Technical Report

Event Management System

Brandon Cossin
bcossin@kent.edu

Tyler Gargasz
tgargas2@kent.edu

Joel Lee
jlee131@kent.edu

Chris Seitz
cseitz5@kent.edu

Nolan Spencer
nspence9@kent.edu

Stakeholder
Alfred Shaker
ashaker@kent.edu

April 20th, 2022

Table of Contents

Figures

1.0 Introduction

1.1 Motivation

Kent Hack Enough, a student-run Hackathon at Kent State University, has an aging event management system that needs a fresh coat of paint and inner workings. As such, our group is interested in creating a replacement event management system for the event. Although this project was initially rooted in an interest in being applicable to Kent Hack Enough, it is intended to be a generic event management system that could be utilized in a variety of other events or organizations that all have similar requirements.

1.2 Problem

An event management system is needed that has a frontend that displays information about the event and allows people to fill out a form and apply for the event. This information needs to be stored in a database and viewable on a staff management portal that allows registrants to be approved, receive emails, and have their information edited and/or moderated by staff members. With all of this factored in, the system will have all the necessary components to create an event and manage it at the host's discretion.

There are many event management systems available to use, but they involve such a magnitude of configuration and scope that they can be overwhelming to hosts of small events. Our system is intended to be a simple all-in-one solution that accomplishes the minimum necessary to run an event such as Kent Hack Enough, with such capabilities being universal among many student-run or non-commercial events. Our system must be easy to use, easy to configure, self-contained, easy to make minor source code adjustments, and completely themeable.

1.3 Customer

The customer's objectives involve those needed to run an event of any scale.

1. Provide Events
2. Facilitate User Registration
3. Management of Data through an Interface
4. Exporting data

Potential customers may be student organizations, companies, and other entities that run events and would be interested in such an event management system with an accompanying front end.

In accordance with these objectives, Alfred Shaker, Ph.D. Student & Part-Time Instructor, has a common interest in our project as President of Kent State's Fighting Game Community and has agreed to be a stakeholder in our project and provide guidance and feedback on the construction of this event management system.

1.4 Users

There are two types of users for the system, Users and Staff.

The term *User* describes someone who is utilizing the website and is not part of the group of event organizers. Users may want to learn more about the organization, see what events are being hosted, and sign up for those events.

The term *Staff* describes someone who is part of the organization running the website and has access to the Staff Portal. Staff will facilitate all behind-the-scenes operations of the system through the portal.

There should be no more than a thousand expected users for any event and no more than a dozen or so staff members for the event. The users should be assumed to have very few computing skills, yet enough to be able to fill out a google form and traverse a basic website. The staff members should also be assumed to have no knowledge of development or anything beyond utilizing dropdowns and search bars to traverse the staff management portal and make adjustments to data within it. Registrants must be able to easily find details about the event and fill out form entries to apply for participation. Staff members must be able to clearly see how many people signed up for the event, approve registrants, assign roles, send emails, and overall control all aspects of the event and the information visible on the main website through the staff portal.

1.5 Constraints

The event management system must be able to be easily edible with someone with basic programming skills, with anything beyond the functionality or styling of the website being able to be completed within the staff portal or by editing documents within the source code of the system.

The event system must be able to run on any server and be containerized such that the entire system of frontend, management portal, database, and load balancer be self-contained. Docker will be used to do this, thus the system must be able to run on Linux systems.

The website and staff portal must be designed such that it is very user-friendly and has at least a bare minimum of accessibility. The source code for the project must be clearly documented and designed such that it is easy for novice programmers to make adjustments to the project.

2.0 Planning

2.1 Elicitation Plan

To begin eliciting requirements for our system, our group looked at the current system for Kent Hack Enough and analyzed the systems already in place while also gathering feedback from Kent Hack Enough's organizers on what could be improved. In addition to this, Alfred Shaker provided exceptional insight into how such a system could be molded to be usable by a variety of other organizations with similar interests to Kent Hack Enough; ultimately leading to the idea that we could derive all of our plans from.

The plan is, for the most part, to reimplement core features of Kent Hack Enough's hackathon system except in a new and improved way, utilizing a new technology stack that was more documented, and to streamline the entire process of adjusting the features implemented in the system. All technologies used were also chosen with the previous iteration in mind; with most being the same or upgrades from what worked well with Kent Hack Enough; such as the migration from JavaScript to TypeScript. With this, our group was able to cultivate the requirements for this project and produce the system that is detailed in this report.

2.2 Assumptions

This project assumes that hosts and customers share the same expectations and requirements as the organizers of Kent Hack Enough. Research has not been conducted to see if this is truly the case, and as such Kent Hack Enough's utilization of such a project may be an abnormality rather than the standard. Nevertheless, there remains potential for other student organizations to be searching for the same software as this project and do not know where to start or are too intimidated by the already existing options.

