

– Independent Work Report, Spring 2023 –

# **Creating an Application for Improved Communication During Tournaments for the Eastern Collegiate Taekwondo Conference (ECTC)**

Joseph Lou

Advisor: Robert Dondero

May 1, 2023

## **Abstract**

*The Eastern Collegiate Taekwondo Conference (ECTC) hosts a few taekwondo tournaments a year for students of college universities in the Eastern United States. At these tournaments, there is a need to send timely notifications to participants, such as when they need to get ready to compete. My project aimed to create a system that would make it easier for administrators to send the notifications and make the notifications themselves a better experience for the recipients. I created the Tournament Notification System (TNS), which ended up being used in an actual ECTC tournament at the end of the semester.*

This paper represents my own work in accordance with University guidelines.

Joseph Lou

## Table of Contents

<b>1. Introduction</b>	<b>3</b>
1.1 Background	3
1.2 Motivation	4
1.3 Prior Work	5
<b>2. Approach</b>	<b>5</b>
2.1 Plan	5
2.2 The System	7
2.2.1 Authentication and Authorization	7
2.2.2 Manage Admins	8
2.2.3 Global Settings	9
2.2.4 Admin Settings	11
2.2.5 Notifications	14
2.2.6 Matches Status	21
2.2.7 Sent Emails	22
2.2.8 Subscriptions	22
<b>3. Implementation</b>	<b>24</b>
3.1 Database Schema	24
3.1.1 Global State	24
3.1.2 Admins	25
3.1.3 Roster Tables	25
3.1.4 Matches Status	27
3.1.5 Emails Sent	27
3.1.6 Team Subscriptions	28
3.2 Reading from Google Spreadsheets	28
3.3 Sending Notifications	30
3.4 Mailchimp	30
3.4.1 Templates and Campaigns	30
3.4.2 Custom Email Content	33
3.4.3 Sending Batch Notifications	34
3.4.4 Batch Operations	35
3.5 AJAX	35
3.6 Frontend	36
3.6.1 Responsiveness	36
3.6.2 Success and Error Messages	36
3.7 Code	37
3.7.1 Organization	37

3.7.2 Statistics	39
3.8 Deployment	39
<b>4. Evaluation</b>	<b>40</b>
4.1 Formal Evaluations with Users	40
4.2 Real Use at the 2023 UVM Tournament	42
4.2.1 Training Session	43
4.2.2 Scaling Issue	43
4.2.3 Campaigns and Segments Issue	44
4.2.4 Survey	46
<b>5. Conclusion</b>	<b>47</b>
5.1 Future Work	47
5.2 Lessons Learned	48
5.3 Project Self-Assessment	49
<b>Appendices</b>	<b>50</b>
A. Service Accounts	50
B. Roster Spreadsheet Specifications	50
C. Subject Placeholders	51
D. Conversation with Mailchimp Support	53
E. Evaluation Notes	58

# 1. Introduction

## 1.1 Background

To better understand the motivation for the system and all the relevant parts of what the project aimed to achieve, we will need to first discuss the background for the problem.

Taekwondo is a Korean martial art. The Eastern Collegiate Taekwondo Conference (ECTC) (<https://www.ectc-online.org/about-us>) is a collegiate taekwondo league that hosts five Olympic-style tournaments each academic year at various universities in the Eastern United States, including a tournament at Princeton, typically in the spring (this year's was on Sunday, February 26, 2023). At these tournaments, athletes from many colleges and universities come together to compete in two events called poomsae (forms) and kyorugi (sparring). Poomsae has three divisions split between belt levels (PA, PB, PC), and sparring has six divisions split between belt levels and gender (Men's [ABC], Women's [ABC]), for a total of nine divisions. Each division uses a single-elimination bracket (Princeton COS Professor Matt Weinberg, a long-time member of the league, helped develop the bracket generation algorithm) between teams of at most three members. Each match consists of the members of both teams facing off one-by-one against someone from the opposing team, and the team with the most wins (best of 3) advances<sup>1</sup>. Each match takes place at a ring, the number of which varies depending on the amount of space the tournament venue has and the number of competitors (more competitors means more teams, which means the tournament will need to go faster to end on time or at least get through the majority of the matches, so more rings are needed). To ensure a smooth process of getting the proper competitors for each match to the proper ring and assigning matches to

---

<sup>1</sup> I am omitting some rules here, such as sparring teams also being restricted by weight classes, since they are not relevant for the paper. If you are interested, you can visit <https://www.ectc-online.org/rules> to see the official rules of ECTC tournaments.

rings whenever one becomes available, the teams for matches are first called to a holding area, where a representative will find all the team members for a match and then lead them to their assigned ring to compete. The most important and most frequent notification is therefore the one that calls specific match numbers to holding.

The ECTC currently uses a collection of Google Spreadsheets to aid them during tournaments. A registration spreadsheet is sent out to each school that is participating at a tournament, where they will list all their competing athletes and how their teams among the nine divisions are composed. Teams have names in the format “<School> <Division><Team Number>”. For example, Princeton may have two teams of black belts competing in poomsae, which would be teams Princeton PA1 and Princeton PA2, and two teams of green belts competing in sparring, which would be teams Men’s B1 and Women’s B1. After registration, all the relevant information is imported into a main spreadsheet called the Tournament Management System (TMS), which has an impressive use of formulas and Google Apps Script integrations to almost work like a live updating webpage. During the day of the tournament, members of the Tournament Committee (TC) use the TMS spreadsheet to keep the tournament running, assign matches to rings, save results, and update the brackets in real time.

## **1.2 Motivation**

The ECTC sends emails for notifications. They used to use Google Groups, but due to limitations on how many contacts can be added per day, they switched to an email service called Mailchimp (<https://mailchimp.com/>). Because so many notification emails need to be sent, the Communications Manager (who is in charge of sending the notifications) simply sends all emails to everyone at the tournament, with a subject like “Matches XXX-XXX: Report to Holding”. Receiving the hundreds of emails in my inbox every tournament despite only being on a single

team, I wondered if there could be a way to send targeted email notifications to only the relevant people. If I have to compete in Match 920, for example, then Matches 901-919 being called to holding doesn't really pertain to me. This was the basis for my project idea.

### **1.3 Prior Work**

Although there do exist third-party tournament management systems, since the ECTC already had a dedicated and working workflow, I thought that integrating into it would be the best way to incrementally improve some tedious parts of the process. In particular, sending emails on Mailchimp isn't super straightforward, involving steps such as setting the content from a template (which goes through a few screens), setting the recipients, setting the subject, and then triggering a send. The subject for each email only changes the match numbers, but otherwise the email content is the same (a generic template that asks the intended recipients to report to holding). It is possible to define a limited set of recipients for an email, but it does require a bit of work as well, so it makes it simpler to just send to everybody. Whenever a new batch of matches can be called to holding to be assigned a ring and compete, the Communications Manager must go through all these steps to send out the notification. This was the particular point that I wanted to focus on for at least the first parts of my project. If I had time, there were also some other ideas floating around that I was willing to tackle.

## **2. Approach**

### **2.1 Plan**

I started by contacting members from the TC, William Estey, Carissa Fu, and Tara Sarathi, to gauge interest in a project like this. Being tired of the manual work, they were immediately interested, and we set up weekly calls to stay updated on the progress of my work. Their input on

what was and was not needed or wanted was very helpful in narrowing down the scope of the project and figuring out what exactly I would work on to be most helpful to the actual users.

In the end, we decided on a project that would focus on simplifying the process of sending notifications, with more features added on if time permitted. (The users had many ideas of additional features they wanted, so running out of things to do wasn't an issue.) The benefits of the system would include:

1. Being able to send personalized notifications to only the relevant people for each match
2. Using less of the Mailchimp email quota<sup>2</sup>
3. Making the workflow of sending notifications more efficient

In order to properly send notifications to certain people, I needed to know all the information about each team. Since we have a single-elimination bracket, the teams competing in later matches aren't clear until previous matches have finished and the winner advances. Thus, I knew I needed a design that would be able to fetch the updated competing teams for a given match number. Once I knew the team names, I also needed a way to find out the athletes that belong to each team, specifically their email addresses, and finally I could send a personalized email to just the people in that match.

The TMS spreadsheet conveniently had all the information I needed. My users and I agreed on a roster format, and they would create a roster worksheet that I could query to get the full roster, including all the schools, user emails, and teams. There was also another worksheet in the TMS that updated in real time with the competing team names for each match based on the results from previous matches in the bracket, which is exactly what I needed. I was also able to gain access to their existing Mailchimp account, where I created an API key to connect with the

---

<sup>2</sup> The Mailchimp account has a quota of sent emails. Every email sent to a single email address counts as a distinct email sent, even if all the emails were sent at once with the same content body. Therefore, personalizing the recipients would use less of this email quota.

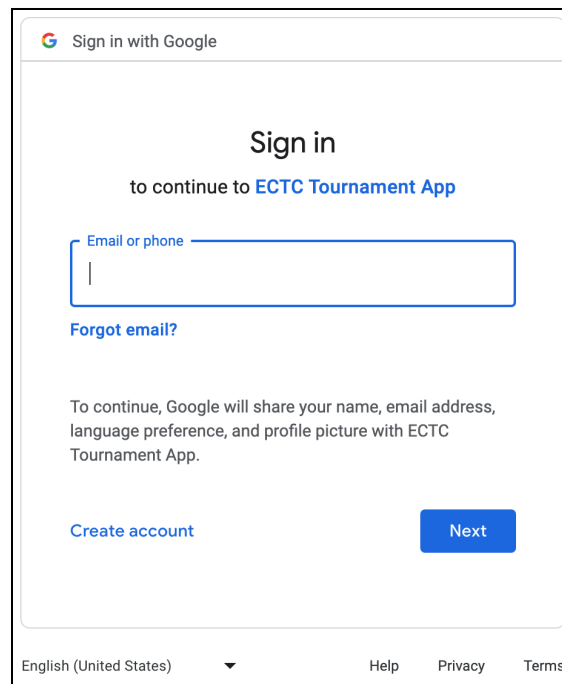
Mailchimp Marketing API. Using some method of accessing data from a Google Spreadsheet, I would then have all the tools I needed to achieve the goal of personalized notifications.

## 2.2 The System

In this section, I will walk through very generally how to use the system, as if the reader were an intended user of the system that only has the base knowledge of tournaments as described in the [1.1 Background](#) section. Each subsection (with exception of the first subsection) will focus on a single page of the application.

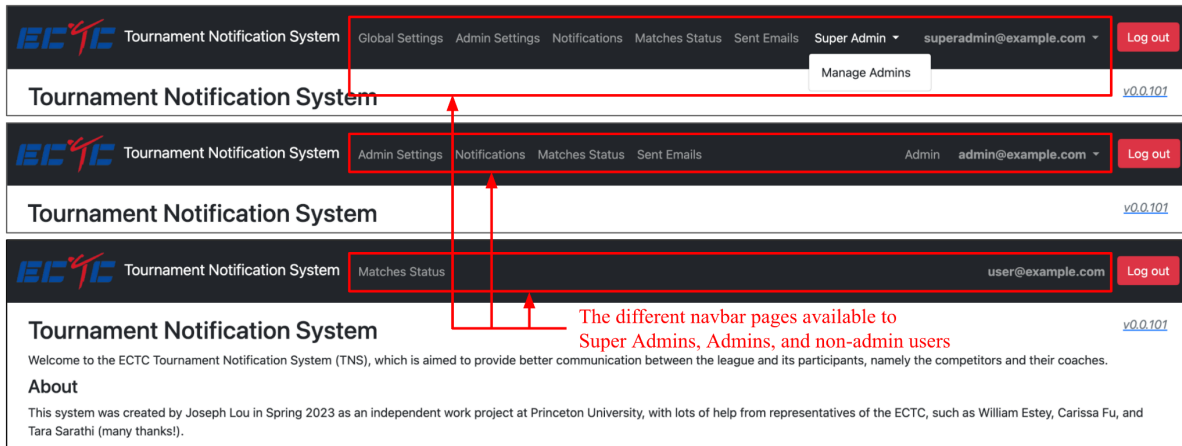
### 2.2.1 Authentication and Authorization

To authenticate users, I use Sign in with Google (see [Figure 1](#)). For authorization, since sending notification is entirely an admin task, I check that the logged in user is a Super Admin or an Admin for the site. If so, they will be able to access all the relevant pages of the application. Otherwise, the user can still view the Matches Status page, which is public. See [Figure 2](#) for a comparison of the available navbar pages for each type of user.



