Satori: Open-source Course Management System

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Megan Marshall

with

Madison Flynn
Jelena Liu
Daniel Mizrahi

May 6, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Megan Marshall

Capstone advisor: Aaron Bloomfield, Department of Computer Science

Satori: Open-source Course Management System

Madison Flynn
University of Virginia
mrf7pc@virginia.edu

Jelena Liu
University of Virginia
jl2gb@virginia.edu

Megan Marshall
University of Virginia
mem5ak@virginia.edu

Daniel Mizrahi
University of Virginia
drm7wb@virginia.edu

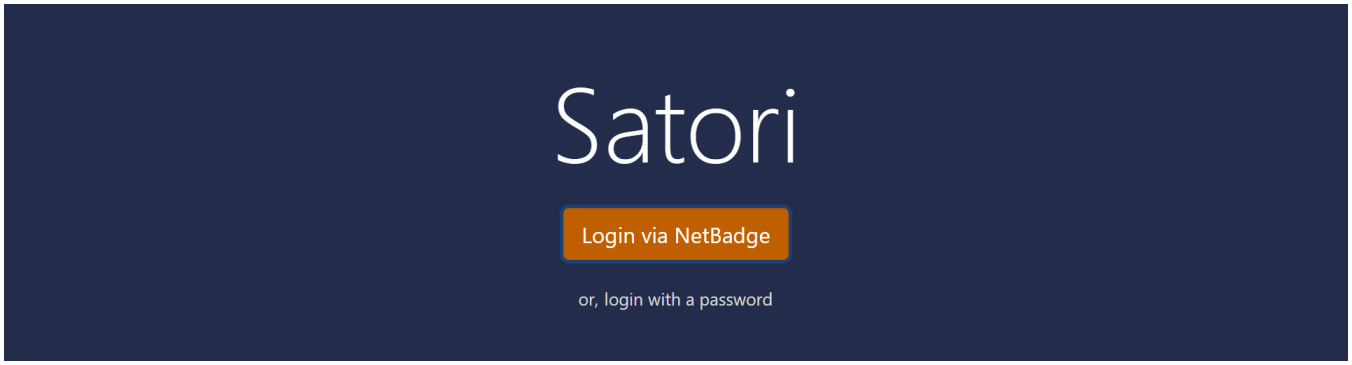

Figure 1. Satori Landing Page

Abstract

Satori is an open-source course management system that targets instructors who utilize online tools to supplement their teaching. The system includes capabilities for an office hours queue and a ticketing system for support requests. Infrastructure exists in the project that allows for the creation of new tools in the future, allowing professors to modify and personalize Satori to fit their needs. The development of this system aims to help courses run smoother for both students as well as course staff by providing an intuitive and comprehensive platform.

1 Introduction

This project focuses on creating a course management system that can be used in various subjects, but development was targeted around a University of Virginia course: CS 2150: Program and Data Representation. Currently, CS 2150 employs "Course Tools," a platform made in the early 2000s, which hosts an office hours queue and a support request system. The new system, named Satori, will have the same functionalities, with the possibility of additional features being added in the future.

The current "Course Tools" system works properly, but faces some issues. As the class size of CS 2150 continues to grow, reaching over 500 students in the Fall 2020 semester, it has become more difficult for the initial office hours queue to handle the sheer volume of students during busy sessions. The system becomes glitchy, with problems including students being kicked off the queue randomly, the queue freezing at inconsistent times, and the website having a slow response time during busy office hour times.

Furthermore, the support request tool also has a slow response time when there is a large amount of student request tickets in the system and is not compatible with Gradescope, a system the instructors use for automatic assignment grading and feedback. Our group hopes to improve the experiences of students in the course by mitigating problems the current system has.

In the 2019-2020 academic year, another group began the development of Satori, and created the overall architecture as well as the office hours queue. This year, our group is focused on the creation of the new ticketing/support request feature, as well as creating tests and fixing bugs along the way. In order to allow faculty to analyze office hours sessions and adjust staffing, we are also working on incorporating statistics into the system. The existing Satori platform was created on Django 3.2 using Python 3.7 and MySQL is used to create and manage the database. It is hosted in a docker container and run on a UVA Computer Science department server, Pegasus.

