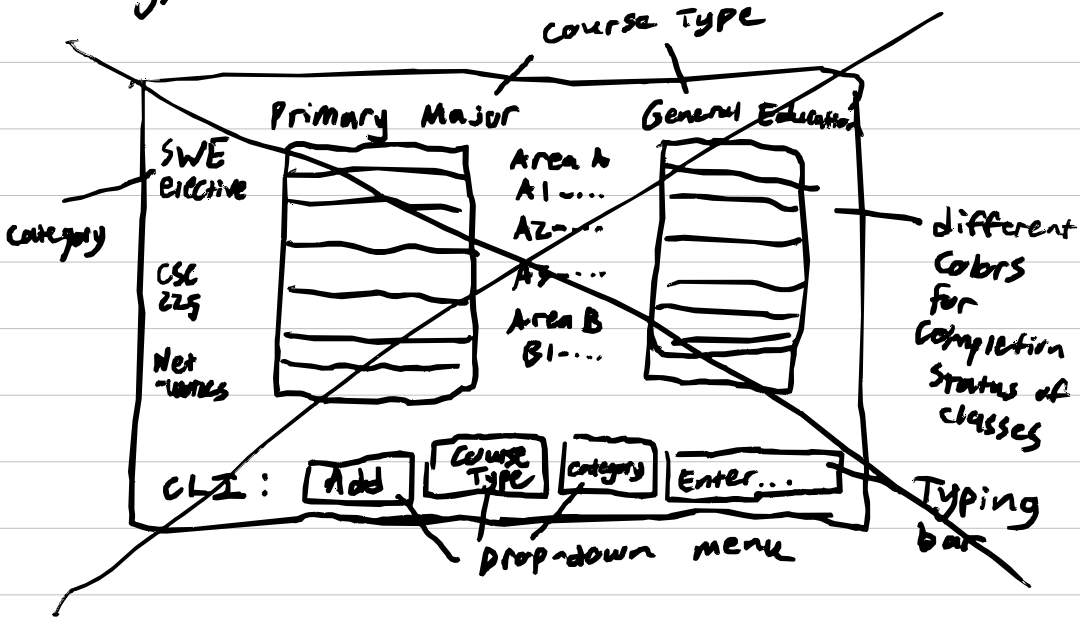
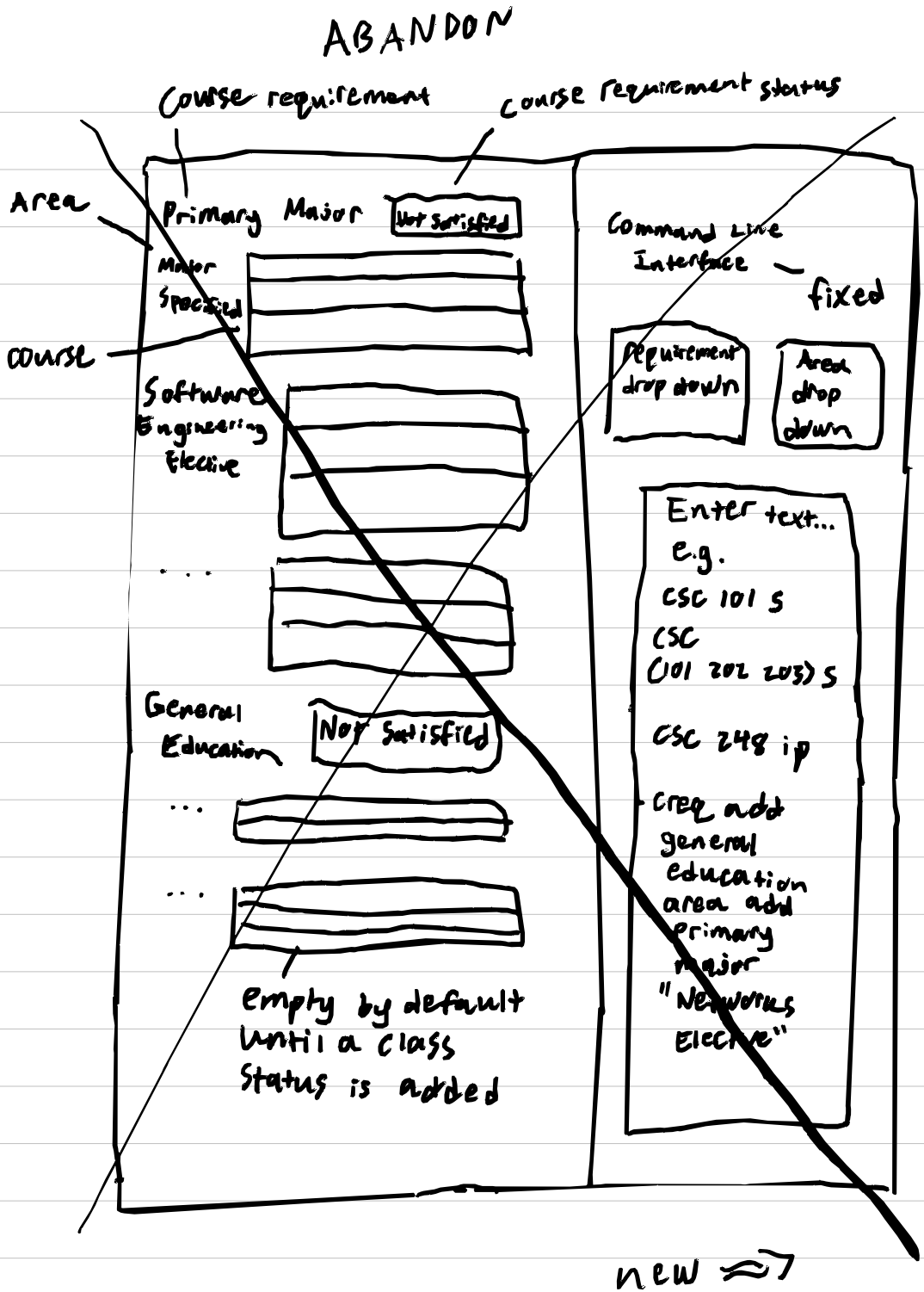
Goal: create a website that improves upon Cal Poly SLO's degree progress report, allowing me to see what requirements I have left. It should be easy to use and flexible for when my school switches to semester system.