**Figure 1: The Google authentication prompt that users see.**

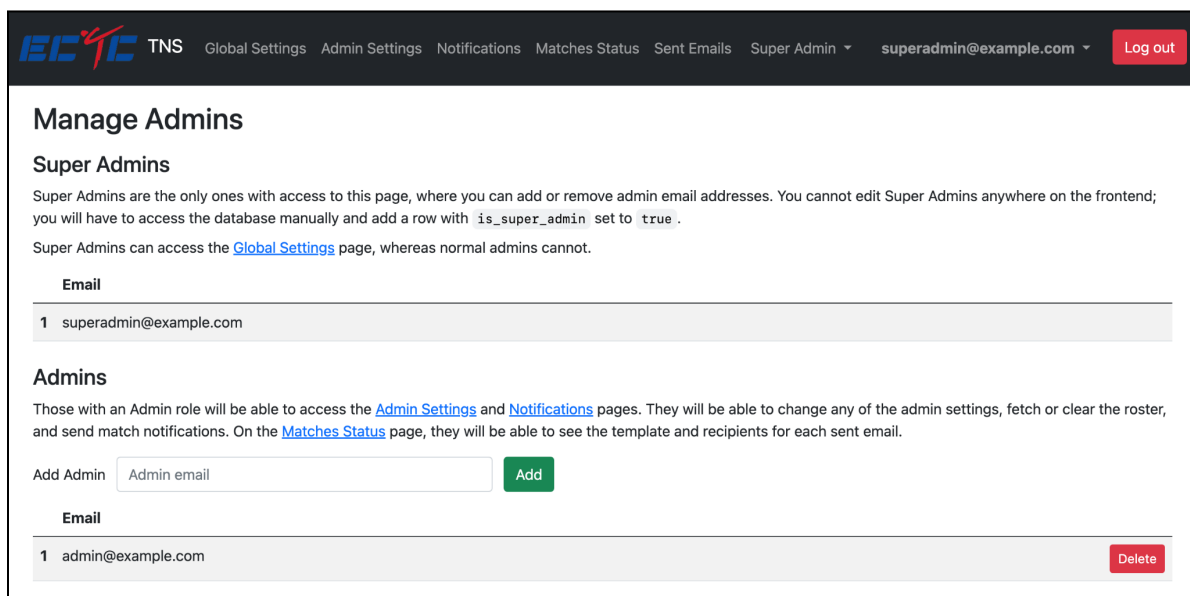




**Figure 2: The home page as a Super Admin, an Admin, and a non-admin user, respectively.**

### 2.2.2 Manage Admins

A Super Admin user is able to add or remove Admin users through the Manage Admins page (see [Figure 3](#)). To avoid the headache of recursive higher levels of admin permissions to add other admins, there is no way to add Super Admins on the frontend, and the first Super Admin was seeded as the ECTC's main email address. To edit Super Admins, one would have to manually update rows in the production database. I assumed this would not be a common task, and my users agreed that this seemed reasonable.



**Figure 3: The Manage Admins page.**

### 2.2.3 Global Settings

A Super Admin user also handles the one-time global settings that only a Super Admin would have access to or would need to set. In particular, in order to access data from a Google Spreadsheet, I am using a Google service account, which the user must provide in the form of a credentials JSON file<sup>3</sup>, and to be able to send emails through the Mailchimp account, I need a Mailchimp API key. These values are unlikely to ever change throughout the lifespan of the application. Even if they did change, only the Super Admin would be expected to be able to reset them; regular Admins should not need to worry about these higher-level settings.

On the Global Settings page, a Super Admin may input the required information (see [Figure 4](#)). The rest of the application will not work without these settings set, as demonstrated in [Figure 5](#). Since regular Admins cannot access the Global Settings page, they will instead be able

**Service Account**

The service account is used to access Google Spreadsheets. Either make your spreadsheet publicly accessible or share the spreadsheet with the service account email with at least view permissions.

Current service account: *None*

**Change Service Account**

Please see [this page](#) for how to create a service account. Upload the credentials JSON file here.

Choose File No file chosen

**Mailchimp**

**API Key**

We use Mailchimp to send email notifications. To allow the system to access your Mailchimp account, we will need an API key. Please see [this article](#) for information about them. API keys must be secure. Because of this, once you provide your API key, you won't be able to see it on this site again (similar to when you generated the API key), so be sure to store it somewhere else. You can always clear or set another API key in case the one you provided is no longer working.

You currently do not have an API key set.

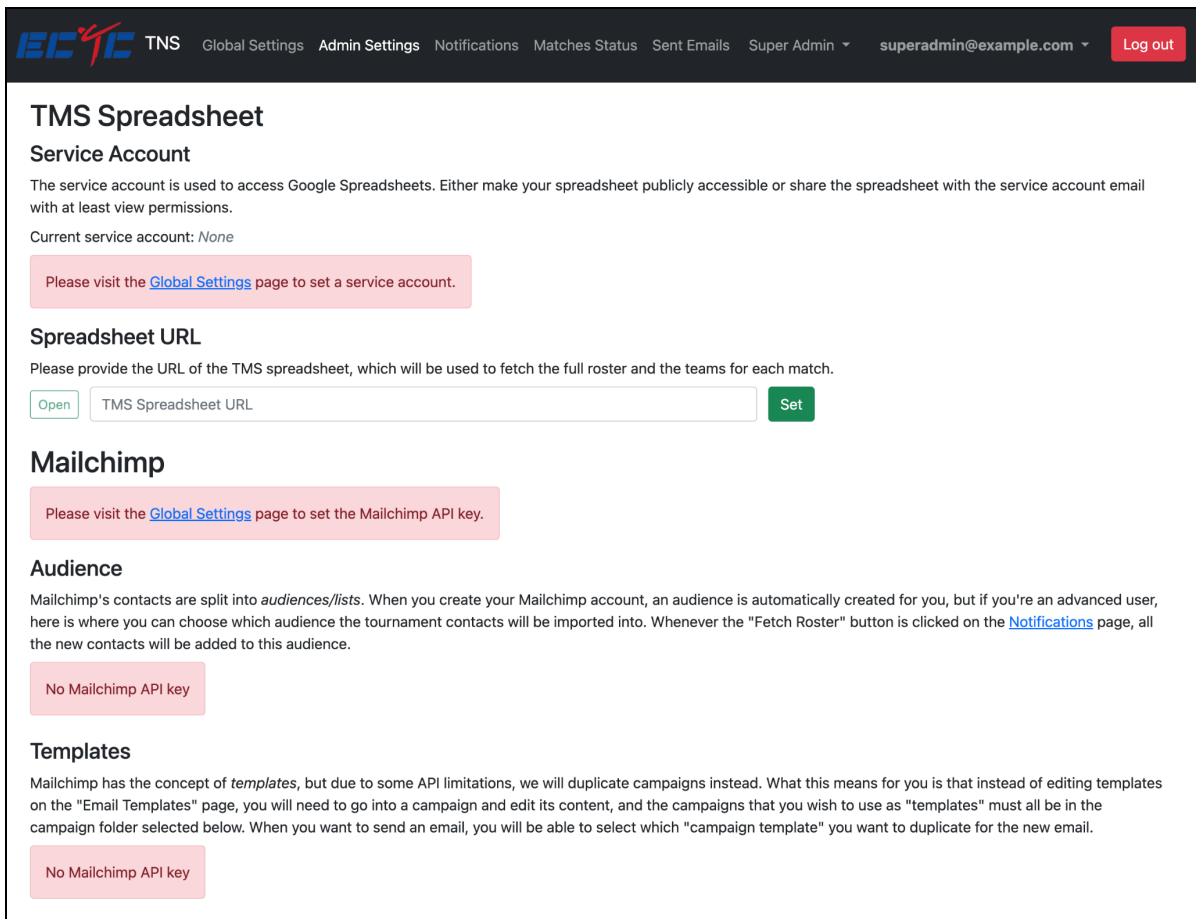
Set Mailchimp API Key

**Danger Zone**

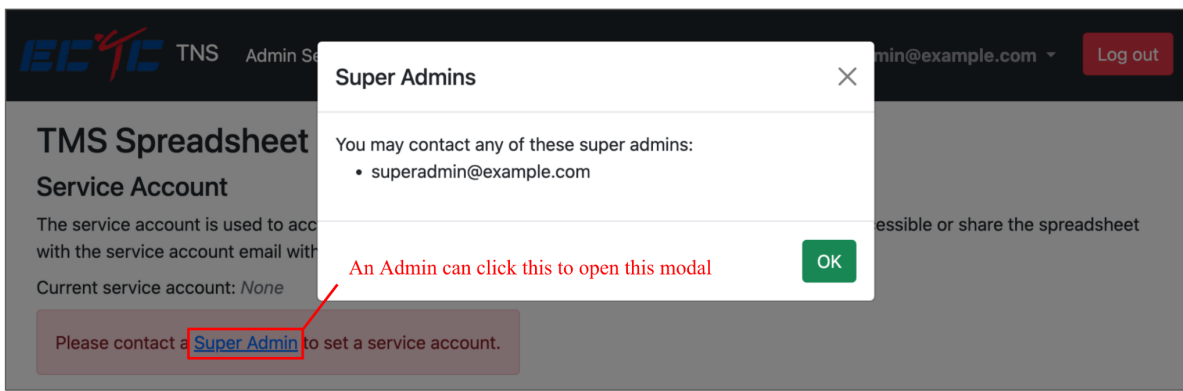
As a Super Admin, you have the ability to clear all the data currently saved in the databases, which includes the TMS spreadsheet url, the selected Mailchimp audience and template folder, the Mailchimp audience tag, the full roster, and all saved sent emails. (To clear the service account and Mailchimp API key, see the sections above.)

**Figure 4: The Global Settings page.**

<sup>3</sup> See [Appendix A](#) for more information about service accounts and how to create them.



**Figure 5: The Admin Settings page (for Super Admins) when the Global Settings are not set.**



**Figure 6: The Admin Settings (for Admins) when the Global Settings are not set.**

to see a modal popup that lists the Super Admins they can contact if there are any missing global settings, as demonstrated in [Figure 6](#).

In addition, a Super Admin may clear all the currently saved data. This should be done before each tournament so that the system can start with a clean slate. The global settings will not be cleared because those are one-time settings that would likely be used throughout the lifespan of the system, so the user should manually clear those if they wish to reset them.

A Super Admin also has the option of clearing all the user subscriptions to team notifications (see section [2.2.8 Subscriptions](#)). This gives the Super Admin the ability to clear all the database values from this page.

#### **2.2.4 Admin Settings**

An Admin user (which includes Super Admin users) may access the pages relevant to sending notifications. In the Admin Settings page, they may configure the system for a tournament (see [Figure 7](#)). This includes the link to the TMS spreadsheet, so the system can fetch the required information, and some Mailchimp settings for sending emails.

On Mailchimp, an “audience” (or “list”) is a collection of contact emails. Each contact may have optional “tags”, which the TC has been using to organize contacts by tournament (e.g., athletes who competed at the Princeton Tournament were tagged with “2023 Princeton Tournament”). In order to send an email, one must create a “campaign”, which can be sent to an entire audience or to the contacts tagged with a specific “segment” (another name for tags). The content of the campaign, which recipients will see as the body of the received email, can be populated by “templates”, which can be created by the user. Templates may be added to “template folders”, and sent or draft campaigns may be added to “campaign folders”.

EETC TNS
 Admin Settings
Notifications
Matches Status
Sent Emails
Admin
admin@example.com ▾
Log out

## TMS Spreadsheet

### Service Account

The service account is used to access Google Spreadsheets. Either make your spreadsheet publicly accessible or share the spreadsheet with the service account email with at least view permissions.

Current service account: **serviceaccount@example.com** Copy

### Spreadsheet URL

Please provide the URL of the TMS spreadsheet, which will be used to fetch the full roster and the teams for each match.

Open

Set

## Mailchimp

### Audience

Mailchimp's contacts are split into *audiences/lists*. When you create your Mailchimp account, an audience is automatically created for you, but if you're an advanced user, here is where you can choose which audience the tournament contacts will be imported into. Whenever the "Fetch Roster" button is clicked on the [Notifications](#) page, all the new contacts will be added to this audience.

**Audience Name**

---

**From Name** -

---

**From Email** -

---

**Default Subject** -

---

**Members** -

---

**Last email sent** -

---

For ease of organization, you can have all the fetched contacts from the TMS spreadsheet be tagged with this optional label when they are imported into Mailchimp.

Audience Tag  Set

### Templates

Mailchimp has the concept of *templates*, but due to some API limitations, we will duplicate campaigns instead. What this means for you is that instead of editing templates on the "Email Templates" page, you will need to go into a campaign and edit its content, and the campaigns that you wish to use as "templates" must all be in the campaign folder selected below. When you want to send an email, you will be able to select which "campaign template" you want to duplicate for the new email.

Folder Name

**Figure 7: The Admin Settings page.**

The current flow for the Communications Manager is to create an email template before the tournament (if one doesn't already exist), then create campaigns using that template during the tournament, sending to all the contacts tagged with the current tournament. Since contacts that went to previous tournaments but possibly not the current tournament are in the audience, filtering the recipients using the tournament tag is necessary to avoid unnecessarily spamming people.

It would be ideal for me to emulate this flow as best as I can, cutting out the repetitive tedium of having to keep creating campaigns for every notification. In addition, by automating tasks with the API, I could easily create a segment that contained only the relevant recipients for a match notification and send a campaign to only that segment (thereby not sending the email to people who don't need to receive it).

However, the Mailchimp API had a bug relating to user-defined templates, and I was unable to create a campaign and set its content to be a user-defined template. I still decided to name everything on the Admin Settings page as if we were using actual templates, but the requirements are a little different. The ideal user flow would be:

1. The user creates one template for each email body content they want to use. Each template should be descriptively named so the user can know which one they want to select from a dropdown.
2. The user puts all these templates in a single template folder.
3. The user selects the desired template folder on the Admin Settings page, so that the system can fetch all the relevant templates to use for the notification emails. When the emails are sent, the selected template will be used as the body content of the email.

Instead, the workaround<sup>4</sup> I went with changed this flow to be:

---

<sup>4</sup> See section [3.4.1 Templates and Campaigns](#) for more technical details and challenges on this workaround.

1. The user creates one draft campaign for each email body content they want to use, with the content already set. Each campaign should be descriptively named so the user can know which one they want to select from a dropdown.
2. The user puts all these campaigns in a single campaign folder.
3. The user selects the desired campaign folder on the Admin Settings page, so that the system can fetch all the relevant “campaign templates” to use for the notification emails. When the emails are sent, the selected “campaign template” will be replicated, which will result in the newly created campaign having the desired body content.

The Mailchimp settings on the Admin Settings page allow an Admin user to select the audience they wish to use, specify an optional tag name for all the imported contacts, and select a template folder (which is actually a campaign folder for the workaround) to use for the notification email campaign templates. When the teams roster is fetched from the TMS spreadsheet, all the emails will be added to the selected Mailchimp audience and tagged with the specified tag name, if it was given. When sending a notification, the user will be able to choose a template (actually a draft campaign) from the selected template folder (actually a campaign folder) to use for the body of the email.

