## Project 3 – The SupeAble Admin Application

Document Revision 3.2

## Expectations

* Instructor Guidance in Class – Low
* Independent Effort – High
* Originality – Low
* Teamwork – *None*

## Overview

Same as Project 2.

## Scenario – the SupeAble Admin Application

As previously noted, our SupeAble solution has two parts: a mobile client used by attendees to report their attendance at an event, and an (administrative) event manager panel that will allow non-technical users to create a new meeting, open / close attendance tracking meetings, and see relevant attendance information.

This project (project 3) will focus on the admin client experience. We are assuming that most users will want to administer events using a desktop / laptop. So, we will use a different template in project 3.

## Disclaimer

As in project 2, this document, and the project itself, will evolve over time.

## Instructions for the Week of Monday 1/30/2023

### Introduction

A good amount of planning has already been done for you. Let’s start this effort by looking at what has already been done.

Investigate the following items (they can be found on the MIS Community Site!)

* The UI Sketch (Discussion items are in **blue**)
* The Desktop Template (This is an example of SPA: Single Page Architecture.)

Together as a Class

1. Download the desktop template, rename the folder to from mis3502\_desktop\_client\_template to “supeableadmin” plus your name, plus some random number.

For example: supeableadmin\_shafer\_912

1. Upload your SupeAble Admin folder to your s3 bucket. Continue your work by making edits through the AWS Toolkit extension of VS Code.
2. Review the contents of the desktop template. Note the similarities and differences from what you say in the project 2 template.
3. Following along with the instructor change the title, logo, and footer on your index.html page.
4. There are a number of tags with the class “card” in div-AAA. You can delete all but one of those.

### On your Own

1. Edit your SupeAble admin solution until it matches the UI Sketch.
2. The creation/management of the tag with the id of container1col is the trickiest part. This is what will be presented at a meeting / seminar and posted on a large overhead screen.

The next page shows what my mock-up code looks like:

|  |
| --- |
| <!-- begin container-1col -->  <div class="container-fluid" id="container-1col" style="display:none;">      <div class="row">          <div class="col-md-12" id="div-display">              <div id="scan1" style="width:45%;float:left" class="text-center">                  <h1>Get the App</h1>                  <!-- intentionally made the "Get the app" QR code a little smaller -->                  <img src="https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=Example" style="width:70%">              </div>              <div style="width:10%;float:left;" >                  &nbsp;              </div>              <div id="scan2" style="width:45%;float:left;" class="text-center">                  <h1>Event Code</h1>                  <img src="https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=Example" style="width:80%">              </div>          </div>      </div>      <div class="row">          <div class="col-md-12" id="div-display">              &nbsp; <!-- Extra white space at bottom of the page -->          </div>      </div>  </div>  <!-- end container-1col --> |

1. Make sure that your copy of the non-functional prototype works. You should be able to navigate from one element of the interface to another. Every button click and every menu choice should direct the user to the correct visual element.
   1. The exception to this is the “Event Data”, “Archive” and “Unarchive” buttons on the manage div.
2. You are ***not*** expected to dynamically integrate the QR code generator.
3. You are ***not*** expected to dynamically add new event cards.

### Do I need to turn in my work?

No. But when we next return to working on Project 3, you will be expected to have a working prototype with clickable buttons / links and no lingering CSS issues or presentation issues.

### What should it look like when I am done?

If you want an idea of what your prototype should look like, please see this short youtube video.

<https://youtu.be/c0CV-0dGq3M> (this video has no audio )

**CONTINUED …**

## Instructions for Week of Monday 2/6/2023

### Announcements:

The SupeAble APi has been updated! It will now prevent two users with the same username from being created!

### Things to do:

1. With assistance from the instructor, students will create a database and tables on their personal database accounts.
   1. Students will learn how to issue these two commands:
      * CREATE DATABASE <<DATABASENAME>>
      * USE <<DATABASENAME>>
   2. Students will also learn how to use the create table wizard in MySQL Workbench
2. Your instructor will also review several of SQL commands currently in use by the existing Web Service. Here are some of them:

|  |  |
| --- | --- |
| 1) | SELECT userid, username, email, sms, usercreatedate, lastlogin  FROM users  WHERE username = ? and password = ? |
| 2) | SELCT count(\*) as qty FROM users WHERE username = ? |
| 3) | INSERT INTO users (username,password,email) VALUES (?,?,?) |
| 4) | SELECT history.\* , eventname  FROM history  JOIN events ON history.eventid = events.eventid  WHERE userid = ?  ORDER BY scandate DESC |

1. Review the spreadsheet where the existing web service features are documented. ***Compare it to the UI Sketch for the SupeAble Admin (Desktop) Interface.***
2. We don’t have all the features we need! What features should we add? When you describe a new feature, be sure to give enough detail so that you can easily:
   1. Remember how the API should be built.
   2. Remember how the API should be used.
   3. To support “a” and “b” above you should at least specify the HTTP method (GET, POST, PUT, PATCH, DELETE) and the specific attribute names.
3. How do I know which HTTP method to use?