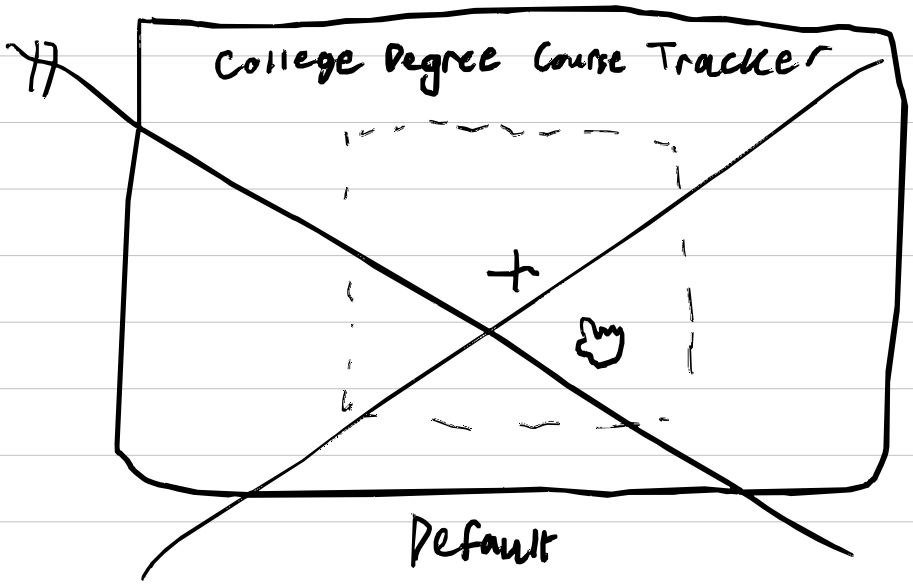


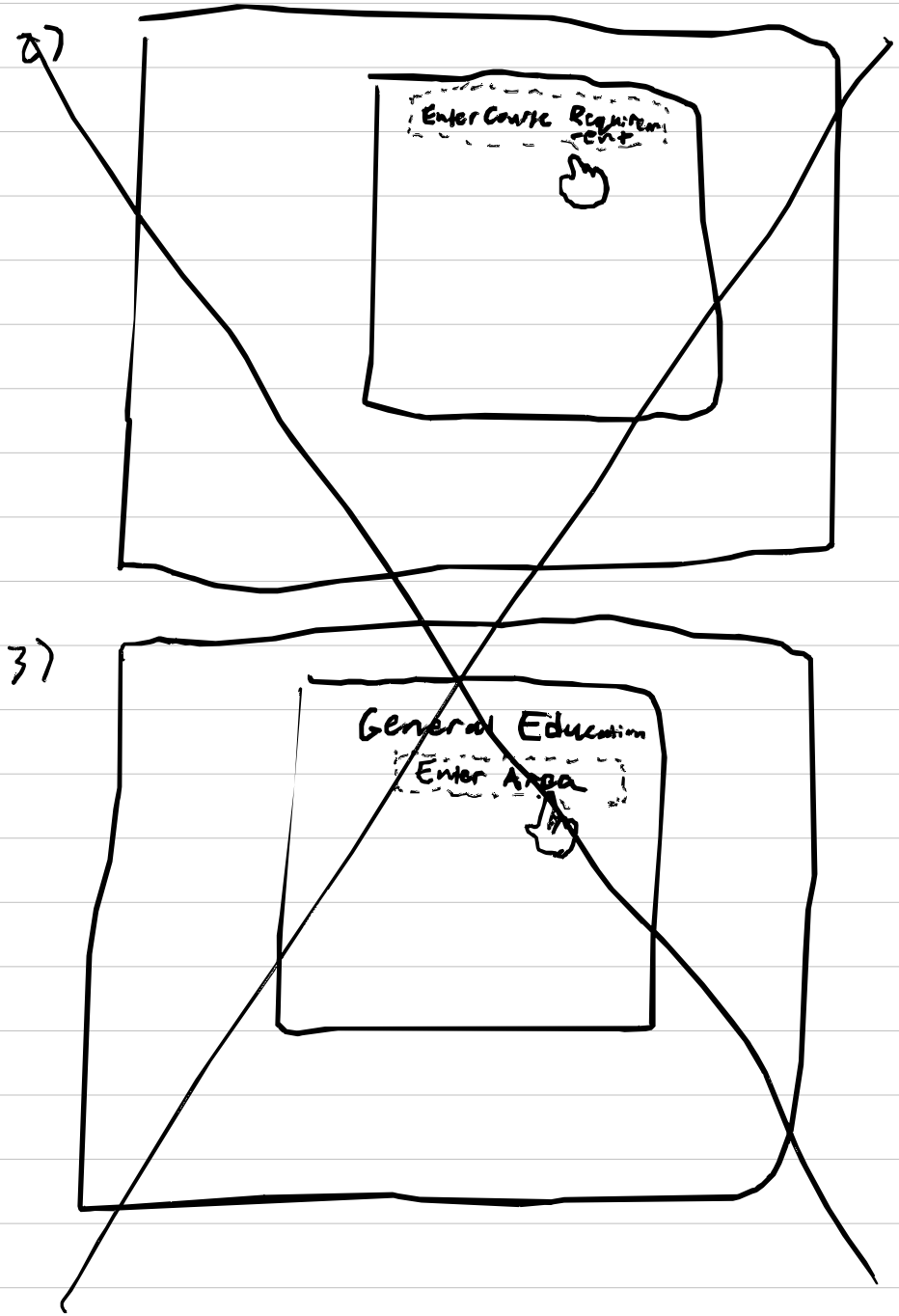

ABANDON
Course requirement
course requirement status
Area
Primary Major
Not satisfied
Major Specified
course
Software Engineering Elective
General Education
Not satisfied
empty by default until a class status is added
Command Line Interface
fixed
requirement drop down
Area drop down
Enter text...
e.g.
CSC 101 S
CSC (101 202 203) S
CSC 248 ip
creq add general education area add primary major "Networks Elective"
new