### **2.2.5 Notifications**

The Notifications page will not work without all the Admin Settings being set, since all those settings are required for the page to work.

The first step for an Admin user on the Notifications page would be to fetch the roster. Before the roster is fetched, the page also will not work (meaning all inputs relating to sending notifications will be disabled; see [Figure 8](#) and [Figure 9](#)), since it would not know the team members for each team, and thus it wouldn't know who to send emails to.

EFC TNS
 
[Admin Settings](#)
[Notifications](#)
[Matches Status](#)
[Sent Emails](#)
[Admin](#)
admin@example.com
Log out

## Notifications

### Teams Roster

The roster in the TMS spreadsheet will be fetched whenever the "Fetch Roster" button is pressed. There should be a worksheet called `FULL ROSTER` with the full teams roster. For full specifications (including what else the button does), see help [here](#).

Fetch Roster

Last fetched: *never*

### Send Match Notifications

**Roster has not yet been fetched**  
Please fetch the roster before you can send notifications.

Enter the match numbers below that you want to send notifications for. The matches will keep being added to the matches queue, which you can edit as needed, until the "Send Notification" button is pressed. To restore the state of the last queue, click the "Last Query" button.

The TMS spreadsheet should have a worksheet called `Communications` which will be queried to find the competing teams for each match.

Add Matches 

Fetch Matches Info

The match numbers should be separated by spaces or commas. You may use dashes for match number ranges.

Match	Division	Round	Status	Blue Team	Red Team	Valid	
No matches							

Template *Cannot fetch templates*

Subject

You must include the text `{match}`, which will be automatically replaced with the match numbers above for each email. For the full placeholder list, subject specifications, see help [here](#).

Also Send To

- School Coaches
- School Spectators
- Team Subscribers

Note that if there are multiple matches with the same school or team being sent, these people will receive multiple notifications as well.

Send Match Notification

### Send Blast Notifications

You can send a blast notification to either the entire audience or to a specific division.

Template *Cannot fetch templates*

Subject

Recipients Entire audience *(No divisions)*

Send Blast Notification

**Figure 8: The Notifications page before the roster is fetched.**



**ECFC** TNS Admin Settings Notifications Matches Status Sent Emails Admin admin@example.com Log out

## Notifications

### Teams Roster

The roster in the TMS spreadsheet will be fetched whenever the "Fetch Roster" button is pressed. There should be a worksheet called **FULL ROSTER** with the full teams roster. For full specifications (including what else the button does), see help [here](#).

You can view the full roster here: [Full Roster](#)

**Fetch Roster** **Warning:** Fetching again will override the current roster.

**Clear Roster**

Last fetched: 2023-04-29 18:03:28 ([logs](#))

### Send Match Notifications

Enter the match numbers below that you want to send notifications for. The matches will keep being added to the matches queue, which you can edit as needed, until the "Send Notification" button is pressed. To restore the state of the last queue, click the "Last Query" button.

The TMS spreadsheet should have a worksheet called **Communications** which will be queried to find the competing teams for each match.

Add Matches  **Fetch Matches Info**

The match numbers should be separated by spaces or commas. You may use dashes for match number ranges.

Match	Division	Round	Status	Blue Team	Red Team	Valid	Remove All
No matches							

Template

Subject

You must include the text `{match}`, which will be automatically replaced with the match numbers above for each email. For the full placeholder list, subject specifications, see help [here](#).

Also Send To  School Coaches  
 School Spectators  
 Team Subscribers

Note that if there are multiple matches with the same school or team being sent, these people will receive multiple notifications as well.

**Send Match Notification**

### Send Blast Notifications

You can send a blast notification to either the entire audience or to a specific division.

Template

Subject

Recipients

**Send Blast Notification**

Figure 9: The Notifications page after the roster is fetched.

Fetching the roster will query a worksheet in the TMS spreadsheet called **FULL ROSTER**. All the users and teams found in the roster will be saved in the system's database to minimize the number of repeated queries to the actual TMS spreadsheet. All the emails will be added to the Mailchimp audience and tagged with the optional tag. Every time the roster is fetched, the existing saved roster in the database will be wiped and replaced, so that any changes to the actual roster will be reflected in the system. Successfully fetching the roster also creates fetch logs, which includes possible warnings or errors that occurred in the rows of the worksheet, and a link to the full roster organized by schools, users, and teams. See [Appendix B](#) for the full expected format and specifications of the roster worksheet. There is also a help page linked on the Notifications page for the end users.

This brings us to the core part of the system, sending notifications. In the TMS spreadsheet, there exists a “communications view” worksheet called **Communications** that lists all the match numbers and information about them, including the division, round, status, and the competing teams (which may need to be updated from previous matches). Whenever a match number or range of match numbers is typed into the input box, this worksheet will be fetched to look for the relevant information for the requested matches. This information will then be inserted into the matches queue list, which contains all of the matches that will be sent a notification when the button is clicked. The user may modify this list as they see fit, removing matches with the “Remove” buttons on the right or adding more matches with the input. Above the queue is a matches query that exactly matches the matches currently in the matches queue, such that the user may enter the query into the input box and get the same result back (assuming the information in the worksheet hasn't changed). The “Last Query” button also does the same thing. See [Figure 10](#) for the matches queue and the corresponding inputs for interacting with it.

### Send Match Notifications

Enter the match numbers below that you want to send notifications for. The matches will keep being added to the matches queue, which you can edit as needed, until the "Send Notification" button is pressed. To restore the state of the last queue, click the "Last Query" button.

The TMS spreadsheet should have a worksheet called **Communications** which will be queried to find the competing teams for each match.

Add Matches

The match numbers should be separated by spaces or commas. You may use dashes for match number ranges.

Matches query: 701-702  Current matches in queue Match numbers input

Match	Division	Round	Status	Blue Team	Red Team	Valid	Remove All
701	PA	Round of 64	In Holding	Princeton PA1 <input type="button" value="Show Members"/>	MIT PA2 <input type="button" value="Show Members"/>	Yes	<input type="button" value="Remove"/>
702	PA	Round of 64	Canceled	Princeton PA2 <input type="button" value="Show Members"/>	Waiting for Match #701...	Red team invalid	<input type="button" value="Remove"/>

Matches queue

**Figure 10: The matches queue and the inputs for interacting with it.**

When the user is satisfied with the matches queue and wishes to send a notification for all of the listed (valid) matches, they can go down to the sending section on the left (see [Figure 11](#)). Here, they may select a template for the email content (from the selected template folder on the Admin Settings page), specify a subject, and choose additional recipients.

The subject requires a bit of attention at first. Since the point of the system is to personalize the notification emails, it also makes sense to personalize the subjects of the emails.

Template

Subject

You must include the text `{match}`, which will be automatically replaced with the match numbers above for each email. For the full placeholder list, subject specifications, see help [here](#).

Also Send To  School Coaches  
 School Spectators  
 Team Subscribers

Note that if there are multiple matches with the same school or team being sent, these people will receive multiple notifications as well.

**Figure 11: The sending section.**

Currently, one campaign is created per batch of matches, such as “Matches 701-704: Report to Holding”. Now, since we are creating one campaign per individual match in order to personalize the recipients, we can also change the subjects to include only the relevant match number. I still wanted to let the end user have control over the subject content (also lightly requested by my users), so the subject input allows placeholder values such as `{match}`, `{division}`, `{round}`, and `{team}`<sup>5</sup>, which will be automatically populated with the proper values for each campaign for each match. After a successful send, the last used subject is remembered, so the user may only need to set this once, but it is available to be changed if they wish.

The roster supports coaches, athletes, and spectators, and here the Admin user may also opt to send the notification emails to other recipients for the school of the team in each match. For instance, it is reasonable to assume that the coach for a school would want to know all the times one of their teams is being called to holding, and similarly for spectators. The Admin user may also send the notifications to any subscribers to the teams in the matches being sent. See section [2.2.8 Subscriptions](#) for more information on team subscriptions.

The final part of the Notifications page is the ability to send blast notifications to a large group of people, such as an entire division. When the audience tag from the Admin Settings page is not set, the user may also send a blast notification to the entire audience, which may include contacts that are not in the current tournament. A confirmation dialog will display a warning about this when attempting to send to an entire audience (see [Figure 12](#)). When the audience tag is set, the user will instead have the option to send a blast notification to all the contacts with that tag (see [Figure 13](#)). In either case, an Admin may send a blast notification to any of the divisions seen in the roster (see [Figure 14](#)).

---

<sup>5</sup> See [Appendix C](#) for the full list of placeholders, what they are for, and how they work.

**Send Blast Notifications**

You can send a blast notification to either the entire audience or to a specific division.

Template:

Subject:

Recipients:

**Send Blast Notification** ✕

Are you sure you want to send a blast notification email to all contacts in the selected audience?

If you specify the audience tag on the [Admin Settings](#) page, this button will instead send a notification email only to the contacts with that tag.

**Figure 12: Sending a blast notification when the audience tag is not set.**

**Send Blast Notifications**

You can send a blast notification to either the `2023 Princeton Tournament` tag or to a specific division.

Template:

Subject:

Recipients:

**Send Blast Notification** ✕

Are you sure you want to send a blast notification email to all contacts with the tag `2023 Princeton Tournament` ?

**Figure 13: Sending a blast notification when the audience tag is set to “2023 Princeton Tournament”.**

**Send Blast Notifications**

You can send a blast notification to either the entire audience or to a specific division.

Template:

Subject:

Recipients:

**Send Blast Notification** ✕

Are you sure you want to send a blast notification email to all contacts in the **PA** division?

**Figure 14: Sending a blast notification to the “PA” division.**

## 2.2.6 Matches Status

The Matches Status page is a public page, but only Admin users can see the full information (see Figure 15). In the TMS, each match has a status, from waiting for previous matches to “In Holding” to “Competing” to “Done”. (There are more statuses, but they are not relevant to the actual system.) For the user’s convenience, the status for every match is saved whenever the **Communications** worksheet is fetched (when fetching match information) and then displayed on this page. The table of statuses allows the user to easily see if a match has had an email sent to it, including all the relevant information about the email. The public version of the page only shows the email subject and when it was sent, while the admin version of the page also shows the name of the Mailchimp template used and all the recipients of the email. This page was an important feature requested by my users to see if any matches had been accidentally skipped.

The figure shows two screenshots of the 'Matches Status' page. The top screenshot is for an Admin user, showing a table with columns for Match, TMS Status, Mailchimp Template, Email Subject, Recipients, Email Time Sent, and Any Email Sent. A red box highlights the 'Mailchimp Template', 'Email Subject', and 'Recipients' columns, with a red arrow pointing to a note: 'Only Admins can see these columns'. The bottom screenshot is for a non-admin user, showing the same table but with the 'Mailchimp Template' and 'Recipients' columns hidden.

Match	TMS Status	Mailchimp Template	Email Subject	Recipients	Email Time Sent	Any Email Sent
901	Done	[TEMPLATE] Report to Holding	Match 901: Report to Holding	Show	2023-04-22 08:21:55	Yes
902	Done	[TEMPLATE] Report to Holding	Match 902: Report to Holding	Show	2023-04-22 08:21:59	Yes
903	Canceled	[TEMPLATE] Report to Holding	Match 903: Report to Holding	Show	2023-04-22 08:22:02	Yes

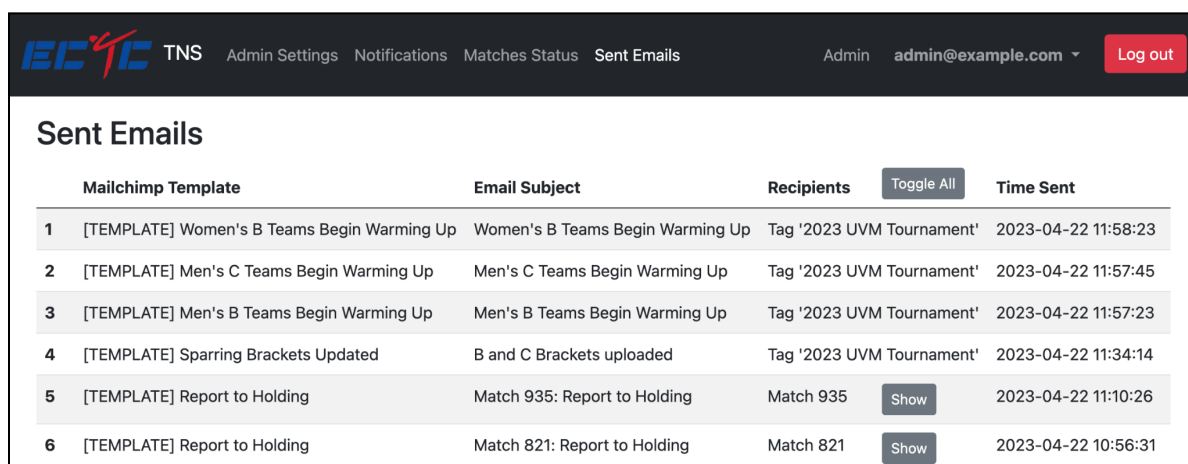
  

Match	TMS Status	Email Subject	Email Time Sent	Any Email Sent
901	Done	Match 901: Report to Holding	2023-04-22 08:21:55	Yes
902	Done	Match 902: Report to Holding	2023-04-22 08:21:59	Yes
903	Canceled	Match 903: Report to Holding	2023-04-22 08:22:02	Yes

**Figure 15: The Matches Status page for an Admin and non-admin user, respectively (from the 2023 UVM Tournament).**

## 2.2.7 Sent Emails

The Sent Emails page is a protected Admin page and lists all the emails that were sent by the system, including blast notifications (which don't appear on the Matches Status page) (see [Figure 16](#)). The displayed information is similar to the Admin view of the Matches Status page.



