# Project 2 – Attendance Tracking Client

Document Revision 3.1

## Expectations

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

## Overview

An Event Management System (EMS) exists to help people organize and conduct meetings of all kinds. Meetings may be classes, lectures, workshops and/or social events. Events may consist of a single meeting, or of many meetings. Multiple meetings are sometimes organized into tracks. A “track” is a succession of meetings with a common theme.

An easy-to-use EMS, provided at low cost or no cost, would be of value to a variety of organizations large and small.

One aspect of such a system would be attendance tracking. This will be our focus this semester.

## Scenario – the SupeAble Client

This semester students will focus on the development of an attendance tracking solution. This effort is only one part of a larger EMS solution (described above) that will be developed over the next 2 years.

The purpose of attendance tracking is … at the barest possible minimum … to determine the number of attendees that were present at a meeting. But beyond that, event managers may to know the history of meeting attendance, the history of an individual attendee’s attendance, and to have some assurance that those persons reporting attendance were, in fact, present.

No solution is foolproof – but our solution should offer precautions/restrictions that are sufficient to provide event managers with confidence that the data collected by the system can be trusted. This will make our solution “suitable” for a wide variety of possible uses. We will call our attendance tracking solution SupeAble ... because SupeAble rhymes with "suitable"!

Our solution will have two parts: a mobile client used by attendees to report their attendance at an event, and an 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 2) will focus on the mobile client experience.

## Disclaimer

This document, and the project itself, will evolve over time. That is the nature of programming and/or software development projects. Requirements are assessed, prototypes are built, the quality of the prototype is evaluated, and then the whole process repeats again and again until you have something called a “minimum viable product” (MVP). The MVP has “just enough” features to be useful.

The MVP is launched and as demand, time and budget dictate, the whole development process starts over.

This iterative process is called the “Agile Methodology” and it can be illustrated as follows:

Shape, arrow

Description automatically generated

[This Photo](https://courses.lumenlearning.com/boundless-business/chapter/an-introduction-to-management/) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

CONTINUED…

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

### Together as a class

A good amount of planning has already been done for you. Let’s start this week’s 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 Pseudo Code (One item is left for you to think about.)
* The Scanner “Proof of Concept” code (will only work on https sites!)
* Sample QR codes
* The Mobile Template (This is an example of SPA: Single Page Architecture.)

More Instructions

1. Download the mobile template, rename the folder to from mis3502\_mobile\_client\_template to “supeableclient” plus your name, plus some random number.

For example: supeableclient\_shafer\_912

1. Upload your SupeAble client to your s3 bucket. Continue your work by making edits through the AWS Toolkit extension of VS Code.
2. Following along with the instructor to create a non-functional prototype. It should be complete enough so that a non-technical user can easily interact with it and envision what the finished product will be like.
3. At a later date, your instructor will also lead the class through the integration of the QR code scanner.

### On your Own

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.
2. You are ***not*** expected to integrate the QR code reader. Instead use a large red button (BS4 class btn-danger) as a placeholder.
3. Update your copy of the pseudo code document. Complete the pseudo code for the “Make Account” feature.

CONTINUED …

### Turn in your work

This week’s work will ***not*** be graded. But … as a form of ***participation*** … you should provide some updates to canvas. This way you can ensure that you are making progress towards the project’s solution, and the instructor can gauge the performance of the class.

* Upload your URL to the “SupeAble URL 1” assignment.
* Upload your Pseudo Code to the “SupeAble Pseudo Code 1” assignment.

You work will ***not*** be graded. But ***documenting your effort*** in this way ***counts towards your participation*** in the class.

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

* First, don’t forget to do your pseudo code work (see step 7!)
* If you want an idea of what your prototype should look like, please see this short youtube video.

<https://youtu.be/9WRWHgUGAFA> (this video has no audio )

**CONTINUED …**

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

### Review Recent Improvements

We’re going to start adding some functionality to the prototype we created in class this week. Before we begin, we should get caught up on all the recent improvements on this project.

* Our “proof of concept” code for the QR code scanner has been improved. It now has a zoom feature. Let’s test it!
* A database schema has been developed. We will review that together.
* A web service has been constructed.
  + <https://misdemo.temple.edu/supeable>

Here are the features of our brand-new web service:

Text, letter

Description automatically generated

* We should test this new web service. We’ll use one of these tools to test our API:
  + <https://apirequest.io>
  + <https://reqbin.com/>
  + Thunder Client (a Visual Studio Code extension )

### Let’s Discuss

* How are features of the API related to the UI Sketch we examined last week?
* Why do some features use the GET method and others use the POST?
* What sort of database operations do you think might be going on behind the scenes?
* Is there one SQL statement for every API feature?