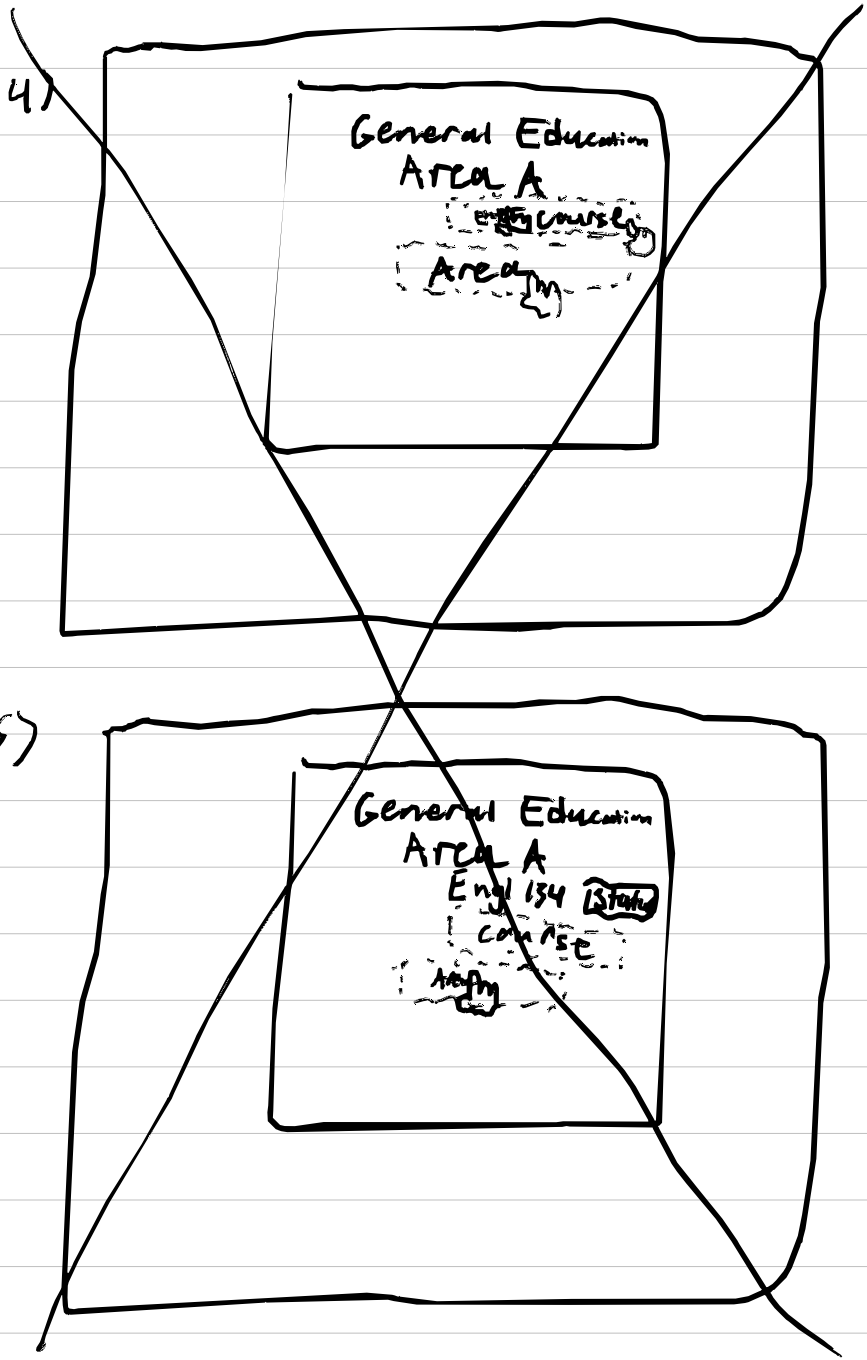
College Degree Course Tracker

Minimum Viable Project:

- add course requirements (or update/delete)
- add requirement areas to course requirements
- add courses to requirement areas
- mark courses as satisfied, in progress, or not satisfied
- store course requirements, areas, courses, and status in persistent storage (JSON?)