	Mailchimp Template	Email Subject	Recipients	Toggle All	Time Sent
1	[TEMPLATE] Women's B Teams Begin Warming Up	Women's B Teams Begin Warming Up	Tag '2023 UVM Tournament'		2023-04-22 11:58:23
2	[TEMPLATE] Men's C Teams Begin Warming Up	Men's C Teams Begin Warming Up	Tag '2023 UVM Tournament'		2023-04-22 11:57:45
3	[TEMPLATE] Men's B Teams Begin Warming Up	Men's B Teams Begin Warming Up	Tag '2023 UVM Tournament'		2023-04-22 11:57:23
4	[TEMPLATE] Sparring Brackets Updated	B and C Brackets uploaded	Tag '2023 UVM Tournament'		2023-04-22 11:34:14
5	[TEMPLATE] Report to Holding	Match 935: Report to Holding	Match 935	Show	2023-04-22 11:10:26
6	[TEMPLATE] Report to Holding	Match 821: Report to Holding	Match 821	Show	2023-04-22 10:56:31

**Figure 16: The Sent Emails page (from the 2023 UVM Tournament).**

## 2.2.8 Subscriptions

The Subscriptions page, which is available through the dropdown on the user's email, allows the user to subscribe to the notifications for a team, allowing them to receive any notifications that the team gets. For instance, I may be interested in knowing when my teammates are going to be competing, so I could subscribe to all the Princeton teams. The subscriptions are listed by school and division, and there are pagination buttons for easily finding what the user may be looking for (see [Figure 17](#)).

To prevent unknown users from logging in to the system and subscribing to some teams, this feature is only enabled for Admin users and users that are in the fetched tournament roster<sup>6</sup>.

<sup>6</sup> There is also a technical reason for this. Mailchimp campaigns will only send to recipients that are actually contacts in the audience, so if we try to send an email to an unknown user, it wouldn't work anyway (see section [5.1 Future Work](#) for an unforeseen bug relating to this). It also makes sense from a security perspective to not allow random users to receive these emails.

This means that coaches, athletes, or spectators in the roster will be able to subscribe to any teams they wish, but non-participants will not have access (see [Figure 18](#)).

Note that the Admin user sending the notifications has full control over the recipients and can prevent team subscribers from receiving notification emails, as seen in [Figure 11](#). This means that users who subscribe to certain teams may still not receive all notification emails.

The screenshot shows the 'Subscriptions' page for an Admin user (admin@example.com). The page title is 'Subscriptions' and it includes a navigation bar with 'Admin Settings', 'Notifications', 'Matches Status', and 'Sent Emails'. Below the title, there is a message: 'Here, you may subscribe to the notifications for a team. You will receive any and all notifications that they receive.' The page features a grid of team and notification type filters. The teams listed are Brandeis, Brown, Cornell, Cortland, Dartmouth, Harvard, MIT, NYU, Northeastern, Rutgers, Tufts, U Albany, U Conn, UMBC, and UVM. The notification types are PA, PB, and PC. Each team has a 'Toggle' button and a list of checkboxes for each notification type. For example, Brandeis has checkboxes for Brandeis PA1 and Brandeis PC1. The user is currently logged in as 'Admin'.

**Figure 17: The Subscriptions page (from the 2023 UVM Tournament).**

The screenshot shows the 'Subscriptions' page for a user not in the roster (unknown@example.com). The page title is 'Subscriptions' and it includes a navigation bar with 'Matches Status' and 'Log out'. Below the title, there is a message: 'Sorry, you cannot subscribe to notifications for a team because your email ( unknown@example.com ) is not in the tournament roster.' The user is currently logged in as 'unknown@example.com'.

**Figure 18: The Subscriptions page for a user not in the roster.**



### 3. Implementation

For the implementation of this system, I used a Python Flask server with a PostgreSQL database, connected with the Flask extension Flask-SQLAlchemy. The frontend was served through Jinja2 templates, and the styling made use of the Bootstrap framework. The final project is deployed through Render (<https://render.com/>), and the production database is also hosted on Render.

The source code is available at <https://github.com/josephlou5/ectc-tournament-app>. The application is live at <https://ectc-notifications.onrender.com><sup>7</sup>.

There were some interesting bugs that were discovered during a real use of the system. Please see section [4.2 Real Use at the 2023 UVM Tournament](#) for a detailed description of the bugs and their solutions, as well as how the real use of the system went.

The rest of this section will describe particular design choices or challenges that I faced.

#### 3.1 Database Schema

I used a PostgreSQL relational database for storing data in the system. This section describes the table schemas and their relationships.

##### 3.1.1 Global State

There were some pieces of information that were global to the entire system, such as the service account used to access Google Spreadsheets (see section [3.2 Reading from Google Spreadsheets](#)), the Mailchimp API, the last used template for sending a notification email, etc. To store this, I decided to use a table called `GlobalState` that had a single row with many columns, essentially acting as a symbol table of the “global variables” of the system. Whenever I

---

<sup>7</sup> Due to the limitations of the free tier of a PostgreSQL database on Render, my production database will expire on June 23, 2023. Although the web service will still be live, it won't have a database to connect to and will therefore be pretty useless. My users have expressed interest in using the system again in the future, so the system may move to their own domain, but this link will not be valid for long after this paper is written.

needed to get this global state, I fetched the first row of the table; if it did not exist, I created one.

In this way, only the first row of this table was used. See [Figure 19](#) for the full schema.

GlobalState					
id (primary key)	Integer	The id of the only row.	mailchimp_match_template_id	String	The id of the last used Mailchimp template for match notifications.
service_account	String	The service account JSON credentials as a string.	mailchimp_match_subject	String	The last used subject for match notifications.
tms_spreadsheet_id	String	The id of the TMS spreadsheet.	mailchimp_blast_template_id	String	The id of the last used Mailchimp template for blast notifications.
roster_last_fetched_time	DateTime	The last time the roster was fetched from the TMS spreadsheet (in UTC).	mailchimp_blast_subject	String	The last used subject for blast notifications.
mailchimp_api_key	String	The Mailchimp API key.	send_to_coaches	Boolean	Whether to also send match notifications to coaches.
mailchimp_audience_id	String	The id of the selected Mailchimp audience.	send_to_spectators	Boolean	Whether to also send match notifications to spectators.
mailchimp_audience_tag	String	The Mailchimp audience tag.	send_to_subscribers	Boolean	Whether to also send match notifications to subscribers.
mailchimp_folder_id	String	The id of the selected Mailchimp template folder.			

**Figure 19: Schema for the GlobalState table.**

### 3.1.2 Admins

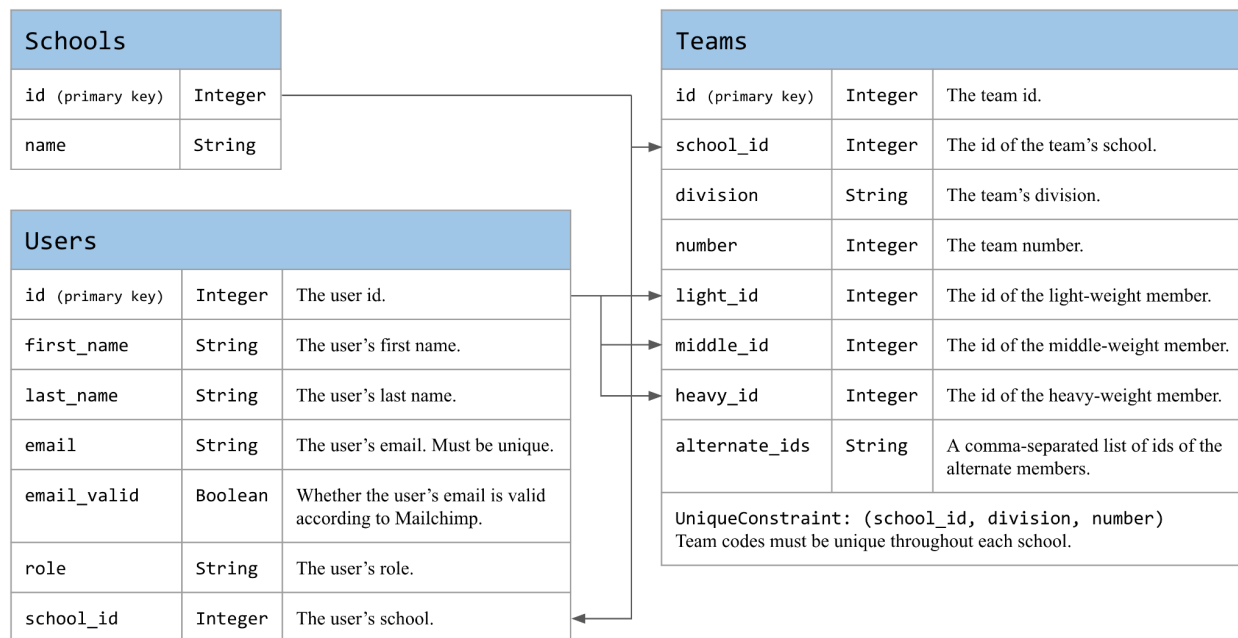
For authorization, I used an Admins table (see [Figure 20](#)). To identify Super Admins, there is an additional boolean column `is_super_admin`.

Admins		
email (primary key)	String	The email of the admin.
is_super_admin	Boolean	Whether this admin is a Super Admin.

**Figure 20: Schema for the Admins table.**

### 3.1.3 Roster Tables

To store the roster, I used a Schools table, a Users table, and a Teams table. Each user belongs to a school. Each team belongs to a school and is composed of multiple users. See [Figure 21](#) for the full schema.



**Figure 21: Schema for the roster tables: Schools, Users, and Teams.**

I also supported different roles for each user. Specifically, I supported having coaches, athletes, and spectators in the roster. Only athletes could be on a team (there is no database constraint for this, however). Coaches and spectators are not necessary for the system to work.

Each team can also be composed of zero or more alternate members. I could have created a `TeamAlternates` table with a `team_id` and `user_id` column, where each row represents a user being an alternate on a team, but I decided to have the alternates be represented as a comma-separated list of user ids in the `Teams` table instead. Because of this workaround, there is no formal relationship between the `Teams.alternate_ids` column and the `Users.id` column.

Whenever the roster was fetched, these tables would be wiped by dropping them and then adding them again, which would effectively reset the internal id counters back to `1`. Since the rest of the application doesn't depend on any of these ids (they are only dependent on each other), this operation was safe to do.

### 3.1.4 Matches Status

As mentioned above, whenever the matches are being fetched, their statuses from the TMS spreadsheet are saved to be displayed on the Matches Status page. The `TSMMatchStatuses` table simply saves these statuses for each seen match number, as well as when each status was last updated (see [Figure 22](#)).

TSMMatchStatuses		
match_number (primary key)	Integer	The match number.
status	String	The status from the TMS spreadsheet.
last_updated	DateTime	When this status was last updated (in UTC).

**Figure 22: Schema for the `TSMMatchStatuses` table.**

### 3.1.5 Emails Sent

The `EmailsSent` and `BlastEmailsSent` tables store the necessary data for the Matches Status page and the Emails Sent page. See [Figure 23](#) for the full schema.

In the `BlastEmailsSent` table, at most one of the `division` or `tag` columns should be set, indicating the recipients of the blast email. If neither is set, the email is understood as being sent to the entire audience that was selected at the time (which is not saved in this table).

EmailsSent			BlastEmailsSent		
id (primary key)	Integer	The row id.	id (primary key)	Integer	The row id.
match_number	Integer	The number of the match for which this email was sent.	template_name	String	The name of the Mailchimp template used.
template_name	String	The name of the Mailchimp template used.	subject	String	The subject of the sent email.
subject	String	The subject of the sent email.	time_sent	DateTime	When the email send was initiated (in UTC).
time_sent	DateTime	When the email send was initiated (in UTC).	division	String	The division the email was sent to.
recipients	String	A semicolon-separated list of recipient email addresses.	tag	String	The Mailchimp tag the email was sent to.
			At most one of <code>division</code> or <code>tag</code> should be set; the other should be null.		

**Figure 23: Schemas for the `EmailsSent` and `BlastEmailsSent` tables.**

### 3.1.6 Team Subscriptions

The team subscriptions are saved in a table called `UserSubscriptions` (each row is the subscription of a user to a team, which is a justification for the slightly misleading name) (see [Figure 24](#)). Each row saves the unique team identifier of the school, division, and team number. The school is saved as the school name instead of the school id for two reasons: 1) subscriptions can theoretically carry across tournaments (if a Super Admin doesn't wipe them); and 2) if the roster is fetched again and a school's id changes, the subscriptions would break if there was a dependency on the `Schools` table.

UserSubscriptions		
id (primary key)	Integer	The row id.
email	String	The user's email.
school	String	The team's school.
division	String	The team's division.
number	Integer	The team number.
UniqueConstraint: (email, school, division, number) Each row must be unique; a user can only be subscribed to a team once.		

**Figure 24: Schema for the `UserSubscriptions` table.**

## 3.2 Reading from Google Spreadsheets

Since all the information I needed was in the TMS spreadsheet, and I didn't want to make users duplicate information by manually entering the roster and match team names into my system, I needed to be able to read from a Google Spreadsheet. In addition, minimizing the amount of work that the end users would have to do to enable me to read data from a spreadsheet would be ideal.

Through other work, I have become familiar with a Python package called `gsread` (<https://docs.gsread.org/en/latest/>) that allows a client to use either their own Google account or a service account to create, view, and edit Google Spreadsheets. However, to use your own Google account, you must go through a process of getting some API credentials, which isn't entirely straightforward, and then you must be continuously re-authenticated whenever your session expires. To use a service account, you must go through a similar process to get API credentials, but there is no need to re-authenticate. Since this option required users to have some sort of credentials file that would need to be inputted into and saved by the system, I wanted to consider some alternatives.

