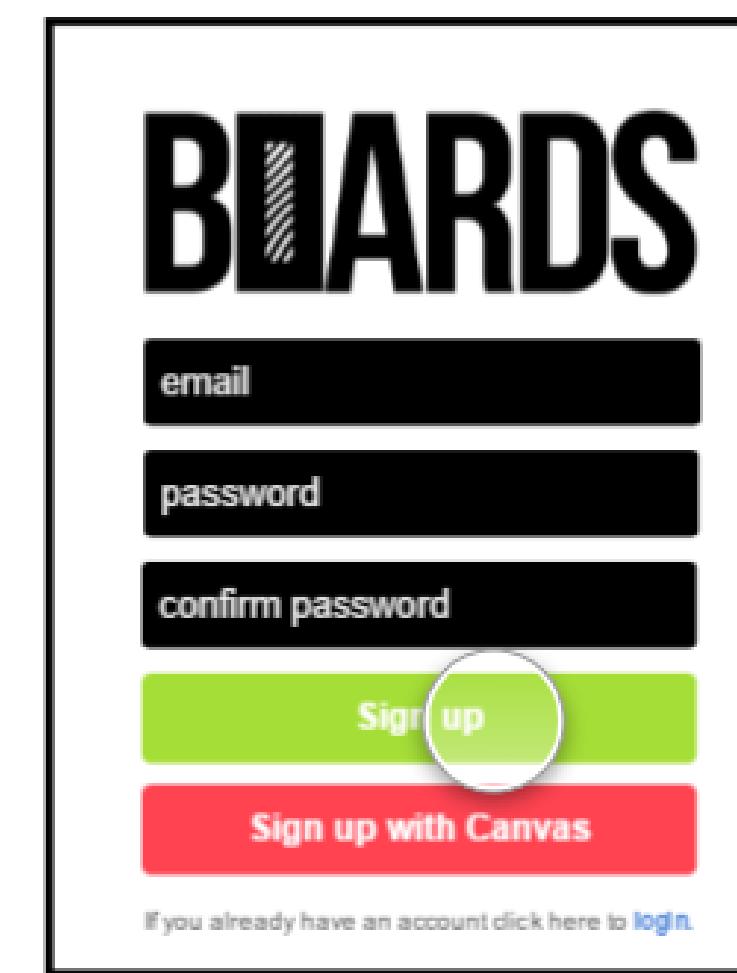
PS9-4 Dynamic Prog 1 11/19/16

PS9-5 Dynamic Prog 2 11/19/16

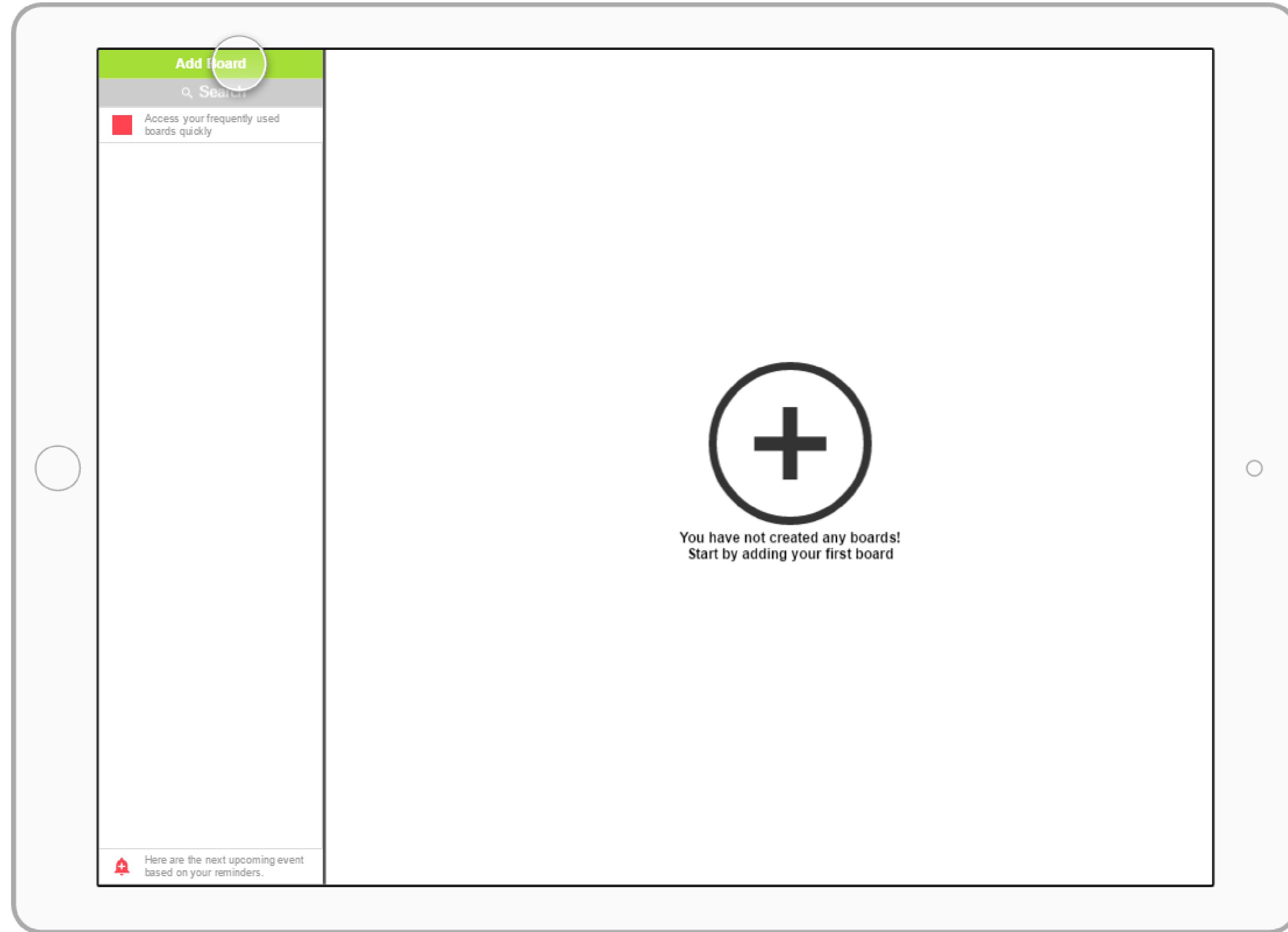
<http://boards.com/joel...> Private

Home

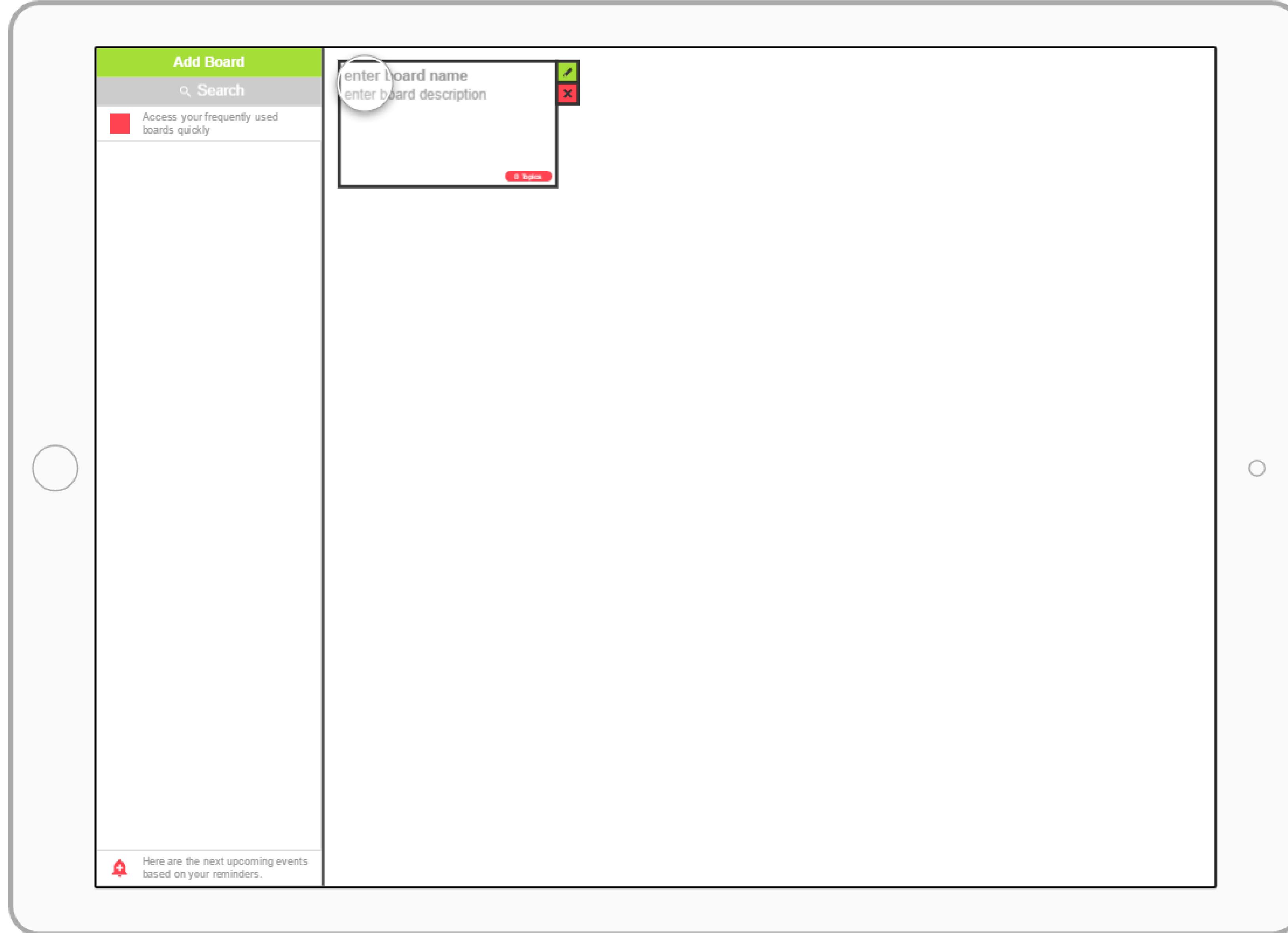
## TASK 1: GETTING HELP



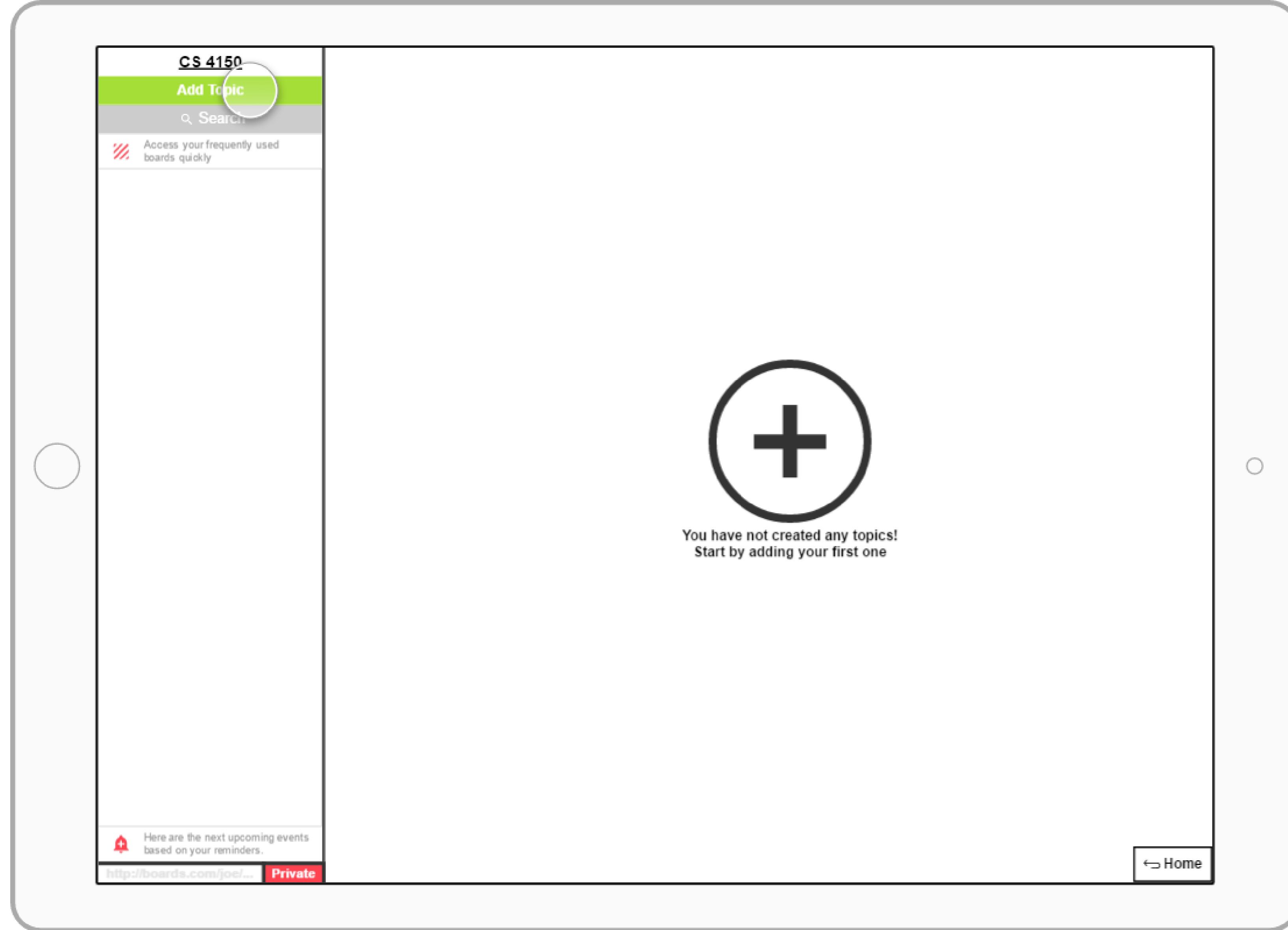
**TASK 2: ORGANIZE COURSE MATERIAL**



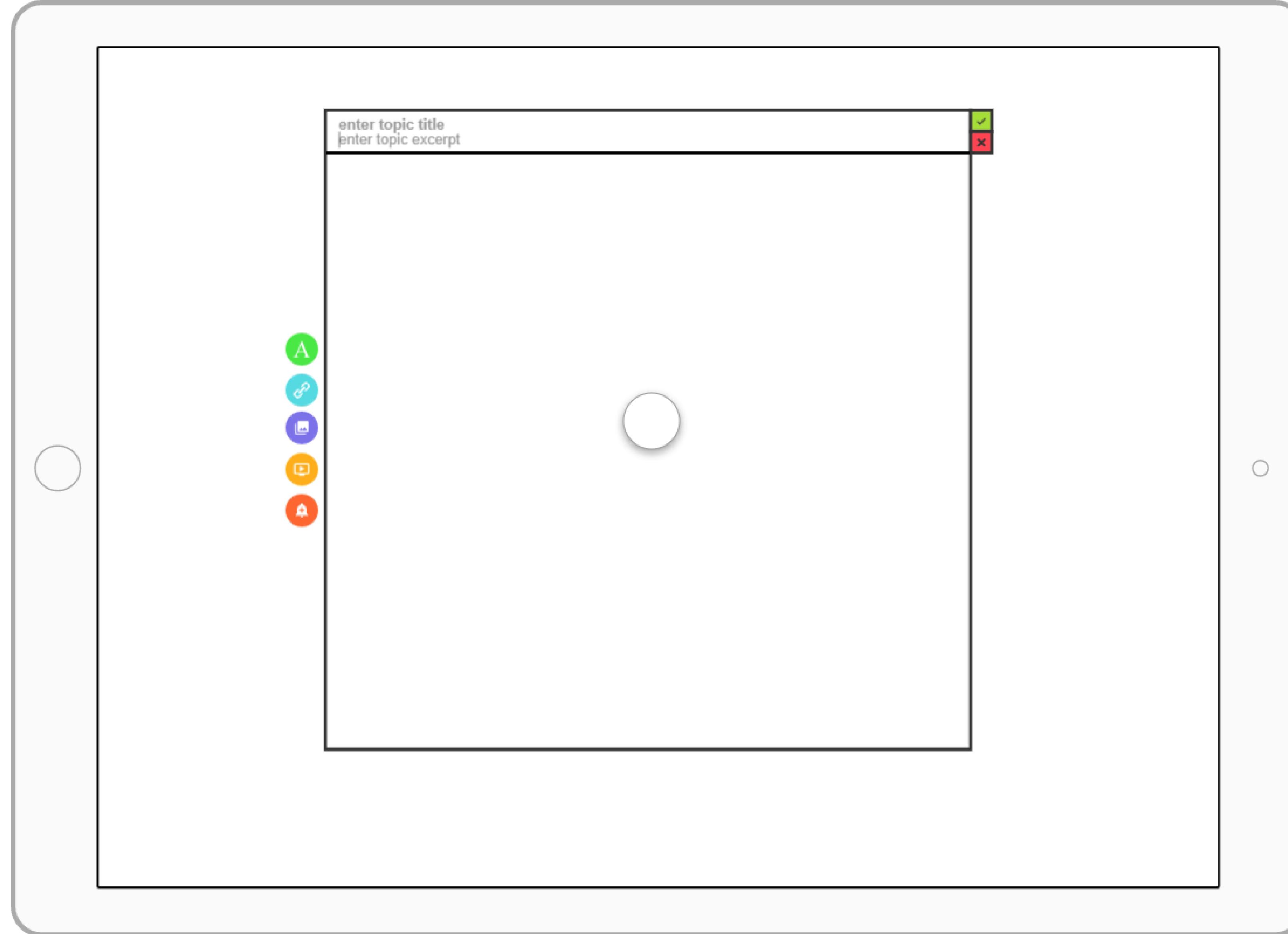
## TASK 2: ORGANIZE COURSE MATERIAL



## TASK 2: ORGANIZE COURSE MATERIAL



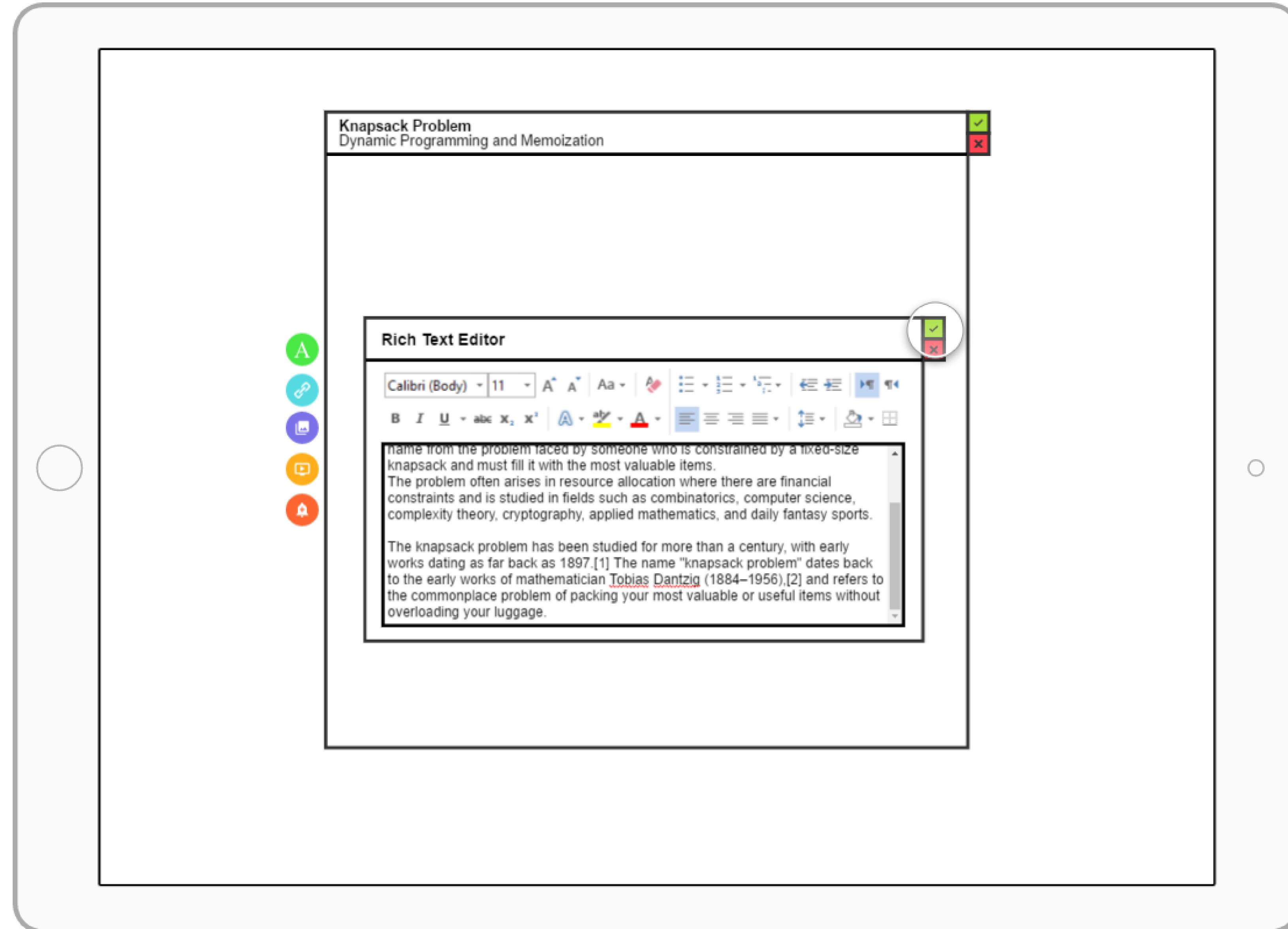
## TASK 2: ORGANIZE COURSE MATERIAL



## TASK 2: ORGANIZE COURSE MATERIAL

The image shows a mobile application interface with a light gray background. In the center is a white rectangular area containing a rich text editor. The editor has a title bar labeled "Rich Text Editor" and a toolbar with various formatting options like Calibri (Body), font size 11, bold, italic, underline, and