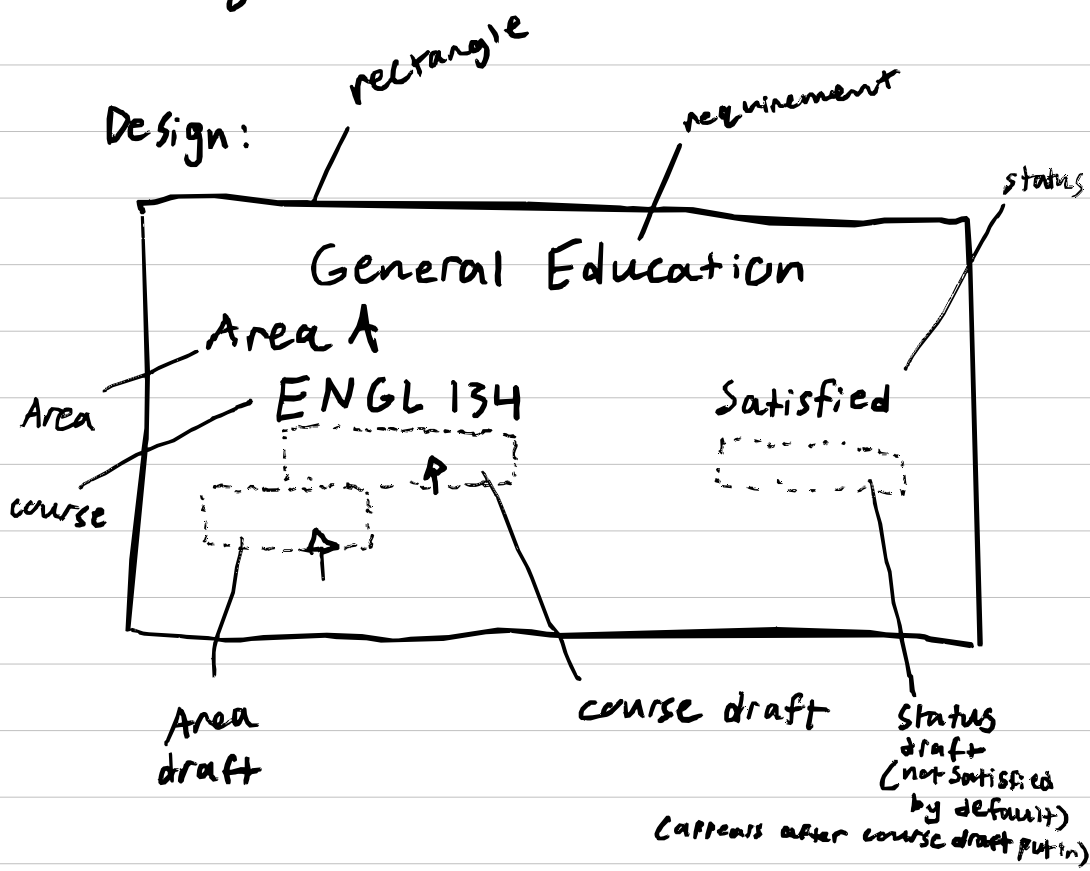


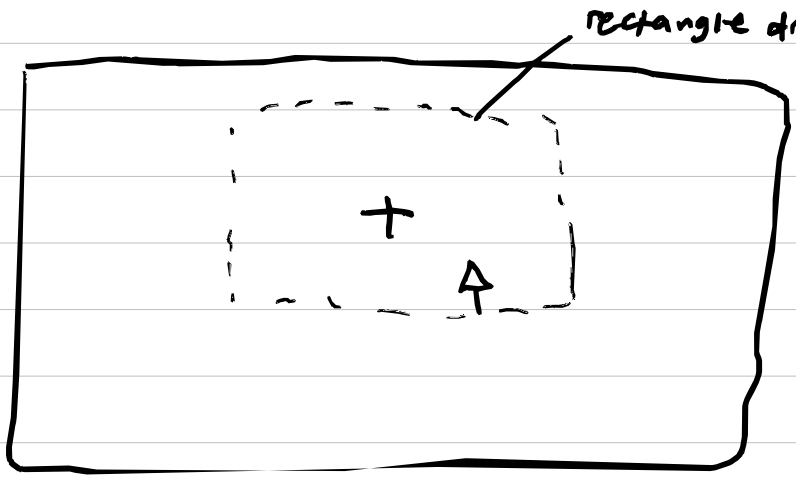

2)
Enter Course Requirement
3)
General Education
Enter Area


4)
General Education
Area A
Area
5)
General Education
Area A
Engl 134
course

Program

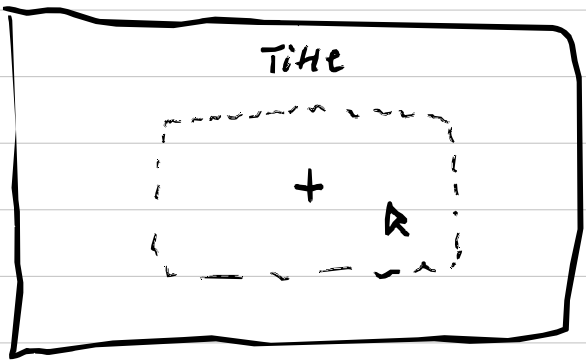
Design:





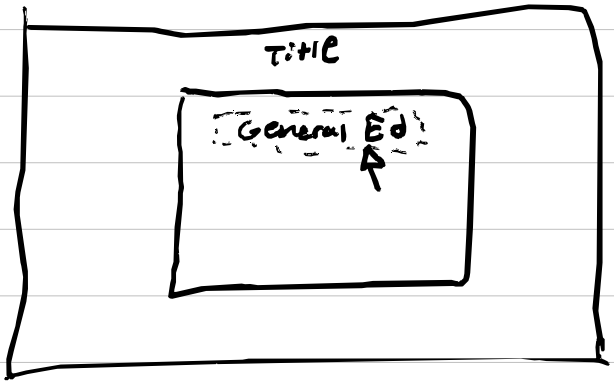
Scenario:

1) User hasn't created any rectangles



Key	Val

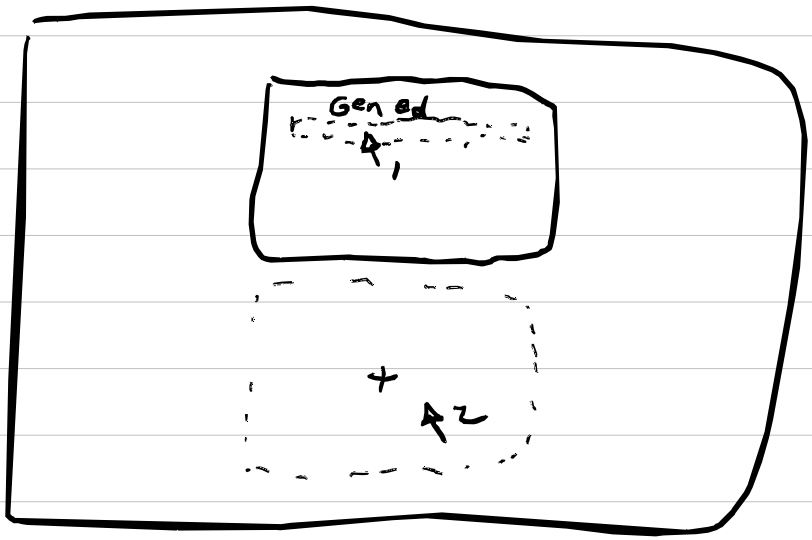
2) User creates rectangle
↳ prompted to input requirement