I found an API service called SheetDB (<https://sheetdb.io/>). I would need to create an API key, then call their API methods with a spreadsheet id as a query argument. Importantly, nothing else would be required, so I could create my own API key and pass it as an environment variable to the server, removing the need for the end user to even think about how to access Google Spreadsheets. I also only needed to read data for the purposes of my project, so this seemed like a promising solution. However, the free plan only allowed for 500 requests per month, which definitely wouldn't be enough, and the next tier was \$29.99 for 10k requests per month, which might be enough but seemed like it wouldn't be worth the cost.

In the end, I decided to go with the slightly complicated but free option with full functionality. Because of the drawbacks mentioned above, I decided to have the end user use a service account rather than their own email. This would also mean the system could be more flexible, since the account accessing the spreadsheets would not be tied to a particular user email address. Please refer to [Appendix A](#) for more information about service accounts, including how to create one.

### 3.3 Sending Notifications

This project was inspired by an issue I saw with email notifications, but there was no need to also constrain myself to emails. My users suggested the possibility of expanding the notification system to mobile push notifications, for example. However, since I only had a semester to work on this and I was very familiar with web services and very unfamiliar with mobile applications, I decided to stick with the email notifications that were currently being used rather than learn how to create a mobile app so that I could add push notifications. Thus, I needed a service that could send emails.

I initially looked for my own email service that I could use. I considered options such as Twilio SendGrid (<https://sendgrid.com/>); Mailgun (<https://www.mailgun.com/>); Flask-Mail (<https://pythonhosted.org/Flask-Mail/>), a Flask extension that allows you to send emails via SMTP; and Yet Another Mail Merge (YAMM) (<https://yamm.com/>). However, there were always drawbacks to them, such as the price or email quotas. Eventually, I decided to piggyback on the service that the TC was already using, Mailchimp (<https://mailchimp.com/>). I discovered that Mailchimp had a Marketing API<sup>8</sup> and a convenient Python client library<sup>9</sup>, so this was a very good choice.

### 3.4 Mailchimp

#### 3.4.1 Templates and Campaigns

As mentioned above, I had some issues with a bug in the Mailchimp API which prevented me from implementing the ideal flow that I wanted and that would be easiest for the users to understand. Specifically, the ideal flow would be to ask the user to create Mailchimp templates for each email body content that they wanted to use, then present them with a dropdown of those

---

<sup>8</sup> <https://mailchimp.com/developer/marketing/api/>

<sup>9</sup> <https://github.com/mailchimp/mailchimp-marketing-python>

templates when sending a notification. On the server side, I would then need to create a campaign and set its content to be the selected template, as a user would do on the Mailchimp UI.

When creating (`POST /campaigns`<sup>10</sup>) or updating (`PATCH /campaigns/{campaign_id}`<sup>11</sup>) a campaign using the Mailchimp API, there is a body parameter `settings.template_id` that you can set to specify the template you wish to use for the campaign. I tried to create a campaign with a test template id, but when I checked the campaign on the Mailchimp UI, it had no content. I then tried to create a campaign on the Mailchimp UI and use the update endpoint to set the template id, but that again had no effect. There were no indications of why these steps did not work, and it was a pretty serious issue because my entire project depended on being able to send notifications with content.

Next, I tried the “set campaign content” endpoint (`PUT /campaigns/{campaign_id}/content`<sup>12</sup>), where you could either provide a template id or some raw HTML to set the content of the specified campaign. I tried the template id again, and this time received a response with a perplexing error: “A deep, internal error has occurred during the processing of your request. Please contact support.” This was very surprising, but I tried the other method of setting the raw HTML, and that worked. This left me with two options: 1) extract the HTML of the selected template, then set the HTML of the campaign; or 2) instead of using Mailchimp templates, have the end user create their own HTML templates on my site, which I would save, then allow them to select from a dropdown of their created templates, and I could use the content of those to set the HTML of the campaign. Obviously, option 2 seemed overly complex, and there were no guarantees that the users would know HTML (Mailchimp

---

<sup>10</sup> <https://mailchimp.com/developer/marketing/api/campaigns/add-campaign/>

<sup>11</sup> <https://mailchimp.com/developer/marketing/api/campaigns/update-campaign-settings/>

<sup>12</sup> <https://mailchimp.com/developer/marketing/api/campaign-content/set-campaign-content/>



provides a very nice template builder which requires no coding knowledge). It would also introduce the unnecessary complexity of dealing with HTML, including parsing, XSS protection, and validation to name a few, not to mention I didn't know Mailchimp's expected format of HTML or supported tags, for example. Option 1 seemed like the way to go, until I realized there was no endpoint to get the HTML content of a template.

Finally, I discovered the replicate endpoint (`POST /campaigns/{campaign_id}/actions/replicate`<sup>13</sup>), which would allow me to replicate an existing campaign, including its content. In this way, I wouldn't need to deal with setting any content. Unfortunately, this meant a worse tradeoff for my users, since they would have to create all the "template campaigns" beforehand, and if they wanted to change the content of one of these "templates", they would need to edit the content of a campaign. This forced me to have to clearly explain this in the system, since it was a very important part of being able to select the proper templates to use when sending emails.

I also contacted Mailchimp Support about the internal error when trying to set the content of a campaign using a template id. They were not able to immediately identify the problem, and they said they would have to move the conversation to email when they found the issue. After three and a half weeks, they finally replied and said that the new builder templates are not supported by the API, and the best solution would be to use the Classic Builder when creating the template<sup>14</sup>. When creating user-defined templates, the Mailchimp UI automatically uses their new template builder (which is in beta at the time of writing), so the existing templates on my users' Mailchimp account were all built with the new template builder, not the Classic Builder. At this point, my project was near completion, and it didn't seem good to ask my users to do

---

<sup>13</sup> <https://mailchimp.com/developer/marketing/api/campaigns/replicate-campaign/>

<sup>14</sup> See [Appendix D](#) for the full transcript of my conversation with Mailchimp Support.

extra work to respond to the shortcomings of the Mailchimp API, so I opted to stick with the workaround I described above.

### 3.4.2 Custom Email Content

Since the goal was to personalize the emails for certain recipients, it would also be good to be able to customize the email body content as well. For example, my users suggested that perhaps the subject could contain the match number and the body could contain the team member names as well as the opposing team name and members. However, looking into Mailchimp to figure out how to do this, I ran into a few obstacles.

Unfortunately, there didn't seem to be a way to edit the content of a template through the API. As mentioned above, editing the content of a campaign would also be a hassle, even through raw HTML. Mailchimp does have a concept of "merge fields"<sup>15</sup>, which are fields associated with each contact that can be injected into the body of an email with the syntax `*|FIELDNAME|*`, such as addressing each recipient by their first name when sending a single campaign. However, in order for this to be useful for me, I would have to edit the merge fields of each recipient of a campaign to be the data that I wanted to display for all of them (e.g., I would edit the merge fields called `*|TEAMMEMBER1|*`, `*|TEAMMEMBER2|*`, `*|TEAMMEMBER3|*`, etc. for each recipient of a match email in order to properly show all the team members). However, this seemed like a waste of time and resources with all the repeated information, not to mention the uncertainty with some teams not having three members and some teams having alternates. It did not seem like I could take advantage of this feature in an efficient or simple way.

I tried finding out if there were any other options for dynamically changing the content of a template on a per-campaign basis, but did not find anything. There seemed to be something about customizing the content of a template, but templates weren't working for me, and this

---

<sup>15</sup> <https://mailchimp.com/developer/marketing/docs/merge-fields/>

seemed like a paid feature that the Mailchimp account didn't have access to. Not having this feature wasn't too much of a drawback for the overall system, but it definitely would have been nice to be able to implement this for the users.

### 3.4.3 Sending Batch Notifications

When sending a notification for a match, I needed to create a campaign with the proper recipients. To set the recipients of the campaign, I had to put all the contacts into a segment (or tag), then direct the campaign to send to that segment. After using the send campaign endpoint (`POST /campaigns/{campaign_id}/actions/send16`), Mailchimp would need to prepare the campaign, then send it. Because of this, the time I triggered the send wouldn't necessarily be when the emails were actually sent to the contacts, but this was a minor issue that I wasn't very bothered with.

Thus, when sending a notification for multiple matches at once (which the "Send Notifications" button supports), I had to create one campaign per match, setting the segment with the proper recipients for each one. Since segments are actually created objects on Mailchimp and stick around even if I only wanted to temporarily use one for the purpose of a single campaign, I decided to reuse a singular segment called "[TNS] Email Segment" that would be edited for every created campaign. Whenever a notification was triggered, I would look for an existing segment with that name, and if it didn't exist, I would create it. Then, for every match, I would create a campaign (by replicating the selected template campaign, for the workaround), replace the contacts in the segment with the intended recipients for this match, direct the campaign to send to the segment, then trigger a send for the campaign. Importantly, I was reusing a single segment for every single campaign that was being sent. This proved to be an issue later; see section [4.2.3 Campaigns and Segments Issue](#) for what happened.

---

<sup>16</sup> <https://mailchimp.com/developer/marketing/api/campaigns/send-campaign/>

### 3.4.4 Batch Operations

The Mailchimp API has a set of batch operations endpoints<sup>17</sup> that allow the user to run many operations at once and asynchronously. This seemed like an ideal feature to take advantage of, especially for large operations such as adding all the new contacts to Mailchimp or sending a large batch of match notifications. However, when I tested this with a small batch of adding three contacts to the Mailchimp audience, it did not seem to work. I checked the status after 20 minutes and it was still pending, so I decided that this would not be a reliable feature to use. It would be bad for the users to have to wait so long for even a simple operation to finish, and I would also have to implement some way for my server to save the batch id and continuously check its status until it finished. In the end, I did not use batch operations.

## 3.5 AJAX

Although there are many inputs in this system, as seen in section [2.2 The System](#), I did not use any forms. Instead, I made heavy use of AJAX requests to `GET`, `POST`, and `DELETE` endpoints. The responses returned by these AJAX endpoints were usually JSON data with the format `{"success": True}` or `{"success": False, "reason": "error message"}`. Upon a success, I would sometimes trigger a page refresh so that new data would be shown (e.g., accepted inputs on the Global Settings and Admin Settings pages), or I would display a “Success” message somewhere on the page that would disappear after some time (e.g., dropdowns on the Admin Settings page; subscribing to a team on the Subscriptions page). Upon an error, I would usually be able to trigger a visual validation error through Bootstrap’s `.is-invalid` class on `<input>` elements. This scheme was used on almost every page with user interaction and in every endpoint meant to be called with AJAX.

---

<sup>17</sup> <https://mailchimp.com/developer/marketing/api/batch-operations/>

## 3.6 Frontend

I used the Bootstrap framework for styling. I copied my Bootstrap boilerplate and template code from a previous project I had made which used Bootstrap 5.0. At the time of creating this project, Bootstrap 5.2 had already been released, with a few nice updates such as more colors. However, by the time I realized this, I didn't want to go back through all my pages and figure out what needed updating, so I stayed on the lower version.

### 3.6.1 Responsiveness

Bootstrap is developed “mobile first” with responsiveness built in. Since my project is meant to be used in a tournament context, in which the users of the system may have to be mobile for various reasons, sometimes without a laptop handy, I also decided to try to make the frontend responsive to different window and screen sizes. One example is in [Figure 15](#): at a certain point, the main title in the navbar will shrink from “Tournament Notification System” to “TNS”.

Sometimes the components on the page were weirdly positioned and hard to work with, especially when elements started to wrap to the next line, but I'm pretty satisfied with how everything turned out. There are a few exceptions, such as the matches queue table on the Notifications page: when the members of each team are shown, the result is a little squished, especially with long email addresses, and when the screen size gets smaller the table starts to really look awful. To fix this, it might require some design changes of the table layout.

### 3.6.2 Success and Error Messages

In order to make using the system as clear as possible for users, I made sure to display descriptive and appropriate success and error messages for almost every action performed by the user. For example, [Figure 25](#) shows a success message after sending a match notification. These

### Send Match Notifications

Successfully sent match email notifications ✕

Enter the match numbers below that you want to send notifications for. The matches will keep being added to the matches queue, which you can edit as needed, until the "Send Notification" button is pressed. To restore the state of the last queue, click the "Last Query" button.

The TMS spreadsheet should have a worksheet called `Communications` which will be queried to find the competing teams for each match.

Add Matches

The match numbers should be separated by spaces or commas. You may use dashes for match number ranges.

Fetch Matches Info

Last Query

Match	Division	Round	Status	Blue Team	Red Team	Valid	
No matches							

Template

Subject

You must include the text `{match}`, which will be automatically replaced with the match numbers above for each email. For the full placeholder list, subject specifications, see help [here](#).

Also Send To

- School Coaches
- School Spectators
- Team Subscribers

Note that if there are multiple matches with the same school or team being sent, these people will receive multiple notifications as well.

Send Match Notification

#### Send Blast Notifications

You can send a blast notification to either the entire audience or to a specific division.

Template

Subject

Recipients

Entire audience

PA

PB

PC

Men's A

Men's B

Men's C

Women's A

Women's B

Women's C

Send Blast Notification

**Figure 25: A success message after sending a match notification.**

were achieved either through Flask's `flash()` function<sup>18</sup> or through messages returned in the response of AJAX calls (see section 3.5 AJAX). Bootstrap's alert components<sup>19</sup> were very nice for this purpose. I also used JavaScript's `setInterval()` function to make these alert messages disappear after some time.

