SI Attendance Management
Final Report

May 1709

Jon Haut - Team Lead
David Lowry - Webmaster
G. Leo Southwick - PHP Developer
Levi Partridge - Utility Developer
Sam Christy - Python Developer
Xirui Wei - GUI Developer

Client - Iowa State Supplemental Instruction Program
Advisor - Dr. Simanta Mitra

Email - may1709@iastate.edu
Website - may1709.sd.ece.iastate.edu

Table of Contents

1 - Introduction

1.1 - What is Supplemental Instruction?

Supplemental Instruction (SI) is an internationally recognized academic support program offered by Iowa State University. Students who previously excelled in a course can become student leaders and hold study sessions to help students currently in the class find success. SI's main source of income is funding from the university, and in order to receive their funding the organization needs to present data to show that Supplemental Instruction improves students' grades. This is done by collecting SI session attendance data for each student and pairing that with their academic success in the given class.

1.2 - Problem Statement

In the past, student leaders have collected this data by hand or by using ISU's Attendance tracking system (A-Track) This data is then manually copied into an excel spreadsheet on a shared Box account, a very tedious process. A-Track's system provides support for checking students into events using a card reader, however the system is cumbersome, not mobile-friendly, and works much better as a back end system.

1.3 - Goals

The primary focus of our project is to eliminate all of the tedious work student leaders are tasked with so they can focus on providing better sessions for their students. User experience also became a focal point of our work. From start to finish we want the user's experience to be straightforward and intuitive. Part of improving this experience is making sure the site is fully responsive and looks great on the mobile devices that SI has outfitted with card readers.

2 - Requirements

2.1 - Functional Requirements

1. Users accounts should be used to prevent unauthorized use of the application.
    a. Users will be created by Iowa State University
    b. Existing users should be able to log-in from the login page.
    c. User passwords should be hashed before being sent to the server.
    d. Users should be able to log out of their account from any page on the site.
2. The system should get attendance data for specific events using the A-Track API.
3. The system should properly create A-Track events using the API.
    a. Each event's ID, start date, status, and proctor list should also be stored in the database.
        i. An event's initial status should be "Upcoming".
    b. The current user will automatically be made a proctor for the event.
4. The user should be able to remove events from our database if they choose to.
5. The system should be able to check a student into an event in A-Track's system using the API.
    a. The SI department's card swiper app should be able to access this service.
6. The home page should display all events for which the current user is a proctor.
    a. These events should be categorized as "Upcoming", "Completed" or "Processed".
7. The show event page should display individual events
    a. A list of checked-in students should be displayed
    b. A button should select and display a random student from the list
    c. A button should initiate the processing of the event that is being shown
    d. A button should redirect to the edit event page
8. The edit event page should allow the user to edit the name, date, or duration of an event
    a. A button on the edit event page should allow a user to delete that event.
9. The create event form should be accessible from the home page.
    a. The name of an event should be auto-generated using the user's name, class, and the event type.
    b. "Session Type", "Start Date" and "Duration" should be required fields.

    c. The form will not be processed if any of the required fields are missing
10. The app will allow students to check into an event in the following ways.
    a. Using student's Iowa State net ID or email address.
    b. Using student ID number.
    c. Using the SI department's card swipers and iOS devices.
11. Attendance data should be processed and prepared for insertion into the analysis spreadsheet stored on CyBox.
    a. Users should be able to manually process specific events.
    b. Users should receive a notification when an event has finished processing
12. Interface with CyBox to keep Iowa State's SI analysis spreadsheet up to date with student attendance.

2.2 - Non-Functional Requirements

1. Speed up the data transfer
    a. When using our solution, the time to transfer the data from A-Track to CyBox needs to be significantly faster.
2. No installs
    a. SI leaders do not need to install any new programs onto their computers.
    b. SI leaders do not need to download any excel files locally, all handled via the website
3. Security
    a. Due to the sensitivity of the data we will need to be accessing and storing all data in a secure fashion
4. Scalability
    a. The solution needs to work for all different SI courses, not just one
    b. Needs to be able to handle the addition of new SI courses
5. Usability
    a. The user interface is easy to operate and very intuitive even for non-tech savvy SI leaders
    b. Any SI leader can operate the site with minimal assistance

3 - Design and Implementation

3.1 - Laravel

3.1.1 - Overview

The Laravel framework is a Model, View, Controller (MVC) PHP framework we utilized for our project. One of the main features that we used was the customizable package management system, allowing for an extremely configurable application. Anytime we needed a new package installed, we just needed to install it via npm or composer and then the package managers handled all dependencies for us. One of the biggest benefits of this, compared to raw PHP, was in user authentication and database calls, we were able to quickly install both authentication and database models with just one command. The Laravel framework itself is also highly customizable. It comes with plenty of useful services out of the box. Not only are they all configurable, but in most cases there is official documentation on how to configure the service. Since none of us had used Laravel before it was very refreshing that virtually any issue we ran into was a quick Google search away from the solution. Laracasts is the framework's official question and answers page with extensive documentation, and there are also lots of StackOverflow answers and other third party how-to documentation.