2 Related Work

As our group focused on developing a new ticketing feature on Satori, we looked into the existing methods students could use to request extensions or bring up other issues to instructors and teaching assistants. Many of these are able to

accomplish what is necessary for student requests, but lack cohesiveness or may be harder to use. We hope to incorporate the beneficial characteristics of similar systems into ours as well as improve on disadvantages they may have.

2.1 Course Tools

As mentioned above, the existing system that has been used was not built to support the growing enrollment size of CS 2150. In addition to the office hours queue and ticketing system, it used to also contain gradebook and assignment submission tools. However, the latter features have been replaced with Gradescope [1]. Approving extensions through the ticketing system and actually granting them has become less convenient due to the incompatibility between Gradescope and "Course Tools." It is also written in PHP, which is losing popularity, while Python has become one of the most popular languages [2]. With our project, we hope to replicate the capabilities of the "Course Tools" system and continue to build on them. We also hope to improve on the design of the platform.

2.2 Email

For many courses at UVA and other universities, the most common method of communication between students and course staff is through email. However, many instructors have to go through many emails a day, and it may be hard to sort through which pertain to extensions or issues students bring up. Also, when students email a single instructor, all responsibility of responding falls onto that individual, while a ticketing system allows for multiple possible respondents. Most students are already familiar with sending and receiving emails, and we hope that our new system is as straight-forward and intuitive as communicating through email.

2.3 Kytos

Kytos is another system that was created by UVA faculty in order to provide course tools to students for various courses such as CS 1110: Introduction to Programming. It contains features like the office hours queue, assignment submission and a gradebook. However, there is no general ticketing system that allows students to easily communicate with course staff without making comments directly related to a specific assignment. Our team hopes to make Satori as cohesive as Kytos while providing additional features as well as improving on the interface.

3 System Design

Our work on Satori can be spilt up into two primary systems: support requests and statistics. Support requests can be further broken down into two sections: tickets and workflows. Careful consideration went into designing each of these systems and their parts.

3.1 Tickets

One of the major features of our system is support requests. This ticketing system should be user-friendly for students, teaching assistants, and instructors to use. It also should be a efficient for students to submit their issues and for the course staff to respond to them. Tickets serve as an easier way for students to communicate with the course staff rather than using email as multiple staff members can answer tickets, issues can be solved quickly, and it centralizes student's needs all in one place. Making support requests fast and reliable is a top priority. This is important, especially when there are a lot of tickets displaying on the web page that need quick responses, because the more time students wait for a response from the ticketing system, the more time they waste not getting an answer to their question or a response to their issue. Additionally, the more time the course staff is waiting for a response from the ticketing system, the less they are able to give quick feedback or extensions. Testing was done to test permissions and to ensure that submitting tickets are fast.

Two important considerations when designing the tickets are the way in which tickets are ordered for course staff to respond to and being able to see past tickets/responses. The most straightforward solutions to the first consideration are to order them by when the ticket was submitted or by ticket ID. These both go hand in hand as a unique ID is given to a ticket when it is created, thus the higher the ticket ID the more recent the ticket was submitted and vice versa. We also needed to take into account if a student replied on an already resolved or stalled ticket, which our solution works for as the course staff would want to completely resolve the issue of an earlier ticket before a later one. At first, it could be a possible solution to show all tickets regardless of their status, which would work for students as they would at most submit a dozen or so tickets, however as the number of tickets gets larger throughout the semester, the web page would take longer to load all of the tickets. Knowing this, a separate table was created for resolved tickets and if the course staff wanted to look back at these tickets they will load once a dropdown for the table was opened.

As part of our larger goal of making this system easily adopted by other courses, an intuitive user interface was a priority in the development process. The ticketing system should not require a lot of background experience or a technical background to operate. We designed the forms, buttons, labels, tables, and links to be clear to know how to use.