## 3.7 Code

### 3.7.1 Organization

I mostly organized my code by purpose, then by permissions of the type of user who that code was for. All the database models and helpers are grouped into the `src/db/` folder, such that I can simply `import db` in other files and access my database getter and setter methods. I have

<sup>18</sup> <https://flask.palletsprojects.com/en/2.3.x/api/#flask.flash>

<sup>19</sup> <https://getbootstrap.com/docs/5.0/components/alerts/>

a `src/static/` folder with logo files, shared global JavaScript functions, and global styles. (This folder also stores the public changelog, which I started after fixing the bug described in section [4.2.3 Campaigns and Segments Issue](#).) The fetch roster logs, which are saved in a JSON format, are also located in the static folder when it is created. The `src/templates/` folder is grouped by user permission or page, with a few shared templates such as `layout.jinja`, `navbar.jinja`, and `macros.jinja`. The `src/views/` folder is similarly grouped, using a custom helper class to add all the routes into the main Flask app in `src/app.py`<sup>20</sup>. I also have a `src/utils/` folder for utilities that serve a certain purpose, such as those for fetching from the TMS spreadsheet, those for communicating with the Mailchimp API, and general server utilities like generating an HTML response from a rendered template. As the main bulk of the server code, the `src/views/notifications.py` file was getting so long (over 1500 lines) that I had to create a separate `src/utils/notifications_utils.py` file just to contain any helper functions that weren't directly related to a route, such as a handwritten parser for match number queries and a subject placeholder validator.

In the root directory of my code, I also added the scripts `build.sh` and `start.sh` for deployment. In the past, I had often put the build and start commands directly into the deployment service, but doing it this way will allow me to better understand what I need to do the next time I make a web service (and also make it easier to deploy the system elsewhere). In the same spirit, I also created a local `runserver.sh` script that launches a local development server with all the proper environment variables set, which was mostly for my own convenience. (Because this included sensitive environment variable values such as those for Authentication through Google, I could not commit this to the repo.)

---

<sup>20</sup> An alternative to my custom helper class would be to use Flask Blueprints. However, this is a concept that I was not familiar with and did not research for use in this project.

### 3.7.2 Statistics

Language	Total Files	Total Lines
Python	48	7781
Jinja (not including <code>&lt;script&gt;</code> tags)	21	3319
JavaScript	11	2081
CSS	1	87
<b>ALL</b>	81	13,268

Folder / File	Language	Files	Lines
Database Migrations	Python	20	939
<code>src/db/*</code>	Python	9	1836
<code>src/static/shared.js</code>	JavaScript	1	332
<code>src/static/style.css</code>	CSS	1	87
<code>src/templates/*</code>	Jinja + JavaScript (in <code>&lt;script&gt;</code> tags)	21	5076 (3319 + 1749)
<code>src/utils/*</code>	Python	8	2546
<code>src/views/*</code>	Python	8	2269

## 3.8 Deployment

I deployed the application as a web service on Render. In the free tier, web services spin down after 15 minutes of inactivity, then are spun back up upon the first new request, which could lead up to an initial response delay of 30 seconds<sup>21</sup>. This didn't seem ideal to me, especially in the context of a tournament needing to handle a lot of things. I had received funding from the School of Engineering and Applied Science (SEAS) for a subscription to an email service, but since I was using my users' own paid Mailchimp account and therefore wasn't using the funds for

<sup>21</sup> Source: <https://render.com/docs/free>



anything, I decided to upgrade to the Render starter tier for \$7/month. The application therefore did not have any downtime, and every request was handled immediately.

## **4. Evaluation**

### **4.1 Formal Evaluations with Users**

I performed formal evaluations with three users, student volunteers from the Princeton Taekwondo Team that had the appropriate knowledge and tournament experience to understand (and appreciate) the purpose and goal of the system. Through these evaluations, I was able to pinpoint some potentially confusing parts of the system and additionally gain some insights on how real users figured out how the system worked and what it did.

For each of the users, I asked them to perform a list of tasks that would generally step them through using the system as an Admin user. (I did not evaluate the Super Admin role because the Manage Admins page and Global Settings page are pretty straightforward and are not used regularly.) Throughout the evaluation, any general comments or suggestions were noted, as well as any issues with the tasks.

The tasks were:

1. Given the TMS spreadsheet link (a testing spreadsheet), complete all the admin settings
2. Send a match notification for matches 701-705 to report to holding
3. Send a match notification for matches 701 and 801 for final call to holding
4. Send a blast notification to the Women's A division for Women's A team to start warming up
5. Send a blast notification to everyone about a lost item
6. Check whether Match 702 received an email
7. Check the list of all emails sent

The most prominent issue that came up during these evaluations was how to define a match notification subject. Since my backend was in Python, I used the style of Python format strings for the subject placeholders, allowing me to use existing language features to inject variables into a string. I explained a bit on the frontend what these placeholder variables were for and what their values would be replaced with. However, the users in my evaluations did not immediately understand how they worked.

In the subject of match notifications, you must include the placeholder `{match}` so that the email subject contains the match number that it is being sent to. There is some help text underneath the subject input box which says as much, and the help page also states this. However, the evaluated users either did not read the small help text or did and still didn't understand it. They tried inputs such as "Match {701}", "Match {701-705}", and "Match {match 701}", all of which received errors due to unknown placeholder variable names. I either had to step in and explain that the `{match}` text itself was required (as it appears; no editing necessary on your own) or the user eventually discovered the correct format through trial and error.

Based on this feedback, I changed the help text so that it was clearer in explaining that the user should input the text `{match}` and that it would be automatically populated when the emails were sent, implying that the user did not need to type any match numbers themselves. However, I asked the real users after they used the system if they experienced any problems with this, and none of them did. This could have been because I hosted a training session for them, so the tricky parts of the application were already explained to them. The evaluated users did say that once they understood, it made sense, but before that it was unclear.

Another common request from my evaluations was to have the subject automatically populate based on the selected template. I explained the issues with this, namely that it would

override a subject that the user already typed, which they may have wanted to keep. I do understand the appeal of not having to type a different subject for each template, especially in the blast notification section where the subject may very well be exactly the same as the template name, but this would be something that requires a bit more thought. For example, a button could be added next to the subject input box that would replace the text with the template name, thereby leaving it up to the user if they would like to automatically set the subject to the template name (which cannot be copied and pasted, since it is in a dropdown).

Some positive feedback I received from my evaluations was that the system generally seemed easy to use and useful for its purpose. One user really liked that there were very clear confirmation messages that told you when an action was performed successfully, as discussed in section [3.6.2 Success and Error Messages](#). Someone also said that there was enough text on the main pages so that it wasn't overwhelming but always answered whatever they were wondering; a memorable quote from their evaluation was, "It would be nice if you had examples – oh you do."

Please see [Appendix E](#) for the full notes from these three evaluations.

## **4.2 Real Use at the 2023 UVM Tournament**

One of the main goals of the project was to be able to use the system in a real tournament. In the Spring 2023 semester, there were only two ECTC tournaments: one at Princeton on February 26, 2023 and one at the University of Vermont (UVM) on April 22, 2023. The Princeton Tournament was definitely too soon for my project to be fully working, so we aimed for the UVM Tournament, which had really good timing anyway because it was only a week before this written report was due, which meant that the project should be pretty much finalized by that

time. Unfortunately, I was unable to make it to this tournament in person, but I gave out my contact info to my users so that they could call or message me with any potential issues.

#### **4.2.1 Training Session**

We first hosted a training session for the specific volunteers that were going to be using the system at the tournament. I spent an hour walking through the features of the system and explaining the relevant aspects of how each part worked. I answered any questions they had, but they seemed to understand it quite well, and were excited to be using it at the tournament. I got into a few technical details with one of my users to explain the limits of what they had to do; previously, all the emails of the tournament participants were imported manually into Mailchimp, but I explained that the “Fetch Roster” button would automatically do that, so now there was no work needed on Mailchimp other than the setup of the “template campaigns”.

Speaking of which, I was afraid that the Mailchimp template workaround would be the hardest thing to understand (I didn’t go over it with the evaluated users because they were not familiar with Mailchimp), but fortunately the users seemed to understand it well enough, and on the system frontend, the workaround presents itself as if it were the actual flow that they would understand.

#### **4.2.2 Scaling Issue**

Early in the morning, I received a call that the roster didn’t seem to be fetching. While looking at the system, I also noticed that there was no link to the fetch roster logs, which would only happen if the entire fetch failed in some way. Curious, I checked the server logs, and saw that while trying to add new contacts to Mailchimp, the server was crashing with a timeout error (triggered by Render, it seems). I realized that I was performing a redundant and expensive operation while adding the contacts to Mailchimp. Essentially, I was blindly adding all emails

that I saw in the roster to Mailchimp. However, this is unnecessary because if the roster was fetched before then all these contacts would have been added to Mailchimp then. My quick fix was to take a set difference between the current emails seen in the roster and the previous emails currently in the roster, and only add the new emails seen to Mailchimp. Since I was making one API call per added user (see section [3.4.4 Batch Operations](#) for why I couldn't add them all at once), this was a scaling issue when there were too many new contacts.

After the franticness of fixing bugs as soon as possible, I came up with an even better solution to this problem. By making a few additional (paginated) API calls to Mailchimp, I could easily figure out exactly who was already a contact in Mailchimp without having to rely on the previous state of my saved roster. I then would only need to add the missing contacts. This also has the added benefit of not doing additional work for users spanning multiple tournaments.

#### **4.2.3 Campaigns and Segments Issue**

Later in the day, I received a message that school coaches seemed to not be receiving any notification emails. I checked my Sent Emails page, which said that the coaches were receiving the emails. I further investigated on Mailchimp itself, and here I discovered a huge bug. Emails being sent in batch were really all being sent to only one set of recipients, while all the other recipients were not receiving any emails.

Recall the way that I reused a single segment for all created campaigns, as mentioned in section [3.4.3 Sending Batch Notifications](#). When specifying the recipients of a campaign, you give the campaign the segment id, which is reasonable. After triggering the campaign to send, Mailchimp prepares the campaign for sending, which may take a few moments, and then at some point, looks up the recipients in the specified segment and sends the email to them. However, since I was creating campaigns, setting the contacts in the shared segment, and triggering

campaign sends in such quick succession, the campaigns internally must have looked up the recipients from the segment after I had already finished “sending” all the campaigns. This meant that all the campaigns were actually using the recipients list only meant for the last campaign, since that was the most recent contacts list in the segment when it was queried.

This was obviously a huge issue, since most emails were not being sent to the intended recipients, and a small subset of users were getting a bunch of emails they weren't supposed to. Fortunately, the fix for this was pretty simple: instead of reusing a single segment for every batch of  $n$  matches, use one segment per match, resulting in  $n$  segments for a single batch of  $n$  matches. I changed the name of the shared segment I was using to include an index at the end, so campaign  $i$  would use a segment with name “[TNS] Email Segment { $i$ }”. This ensured that the campaigns in a single batch were all distinct, resulting in all of them reaching the proper recipients. However, this still left the possibility of the segments clashing with each other if two batches were sent in quick succession, but I just had to warn my users to not do that. A few possible fixes for this last issue include:

1. Create temporary segments, then delete them after the campaign using it is sent;
2. Pre-tag each contact on Mailchimp with the appropriate teams they are on, then send campaigns to those predefined segments (i.e., if all the segments are predefined, I won't need to change the contacts in a segment, and there would be no fear of sending a campaign to a segment whose contacts changed between triggering the send and Mailchimp actually sending the email); or
3. Create a segment for each match number, which should always have the same teams and therefore the same recipients.

#### 4.2.4 Survey

After the UVM Tournament, I sent a survey to the users of the system to ask about their experience and perform a mini-evaluation of the system in a real setting.

The overall consensus seemed to be pretty positive; they all said they would love to use the system in the future. Anyone who used the system was happy with the ease of using it and how it simplified the process of sending notification emails, but the bugs that we ran into during the day definitely made the experience a little rocky. Thankfully, the tournament still ran smoothly despite the setbacks (at least from what I could tell, there weren't any huge issues on the tournament side).

I also made a point of asking these users whether they found the subject placeholders confusing or hard to understand, as my evaluated users had experienced. I had done a training session with them, but at that time I didn't really think that I had to bring special attention to how the placeholders worked. However, all of them said that at least the `{match}` placeholder was clear to them, but they didn't try using any of the other available ones.

A pain point that was brought up multiple times (and even during the tournament itself) was the handling of when certain information is missing from the roster. Currently, fetching the roster will ignore any invalid rows, which includes when a user's email is missing. A consequence of this is that if this person was the only member of a team, then that entire team wouldn't be added to the roster, which could lead to issues later on because one of the teams for a match wouldn't be found, and then the entire match would be invalid, meaning the other team wouldn't be able to receive the notification either (the match would be skipped when sending notifications). However, making the fetcher more lenient could lead to other potential issues as well, since I use a person's email to unique identify them; if a row is missing the email, there

would be no way for me to accurately connect the row with another person, since I don't want to assume that rows with the same first and last name necessarily represent the same person. This also introduces the possibility of a team not having any valid emails to send to, which would also need to be handled. In summary, I didn't think I would be able to fix this within this semester, but after more thought is put into it, perhaps future work could be done to address this very important issue.

Another thing that someone mentioned was a bit of confusion around the subscriptions feature. Apparently some people at the tournament had come up and asked how to subscribe themselves to other teams, but couldn't find the subscriptions page even when they logged in. That would probably mean that the email they logged in with wasn't in the roster, but I wasn't aware of this until I saw the survey results, so there was nothing I could have done about that specific instance. Perhaps making the instructions and restrictions on the subscriptions page clearer would aid in future understanding of it.

Please see Appendix E.4 for the answers to the survey.

## **5. Conclusion**

### **5.1 Future Work**

There are still a few things in the system that could be worked on. As of this writing, I have 3 issues open on the GitHub repository:

- Only missing contacts should be added to Mailchimp (discussed at the end of section [4.2.2 Scaling Issue](#))
- Fetching the roster may result in missing teams if the team is composed of invalid rows (discussed in section [4.2.4 Survey](#))



- Admins do not receive emails for subscribed teams
  - Admin users have access to the Subscriptions page and therefore may subscribe to team notifications, but they might not be added as contacts in the Mailchimp audience, which means any campaigns sent to them will simply ignore those email addresses.