alignment tools. Below the toolbar is a text input field with the placeholder "Enter or paste your text". To the left of the editor is a vertical sidebar with five circular icons: a green "A", a teal key, a purple document, an orange video camera, and an orange bell. At the top of the central area is a header bar with the text "Knapsack Problem" and "Dynamic Programming and Memoization" followed by two small colored boxes (green with a checkmark and red with an X). The entire interface is enclosed in a rounded rectangle.

## TASK 2: ORGANIZE COURSE MATERIAL



# TASK 2: ORGANIZE COURSE MATERIAL

## Knapsack Problem Dynamic Programming and Memoization

The knapsack problem or rucksack 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. The knapsack problem has been studied for more than a century, with early works dating as far back as 1897. The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1956), and refers to the commonplace problem of packing your most valuable or useful items without overloading your luggage.

A 1998 study of the Stony Brook University Algorithm Repository showed that, out of 75 algorithmic problems, the knapsack problem was the 18th most popular and the 4th most needed after kd-trees, suffix trees, and the bin packing problem. Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials,[4] selection of investments and portfolios, selection of assets for asset-backed securitization,[6] and generating keys for the Merkle–Hellman and other knapsack cryptosystems.

### [Wikipedia Knapsack Problem](#)



**TASK 2: ORGANIZE COURSE MATERIAL**

**Knapsack Problem**  
Dynamic Programming and Memoization

