Satori: A Course Management System

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Ramya Bhaskara

Spring, 2022

Technical Project Team Members

Joshua Mehr

Cristian Scruggs

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

Aaron Bloomfield, Department of Computer Science

Satori: A Course Management System

Ramya Bhaskara
University of Virginia
rsb4zm@virginia.edu

Joshua Mehr
University of Virginia
jmm3vn@virginia.edu

Cristian Scruggs
University of Virginia
cms3zd@virginia.edu

Abstract

CS 2150: Program and Data Representation, currently hosts the third largest number of computer science students at the University of Virginia (393 students in the Spring 2022 semester) but lacks a modern course tool to enhance student's learning experiences. Specifically for office hours where students must enter an outdated online queue in order to be helped by a teaching assistant, this results in long queue times for students waiting for help. This technical report outlines the work done to create a modern web application to better assist with students seeking help with the course work. The web application was written using the Django framework and hosts a multitude of features that allows for less waiting for the students and less stress for the Teaching Assistants who lead office hours. The current web application has proven useful for handling the workload of CS 2150.

I. Introduction

This paper details the continued development of Satori, an application that will be used as the primary course management system for CS 2150 (Program and Data Representation) at the University of Virginia. The current system in use (referred to as 'Course Tools'), includes two primary functionalities: a support request management system and an office hours queue. While Course Tools was a functional solution in the initial years the course was offered, as the course has grown in the last decade or two, these course tools no longer serve their purpose.

CS 2150 is a required course for all computer science majors and minors, making its enrollment approximately 400 students per semester. These large enrollment sizes have overwhelmed the old system, causing long response times, data consistency glitches, and frequent hangs that render it unusable for long periods of time. The site is also restricted by a very primitive user interface. These problems make it difficult for students to get help on assignments and for teaching assistants to manage office hours. In order to address these issues, we present the development of an improved web-based application that facilitates office hours and eases the handling of support requests.

II. Background

The front and back end of Satori is built using Django 4.0.4, which is a Python based web development framework, and the back end consists of a MySQL database. This version of Django requires that we run it using a python version ^3.8. To help with organizing and gathering our code, we used GitHub, a version control tool for software development. This is also suitable considering the nature of this project. Ultimately, Satori will be deployed in a Docker container to facilitate the linking of the login system with NetBadge, which is UVAs centralized login tool. It will ultimately by deployed on UVA Computer Science servers. More specifically, the production version will be deployed on the Andromeda server, and the development version will be deployed on the Pegasus server (though these are subject to change).

Another consideration is changing course requirements in the Department of Computer Science. It was recently announced that the department is introducing a new curriculum for all new students, which means that CS 2150 will no longer be offered after the Fall 2022 semester. As such, though the development will be done with the particular use case of CS 2150 in mind, the ultimate goal is for this tool to easily adapt to handle additional courses in the future.

III. Related Work

A. Course Tools

As previously mentioned, the Course Tools system was suitable to CS 2150 in the initial years of the course being offered, but, as the class size increases, Course Tools becomes increasingly insufficient and dated. Before the widespread adoption of Gradescope, the Course Tools had assignment submission and grades. Now that submission and grading are dealt with elsewhere, the only use of Course Tools is for Support Request Tickets and the Office Hours Queue. The previous tools were made in PHP, which although works for the intended use, is very much falling out of the mainstream of coding languages. By switching to Python, the underlying code should be much more accessible to those who plan to work on it in the future. Using the Django framework, we should be able to replicate and potentially improve upon the functionality of Course Tools while also making it more visually appealing and professional.

B. Previous Work on Satori

The development of Satori began in 2019 with the goal of replacing Course Tools and handling all aspects of the course, including support requests, office hours, assignments, exams, and grades. However, shortly after development began, the course adopted Gradescope for assignment submission, and the new system no longer needed to implement a separate system for assignments, exams, and grading. This made it

so that the previous system had a lot of unnecessary built in complexity, making the codebase very difficult to work with. This caused a multitude of issues, including the inability to add courses and create queues. With this new set of issues, the team decided to restart development from scratch to remove some of this complexity from the system and make the system more scalable. The goal of our work was to redesign the old system to remove some of that built in complexity that made it unusable.