There was also another requested feature from the survey after the UVM Tournament to have the ability to send an email with custom body content. That is, there would be a text box input where the user would be able to type in exactly what they want the body of the email to say. This feature would mainly be to completely remove the need to use Mailchimp during a tournament at all, since it would cover pretty much all the cases where my users would need to use Mailchimp to send an email. This is somewhat related to section [3.4.2 Custom Email Content](#), but more so about completely setting the body content rather than customizing it on a per-match basis.

In addition to these, there is also potential to expand the system by adding more features beyond notification sending. The users that I regularly met with had also expressed interest in other features that I did not have time to implement, such as live school ranking results. This is definitely something I would have liked to work on if I had more time.

## **5.2 Lessons Learned**

While I have created web services in the past (even with the same technologies and tech stack), I still learned a lot throughout the creation of this particular project. This was on the larger side of any projects I had done, so code organization and modularity were very important throughout. Since I was also integrating with multiple other sources (Google Spreadsheets and Mailchimp), having all the utilities split between different files was very useful. I also experimented a lot

more with Bootstrap and Jinja2 macros, which helped me achieve a modularly-built frontend that I was happy with, though UI/UX design is not my strong suit.

### **5.3 Project Self-Assessment**

Overall, the project went very well. I created a system to organize the team information of a tournament and send personalized emails. I was able to add many features to simplify the life of the Communications Manager at an ECTC tournament, and it was (mostly) successfully used at a real tournament. Although there were a few issues keeping it from the perfect application, reactions to the system were very positive, and the users have expressed interest in continuing to use the system in the future.

## Appendices

### A. Service Accounts

A service account is a special type of Google account intended for non-human users that can be authorized to access data using Google APIs. Just like any other Google account, documents must be shared with it for it to have access, and it can also view any publically accessible document. It can create, view, edit, and delete documents, but for the purposes of my project, I only needed one with read scopes.

There are some nice instructions in `gsread`'s documentation about how to create a service account (<https://docs.gsread.org/en/latest/oauth2.html#for-bots-using-service-account>). I also put this link on the Global Settings page for the Super Admin who needs to create it. Going through the steps will result in a credentials JSON file, which the Super Admin should upload on the Global Settings page.

### B. Roster Spreadsheet Specifications

These specifications are also available to the user through a help link on the Notifications page.

In the given TMS spreadsheet in the Admin Settings page, there should be a worksheet called `FULL ROSTER` with the full teams roster. The first row should be a header row with the following headers:

- **First Name:** The first name.
- **Last Name:** The last name.
- **Email:** The email. Must be unique in lowercase.
- **Role:** The possible roles are “Coach”, “Athlete”, and “Spectator” (case insensitive).
- **School:** The school.
- **Team Code:** The team code (e.g., “PA1”, “Women’s B2”, etc).

- **Fighting Weight Class:** The possible weights are “Light”, “Middle”, “Heavy”, and “Alternate” (case insensitive).

The headers themselves are case insensitive and must be unique in the headers row. The other columns will be ignored.

The email, role, and school columns are required for each row, and at least one of the name columns are required for each row.

Coaches and spectators (technically not required) should be specified with values in the required columns. Since they aren't on any teams, the “Team Code” and “Fighting Weight Class” values will be ignored for those rows.

Athlete rows represent the athlete's role on a single team, so the same athlete may have multiple rows if they are on multiple teams. The “Team Code” value is required, and the “Fighting Weight Class” value is required for sparring teams. Poomsae teams should not have a “Fighting Weight Class” value unless it is “Alternate”.

When the “Fetch Roster” button is clicked, the roster will be fetched as specified. Logs will be generated for each event that happens, such as adding a school, user, or team. Errors and warnings will also appear for invalid rows, which may be helpful to check; for example, if a row is missing an email, if a school doesn't have any athletes or teams, etc. The existing saved roster is also wiped so that roster changes will be reflected.

After the fetch, all the contacts will also be added to Mailchimp so that they can receive notification emails. If an audience tag was given on the Admin Settings page, these contacts will also be tagged with that tag.

### **C. Subject Placeholders**

These specifications are also available to the user through a help link under the subject input.

The email subject for match notifications can be customized on a per-match or per-team basis by using placeholder variables. These placeholders will be automatically replaced with their corresponding values in the subjects of the sent emails. The available placeholders are:

- `{match}`: The match number (required in the subject).
- `{division}`: The division (as seen in the TMS).
- `{round}`: The round (as seen in the TMS).
- `{blueteam}`: The name of the blue team.
- `{redteam}`: The name of the red team.
- `{team}`: The name of the team receiving the email (will be different for both teams).
  - This will cause two campaigns to be sent for each match so that each campaign can have different subjects. Any additional recipients (such as coaches, spectators, or team subscribers) will receive the email for only the relevant team.

Placeholders are case insensitive and may be given multiple times. Invalid placeholders will be rejected.

For example, a simple subject could be “Match `{match}`: Report to Holding”. If sent for matches 701 and 702, this would result in the actual email subjects “Match 701: Report to Holding” and “Match 702: Report to Holding”.

To prevent XSS attacks, and because I wasn’t sure what the valid character set for a Mailchimp subject was, the subject is restricted to letters, digits, spaces, and any of the following characters: `-_+. , !#&() [ ] | : ; ’ ” / ?`.

On the server, I used a handwritten parser to go through a given subject and validate the placeholder values and valid characters. I then used the Python `str.format()` method to inject the proper values for each team’s subject. If the subjects were the same for a match, then the

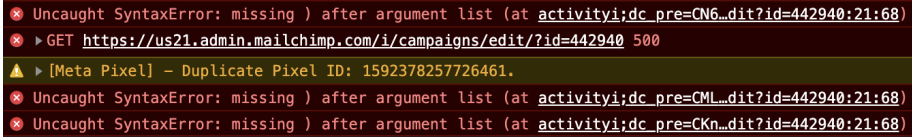
recipients were combined and it was sent as a single campaign. Otherwise, two separate campaigns were created and sent.

## D. Conversation with Mailchimp Support

I contacted Mailchimp Support for help with a “deep, internal error” when trying to set the content of a campaign using a template id. If they could resolve this issue, I would be able to implement the ideal flow I wanted for my users. Unfortunately, they were not able to. Here is the full conversation I had with Mailchimp Support, with the issue finally being resolved three and a half weeks (25 days) after initially bringing up the issue.

2023-03-23 13:17:41	Me: Hello. I am trying to create a campaign with a saved template through the API, but upon trying to send it says that the campaign is not ready to send, and looking at the campaign on the website shows that there is no content (and no template) for the campaign. I tried using the `set_content` endpoint with the template id and got an error message "A deep, internal error has occurred during the processing of your request. Please contact support." I can share the exception and instance in the error message too if that would be helpful. A workaround I've considered is using `set_content` to set the html myself (but I haven't figured out how to get the html content of a template), but I would prefer to go with attaching a Mailchimp template to a campaign.
2023-03-23 13:18:42	Mailchimp Support: Thanks for contacting Mailchimp Support. You may experience longer than average chat wait times. Thank you for your patience, and we look forward to assisting you.
2023-03-23 13:22:42	Mailchimp Support: We apologize for keeping you waiting. We're making every effort to assist our customers as quickly as possible, and we look forward to helping you as soon as we can.
2023-03-23 13:27:42	Mailchimp Support: We appreciate your patience while we experience an increase in support requests. We're working as quickly as we can, and one of our agents will be with you soon.
2023-03-23 13:32:55	*** Dragutin joined the chat ***
2023-03-23 13:34:10	Dragutin: Hello! Thank you for contacting Mailchimp Support. I understand you're chatting in for assistance with errors you're getting as you're trying to create and send a campaign with the API; is that correct?
2023-03-23 13:34:16	Me: Yes

2023-03-23 13:34:53	<p>Dragutin: Thank you! I am happy to help as much as I can.</p> <p>How long have you been getting that alert about the deep internal error?</p>
2023-03-23 13:35:29	<p>Me: Since yesterday. Other endpoints work fine. It seems to only be related to trying to add a template id to an existing campaign. Also, once I get this error, trying to view the campaign on the website shows an error page as well.</p>
2023-03-23 13:36:43	<p>Dragutin: Thank you! I am monitoring the API on my end right now. Can you try to add the template ID to the existing campaign now?</p>
2023-03-23 13:37:12	<p>Me: Sure! I'll create a test campaign.</p>
2023-03-23 13:37:59	<p>Dragutin: Thank you! Let me know when you get that done and we'll go from there.</p>
2023-03-23 13:38:07	<p>Me: (Sorry, my environment is taking a bit to start up.)</p>
2023-03-23 13:38:17	<p>Dragutin: Not a problem at all!</p>
2023-03-23 13:41:04	<p>Me: The campaign id is "b7d47f1f51". I successfully created it without a template id. I then used the `set_content` endpoint to add a template with id 31379 (a user template). I got the following error:</p> <pre>{   'detail': 'A deep, internal error has occurred during the processing of your request. Please contact support.',   'exception': '7fae68ac464d695af8336d3ae84f253e',   'instance': 'b57755cf-3c76-63ba-ad5f-5552f843e744',   'status': 500,   'title': 'Internal Server Error',   'type': 'about:blank' }</pre>
2023-03-23 13:41:30	<p>Me: (I am using the `mailchimp_marketing` Python package.)</p>
2023-03-23 13:42:56	<p>Dragutin: Thank you so much! I am reaching out to an internal specialist on this for you right now. Please stand by.</p>
2023-03-23 13:43:04	<p>Me: Thank you!</p>
2023-03-23 13:44:16	<p>Dragutin: Quick question, do you have the full timestamp on that error on your end?</p>
2023-03-23 13:45:12	<p>Me: Sorry, no... I'm using a Google Colab Notebook and it says the execution finished at 13:40 pm Eastern, if that helps</p>
2023-03-23 13:46:21	<p>Dragutin: It does indeed! Please stand by.</p>
2023-03-23 13:57:50	<p>Dragutin: I am still here and awaiting a response from the specialist. I</p>

	appreciate your patience!
2023-03-23 13:58:00	Me: Thank you very much for your help
2023-03-23 14:09:50	Dragutin: Do you see anything else on your end about any other kinds of errors?
2023-03-23 14:10:12	Me: This is the only error I've seen throughout my testing
2023-03-23 14:13:26	Me: I just tried viewing the campaign I created on the website and I saw these errors in the console. Maybe this could help?
2023-03-23 14:13:44	Me: Uncaught SyntaxError: missing ) after argument list (at activityi;dc_pre=CN6v6sTT8v0CFQ2inwodLa0Hfg;src=9894771;type=klp;cat=entir0;match_id=182378926;ord=1678932409004.3ikqgtd;gtm=45fe33k0;gcs=G111;auiddc=23738992.1678570977;u1=182378926;u10=undefined;u11=en;u12=undefined,undefined,undefined,undefined;u22=admin;u3=undefined;u4=1842491818.1678570977;u5=1678932409004.3ikqgtd;u6=2023-03-23T14:12:35.218-04:00;u7=page_view;u8=campaigns;u9=edit;gdid=dYWJhMj;~oref=https://us21.admin.mailchimp.com/campaigns/edit?id=442940:21:68)
2023-03-23 14:14:00	main-VNTK3OYP.js:1 GET https://us21.admin.mailchimp.com/i/campaigns/edit/?id=442940 500
2023-03-23 14:14:24	Dragutin: Thank you! I will relay that to the internal specialist as well.
2023-03-23 14:14:32	Me: 
2023-03-23 14:14:58	Me: Thank you
2023-03-23 14:34:23	Dragutin: I am still here and working with the specialist right now!
2023-03-23 14:34:34	Me: Thanks for the updates!
2023-03-23 14:42:39	Dragutin: Can you tell me exactly where you're placing the call?
2023-03-23 14:42:52	Me: What do you mean by "where"?
2023-03-23 14:43:25	Dragutin: Are you using an integration or are you doing it directly in Python?
2023-03-23 14:43:35	Me: Directly in Python
2023-03-23 14:44:14	Dragutin: Thank you! So we're going to need to go ahead and take this case to email, so our specialists can investigate further.