3.1.2 - Routing and Middleware

Routing is used for handling the interactions between the Model, View, and Controller. This enables the ability to define actions based on what endpoint the client sends their request to. Route parameters can be defined for routes that get passed directly into the controller action. Laravel's architecture allows for MiddleWare to be applied to requests on any route. Middleware is code that is executed on an HTTP request before it is handled by our application. One example of middleware is checking that route parameters meet certain conditions before handling a request. Our project uses middleware for several different features, mostly security related.

CSRF Token middleware is used on every post request route on our site. The goal of utilizing this feature is to protect against cross-site request forgery attacks. Each form on our site is embedded with a CSRF token unique to the client's session that generated it. The framework keeps track of active CSRF tokens and prevents post requests that it doesn't have a matching token for.

The purpose of the Auth middleware is to block guest clients who send a request to a user only route, then redirect them to the login page. Since our web app requires user information to function every route uses the Auth middleware. This middleware in particular is extremely customizable. In addition to security purposes Auth also acts as a User model that we can use to access the user data stored in our database.

3.1.3 - Illuminate DB

Illuminate is Laravel's database interaction framework. This allows us to perform database actions without writing mySQL scripts, instead we can just call functions on our data models that will generate and run scripts for us. Illuminate also provides database migration classes that make code first migrations simple, which allowed us to keep our local and remote databases in sync. More importantly code first migrations eliminate all the work of making sure our code and database schema are properly aligned.

3.1.4 - Queues

The core purpose of our application is the backend excel sheet manipulation. This action is rather time consuming, however so it is important it can be run in the background and not block the user interface. Our solution is to add these jobs to a processing queue. Laravel provides a Queue framework that works with a number of popular drivers, such as Beanstalkd, IronMQ, Amazon SQS and Redis. However it is unlikely that our app will see the kind of traffic that would justify using a 3rd party service, so we use a jobs table in our database. We then use Supervisor to run worker daemons that take jobs from the queue and handle them. If a job fails it is pushed back onto the queue for processing later. The workers allow us to specify a number of tries that jobs are allowed to fail, after this quota is met they will be moved to the failed jobs. This has been especially useful in development as it allows us to see which jobs and when.

3.1.5 - Event Broadcasting and Notifications

Client to server event broadcasting is an important feature for this application. Because we are queuing the user's jobs they don't have to wait for their session to process, unfortunately that means that they also wouldn't know when the process is finished. To solve this problem we implemented event broadcasting, more commonly referred to outside the Laravel world as websockets. Laravel provides support for implementing websockets through one of several different drivers including Pusher, Redis and socket.io. We chose to use Pusher because it has an excellent free plan that is capped well above the capacity we need it for. For security purposes, all of the broadcasting channels we use are private, meaning they require authentication to listen or broadcast on. Broadcasting is used to instantly update all clients viewing an individual session page when one of them checks in. This means that multiple devices could be used for check-in and they would all remain in-sync.

3.1.6 - Error Logging and Exception Handling

Another great feature of Laravel is the out-of-the-box error and exception handling it provides. The error logging is a lot cleaner than a standard web development debugging process. This built in logging process is using the Monolog library, which allows us to handle logs with a single file or multiple files.

Since this is an MVC framework, Exception handling is much cleaner, handling almost exactly like Java. When an exception is thrown the default action is to not catch it and spit the stack trace out to the user, however, it is easy to catch the exception and perform a smooth redirect. We found this to be the most effective when users were trying to access pages they do not have access to while logged in, now we simply catch the exception and redirect them to the login page.

3.1.7 - Scheduled Tasks

The reason we need a scheduled task has to do with Box's api. There are two supported authentication endpoints for the API, either JWT or OAuth. JWT is much better for machine-to-machine use cases like ours. However, at the time we started this project JWT was an enterprise only feature, so we were forced to use OAuth. This endpoint accepts a refresh token that is good for 60 days and returns an access token that lasts 1 hour along with a new refresh token. The access token can then be used to connect to Box's API. If our application goes for over a month without requesting a new token, the refresh token we have will expire and our site will break. To get around this we used Laravel's command scheduler which uses cronos to run user defined commands on a regular interval.