3.2 Workflows

Workflows are the other integral part to support requests. Once a ticket is submitted, the responses from the course staff and students are called workflows. It needs to provide information quickly and accurately so that the course staff can tend to the issues of students. Making workflows quick

and reliable is a top priority. If a response is lost between a student and the course staff, then the course staff are unable to do their jobs and help the student. Also, if the course staff are waiting for a web response to the workflows page, then they are unable to help multiple students in a timely fashion. This is important, especially as some workflows have a lot of responses between a student and the course staff if their issue is complex. Extensive tests were conducted to test course permissions and accurate functionality of the workflows.

Some important decisions that we made when designing the workflows are how to efficiently search for specific course staff to assign tickets to, how to conduct emailing students when a workflow is updated, how to submit automatic extensions to Gradescope, and how to create comments that are hidden from students. For finding course staff that tickets can be assignable to, we first thought that iterating through users with the permission to modify tickets was a good approach. We soon found out that approach took the workflow web page about five more seconds to load than usual, making the user of our system most likely wonder if the web page will be unable to respond or not. After some digging into the code, we found that the users with certain permissions are pre-saved in the permissions model and we decided to grab the course users with the ticket modify permission from there instead of iterating through all users which made the system much faster. For sending emails to students when a workflow is updated, we first though to use Django's emailing template which was not let in by the @virginia.edu's firewall. Instead, we decided to host the emails on our Virginia CS server which solved the issue. For submitting automatic extensions for students to Gradescope, we thought about hooking our system up directly to Gradescope which sounds great in theory, but Gradescope's lack of an API makes it challenging to integrate our project. We also thought about adding submodules to our codebase to integrate the GitHub repository our graduate teaching assistant created to extend due dates on Gradescope for students, but we soon realized this was going to be more of a headache on the development side. Instead, we are integrating the GitHub repository on our server, so we can just implement OS system commands to update the repository with student extensions. One thing to point out here is that the extensions only work for Gradescope and with our larger goal of making the project easily adoptable by other courses, this should be a suitable solution for University of Virginia CS courses as a lot of them have migrated to Gradescope but for the ones who have not then they could possibly plug-in a different GitHub repository instead or this will be a future problem for us to figure out as the project expands. For creating hidden comments, our first quick thought that we had was to use course user permissions to figure out if a user is a student who should not be able to see the hidden comments about their ticket. However, all the permissions that students have, the course staff, especially instructors, also have. We decided that adding a flag to our model to indicate if a user can see the hidden comment or not was the easiest and most efficient solution.

As part of our larger goal of making this system easily adopted by other courses, an intuitive user interface was a priority in the development process. Users should not need a technical background or do not really need to know how workflows work entirely to be able to operate this part of the system. We designed the buttons and tables to be clear and labeled well.

3.3 Statistics

One of our features that helps with the distribution of teaching assistants during office hours and busy lab weeks is statistics. It needs to provide fast and accurate queue aggregate data for a specific date and times throughout the day. This is the top priority for the statistics system and is especially important when there are a lot of students on the queue, ranging from thirty to fifty students, and the wait times are over an hour for students to receive help. If this system does not give reliable information, then the necessary amount of teaching assistant help needed for busy office hour times will not be correctly allocated and students will not be able to get the help they need to get their assignments done on time. Office hours are usually busier when the assignments are more difficult which means that most students are not getting help when they need it most if the statistics system is off. We have conducted testing on this system to see its response to large amounts of queue data.

Two important design decisions when developing statistics are how to aggregate the office hours queue data and how best to represent the data to users. All the information necessary to create average queue wait time statistics are contained within each "QueueEntry" object. Each of these objects contain the time the student entered the queue and the time the student was helped, or if they were removed, along with additional information. Thus, wait time for a given student can be calculated by simply taking the difference between these two times. Using this information, a running weighted average can also be calculated by some fraction of the newest student's wait time and the existing wait time. Every time a new running average is calculated, this average can be placed in a time category based on the nearest half hour this running average was calculated in. Once this process is complete for every QueueEntry on a given day, each half hour category's data points can be averaged to get a single average wait time for a half hour period of time. Then, a line graph can be created by plotting the time of day in half hour increments on the x-axis which map to average wait times in minutes.