IV. System Design

Using the Django "Groups" feature, each user of the application is assigned into the student, teaching assistant (TA), or faculty group depending on their enrollment for each course (discussed below). This allows for each user to have set permissions on which actions they can or cannot perform given their enrollment; for example, only faculty members are allowed to create/delete courses. Permissions are checked redundantly before and after any action is performed by a user to ensure the security of the applications underlying data.

Satori itself is broken up into 3 primary apps: Core, Queue, and Tickets. The Core app mostly addresses the high level management of courses such as creating courses, enrolling students and teaching assistants to said courses, and dealing out permissions. The Queue app allows professors to create any number of queues for courses that they teach. Students should be able to add and remove themselves from queues for courses they are enrolled in, and teaching assistants should be able to remove students from the queue so they can help them. Finally, the Tickets app functions as a central location where students can contact the professors for assistance, and view all relevant communication surrounding their support tickets.

A. Core

The Core app serves as the backbone of the application and handles most of the interactions with the back-end MYSQL database. Primarily used by course instructors, the application defines database tables for Courses, Student Enrollments, and Netbadge Cookies, the latter of which is used to enable students to login with their standard UVA accounts. The features of the Core application primarily involve users interacting with this backend database through a series of GET and POST requests. For example, faculty can create, edit, and populate the class rosters of courses through simple web pages that will only allow the users in the faculty group to perform these actions. The landing page of the webapp is also hosted in the Core app. This is where students and faculty alike can authenticate themselves using their UVA accounts, where a new user entry is made in the database if it is their first time authenticating to the site. Once logged in, users can easily navigate to the other applications of the web app using the links on the landing page which include details about the courses they're enrolled in, the support tickets app, and the queues for each course.

B. Queue

The Queue app is the main course tool of the web application and is where students will enter themselves to wait for help from a TA and where TA's will pull students one by one to assist them. Each course can have any number of queues associated with them but each queue is only associated with a single course. A queue has an associated name, entries, and an "is open" field that can be toggled by course staff when office hours are being hosted. At a high level, when the student enters a queue, a corresponding queue entry is created in the database and the student sees their information appear at the bottom of the queue. As entries are taken by TAs, students move up in the queue based on when they entered until they are taken off the queue and assisted by a TA. For TAs, the queue app allows them to take students off the queue or requeue them to the top of the list if they were not able to provide adequate assistance to the student.

As the developers of the application are TAs for the course themselves, we were able to add additional features to address some frustrating oversights in the previous application. Firstly, a queue now auto-updates its entries without needing to be refreshed by the user. This allows for TAs to not have to waste time constantly refreshing the page to see new entries in the queue. Additionally, the queue app also displays the average wait time for students on the queue based on how quickly entries are being resolved. This allows for students to have a better understanding of how long they will need to wait to be assisted by a TA and TAs to gauge how long they should spend helping a student to most efficiently move through the queue.

C. Tickets

The tickets app allows students to submit support requests to be fulfilled by course staff and is the primary way students will contact course staff for help. A ticket is associated with a single course and is submitted by students or faculty members through POST requests on the tickets homepage. Any number of support requests can be submitted for a single student and each ticket consists of a written request for assistance from course staff. Additionally, each ticket has an associated status; either pending, stalled, deleted, or resolved based on where the course staff is with helping the student with the request. If more information is needed, students and course staff alike can leave comments on each ticket.

Given the large number of students enrolled in the course, it was important that the tickets app function properly in order to keep instructor's email inboxes from being flooded with student requests. A challenge of creating the application was how to best design the user interface for the tickets themselves. Based on anecdotal feedback from other TAs we decided to display the comments for a ticket underneath the original ticket details to have the tickets appear functionally the same as a traditional email thread, making it much easier for users to understand in which order responses were made.