### Together as a class

It all needs to get built, here are some outstanding tasks, in order of priority. All of these are “integration tasks” … we are writing client-side code to integrate out web client with the new web service. ***We probably won’t get through all these items this week***, but this is where we are headed.

1. Enable the login functionality.
2. Enable the “Sign Up” functionality.
3. Integrate the QR code scanner.
4. Validate a QR scan with the web service.
5. Enable the “History” functionality.

### Creating your own server-side code

In class on the week of 3/20, we will complete our ***client-side*** code’s history feature with the inclusion of this function:

|  |
| --- |
| let fixUnixtimestamp = (d) => {      d = d \* 1000;      let newdate = new Date(d);      if (newdate.toLocaleDateString() == "Invalid Date"){          return "Invalid Date";      }      return newdate.toLocaleDateString() + ' ' + newdate.toLocaleTimeString();  } |

Also … in class on the week of 3/20, we will complete the “history” feature of the ***server-side*** code. In doing that, we will:

1. Ensure that each student’s server-side code is referencing the SQL database assigned to them.
2. Point the Project 2 web client to the Project 2 web service that is unique to the student. (This final step will remove all dependency on the server misdemo.temple.edu!)

To do that, we will need these SQL statements:

|  |
| --- |
| -- for code SELECT history.\*, eventname, UNIX\_TIMESTAMP(scandate) as scandate\_epoch FROM history JOIN events ON history.eventid = events.eventid WHERE userid = ? ORDER BY scandate DESC  -- for testing SELECT history.\*, eventname, UNIX\_TIMESTAMP(scandate) as scandate\_epoch FROM history JOIN events ON history.eventid = events.eventid WHERE userid = 1 ORDER BY scandate DESC |

### Wrapping up

We worked through all the above items as a class. It is now your responsibility to bring the project to its conclusion.

#### Some things to do:

* You will need to improve the client-side error trapping a bit.
* You should also consider error scenarios where the Web Service does not respond with success. Those scenarios include 400 status code responses, and the rare / unexpected errors that correspond to 500 status codes.
  + If the Web Service responds with a status code of 400, the web service will provide corresponding error message. You should show that message to the user.
  + If the Web Service responds with a status code of 500, then the user should be notified with an “Unexpected error” message.

In this project students were given a great deal of guidance in class. Very little creativity was expected from any individual student. Consequently, your grade will be determined by seemingly small things. The culmination of all those “small things” suggest the difference between a student who has committed sustained time and effort to creating a working solution, and students who have applied haphazard or last-minute effort.

In a work setting, the culmination of those small things would be the difference between launching a successful / useful solution and a solution that generates unnecessary support requests, dissatisfied customers, and/or possible legal entanglements.

Here is a list of things to check:

1. Does my solution work as shown in this video? <https://youtu.be/55NbL2V2gJE> (No audio!)
2. Is my client-side code using the server-side endpoint URL that is owned / edited by me?
3. 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.
4. Have I checked for (and corrected) any red/critical accessibility problems? Your instructor will use the WAVE web accessibility evaluation tool to inspect your work. This browser extension can be found here: <https://wave.webaim.org/extension/> . You should check your work for any red/critical accessibility problems ***before*** you turn in your work. (**HOT TIP**: There are **two** critical problems in the template itself, and students are expected to find/correct them.)
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. Am I turning the scanner camera off and on as needed? (There are start and stop camera functions provided to help you with this.) What happens if the user reloads the page in the middle of a session?
10. Does my solution look professional?
11. Do I see activity in my database that corresponds to what the user is doing through the web client? For example: if I sign in as user 5 and complete a scan. Do I see scan activity for user 5 appear in the history table?

Here is a rubric for how the project will be graded:

|  |  |
| --- | --- |
| 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. Sign Up 2. Log in / Log Out 3. Scan a QR code (by camera or upload) 4. Show the confirmation message on scan success. 5. Show a “Scan Failed” message on scan failure. 6. Get a list of historic attendance scans |
| 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. |

A final word about creativity / originality. In this project the instructor is \*not\* looking for students to introduce elaborate cosmetic changes to the user interface. (See “Expectations” at the very start of this document.) If you do make such changes, please understand that you will not receive extra credit for your effort.

Also, be advised that such cosmetic changes ***could*** introduce usability and/or accessibility concerns. If they do, then your grade will be lowered accordingly. Similarly, the unnecessary addition of animated images is not welcome, unless it is exceedingly subtle and polished.

The instructor appreciates the spirit of any student who wants to create a fun / novel variation on the project … but such embellishments are not part of ***this*** project’s rubric.

On canvas, you will find an assignment corresponding to Project 2. I expect you to provide me with the URL to your Web Application and the URL to your Web Service API.