Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

My program is written in JavaScript using Adobe Dreamweaver CS4. The purpose of my program is to provide users with a CRUD (Create, Read, Update, Delete) interface using cards that have a title and description. In the video, you can first see how a new entry is created. Also, once a new entry is created, it can be updated by clicking the "Edit" button, changing the information, and clicking "Save". A user can also update an entry's starred status by clicking the star in the card, which toggles the current status. Finally, the video demonstrates how a user can delete an entry by clicking the "Delete" button in the card.

2b)

I completed this project independently. When I began developing my program, I first created an HTML/CSS page with all the visual features I wanted to include, which would soon be functional with JavaScript. I first wanted a user to be able to create a new note. I used a form with JavaScript to collect all the info from the text fields when the "Add Entry" button is clicked. When developing this feature, I encountered an error when trying to display the newly added information. In my "displayNotebook" function I used a for loop with the counter variable of "i" to loop through the array. However, there was one place where I was calling a non-existent variable "id". To fix this, I simply replaced calling "id" with "i". Later on, I developed the functionality for the edit feature. After writing the code for "editEntry", I tried calling it when a user clicked the "Edit" button. However, the code would not work properly because the order of the parameters where I was calling the function was not the same as how I defined them. I solved this by fixing the order and making sure that similar functions had similar parameter orders.

2c)

```
function addEntry(array) {
  var title = document.getElementById("entry_title").value;
  var info = document.getElementById("entry_info").value;

  if (title) {
    array.unshift({title:title, info:info, starred:false});
    document.getElementById("entry_title").value = "";
    document.getElementById("entry_info").value = "";
    toggleElement("entry_form", "flex");
    displayNotebook(array);
  }
}
```

The main algorithm that I selected is “addEntry”. This algorithm has two key parts: “toggleElement” and “displayNotebook”. The “addEntry” function first assigns two variables from the information entered into the form (the title and info). Then, the function uses an if statement to check if the title variable is not empty. “toggleElement” is designed to hide an element if it is visible or show an element if it is hidden, using an if-else statement. In the “addEntry” function, if the “entry_form” element’s display status is equal to “flex” the element is hidden, if not, it is shown. Also, “displayNotebook” takes whatever array is given to it through a parameter, converts it into HTML code, and places it in a div with the id of “notebook”.

2d)

```
function displayNotebook(array) {
  document.getElementById("notebook").innerHTML = "";
  for (var i = 0; i < array.length; i++) {
    var entry = document.createElement("div");
    entry.setAttribute("class","notebook_entry");
    entry.setAttribute("id","entry_" + i);
    var html = "";
    html += "<h1 id='entry_" + i + "_title'>" + array[i].title + "</h1>";
    html += "<h2 id='entry_" + i + "_info'>" + array[i].info + "</h2>";
    html += "<input id='edit_" + i + "_title' value='" + array[i].title + "'
style='display:none' placeholder='Title'>";
    html += "<textarea id='edit_" + i + "_info' style='display:none'
placeholder='Info'>" + array[i].info + "</textarea>";
    html += "<button class='edit_entry' id='edit_" + i + "'>Edit</button>";
    html += "<button class='save_entry' id='save_" + i + "'
style='display:none'>Save</button>";
    html += "<button class='delete_entry' id='delete_" + i + "'>Delete</button>";
    html += "<i class='fa" + determineStar(array[i].starred) + " fa-star'
id='star_" + i + "'></i>";
    entry.innerHTML = html;
    document.getElementById("notebook").appendChild(entry);
  }
}
```

The abstraction I selected is “displayNotebook”. An abstraction in computer programming manages the complexity of the code. My function does this by allowing a programmer to just use one line of code in order to display the notebook, instead of having to understand how the notebook array is constructed. Also, “displayNotebook” eliminates having to call 16 lines of code every time the notebook needs to be displayed. This makes coding and debugging my code a lot easier and manages the complexity.