

Study Share Portfolio

Annie Steenson and Zach Newton

Annie Steenson and Zach Newton

Table of Contents

Annie Steenson and Zach Newton

- I. Evidence of Complex Issues and Higher Levels of Bloom's Taxonomy**
 - A. Study Share Application**
 - B. Features and Implementation**
 - 1. JavaScript Objects (JSON)**
 - 2. AngularJS**
 - 3. Forms**
 - 4. Core Functionality**
 - 5. JQuery**
 - 6. Algorithms**
- II. Evidence of Iteration with Materials**



Stay Organized! Share Collections!

Study Share allows students to create virtual decks of note card for studying. The site is currently under construction. Decks will be lost after window is closed.

[GET STARTED!](#)[CREATE A DECK](#)[CREATE A NOTECARD](#)

Your Decks

[Software User Interfaces](#)[Groceries for Beginners](#)[States and Capitols](#)

Evidence of Complex Issues and Higher Levels of Bloom's Taxonomy

Study Share Application

Written in plain HTML, CSS, and JavaScript, Study Share enable students to create virtual notecards to study from. Students can organize their notecards into decks.

Study Share uses two forms to receive information from the user: one for creating a deck and another for creating a notecard.

When the user submits the form for either a Deck or Notecard, using JavaScript, we store the form information in a JSON (a.k.a. JavaScript Object). The application uses local and temporary memory.

In order for the application to dynamically update, we utilize a framework called AngularJS. With AngularJS, users are able to view decks or notecards they just created.

Created with CSS and JQuery, Study Share's user interface is simple and clean. CSS allows us to use classes to group DOM elements that require the same style. JQuery enables us to create CSS and visual effects using Event Handlers.

For optimization, Binary Search was implemented to find the correct Deck object when switching focusing from one to another.

Overall, we applied our knowledge of CSS, JavaScript, and HTML in order to create an application that uses Forms, JavaScript, DOM Elements, functions, and JavaScript Objects.

Features and Implementation

JavaScript Objects (JSON)

We used JSON, JavaScript Objects, to transfer data about Decks and Notecards. All the Decks are stored in a Deck array. All the Notecards for a particular Deck are stored in an array within the Deck Object.

```
scope.decks = [{
  id: 0,
  name: 'States and Capitols',
  classes: 'American Studies',
  descriptions: 'States on Front, Capitols on Back',
  cards: [
    {
      front: "Nebraska",
      back: "Lincoln"
    },
    {
      front: "Iowa",
      back: "Des Moines"
    }
  ]
}];
```

AngularJS

One skill we worked on from outside of class was using AngularJS. A web application framework created with JavaScript, AngularJS allows us dynamically manipulate the DOM for real-time updates.

```
angular.module('studyShare', []).controller('studyShareController',  
    function StudyShareController($scope) {} );
```

Within Angular's `$scope`, we attach each form element to a model. For example:

- `$scope.deck_name` is the model attached to the form input, "deck_name"
- `$scope.deck_classes` is the model attached to the form input, "deck_classes"