Key	Val
"rectangle-0"	{...} ↑ CSS props + requirement

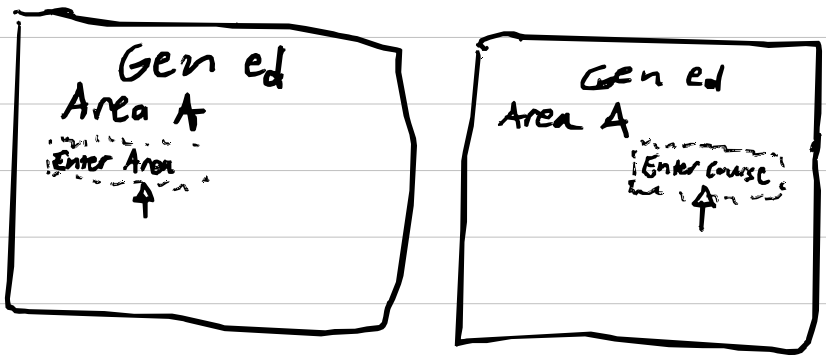
3) reload site and rectangle with properties still there

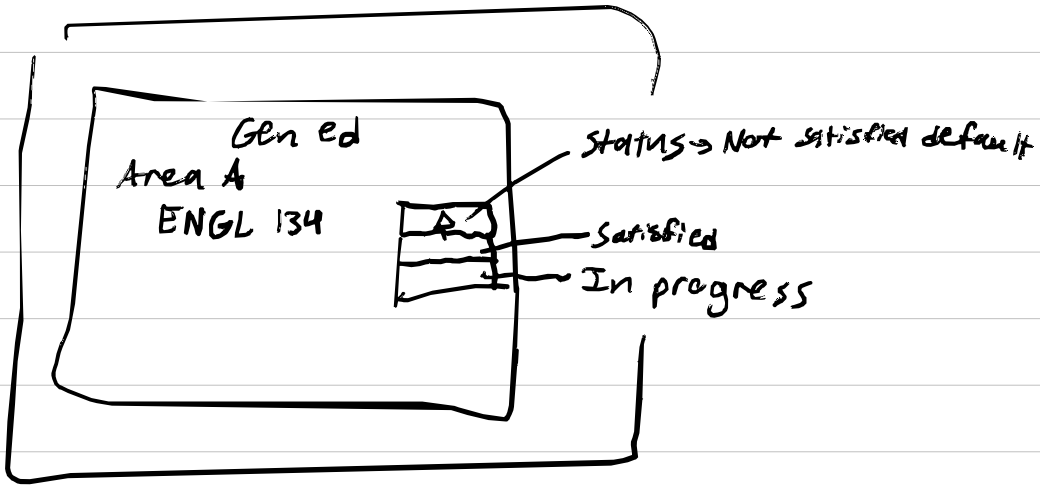
4) Options to put in areas and courses appear whenever cursor hovers over empty space[1] and another rectangle draft appears below (but only when cursor hovers over empty space)[2]

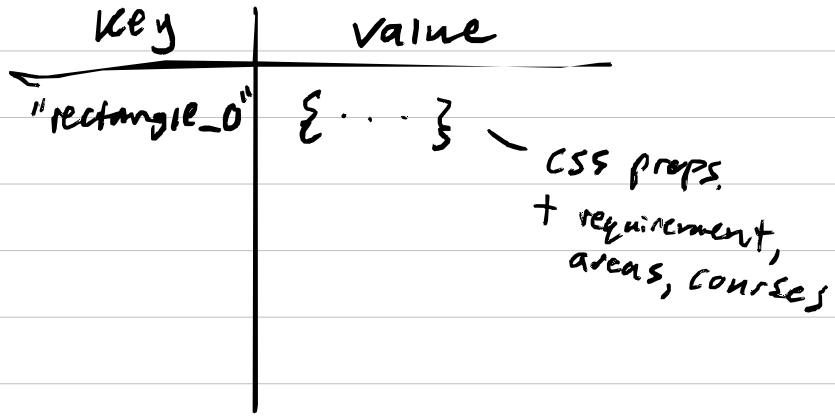


5) User enters Area → two options:

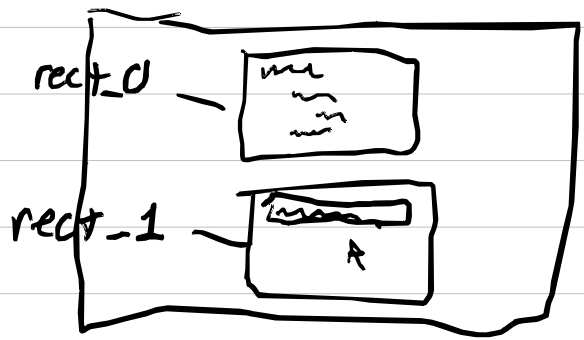
1. course
2. different area



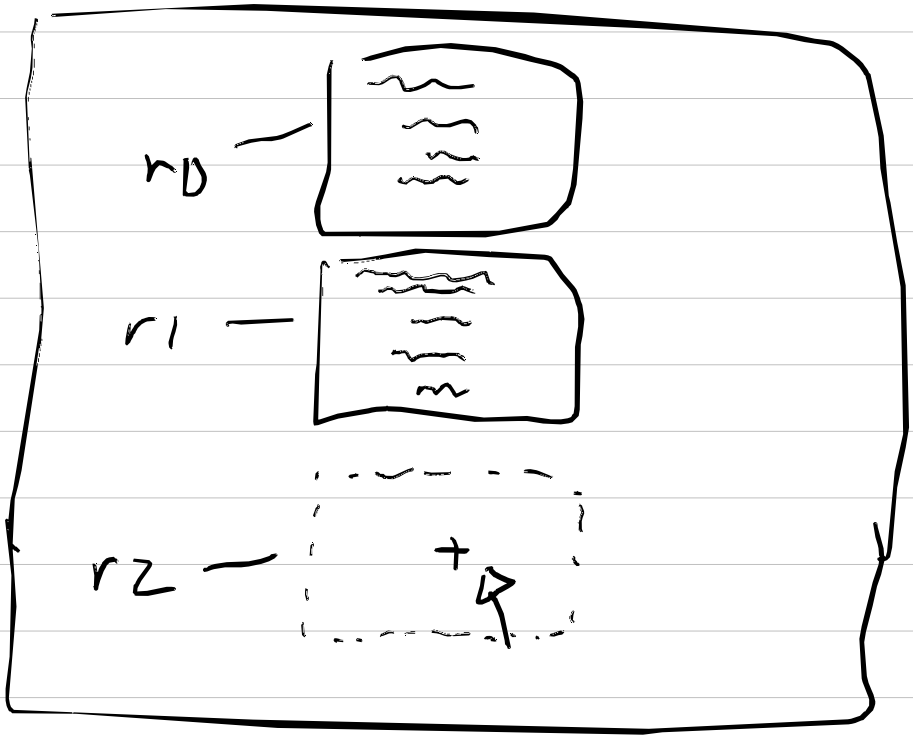




6) User creates new rectangle



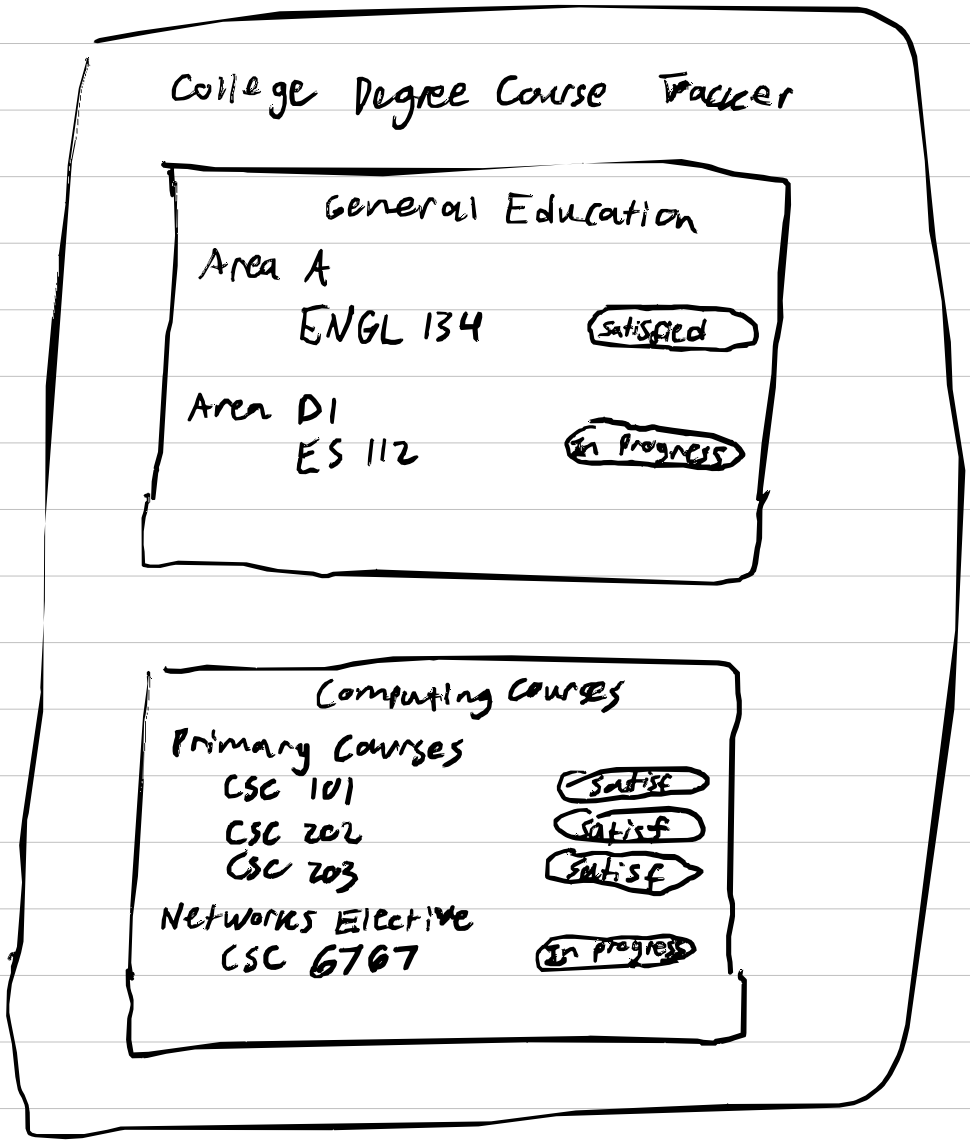
key	val
"r_0"	{..}
"r_1"	{..}



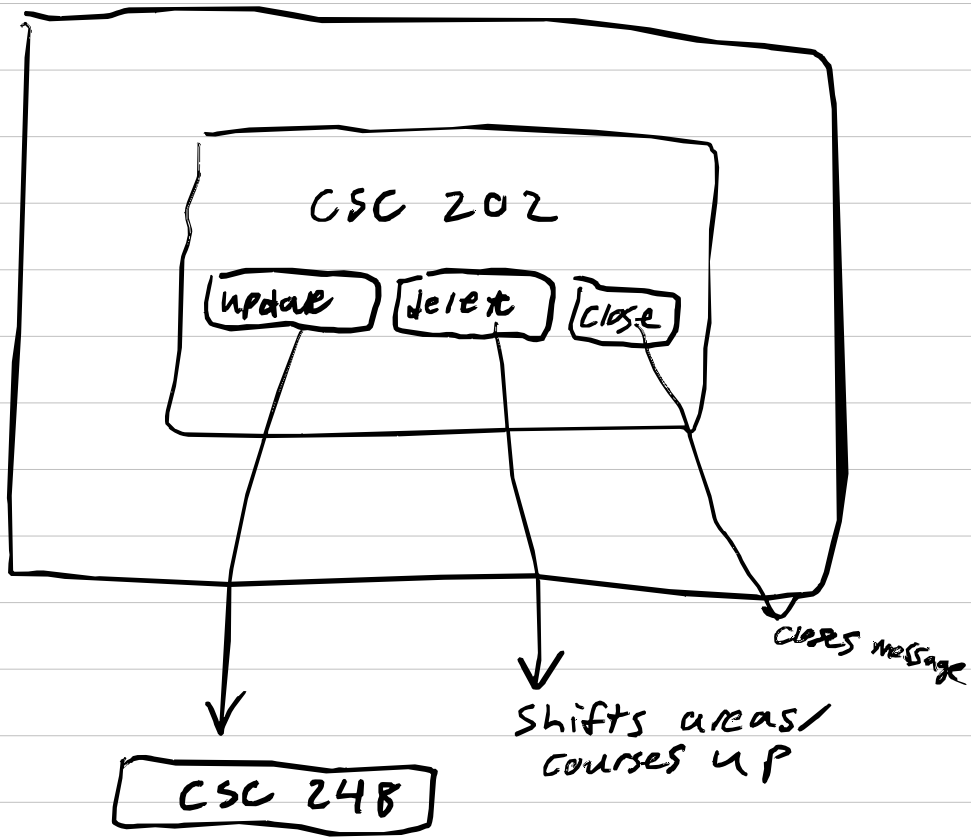
7) User reloads site and rectangle 0 and 1 are still there with correct properties and info.