A priority when designing statistics was that the user interface was intuitive and that data was shown in a comprehensible way. The statistics system should not require any technical background; the user just needs to know the date that they want to observe the queue data for. We designed

the layout to be clearly labeled so that it can quickly be seen where the most help is needed for office hours and to know where teaching assistants should be redistributed to optimize the amount of help and time given to students in the course.

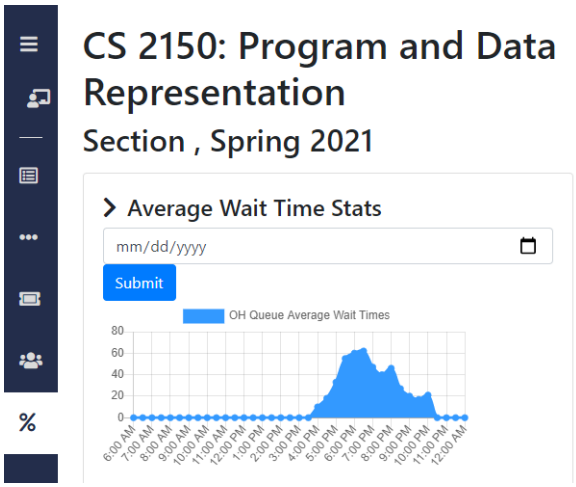

Figure 2. Instructor view of the Statistics page

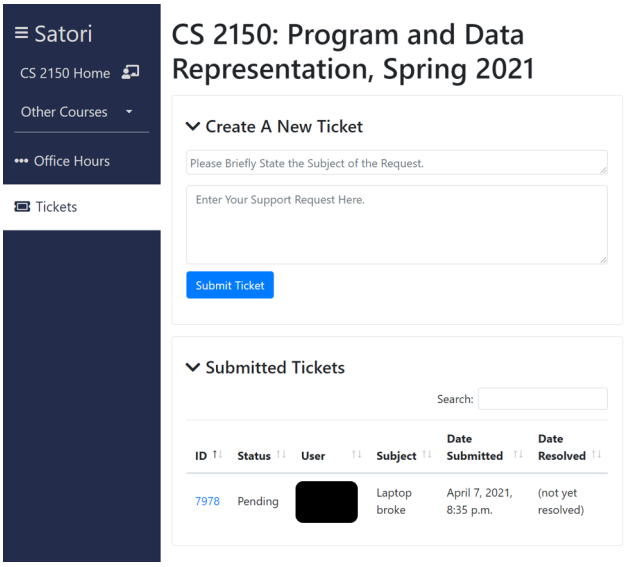

Figure 3. Student view of Tickets

4 Procedure

Satori is very user-friendly so most actions are intuitive. The support ticket system and queue statistics will be further explained as the functionality varies significantly depending on the user's permissions.

4.1 Student Ticket Creation and Updates

The ticket creation form can be accessed on the Tickets page. Students can enter a subject and body for their request before submitting. Their previously submitted tickets are also displayed on this page in the Submitted Tickets table. This section of the page features a short summary of the ticket, which quickly shows important information, including the subject, time submitted, status, and the time it was resolved if applicable. A student can provide an update message to their ticket by clicking on the corresponding ID to open the ticket's workflow page. This new page provides all of the original information about the ticket as well as a record of all of the responses and updates to it. The student can also fill out the form to submit their update or response for this ticket. By submitting a response, the student changes the status of the ticket to pending. The student can also upload files to in their response. This feature can be used if late work or documentation needs to be provided by the student.

4.2 Handling Tickets

Instructors can submit a ticket for a student on the Tickets page. Their ticket creation form looks similar to the student version except instructors also see a drop down containing the names of all of the students in the class. The instructor can select the appropriate student to submit the ticket for. The Tickets page also shows all of the submitted tickets to teaching assistants and instructors organized by ticket status. The pending and stalled tickets are showed in the Active Tickets table while the resolved tickets are shown in the Resolved Tickets table. The ticket tables show all of the information that the student can see as well as the ticket owner, but the course staff can see this information for all students. The Active Ticket table also allows teaching assistants and instructors to assign any ticket to themselves or another user with the correct permissions. They can also release tickets that they are currently assigned to.