3.2 - Shibboleth

We use Shibboleth as authentication for our site. This allows us to use Iowa State netID's and password to login to the site. This also makes the site more secure. In order to configure this we followed Iowa State's documentation on configuring Shibboleth. Little did we know, the documentation was less than 50% complete. We used the official, generic Shibboleth documentation combined with extensive research for all server configurations. Essentially we needed to configure our .htaccess file to force Laravel to ignore any Shibboleth related URL, strictly use https, and then handle the SAML2 response data at the correct location because SAML uses URI to decode the encrypted data. Once we finally had the decrypted data we then had to handle integrate into our existing user database.

This process involved lots, and lots of trial and error and debugging. First, we Shibboleth told us our site was not secure, this was fixed by requesting proper certs. Then, our browser was crashing because of too many redirects. This was our biggest hurdle due to the complexity of Apache's mod_rewrite rules. After this was resolved, we then were strictly receiving encrypted data because we were handling the data at an unexpected endpoint. Once we were able to receive decrypted SAML response, we then got the infinite redirects AGAIN. This time we were able to solve by changing which directory on the server that Shibboleth was protecting. After all of this, we were back in friendly Laravel land, and able to configure routes for handling the response data and creating new users if needed, or logging users in if they already exist in our database. The final issue was that the switch to https prevented us from accessing the Toastr library. We were able to fix this by using secure CDN's (Content Distribution Network).

3.3 - Python

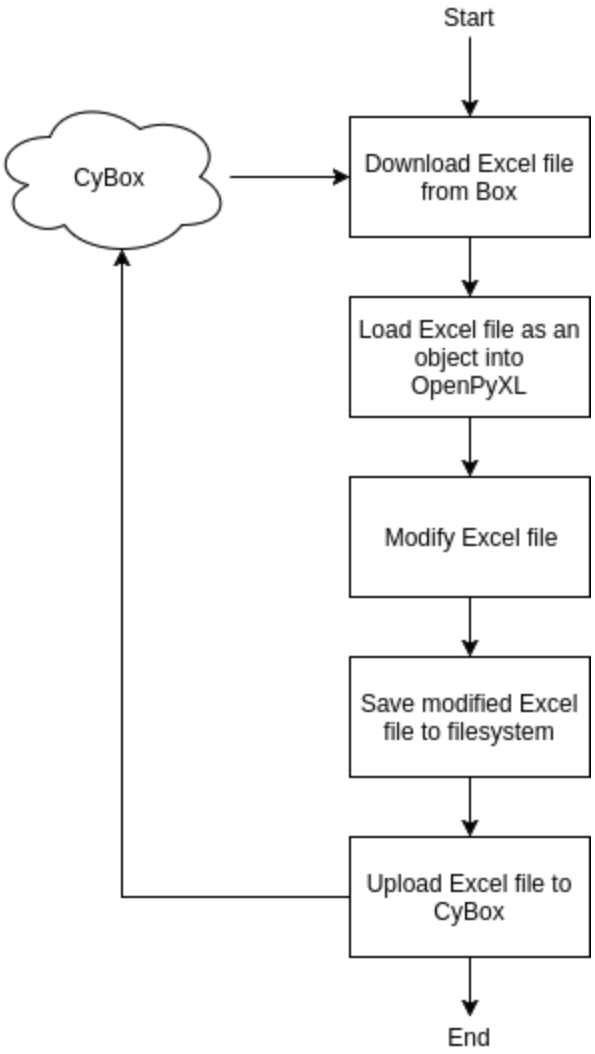
3.3.1 - CyBox API

CyBox is a service provided by ISU that uses the Box.com framework. This is the service that our client uses to store and download the Excel files and we access it programmatically through the CyBox API. The API allows us to follow symbolic links and download the file we need given the student ID of the event creator. We modify the file and upload it back to CyBox. For authentication, The API requires a Client ID and Client Secret which are static, as well as an access token which expires every hour. The expired token can be renewed with a refresh token which expires after 90 days.