| Option # | HTTP Method | When to use it |
| --- | --- | --- |
| 1 | GET | GET requests are “read” operations. They are also safe to perform over and over. They are “safe” because do not change any related data or initiate any costly action. This sort of safety is called “idempotence”.  The HTTP GET method transmits any/all request data in the query string of the URL. The query string part of the URL and it is easily read, and it is also stored in the browser’s cache. Consequently, the HTTP GET method is not used for API requests transmitting any sensitive data.  General Principle – if your database operation is a **SELECT** then your HTTP operation is most likely a **GET**. The exception to this principle would be when the GET operation needs to transmit data securely. |
| 2 | POST | The POST method operations generally create something and may initiate a costly action. POST operations are ***not*** idempotent.  POST operations send any/all related data to the API in the body of the request. This is slightly more secure than sending data in the query string.  General Principle – if your database operation is an **INSERT** then your HTTP operation is most likely a **POST**. |
| 3 | PUT | The PUT method operations are used to replace / redefine something. PUT operations are idempotent.  PUT operations send any/all related data to the API in the body of the request.  General Principle – if your database operation is an **UPDATE** and you are updating every column (other than the primary key) then your HTTP operation is most likely a **PUT**. |
| 4 | PATCH | The PATCH method operations are used to replace / redefine one small part of something. PATCH operations are idempotent.  PATCH operations send any/all related data to the API in the body of the request.  General Principle – if your database operation is an **UPDATE** and you are updating *only one column* then your HTTP operation is most likely a **PATCH**. |
| 5 | DELETE | The DELETE method operations are used to destroy/remove something. DELETE operations are idempotent.  DELETE operations send any/all related data to the API in the body of the request.  General Principle – if your database operation is an **DELETE** then your HTTP operation is most likely a **DELETE**. |

### Do I need to turn in my work?

Yes. You should now have an excel spreadsheet which describes the original four features of the API, plus any new features that you think will be necessary.

Upload your spreadsheet to the corresponding canvas assignment (look under “Project 3”) no later than Monday next week.

Your submission will not be graded but will be considered when assessing your participation in the class.

### Updated instructions for Week of Monday 4/3/2023 and beyond

And … after a long hiatus … we are back! Here is what remains to be done on Project 3.

1. Students must complete the remaining features from the API feature list spreadsheet.
2. Starting next week, your instructor will provide an extensive “start file” for the project 3 UI.
3. Your API features, and the start file, will be integrated with each other in class on 4/10 (or 4/11)
4. ***The project will be due on 4/17 at 11:59 pm.***
5. Your project 3 will be graded using the checklist and rubric below.

### Hot tip!

This query will help you with API feature 8.

|  |
| --- |
| SELECT DISTINCT users.userid, username, email, sms, usercreatedate, lastlogin, eventname  FROM events JOIN history ON events.eventid = history.eventid  JOIN users ON history.userid = users.userid  WHERE createdby = 1 AND events.eventid = 1 ORDER BY username |

### Checklist:

1. Did I create a user in my table with a username of “adam” and a password of “password123” (This will make grading your work a lot easier!)
2. Does my solution work as shown in the instructor-provided video? The latest video is here:  
   <https://youtu.be/3KhnEmQdvqY>
3. Is my client-side code using the server-side endpoint URL that is owned / edited by me?
4. Have I checked for (and corrected) any HTML errors? Your instructor will use the Firefox browser’s view source to check your work. You should do the same ***before*** you turn your work in.
5. Am I showing the content of the responseJSON property of the error object to the user when the web service responds with a 400 status code?
6. Am I showing the message “Unexpected error” to the user when the web service responds with a 500 status code?
7. Have I hidden all input tags that are supposed to be hidden?
8. Have I checked and tested for any HTML layout problems?
9. Does my solution look professional?
10. Do I see activity in my database that corresponds to what the user is doing through the web client?
11. Have I tested the QR codes I am showing the user with my Supeable client (the one I made in Project 2?)
12. Did I use the VS Code Thunder Client to test all 9 features of my web service? Technically, only features 0 through 6 are needed by the current user interface. Still, features 7 and 8 are expected to be functional as they will be used in future semesters!

### Rubric:

|  |  |
| --- | --- |
| Numeric Grade | Description |
| 0 | Student did not turn in work on time. |
| 60 | Student work is not using their own, personal, server-side endpoint URL. This grade is automatic, regardless of any/all other work done on the client. |
| 65 | Student turned in a non-functional solution. It appears that there was an effort to follow along in class, but the solution does not work in one or more significant ways. For the solution to be considered functional, it must have the following core functionality:   1. Log in / Log Out 2. Show a list of events created by the user. 3. Start and event and show the corresponding QR codes    1. The QR code on the left should take you to your Project 2 application.    2. The QR code on the right should change. It is unique for each newly created event. 4. Allow the admin user to create a new event. 5. Flag a current event as archived and / or flag an archived event as current.   The “event data” button as depicted in the original UI sketch, has been dropped from the assignment. |
| 70 | All core functionality is present. Solution works. Found more than two “check list” items to be problematic. |
| 80 | All core functionality is present. Solution works. Found two “check list” items to be problematic. |
| 90 | All core functionality is present. Solution works. Found one “check list” item to be problematic. |
| 100 | No core functionality problems. Solution works. No issues/problems identified. |