Clicking the ticket ID in the table redirects to the ticket's workflow page. On this page, instructors and teaching assistants can see all of the original ticket information and all updates/responses. Instructors can respond to the student and change the status to pending, stalled, or resolved. If teaching assistants and instructors want to converse about the ticket without the student seeing the messages, there is an option to mark the response as hidden to the student. With this option, the responses do not appear when the student views the ticket's workflow page and they do not receive an email. However, if the option is not selected (the default), the student can view the response and will receive an email notifying them of the update.

On the ticket's workflow page, the instructor can also give the student an extension. Underneath the text response are drop downs to select the appropriate assignment. The extension can be given until either a new due date or entered as the number of days after the original deadline.

Figure 4. Teaching Assistant view of Ticket Workflow page

4.3 Statistics

The instructor can view statistics about the office hours queue in the Statistics tab. The instructor needs to select the date and then they can see the average student wait time for every half hour for the office hour queue. The wait time is calculated the same way as the one displayed on the queue page. This information is provided in a graph from 8:00 am to midnight Eastern Standard Time. The Statistics tab does not appear for other users.

5 Results

Throughout the Spring 2021 semester, the Satori web application was utilized at the University of Virginia by the CS 2150: Program and Data Representation course by over 500 faculty and students. The newest functionality of Satori is a vast improvement over the respective existing technological infrastructure. The ticketing system and workflows makes organizing tickets and coordinating support requests with student far easier than before. Likewise, the Statistics feature gives instructors valuable insight into how to best allocate office hours time and teaching assistant resources. Furthermore, with numerous fixes to existing bugs and problems, Satori now runs better than ever.

All of the aforementioned changes and additions to Satori are aimed at giving the students the most friction-less course experience with respect to getting help. Given the circumstances, instructors and teaching assistant time and resources are now more valuable to the students than ever before. A more robust and capable Satori will now be able to keep pace with the ever-increasing capacity of students enrolling in CS and demand for more software tools and features. Therefore, the new Satori has been and will continue to facilitate a more seamless experience of students interacting with instructors and teaching assistants.

6 Conclusions

The Satori web application was an existing course tool built to replace the CS 2150 course management system. Although it had many existing features and functionality, Satori was far from complete. Our 4th year technical capstone project picked up where Satori was left off and made several robust improvements to the existing system in the form of a new Ticketing System, Statistics Insight, as well as countless smaller fixes and improvements to existing code. Now, Satori continues to play a crucial role in the students course experience and serves as an elegant solution to many difficult problems.

7 Future Work

All of the major features of Satori have been included. However, there are a few smaller features that could help instructors customize the system to better fit their classes or make some of these features easier to use. All of the functionality for different users depends on their permissions. These permissions are initially assigned according to their course role on the roster. While the permissions can be changed for each user individually, a feature to update a role's permissions would make it much easier to change the default behavior. It can also be tedious to make another role, such as Secondary Instructor or Super Teaching Assistant, and assign permissions and users to the new role. Allowing the course roles to be better tailored could make it easier for instructors to set up their classes with slightly different functionality.

The course creation process keeps track of the sections of the course each student is enrolled in. However, very few of the features on Satori use the course sections. This could be better integrated to help divide lecture or lab sections and would be especially helpful when multiple professors are teaching a course. That would allow the course to be managed as a whole as well as section by section.

More office hour queue statistics could be included. Right now, it only displays the average waiting time but it could be useful to see how many students were helped in each time interval, the average amount of time it takes for a student to be helped once they are taken off the queue, or the percentage of the class that attends office hours. These statistics could further help the instructor manage staffing during office hours based on student demand. A class's office hours are typically scheduled in advance. The office hour queue could be scheduled to automatically open and close at certain times as opposed to having to be manually opened and closed by a teaching assistant or instructor each time.

References

[1] 2021. *Gradescope*. Retrieved April 7, 2021 from https://www.gradescope.com/

[2] 2021. *TIOBE Index for April 2021*. Retrieved April 7, 2021 from https://www.tiobe.com/tiobe-index/