8) (Add keyboard arrow key navigation later)

Example Web Page



a) Option to update/delete upon clicking field

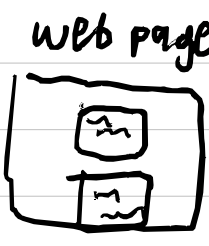


Program Design Flowchart

Pre-existing course data?

Yes

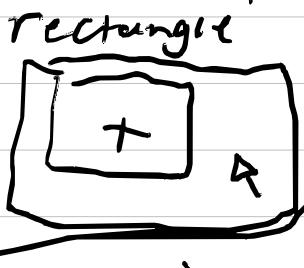
load data onto web page



Add invisible draft rectangles at bottom of course fields and area fields or requirement fields

No

Start web page with draft rectangle



Bold border when cursor outside rect.; Dashed border when cursor inside rect.;

User clicks draft rect.

Prompt to enter requirement appears

User enters words? (no empty string)

yes

No

No new rect. created (no changes)

New rectangle with requirement field added

Add invisible draft rectangle below bottom rectangle

No border when cursor outside rect.; Dashed border when cursor inside rect.;

Requirement field has areas?

NO

Add 1 invisible rectangle for area

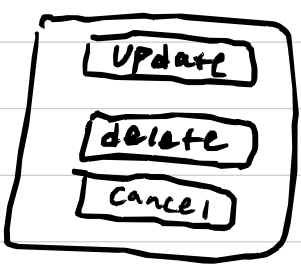
No border when cursor outside rect.; Dashed border when cursor inside rect.;

yes

Add invisible rectangle under bottom course field of every area; Add invisible rectangle under bottom area field of the rectangle

Click on existing field on a rectangle

Option box to update, delete, or cancel



Click update:
- field pops to input new name

Click delete:
- deletes entire rect. if clicked field is a requirement
- deletes entire branch if area
- deletes singular course if course
- ensure that if a course is deleted, move fields below (courses, areas, courses under other areas)