The knapsack problem or rucksack 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. The knapsack problem has been studied for more than a century, with early works dating as far back as 1897. The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1959).

**Reminder**

A 1  
pro  
suff  
pro  
sel  
gen  
Wi  
0

title  
MM / DD / YY  
HH : MM AM PM  
reminder description

2 kg  
\$1 1 kg  
\$10 4 kg

## TASK 2: ORGANIZE COURSE MATERIAL

**Knapsack Problem**  
Dynamic Programming and Memoization

The knapsack problem or rucksack 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. The knapsack problem has been studied for more than a century, with early works dating as far back as 1897. The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1959).

**Reminder**

A 1  
pro  
suff  
pro  
sel  
gen  
Wil  
0

PS9-1  
11 / 19 / 16  
11 : 59 AM PM

Short description|

## TASK 2: ORGANIZE COURSE MATERIAL

**Knapsack Problem**  
Dynamic Programming and Memoization

The knapsack problem or rucksack 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. The knapsack problem has been studied for more than a century, with early works dating as far back as 1897. The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1956), and refers to the commonplace problem of packing your most valuable or useful items without overloading your luggage.

A 1998 study of the Stony Brook University Algorithm Repository showed that, out of 75 algorithmic problems, the knapsack problem was the 18th most popular and the 4th most needed after kd-trees, suffix trees, and the bin packing problem. Knapsack problems appear in real-world decision-making processes in a wide variety of fields, such as finding the least wasteful way to cut raw materials,[4] selection of investments and portfolios, selection of assets for asset-backed securitization,[6] and generating keys for the Merkle–Hellman and other knapsack cryptosystems.

[Wikipedia Knapsack Problem](#)

**0/1 Knapsack Problem Dynamic ...**

**PS9-1**  
Short description for the reminder module. Short description for the reminder module.

11/19/2016

**A**

## TASK 2: ORGANIZE COURSE MATERIAL

CS 4150

Add Topic

Search

Access your frequently used boards quickly

Knapsack Prob

Discussions

Knapsack Problem  
Dynamic programming and memoization

?

PS9-1 11/19/16

Here are the next upcoming events based on your reminders.

PS9-1 11/19/16  
Knapsack Problem

http://boards.com/joel... Private

Home

## TASK 2: ORGANIZE COURSE MATERIAL

# SUMMARY

- Be aware of the scope.**
- Start small, grow gradually.**
- Be conscious of your choices.**
- Testing is never enough.**
- Modularity saves you time.**
- Iterate often, yet be smart.**
- Ask for opinions.**
- If it's not broken, don't fix it.**

# QUESTIONS?