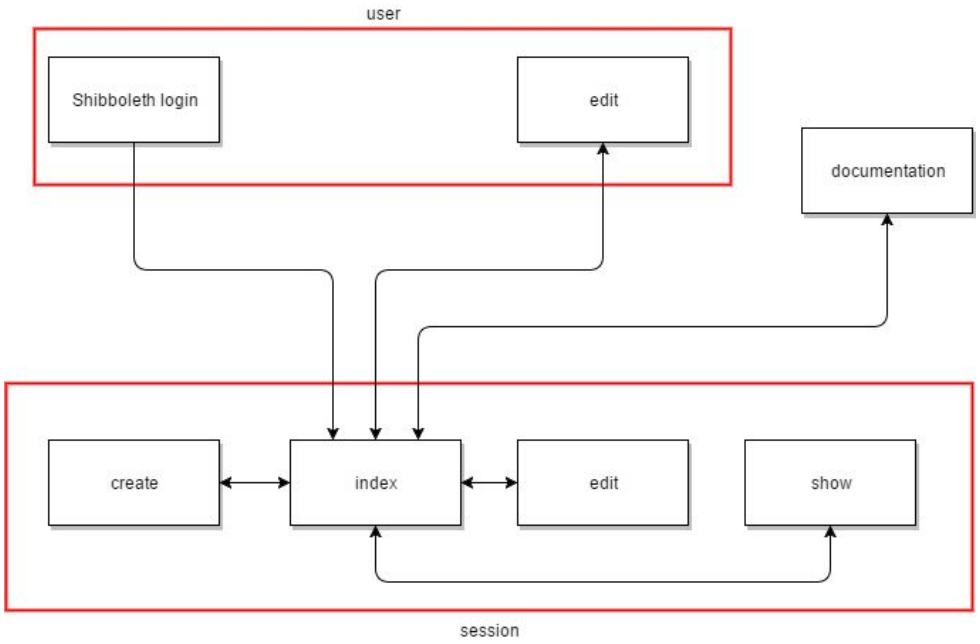
3.3.2 - OpenPyXL

OpenPyXL is a solution for modifying the Excel file once they are downloaded to the server. OpenPyXL allows us to query the information in each of the cells and perform operations like writing to a cell and changing the font of the cell.

