# CIT 215 Final – TM

Github Repo: <https://github.com/TMatis/CRUD-Final-Project-CIT-21500>

Github Pages: <https://tmatis.github.io/CRUD-Final-Project-CIT-21500/>

^For above make sure you are on page “Library” Username is “timo”, Pass is “pi”

# HTML:

<style> is changing the styling for the general CSS file that formats all of the different html pages.

I change the card size and add a min Hight. Make the add\_series into grid column and rows, and make it pretty. Make the series\_container flex and keep the series boxes stay in line. The container is forced to follow a split of 2/8 and 6/8 for add\_series and series containers respectively, it also sets the hight to the viewport size. The series box makes things centered and pretty.

        .container{

            grid-template-columns: repeat(2fr 6fr);

            gap: 10px;

            overflow-y: auto;

            max-height: 100vh;

        }

<body> has most of the set-up content. It has a nav bar, Add\_series, and series\_container.

The nav bar is applied globally and has a minuet differences in which one is “Active”.

The add\_series <div> has the, status box which gets updates as you click buttons, a login box, an Add\_Series box where you add a new series to the array, an info box that gets info added to it, and an update content box that gets content added to it. All box’s except for the login and status are hidden on star up. That changes as the functions are called. It also holds the Status update box, below is the code.

                <div id="StatusBar" class="card" style="height: fit-content; min-height: 100px;">

                    <h2>Status:</h2>

                    <p id="Status\_Info"></p>

                </div>

Below body in the <html> is a footer with secret login bypass button, a second footer for if you log in normally, and a <script> that contains the code for the sticky nav bar.

# JS

## Library\_Code.js

**setup** is the first thing that happens. It gets initialized at the bottom of the document via the $(document).ready(setup). It mostly hides elements until the login process has happened. It also sets up the login process, when you click the relevant buttons.

**authOK** is called when you click login. It checks if the login info is equal to what the user entered.

**login** is of course the login process. It sends entered user name and password to the **authOK** and if it returns true then it shows the content of the page, while also hiding the login box. It also displays a welcome message in the header. It also hides the secret login button. If you login successfully then it updates the status with success, if you enter incorrectly it tells you, you failed. It also initialized the **Add\_Series** function which makes most of the page content.

**bypassLogin** is basically the same as **login** but it is a button on the footer, making for easy login. It does the same thing as login but with “Secret User” header and status update. (very useful for development)

const bypassLogin = () => {

    // Directly show the hidden elements without checking credentials

    $("#series\_container").show();

    $("#Add\_Series .access").hide();

    $("header h2").hide();

    $("#secret").hide();

    $("#backup\_footer").show();

    $(".Series\_Input").show();

    // Create the H3 element using jQuery

    let nameStr = $("<h3>").text('Welcome, "SECRET USER" the account handler!');

    $("header").append(nameStr);

    Add\_Series();

    //updated status bar

    updateStatus('Bypassed login successfully. Welcome, "SECRET USER"!');

};

**updateStatus** is the function that every calls to send an update to the status box. The message is displayed for a default time of 3500 Milliseconds (3.5 Sec.) but it can vary. It highlights the box with a light green to make it visible.

**Add\_New\_Series** is the function for adding new series to the page is handled. It gets the textbox’s ID and sets it to a variable (var). It then splits the book title names to an array and sets it to another var while removing useless characters. It then takes the var’s and puts them into an array var called newSeries. It then pushes that into the Library array, and refreshes the page. It then removes the content in the inputs and updates the status.

    const seriesName = document.getElementById('seriesName').value;

    const description = document.getElementById('description').value;

    const author = document.getElementById('author').value;

    const booksInput = document.getElementById('books').value;

    const RoomInput = document.getElementById('room').value;

    const Book\_CaseInput = document.getElementById('bookCase').value;

    const ShelfInput = document.getElementById('shelf').value;

**Add\_Series** is the function that is called when it refreshed the page. It has a forEach that runs on the Library array. It creates the Series box for each series and populates the box with the CRUD buttons, Series name, on hover which displays the series description, and the on click which displays the book list. It has calls the event **handler** when a button is clicked, and it updates the status.

        let S\_Box = $(`

            <div class="SeriesBox card" id="SeriesBox${index}">

                    <button class="read">Read</button>

                    <button class="update">Update</button>

                    <button class="delete">Delete</button>

                    <button class="duplicate">Duplicate</button>

                <h3 class="SeriesName">${Lib.S\_Name}</h3>

                <p class="S\_Description" style="display: none;">${Lib.Description}</p>

                <div class="BooksOwned" style="display: none;">

                    <h4>Books Owned:</h4>

                    <ul class="Books-List"></ul>

                </div>

            </div>

        `);

**handler** is the function that handles what function is called when the CRUD buttons are clicked.

**updateHandler** is one of the CRUD functions. This function was probably the hardes to get working right (I had to completely delete it and start from scratch to get it to work). It hides and displays the Series\_Input class and the Update\_Content ID at the relevant times so the side bar does not have excessive content. It grabs the series that is being updated, and makes the books a string. It the makes the update form and appends it to the side bar, and also puts in the current values of the array. Then if you click “Cancel” it cancels the update and also updates the status. If you click “Save Changes” then it sets the content of the inputs as the new array information, using a similar method to the **Add\_Series** function. It also updates the status and refreshes the display.

Library[index] = {

            S\_Name: newSeriesName,

            Description: newDescription,

            Author: newAuthor,

            Books\_Owned: newBooks,

            Storage\_Location: {

                Room: newRoom,

                Book\_Case: newBookCase,

                Shelf: newShelf

            }

        };

**viewHandler** is also a CRUD function. It adds an info box to the side bar, which currently displays the location of the series.

**removeHandler** is another CRUD function. It splices the library array and removes the series that the “Delete” button was clicked. It then refreshes the display, and updates the status.

**duplicateHandler** is the last of my CRUD functions. It clones the series where the button was clicked, then it splices the Library array and adds the newly cloned item. It then updates the status and refreshes the display.

## Library\_Array.js

This js file has 2 functions. The first is to house the Library array. The second is to return the Library array to as sudo API.

# CSS

## style.css

This file is the most of the CSS code for all of MY RESUME web pages. It is applied globally, but it has the least authority/priority to change the styling. The pages have edited version of some stylings to make them unique and flow better. It also contains most of the “Fancy” styling like sticky content. It houses the @media screen functions, which change the grids and other styling as the screen size changes. It also holds the bounce function which does not affect this project.

## card.css

This houses the card CSS, and like the style.css file it is applied globally to all the files.

# Wireframe

This is the wireframe diagram I started with: 