Automating Summer Camp Logistics: Building a Custom Schedule Generator and Attendance System for ChugBot

CS4991 Capstone Report, 2024

Jonah Werbel
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jwerbel@virginia.edu

ABSTRACT

Camp Ramah in New England used many outdated systems for camper management, often utilizing paper tools resulting in unmaintainable records and extra work for staff. Using a pre-existing custom software camp developed as the basis, I worked with year-round staff to identify ways to expand its capabilities in daily camp management. After collecting a set of requirements, I designed and developed solutions from scratch. My work built upon the existing work which required my familiarization with the existing code (to understand conventions and prior design choices): PHP (the language used throughout), MySQL (the database system), Docker (used for local testing), and some Amazon Web Services (AWS) systems (for deployment). The solutions were designed, developed, and operational before the camp season began and they performed well during their inaugural season with consistent, near-daily usage streamlining camp operations. Stakeholder feedback will continue to guide updates and adjustments as I continue my software development role with camp.

1. INTRODUCTION

A picture-perfect scene is unfolding at Camp Ramah in New England (CRNE, colloquially known as "Ramah"). Children are happily playing sports, swimming and doing crafts. Counselors are joyfully leading electives and planning programs for their campers. Everyone is having a wonderful day at their home away from home. Suddenly, all counselor phones buzz. "RED ALERT," a text message reads. "CAMPER MISSING."

Sending this message is one of the worst nightmares of Ramah leadership, who far prefer working to provide a traditionally fun "camp" summer where children make memories, try new things, and have fun. However, they are responsible for ensuring the safety of the campers, guaranteeing each one makes it home at the end of the summer.

One important step in keeping campers safe is verifying they are where they are supposed to be. Both in case of emergency and to dissuade trouble, being present keeps eyes on campers and enables quick responses if necessary. I implemented a mechanism to generate individual schedules for campers, telling kids where they should be at any time. Additionally, I created an online, real-time attendance module so activity leaders could verify all kids are present and other staff could check the status of their campers. Together, individual schedules give campers the agency to be present and permission to behave, while the attendance system enables staff to verify camper presence and respond to any absences.

2. BACKGROUND

At Ramah, campers are given the option to choose from several elective options. Each camper ranks their top six choices for three daily periods. Until 2015, the ranking and

assigning process was done manually. This time-consuming process was eliminated when a parent created an automated system called "ChugBot." The Hebrew word *chug,* referring to a class or activity, is used at camp to describe elective periods. Campers log in to rank their preferences for each period.

3. RELATED WORKS

The method for creating individual schedules for campers is based on the idea of a mail merge. Wempen (2008) discusses mail merges and how they generate personalized documents based on a template using a set of structured data. Her book further describes using Microsoft Word to perform one. I extended this idea by creating a module to build campers' customized schedules. After creating the template and inserting proper tags, an individual schedule is automatically made for each camper with their information.

Apandi & Mohamed (2012) provided critical insight into necessary components for a rudimentary attendance system with their development of a classroom system. There is a need for marking and viewing attendance, in this case with different user roles. I also needed to ensure attendance records were easily digestible, communicating quickly who was present or absent. Furthermore, I needed to consider what the intended action would be upon learning the attendance records. Their system generated a report by student; mine generated a report by elective period to easily distribute to other staff members.

4. PROJECT DESIGN

Extending ChugBot to include both a schedule builder and attendance module required a significant amount of work but allowed me to make many design choices and decisions. I consulted with camp staff to determine their priorities but largely relied on my own understanding of camp operations to develop a set of requirements. I implemented each part iteratively, providing updates as work progressed before releasing my code to be reviewed. Finally, I deployed the updates to live ChugBot instances.

4.1 Schedule Builder

The core requirement for the schedule builder was to provide campers with paper copies of their elective assignments. From experience, I know it is helpful for many campers to have full schedules of the day, allowing independence and knowledge of the day. Therefore, I recognized the benefit of allowing customization of the output, enabling creation of individual schedules. As a result, I understood how creating a mechanism to save those templates for frequent usage would minimize repetitive work for staff. The final requirement is specific to operations at CRNE: older campers train in one sport for a final competition during a month-long session but have different electives weekly. Elective assignments are saved in the database by week they begin, so I needed to include a way to include assignments from multiple weeks.

4.1.1 Design Details

The user begins by selecting the age group they are generating schedules for, and if a template was previously saved the user will see an option to select it. The week to use elective assignments from is also selected.

The schedule template itself is typed into a rich text editor below the other options. TinyMCE enables users to type and format text like standard document editors (n.d.) It also interacts with the website, allowing me to include buttons which automatically add placeholders at the location of the cursor (e.g. for name, bunk, electives, etc.) and to later retrieve typed text as HTML (n.d.).

Users needed the ability to override which week certain information is retrieved from, and a set of dropdowns are located below the editor in an expandable window.

When the template is ready and a user generates schedules, a form is submitted to the server. A SQL query is built to retrieve camper information and their elective assignments (for

correct weeks). Then ChugBot loops through each camper and replaces the placeholders in the template with their information, adding the resulting schedule to a viewable page. Printing the webpage maintains page breaks between each camper's schedule for clarity.