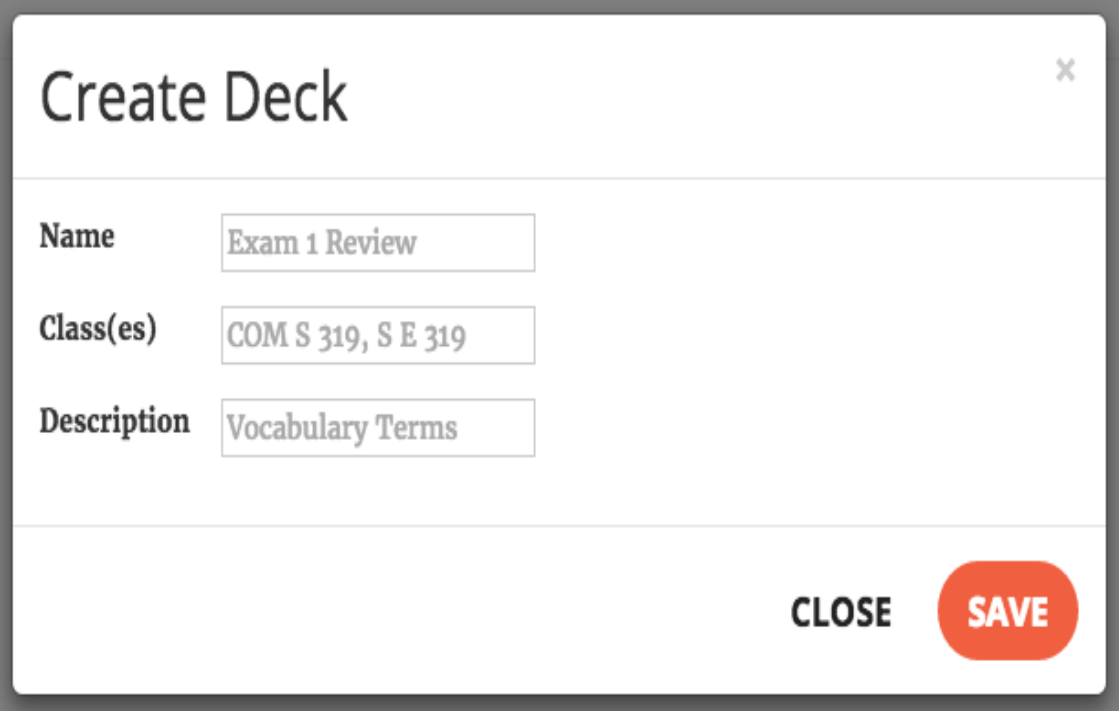
Forms

Study Share provides two separate HTML forms: one for creating a deck and one for creating a notecard.

```
<form>  
  <label for="deck_name">Name</label>  
  <input id="deck_name" type="text"  
    name="deck_name"  
    placeholder="Exam 1 Review"  
    ng-model="deck_name"/>  
  
  <label for="deck_class">Class(es)</label>  
  <input id="deck_class"  
    type="text"  
    name="deck_class"  
    placeholder="COM S 319, S E 319"  
    ng-model="deck_classes"/>  
  
  <label for="deck_description">Description</label>  
  <input id="deck_description"  
    type="text"
```

```
        name="deck_description"
        placeholder="Vocabulary Terms"
        ng-model="deck_description"/>
</form>
$scope.saveDeckModal = function () {
    $scope.active_deck = {
        id: $scope.deck_counter,
        name: $scope.deck_name,
        classes: $scope.deck_classes,
        description: $scope.deck_description,
        cards: []
    };
    $scope.decks.push($scope.active_deck);
    $scope.deck_counter++;

    $scope.closeDeckModal();
};
```



The screenshot shows a 'Create Deck' modal window with a close button (X) in the top right corner. The form contains three input fields: 'Name' with the value 'Exam 1 Review', 'Class(es)' with the value 'COM S 319, S E 319', and 'Description' with the value 'Vocabulary Terms'. At the bottom right, there are two buttons: a 'CLOSE' button and a red 'SAVE' button.

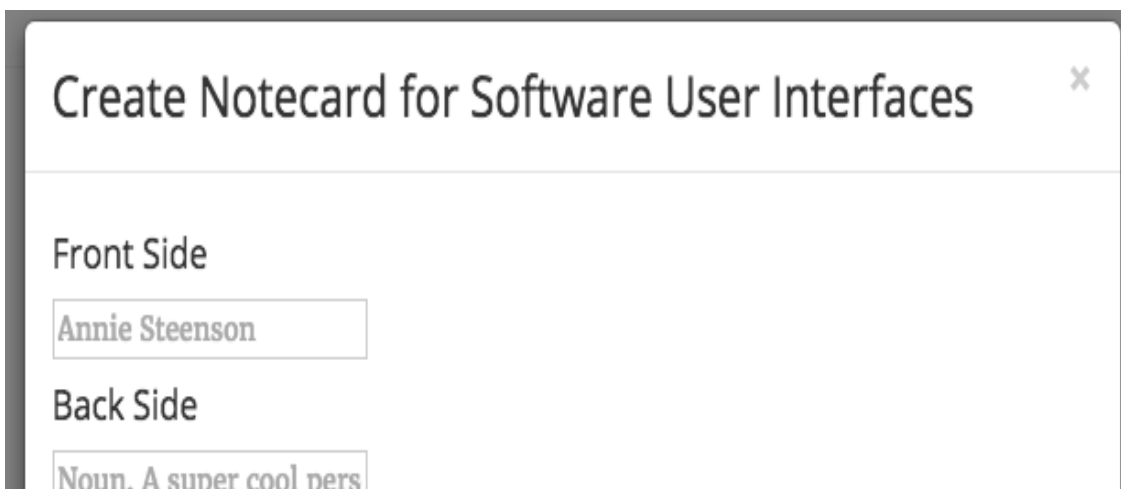
When “Save” is clicked:

1. Either “saveDeckModal” or “saveCardModal” is called.

2. Use the data-bindings created from Angular's directive, ng-model, to retrieve form information.
3. Assign the new Deck JSON to `$scope.active_deck`
4. Push `$scope.active_deck` to the array of Decks, `$scope.decks`
5. Close the modal in use and clear the form input elements.

```
<form>
  <h4>Front Side</h4>
  <input type="text" name="card_front"
    placeholder="Annie Steenson"
    ng-model="card_front"/>
  <h4>Back Side</h4>
  <input type="text" name="card_back"
    placeholder="Noun. A super cool person."
    ng-model="card_back"/>
</form>
```

```
$scope.saveCardModal = function () {
  if ($scope.active_deck == null) {
    $("#form_error").show();
    console.log("null");
    return;
  }
  var deck = $scope.findDeck($scope.active_deck.id);
  deck.cards.push({
    front: $scope.card_front,
    back: $scope.card_back ,
  });
  $scope.closeCardModal();
};
```



Create Notecard for Software User Interfaces

Front Side

Annie Steenson

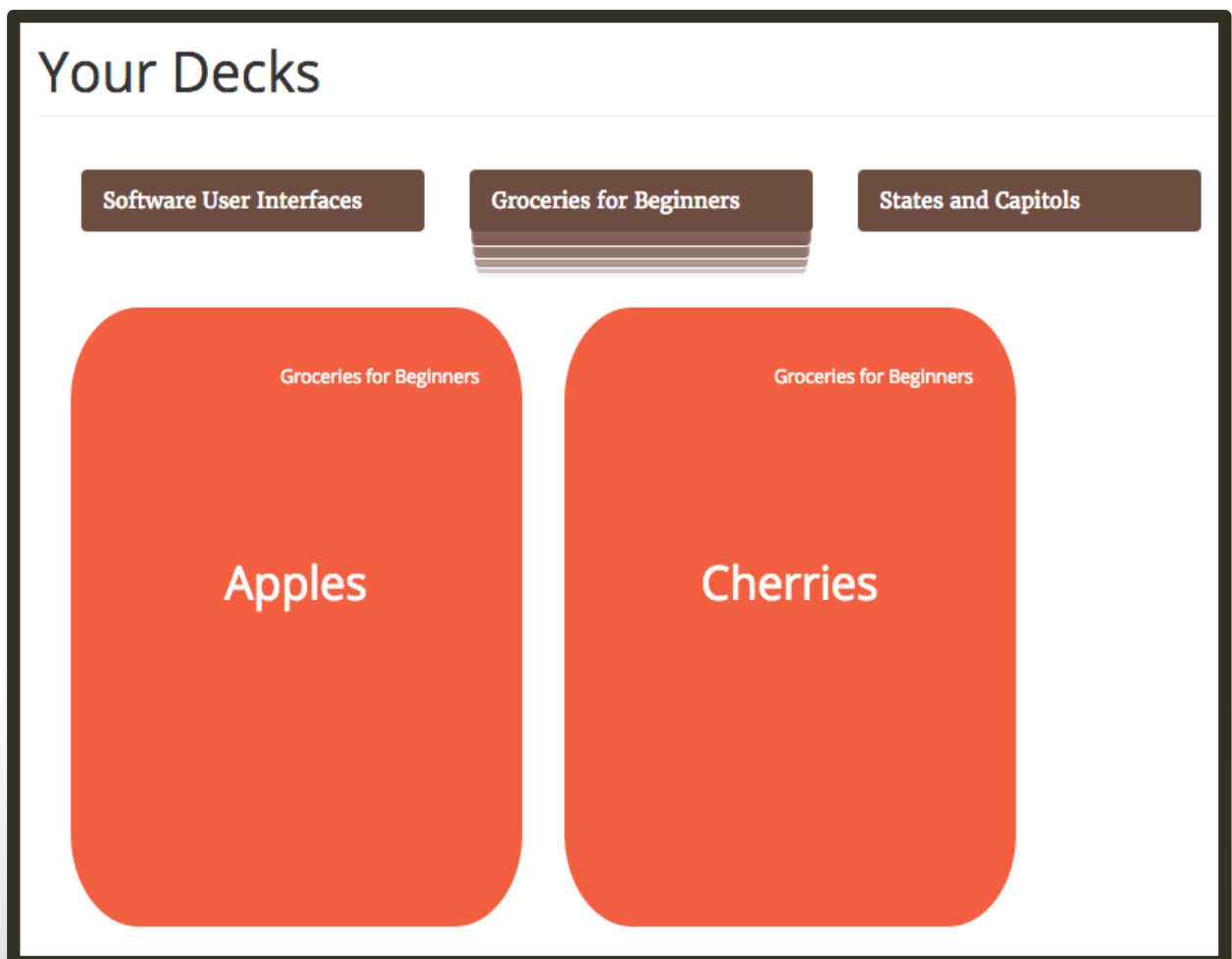
Back Side

Noun. A super cool pers

Core Functionality

The user can view each Deck by click on the Deck's title. A gradient graphic will transform under the currently selected Deck's title. The Deck currently selected is apparent by the gradient graphic blow the name.