2023-03-23 14:44:36	Dragutin: I will reach out at xxxxx@gmail.com as I get updates.
2023-03-23 14:44:55	<p>Me: Here is the code I used that resulted in the error:</p> <pre># create campaign subject = 'Error Test (with template)' response = client.campaigns.create( {   "type": "regular",   "recipients": {     "list_id": AUDIENCE_ID,   },   "settings": {     "subject_line": subject,     "preview_text": subject,     "title": subject,     "from_name": from_name,     "reply_to": from_email,   },   "content_type": "multichannel", } ) campaign_id = response['id'] client.campaigns.set_content(campaign_id, {'template': {'id': TEST_TEMPLATE_ID}})</pre>
2023-03-23 14:45:03	Me: Okay, sounds good.
2023-03-23 14:45:15	Dragutin: Thank you! I will pass that along as well.
2023-03-23 14:45:36	Dragutin: I'll go ahead and end our chat session now; thanks for contacting Mailchimp Support. Have a great day.
2023-03-23 14:45:40	*** Dragutin left the chat ***
2023-03-23 14:45:42	Me: Thank you for all your help!
2023-03-28 08:59	<p>Me: Hello,</p> <p>Good morning! I was wondering if there were any updates on this issue. I can provide more information if needed.</p> <p>Thank you!</p>
2023-03-28 11:48	<p>Dragutin: Hi,</p> <p>Thanks for writing!</p> <p>I am still awaiting a reply from our internal specialists; I will reach out with updates as soon as possible.</p> <p>Dragutin</p>

2023-04-01 12:22	<p>Me: Hi again,</p> <p>Sorry to keep pushing. I was wondering if there were any updates now? I have a project that is depending on this feature to work, and I'm currently having to try some other undesirable workarounds. Any more information would be greatly appreciated!</p>
2023-04-03 15:23	<p>Dragutin: Hi,</p> <p>Thanks for writing! I have reached out to our technical specialists and am awaiting a response.</p> <p>Dragutin</p>
2023-04-14 18:03	<p>Mailchimp Support: Hello,</p> <p>We wanted to reach out one last time regarding the ticket (10868720). Since we have not heard back, we will go ahead and close this ticket.</p> <p>We hope you resolved the issue successfully. If you still need help, reply to this email within the next few days and your ticket will reopen automatically. As always, you can open a new support request via your paid Mailchimp account.</p> <p>Thank you, Mailchimp Support</p>
2023-04-16 09:58	<p>Me: My issue was never resolved. I was told I would receive updates but it's been two weeks and I haven't heard anything.</p>
2023-04-17 12:35	<p>Wyatt: Dear Valued User,</p> <p>Thank you for contacting Mailchimp Support. Hope your weekend was a good one.</p> <p>Let's jump back into the issue. I apologize for the very long delay. We just received information about the error, "<b>A deep, internal error has occurred during the processing of your request. Please contact support,</b>" when using the <b>set_content</b> endpoint with the template id.</p> <p>After some testing was completed, our Mailchimp engineers have concluded that the <b>new builder templates</b> are not supported by the <b>API</b>. The engineers stated the system is unable to interpret which template type it is, thus the error message is being received. Since this is the case, the best work around would be to use the <b>Classic Builder</b> templates instead. <a href="https://www.screencast.com/t/Y5tXG6nLPFN8">https://www.screencast.com/t/Y5tXG6nLPFN8</a></p> <p>I've included the article on how to switch email builders below:</p>

	<p><a href="https://mailchimp.com/help/design-an-email-classic-builder/?_gl=1*1dmluor*_up*MQ..*_ga*MTMwNjE4MDczNS4xNjgxNzQ4ODg2*_ga_N5HD1RTH6E*MTY4MTc0ODg4NS4xLjAuMTY4MTc0ODg4NS4wLjAuMA.#Choose_the_classic_builder">https://mailchimp.com/help/design-an-email-classic-builder/?_gl=1*1dmluor*_up*MQ..*_ga*MTMwNjE4MDczNS4xNjgxNzQ4ODg2*_ga_N5HD1RTH6E*MTY4MTc0ODg4NS4xLjAuMTY4MTc0ODg4NS4wLjAuMA.#Choose_the_classic_builder</a></p> <p>Thank you for being so patient during the troubleshooting process. Please let me know if there is absolutely anything else I can assist with. I am always here to help.</p> <p>Wyatt</p>
--	--

## E. Evaluation Notes

### E.1. Evaluation with Cecilia Kim, 4/21/2023

Task: Given the TMS spreadsheet link (my testing spreadsheet), complete all the admin settings

- Went to Admin Settings page
- Pasted the spreadsheet url
- Selected the test audience
- Set the audience tag
- Selected the template folder

Task: Send a match notification for matches 701-705 to report to holding

- Went to Notifications page
- Saw the “roster not fetched yet” error
- Fetched the roster
- Fetched matches 701-705 (ignore the warnings)
- Selected the proper template
- Tried to set subject without placeholders, resulting in validation errors
  - Used actual match numbers {701-705}
  - Tried {701}
  - I explained the placeholders
- Sent the notification email

Task: Send a match notification for matches 701 and 801 for final call to holding

- No issues

Task: Send a blast notification to the PA division for Women’s A team to start warming up

- Was initially confused about the PA division
  - That's my bad; I didn't have the Women's A division in my test data, but the templates only ask sparring teams to warm up. In future evaluations, I will add a Women's A team for this task.
- Selected the proper template
- Set the subject
- Sent the blast notification email

Task: Send a blast notification to everyone about a lost item

- No issues

Task: Check whether an email was sent for Match 702

- Went to Sent Emails page
- Went to Matches Status page
- Reported no (correct)

Task: Check the list of all emails sent

- Went to Sent Emails page

Any thoughts / comments / suggestions / complaints?

- Subject placeholders is not clear (that the user needs to use them)
- Would be nice if subject auto-populated based on selected template
- "It looks beautiful"
- Very easy to use besides the subject

## **E.2. Evaluation with Yolian Lao, 4/21/2023**

Task: Given the TMS spreadsheet link (my testing spreadsheet), complete all the admin settings

- Went to Admin Settings page
- Pasted the spreadsheet url
- Opened the url, just to see what it does
- Selected the test audience
- Left the audience tag blank ("I don't know what this is, so I'll leave it blank")
- Selected the template folder

Task: Send a match notification for matches 701-705 to report to holding

- Went to Notifications page
- Saw the "roster not fetched yet" error, so fetched the roster

- Fetched matches 701-705
- Selected the proper template
- Entered subject without placeholder
  - Tried {701-705}, {match 701}
  - Eventually got {match} through trial and error, but was surprised that it worked
- Checked “Send to School Coaches”
- Sent the notification email

Task: Send a match notification for matches 701 and 801 for final call to holding

- Entered match numbers, but didn’t fetch yet
- Tried sending, but got an error “No matches”
- Fetched matches, then sent the email with no issues

Task: Send a blast notification to the Women’s A division for Women’s A team to start warming up

- No issues

Task: Send a blast notification to everyone about a lost item

- No issues

Task: Check whether Match 702 received an email

- Went to Sent Emails page
- Went to Matches Status page, then back to Sent Emails page
- Looked for “Match 702” in list of Sent Emails, and reported no
- I explained the Matches Status page

Task: Check the list of all emails sent

- Went to Sent Emails page

Any thoughts / comments / suggestions / complaints?

- “If this were my grandma doing this, she would not be able to figure this out” “But then again she’s not like tech savvy”
- When choosing the division for blast notifications, should add an “everyone” button so it makes more sense
  - In response to this, I changed the “Division” selection to a “Recipients” selection, which includes an option to send to either the entire audience (if no audience tag was set) or the given audience tag.

- “It’s a very nice system. It’ll help on the operating end.”

### E.3. Evaluation with Elise Kim, 4/22/2023

Task: Given the TMS spreadsheet link (my testing spreadsheet), complete all the admin settings

- Went to Admin Settings page
- Pasted the spreadsheet url
- Selected the test audience
- Set the audience tag
- Selected the template folder

Task: Send a match notification for matches 701-705 to report to holding

- Went to Notifications page
- Fetched the roster
- Entered each match number individually (701, 702, 703, 704, 705)
  - “It would be nice if you had examples – oh you do”
  - Saw that a range was possible, so changed to “701-705”
  - Fetched the matches
- Selected the proper template
- Entered a subject, then saw the help text and added “{match}”
- Sent the notification email

Task: Send a match notification for matches 701 and 801 for final call to holding

- No issues

Task: Send a blast notification to the Women’s A division for Women’s A team to start warming up

- “I wouldn’t need to select matches because it’s a blast notification, right?”
- Sent the blast notification email

Task: Send a blast notification to everyone about a lost item

- No issues

Task: Check whether Match 702 received an email

- Went to Matches Status page
- Reported no (correct)

Task: Check the list of all emails sent

- Went to Sent Emails page

Any thoughts / comments / suggestions / complaints?

- Auto-populate subject from selected template
- “Very sleek”
- There is enough text that it’s not overwhelming but it always answers whatever I wanted to know
- User can always see whether something worked or not (confirmations are nice)

#### E.4 Survey after UVM Tournament

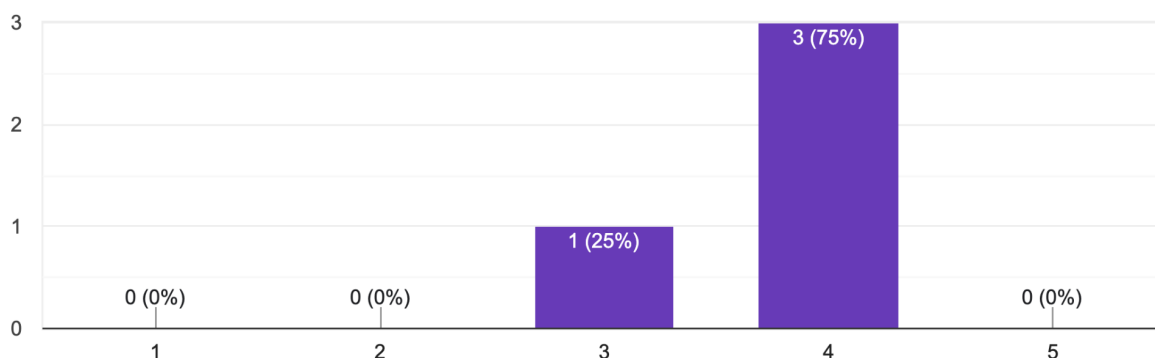
*Name (optional)*

- Vi Belt
- Laney Flanagan
- Rae

*How was the overall experience using the system?*

How was the overall experience using the system?

4 responses



*Could you briefly expand on why you chose the rating you did?*

- It worked well for me! I was one of the people directly assigned to communications, and it sent the majority of emails I had asked it to. I didn't have any major problems, but I also don't really understand the intricacies of the program and so I didn't have real expectations
- I couldn't give it a literally perfect because we did not need to call to address some errors we saw. Once those were fixed, it was fine!

- When all of the rosters were properly loaded for poomsae and minor bugs fixed, it worked perfectly and was super easy to use. I literally did not even have to think when sending out match reports. However, I can't speak to how it worked in the afternoon, since the sparring rosters had not been loaded into the TMS and we couldn't use it :(
- When we were able to use the TNS, it seemed to work great! However, we struggled to find a way to 1)include coaches in emails and 2) to re-fetch the names once the sparring brackets were created and were unable to use it for the second half of the day (sparring). So, explicitly when using the TNS, I would probably give it a 4 rating but lowered that in terms of overall experience since we were unable to use it for a portion of the day.

*Besides the known bugs that I attempted to fix during the day (fetch roster timing out and emails not sending to proper recipients), did you have any other issues using the system?*

- Nothing that is unique, just the not sending emails to coaches/spectators
- We weren't able to fetch the rosters in the afternoon but I think that was a consequence of something Will didn't have time to do....so I don't think this was a problem with your system.
- This was mainly a problem when the coaches hadn't been loaded into the roster yet, but if a team had no valid emails (i.e. 1 person team with a bad email, and no coaches either) or one team dropped out of the tournament, the entire match was invalidated. We assumed that the email would still send to the valid team, but the whole match was scratched. In that situation, it would've been preferable to still be able to send the email (or a modified one--like "representative report to head table") so that the team could be notified.
- I think those were it. I'm not exactly sure what the issue was with being unable to get the proper names fetched to use for sparring.

*Were there any features that caused a bit of confusion when trying to learn how they worked? Or were confusing in general?*

- Not really! I found it very straightforward
- N/A
- A few people asked how to subscribe themselves to other teams and ran into trouble. The person I spoke to logged in successfully and could see match statuses, but couldn't find the subscriptions tab. I could see it on mine as an admin account but not on theirs. Not sure what the problem there was?



- Nope, the features were pretty clear once explained/demo-ed.

*I have found that the subject placeholder variables (such as "{match}") were confusing for people to understand. In particular, they didn't immediately understand that the placeholders should be typed as-is into the subject input box, and they would be automatically populated for each match email that was sent (e.g., the subject "report {match}" would turn into "report 701" for match 701 and "report 702" for match 702, etc).*

*Did you also find the use of these placeholder values confusing? If so, what change to the interface / instructions / help text would have made it easier for you to understand its usage?*

- I did not find them confusing to be honest
- The simplest case of {match} was perfectly fine for me, especially since that was the example in the text box. However, I wasn't confident enough to try any more complicated placeholders. The help box with keys and examples was helpful but it opens over the window--as in, I couldn't easily simultaneously look at the instructions while typing the email. I think that would be nice, especially for someone who doesn't know the syntax well, but the explanations were clear enough.
- I think "match number" would be more clear or maybe "match ###" but also I think I'd like the emails to say "Match 701 report" rather than the above - that's largely personal preference to match how we actually speak at tournaments.

*Would you use the system again in the future?*

4 responses: 4 for Yes, 0 for No

*If yes, do you have any other feature requests, now that you've used the system? If no, why not?*

- I don't, mainly because I didn't have any real issues with it
- not yet because we didn't use the system as it's currently built for the full day.
- It would be nice to clarify under which circumstances the coaches are emailed. We clicked send to "School Coaches" for all of our announcements, and coaches received them fine. Otherwise I'm not sure how coaches would get the emails, since no team specifically has 1 coach assigned to each one (i.e. every athlete has a Team Code like PA1 or something, but no coach would have a Team Code). I'm not sure if I misunderstood it but making that more clear would be a good thing I think.

- I'm wondering if there's a way when a match is considered invalid to still send to the other recipients... for instance if match 701 is loading the blue team info incorrectly, it would be helpful to still be able to send the match to the red team. Also, I don't think it's easy, but a "free text" option would be great.

*Do you have any other comments, suggestions for improvement, feedback, words of encouragement, etc?*

- Nope! Thank you so much for working on this and making our jobs a bit easier. I hated getting all of the emails all day but had no real way of changing it (other than just unsubscribing outright), and this seems like a great solution to that.
- Thank you so much for your time and effort building this system. It definitely makes communications much easier and the morning was a breeze. Ideally we'd be able to continue using this system and troubleshoot issues on our own.
- I tried to give a lot of feedback for your project! Overall, the system was super easy to use--almost foolproof once it was going. I wish the roster problem could have been solved faster so we could use it more! Most of the problems we experienced were on the TC side when it came to rosters/TMS/appropriate templates. Thanks for all your hard work!!!
- This was really cool, thank you for creating it!