4.1.2 Challenges

The largest challenge of this feature was designing the SQL query to retrieve camper information and elective assignments and letting it include assignments from different weeks. I first made the query work for one session, generating a new column per elective in the resulting table. Then, I needed to specify individual weeks to retrieve electives from. I designed the query in a separate document, testing it frequently using a test database in Docker. Once the query worked, I needed to write PHP code to build, execute, and use the results from the query.

The other challenge was how to print the generated schedules. I originally planned to create a PDF of the schedules to download or print but struggled to find a good package to create PDFs using PHP. Additionally, I could not find a package which allows both English and Hebrew characters. I instead wrote the schedules to a new webpage and used CSS to style it, hiding non-relevant features and adding page-breaks after each schedule.

4.2 Attendance Module

The attendance module replaced paper attendance sheets. It needed to allow an elective leader to sign in and mark which campers are present, and separately let a unit head sign in to view who is missing. Elective leaders also must be able to update attendance if kids show up late or go missing mid-activity. It also needed to update if camper assignments switch mid-week, and needed a way for the administrator to indicate which week is currently running. Most importantly, the entire module needed to be simple to use, and the website needed to work on a mobile phone.

4.2.1 Design Details

I extended the existing admin login page to work for two new user roles: elective leader and unit head. I made one generic user for each role with a shared password across staff for flexibility. When those users sign in, their default action is to either take attendance (elective leader) or view attendance (unit head). They are greeted by similar forms to specify date, age group, and time of day, plus elective leaders indicate the activity they run.

After searching for their activity, elective leaders see a list of enrolled kids. They select a checkmark next to each present camper and submit the record by pressing a button below the camper list. If updating attendance, campers previously marked present have their checkbox automatically selected.

Unit heads have two views available: by activity (view currently missing campers) or a historical matrix (view trends/patterns). Both utilize consistent icons and coloring to indicate if a camper is present, absent, or if attendance was not taken for their activity. The view by activity has a table with a row for each camper in the age group (sorted by bunk and name) and their elective assignment. The row is highlighted red if absent, yellow if attendance was not taken, or white if present. Staff can also flip a toggle to only see campers not marked present. At the bottom of the page, a button marked "Copy Missing Camper Report" generates a plaintext report of missing campers and where they should be, easily pastable into a staff group chat.

Alternatively, the matrix view provides a summary over a date range for one age group. Each camper is in one row with columns for each date. The intersecting cell has an icon showing their status, and the cell is highlighted with the same color scheme. Tapping the icon shows what elective the camper was assigned.

Finally, administrators designate active weeks for electives. Once new assignments take effect, they update ChugBot so attendance is taken with correct camper lists.

4.2.2 Challenges

The main challenge was simply the scale of the module. There were many components to build with a lot of pieces dependent on each other, but dividing the work into sections made it significantly more manageable. I was also in constant communication with camp staff to ensure I built what they needed.

Testing also proved to be challenging as I spent much time working on fulfilling the requirements and could not objectively test it myself. So, I enlisted others to beta test. I asked them first to execute basic tasks (take, view, and update attendance) before asking for general feedback, and finally encouraged them to try breaking the system (I ran a test server using Docker with sample data so if something went critically wrong, I could simply launch a new instance). Their feedback was invaluable in finding bugs, updating design choices, and improving directions.

5. RESULTS

Both features were delivered before camp began. Due to unrelated circumstances, the schedule generator feature was not used significantly during the summer. Camp staff experimented with it, but I am not aware of schedules being distributed to campers.

Conversely, the attendance module was widely used. It was introduced to all staff and all attendance records were taken using it. According to the database following the conclusion of the summer, 29,673 individual attendance records were taken across 1,275 individual activities (each day an elective is offered is a unique offering). No data is available for comparison from past summers but in 2024, 26,804 of the 29,673 attendance records were "present" (90.3%).

6. CONCLUSION

My ChugBot work enhanced operations at CRNE, modernizing attendance tracking and providing campers individual schedules to ensure safety, optimize staff time, and provide campers agency over their actions. Attendance can now easily be tracked in real-time and historically with a system designed for camp's needs. And campers can now easily have their own daily schedule information. These features transformed camp administration.

Personally, I enhanced my skills in software design, user-centric problem solving, and iterative development. Working with PHP, MySQL, Docker, and AWS honed my technical skills and provided practice software engineering experience, equipping me with essential tools for future development work.

7. FUTURE WORK

Following the summer, I met with my supervisor to discuss additional work. The only necessary change is an update to handle a use case we did not originally identify for the attendance module where campers in different age groups are in different "weeks" at the same time (P. Kekst, personal communication, October 22, 2024). Otherwise, I anticipate continuing to work on various tasks for the entire system (including updating styling, fixing bugs with "de-duplication" options, allowing electives to be uploaded in a CSV, and other tasks as identified). More Ramah camps are beginning to use ChugBot and my work will also aid their camp operations.

REFERENCES

Apandi, S. H., & Mohamed, R. (2012). Development of attendance management system: An experience. *Faculty of Computer Systems and Software Engineering, University of Malaysia Pahang, 26300*.

TinyMCE. (n.d.). TinyMCE; Tiny Technologies Inc. https://www.tiny.cloud/

Wempen, F., & O'Reilly Online Learning: Academic/Public Library Edition (2008). *Mail and Data Merges Using Word 2007*. Indianapolis, Indiana: Que.