Once selected, the Deck's Notecards will dynamically show and the Notecards in the previously selected Deck will dynamically hide. Each Deck's title is brown.



The front-side of a Notecard is orange and the back-side is brown. To flip over a Notecard, hover over it. Each notecard in a particular Deck shows the Deck's name in the top right corner.

Your Decks

Software User Interfaces

Groceries for Beginners

States and Capitols

POST is one of many request methods supported by the HTTP protocol used by the World Wide Web. By design, the POST request method requests that a web server accepts and stores the data enclosed in the body of the request message. It is often used when uploading a file or submitting a completed web form.

Software User Interfaces

GET Request

Jquery

Powered with JQuery, we grab the DOM element of each Deck title and modify the CSS to make it a stack appearance show under the Deck title when that Deck title element has been clicked.

This was particularly complex because we had to make sure of the finicky `setTimeout()` method that never seemed to work right. However, we eventually got a delayed looping function to occur by calling a `setTimeout()` function inside of another function that then calls itself a set number of times.

This function loop required a solid comprehension of JavaScript scope, similar to what we went over in class.

```
$scope.indicateCards = function (event, deck) {
  element = event.currentTarget;
  cards = deck.cards.length;
  cardList = document.createElement("div");
  cardList.className = 'deck-hover-wrapper';
  element.appendChild(cardList);
  var i = 0;
  var loop = function () {
    setTimeout(function(){
      card = document.createElement('div');
      card.className = 'deck-hover-display';
      card.style.zIndex = cards - i;
      card.style.backgroundColor = 'rgba(109,76,65,' + (0.6-i*(0.6/5)) + ')';
      card.style.width = (element.offsetWidth - i*2) +
        'px';
      card.style.height = (30 - i*2) + 'px';
      cardList.appendChild(card);
      i++;
      if (i < 5){
        loop();
      }
    }, 20);
  };
  loop();
}
```


Once a Deck is selected, all the Notecards in that deck are dynamically shown. We use Angular's ng-click directive, so when a Deck title is clicked `$scope.view` is called. `scope.view` takes a `deck_id` as a parameter so we select the correct Deck DOM element.

```
$scope.view = function (deck_id) {
  var collapseid = null;
  if ($scope.active_deck != null) {
    collapseid = $('#collapse' +
                  $scope.active_deck.id);
    collapseid.hide();
  }
  $scope.active_deck = $scope.findDeck(deck_id);
  collapseid = $('#collapse' +
                $scope.active_deck.id);
  collapseid.show();
};
```

Algorithms

One complex thing that we dealt with as a team was finding a deck within an array of JS objects. This was complex for us because we used a binary search method that tripped us up in several places with traversing an array of JavaScript objects.

```
$scope.findDeck = function (deck_id) {
  var min_index = 0;
  var max_index = $scope.decks.length - 1;
  var current_index;
  var current_deck;
  while (min_index <= max_index) {
    current_index = (min_index + max_index) / 2 | 0;
    console.log("current_index: " + current_index);
    current_deck = $scope.decks[current_index];
    if (current_deck.id < deck_id) {
      min_index = current_index + 1;
    } else if (current_deck.id > deck_id) {
      max_index = current_index - 1;
    } else {
      return current_deck;
    }
  }
}
```

```
}  
}  
return -1;  
};
```

Evidence of Iteration of Materials

In lab 2, we used basic HTML DOM elements and implemented a form that is validated with JavaScript. In lab 2 we used JavaScript to grab information from the HTML form elements. The topics covered in lab 2 were:

- Simple variables (local and global), conditions, loops, methods, and arrays.
- User input validations
- JS cookies
- Event Handling

In lab 3, we create a simple calculator using JavaScript. Users can show and delete the previous value on screen. After entering values in a textbox, users can add subtract and multiply. The topics covered in lab 3 were:

- JavaScript
- Objects
- Functions
- Closures