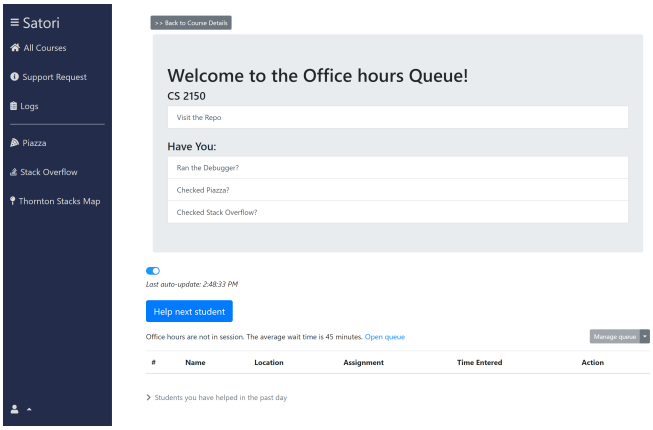

Fig. 1. An image of the Office Hour Queue from the TA perspective

V. Results

The major goals of the web application were twofold. The first was to provide a modern, web application to better handle the demanding requests made for CS 2150 at UVA. The success of this was collected anecdotally by gauging reactions to the new queue from fellow TAs. The reactions have been mostly positive with a few constructive ones that we used to make the application better (like the UI changes to support request comments). The new quality of life features such as displaying an average wait time and the queue auto-updating have proven helpful. Additionally, the new queue is far less prone to crashing than the old one. A visual of the TA view of the queue is seen in figure 1.

The second goal was to leave the project in a place where it can be iterated on and refined by future development teams. This has been done by making sure the code in the repository is simple yet intuitive so that it may be easily understood in the future. Additionally, work has been done to comment the code so that each function's usage is clearly explained. Finally, the last few weeks of development will be devoted to writing clear instructions for maintaining the web app in the future.

VI. Conclusion

Satori is a significant first step in reducing long wait times for students waiting for help in CS 2150 and courses beyond. The features of the web application have proven to be successful in improving office hours for both TAs and students while leaving room for improvements in the future. While there are similar systems to Satori, we are confident that no other queuing system adequately meets the needs of classes in the Computer Science department at UVA, especially considering how large and understaffed some of the classes are. Finally, although the main requirements of Satori have been completed and deployed, there are still some additional features that we would have liked to implement if we had more time with the project.

VII. Future Work

We have already implemented all of the core functionalities intended for Satori. There are a few additional features that we think could be helpful additions whether it be for customizability, a more elegant design, or simple quality-of-life improvements.

One of which would be a page for instructors to view the efficiency of their TAs during office hours in order to ensure that the queue is moving at a reasonable enough speed to reach every student that comes for help. This page could also display other statistics about average wait time, number of students entering the queue, amount of time spent with individual students and much more. Perhaps it could even be separated by office hour blocks since TAs are assigned 2 hour blocks of time where they hold office hours.

Another would be the implementation of an announcement system in which the TAs could broadcast messages to the queue for students to see. This could be helpful in times where there are technical issues or perhaps a simple problem which many students are encountering that could be summed up and solved in a couple sentences. Currently with situations such as this, TAs will either post about it on piazza or hold in person group sessions in which they explain a concept to several students at once. With the addition of an announcements system the large load of students could be decreased for TAs in many of the labs in CS 2150.

We would also like to include either a way for students to input their preferred pronouns into Satori or for the website to take them from the students' listed pronouns in collab. This is a requested feature that we feel would be helpful to ensure that everyone getting help feels more comfortable with how they're addressed without forcing them to explicitly tell the TA helping them every time.

The ability to add additional roles for users would also be helpful to improve Satori's overall adaptability since we plan for it to be used for classes other than just CS 2150 eventually. Some classes with different structures may want special roles for TAs that are doing different jobs or to include roles for Head TAs.

Creating a separate option for students to indicate that they're using virtual office hours and an accompanying link to a Zoom meeting would be a nice quality of life improvement over copying and pasting Zoom links given in the location field of a queue entry.

Acknowledgments

We would like to acknowledge the kind support we've received from all of the TAs of CS 2150 who helped us to stress test our system and gave us constructive feedback on each of the core functionalities that we implemented and how they imagined we could improve them. We would also like to give thanks to all of the students in CS 2150 that made use of Satori for their help in introducing us to some of the bugs in the system.