- See above bulletpoint for areas

Click cancel:
- Box for options disappear and nothing changes on the web page

doesn't allow duplicate requirements

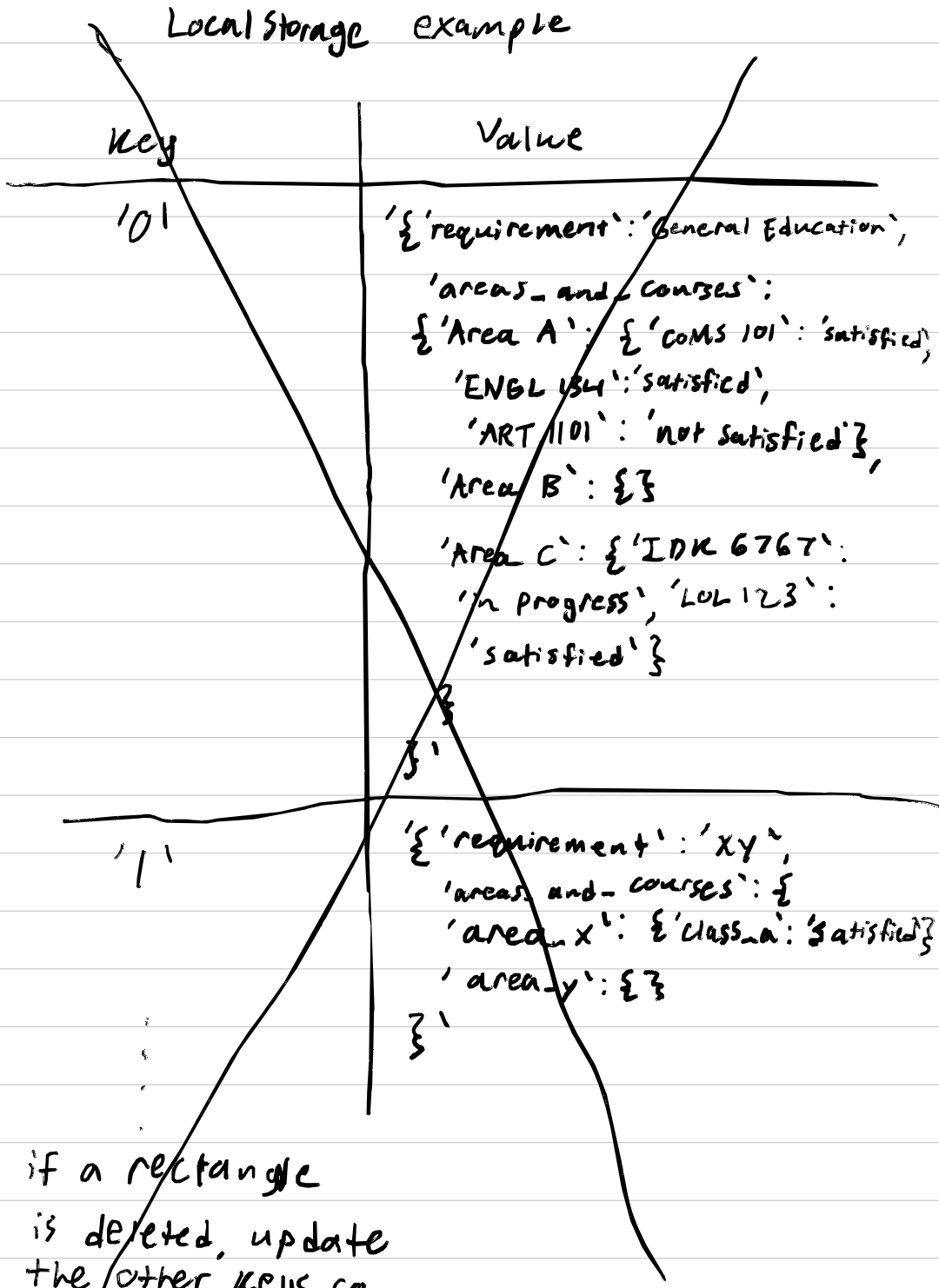
Local Storage for loading website

e.g

key	value
'General Education'	'{'Area A': '{'COMS 101': 'satisfied', 'ENGL 134': 'satisfied', 'ART 101': 'not satisfied'}', 'Area B': '', 'Area C': '{'IDK 6767', in Progress, 'IDL 123', 'satisfied'}'}'
'CS Primary Major'	'{'area', '{'course': 'status'}'}'

Example Syntax

```
Const requirements = Objects.keys
(localStorage); // ['General Education',
                    'CS Primary Major']

Const ge_area_course_values
  = JSON.parse(localStorage.get('General
                Education'))

// {'Area A': '{'COMS 101': 'satisfied',
     'ENGL 134': 'satisfied', 'ART 1101':
     'not satisfied'}'}

Const area_A_course_statuses
=
JSON.parse(ge_area_course_values['Area A']);
```



if a rectangle is deleted, update the other keys so that the order matches Object.keys(localStorage)

i.e., '0' → 0

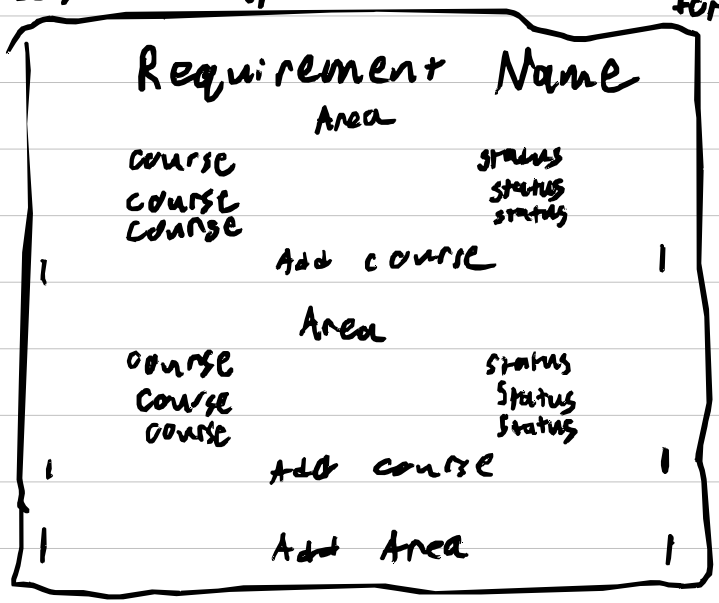
id localStorage key 0

// load rectangles, load areas, load area courses, load status of courses, load draft course rectangles under last course of an area, load draft area rectangle under last area of the requirement or the requirement div if no areas exist for the rectangle

```
function loadWebPage() {
    // load rectangle div
        // using className to access CSS values
    // append requirement div to rectangle div
        // Parse requirements from localStorage
            JSON's 'requirement' key

    // append area divs with
        corresponding course
        and course status divs to rectangle div

    // append draft area divs

    // append draft course divs

    // append draft rectangle div

}
```

+ no more duplicate requirements

+ use a Map instead of Object structure for easier iteration (map_name.size)



* no duplicate areas within requirements, no duplicate courses within areas

Add Area

* cannot add duplicate requirement

Local Storage Example

key	value
"requirement_name"	"{"Area A": {"ENGL 134": "satisfied", "COMS 101": "satisfied" }, "Area B": { }, "Area C": {"course": "in progress" } }"

Code restructuring

Tasks:

1. load data onto web page
2. load draft requirements, area, and courses
3. allow for addition of requirements, areas, and courses
4. allow for deletion of requirements, areas, and courses
5. allow for updating requirements, areas, and courses
6. allow for cancelation of interactions (clicking on drafts)

return to later. need to
✓ create data before testing part 1

1) load data onto web page
   a) only one key value pair in localStorage
      i) key = ''
      ii) value = '{'requirement 1':
{'course 1': 'satisfied',
'course 2': 'not satisfied',
'course 3': 'in progress'
},
'requirement 2':
{'course 1': 'satisfied',
'course 2': 'not satisfied',
'course 3': 'in progress'
},
'requirement 3':
{'course 1': 'satisfied',
'course 2': 'not satisfied',
'course 3': 'in progress'
}
}'

Convert String of entire object to Map

Convert String of course-status Object to Map

b) Convert string of localStorage key into a Map data structure, and then create a rectangle of requirement, areas, courses, and statuses for each key-value pair in the map.

i) function getMapOfData()
```
const obj = JSON.parse(
localStorage.getItem(''));
const map = new Map(Object.entries
(obj));
return map;
}
```

ii) function getMapOfCourseStatus (requirement){
```
const map_of_data = getMapOfData();
const obj_of_course_status
= JSON.parse(map_of_data
    .get(requirement));
const map_of_course_status
= new Map(Object.entries(
    obj_of_course_status));
```

assume requirement's value will be string of an object

```
return map_of_course_status;
```

2) create functions for creating / appending rectangles, areas, courses, and also message boxes.

i) function createBeginningDraftRectangle() {

```
const draft_rectangle
  = document.createElement('div');
draft_rectangle.className
  = 'beginning_draft_rectangle_template';
draft_rectangle.textContent = 't';
```

}