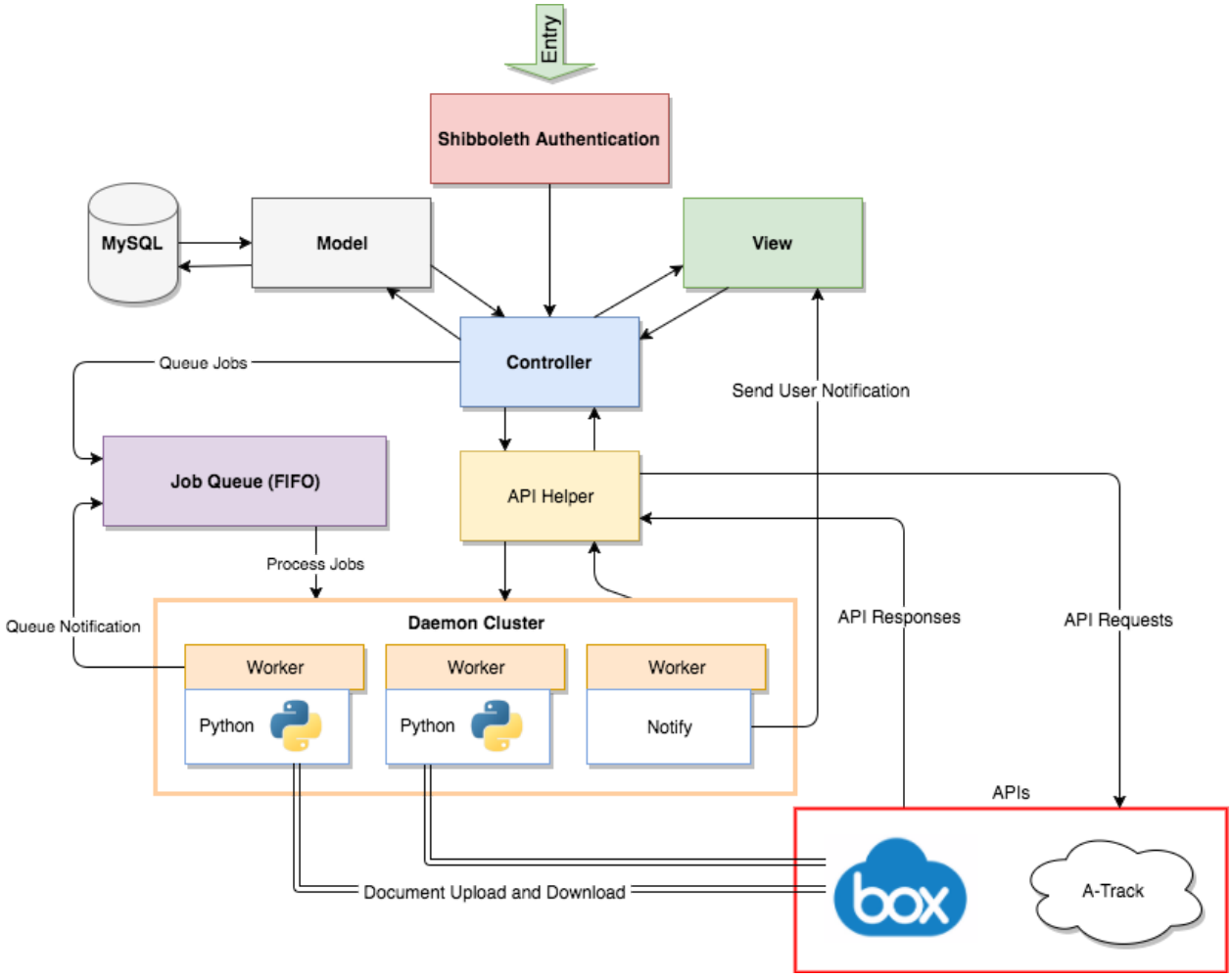
3.4 - Diagrams



This diagram shows the flow of the python process



This diagram shows the navigation flow of the web application.



This diagram shows the Interaction between components of the web application.

4 - Testing

4.1 - Testing Plan

When developing our testing plan, we knew we wanted the site to be tested by current SI Student leaders. We were able to work with a small subset of the current SI Student Leaders known as the Mentors. These were veteran members of the program and the leaders who set the example for the others to follow. This group consisted of eight Student Leaders. The testing phase consisted of four 2-week iterations. We met with our testers every two weeks to demonstrate and newly added features, and to listen to their feedback on the functionality of the site. During these weeks, the live website was set to use our Test database schema. This ensured that the test data was not interrupted in any way by any of the dummy data we were using to develop the site.

4.2 - Results

As expected, there were quite a number of bugs discovered while testing. Some of these include:

- Being unable to login to the application
    - Cause: Not all user data was migrated from the development environment to the test environment.
    - Solution: Migrated the correct data into the test environment
- Cell formulas within the Excel file stopped working
    - Cause: Data inserted into the files from the Python algorithm was in a text format
    - Solution: The Python algorithm was edited to ensure numbers inserted were in a “Number” format
- Processing a session on mobile device caused a crash
    - Cause: Unknown, believed to be caused by the Swipr iOS application
    - Solution: Bug can’t be replicated and was a one-time occurrence. We’ve considered it irrelevant.

Our testers provided multiple ideas for new features or changes to currently features. We were able to implement some of this features, but others proved to be outside the scope of this project. Some examples include:

- Creating a button on the show event page that will select and display a random student that has been checked in to the event.
- Creating a separate page that displays some of the calculated data from the destination excel file.
- Creating a separate program to insert student ID numbers into the excel files, to be run at the end of the semester.

5 - Conclusion

We consider the project to be a great success. Starting this coming semester, F17, all SI Student Leaders will be utilizing the new SI Attendance Management web app. This will allow the SI Program administrators to restrict access to the sensitive student data located in their excel spreadsheets on CyBox. The SI Program administrators have been trained in using the site, and have been provided the documentation they need to train next year’s Student Leaders. Our client is very excited about the completion of the project and is very pleased that we have exceeded all of their expectations. Support of this web application will hopefully be transferred over to the Iowa State Web Dev team. All members of our team have gained valuable

experiences through our design process, and take great pride of the work we have accomplished.

6 - Appendices

6.1 - User Manual

http://may1709.sd.ece.iastate.edu/uploads/finalReports/Si_AttendanceManagmentUserManual.pdf

6.2 - Failed Designs

1. Trying to use a tablesorter library
    a. It initially worked but once we moved to having the datetimepicker and the tablesorter in the same file we ended up getting a jquery conflict preventing us from using one or the other. Ultimately we chose to use the date picker because when we switched to the new vue.js dynamic tables, we weren't able to sort the divs anyway
2. Processing with exceljs
    a. Initially we tried processing events through the node package exceljs. When we first started using it we could read and write simple Excel files which seemed like good progress. After further testing with the actual Excel files we would be dealing with we realized it had a major flaw. We couldn't write more complex cells like formulas and we couldn't format any of the cells with the proper styles. After realizing these problems we switched to using openpyxl the python library because it gave us this functionality we needed to keep from breaking the current formatting of the Excel files.