Additionally, it is assumed that one implementing such a system has the basic technical knowledge to install and configure the system to be tailored to an organization. Although this project is intended to be easy to use and easy to configure, it is also intended as a baseline template for an event webpage that could easily be developed into a fully-featured organization event page.

These assumptions can be verified through extensive testing and outreach to see how effective this system is at achieving its goals in the hands of those not familiar with the development of the project. In the event that these assumptions are invalid, the system will require additional feature additions, guides, and updates in order to redirect its course to that of being effective.

2.3 Risks

There remains a risk that this project is overkill or unnecessary for clients. There is a reason why Google Forms or other forms of data collection are surprisingly effective and popular with student-run organizations or small-scale events. Yet given this risk, there remains potential for such a system to be as fruitful for event organization and hosts as it is for Kent Hack Enough, an event that this team has experience hosting.

In addition to this, the project must be developed on the timeline of the Capstone Project course schedule. This project must be developed within the timeframe of Jan 30, 2022 to Apr 19, 2022, having all features planned and implemented before Apr 14, 2022 in accordance with Capstone's draft submissions on said date. This is a difficult timeframe for a project of this scale and magnitude, and there remains the risk that some features will not have enough time to be implemented or enough time to be implemented at their full potential.

The project must also utilize resources and technology that are readily available and easy to understand and modify, ensuring our project is capable of achieving its goals. To achieve this, various interface libraries and other external dependencies are utilized to speed up development and offload application complexity onto other open-source software. There remains the fact that this process of offloading complexity requires us to use the APIs of said external software, and may negatively impact the performance and usability of our application, especially if refactoring is needed in the future. This project carries a risk of having an overcomplicated technology stack in exchange for speedier development and feature capabilities, which increases the level of knowledge and proficiency required to operate this system.

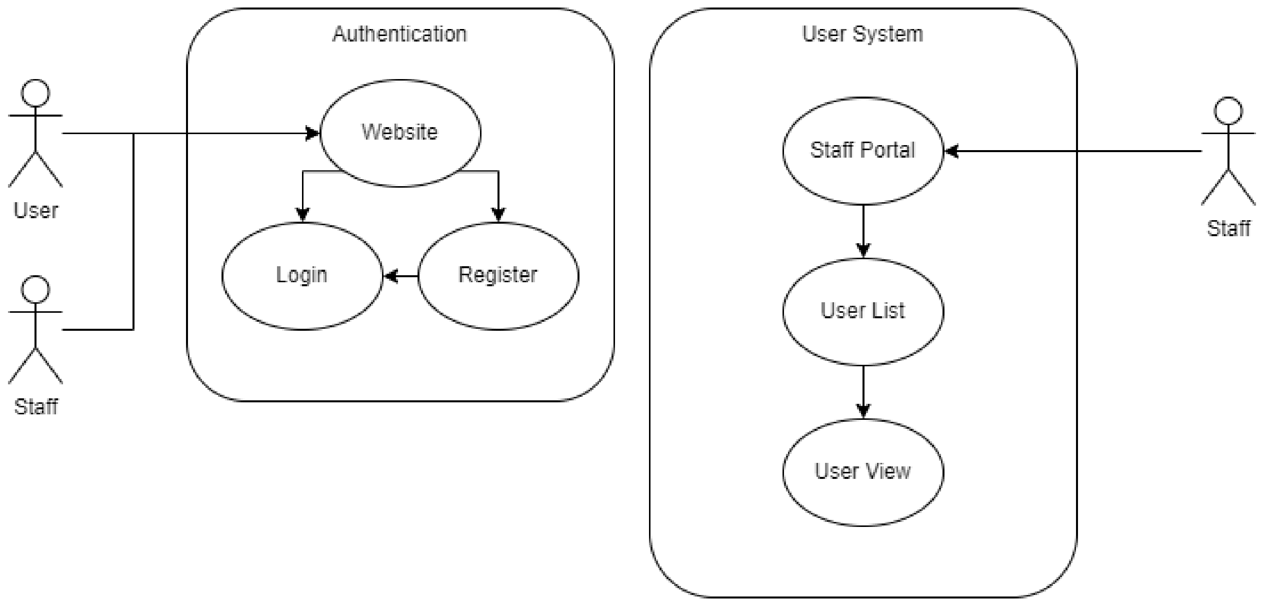
3.0 Use Cases

The application supports a variety of core functions that contribute to solving the overall problem that this project seeks to address. The use cases for these systems are illustrated below.

3.1 Users

In order for our application to be usable, an account system is required. People must be able to register accounts and login or log out of their accounts. Additionally, as this is an event system, all users must show up in the staff portal roster and allow various operations to be performed on them. This allows staff members to have full control over who can sign up for what events hosted by their organization, in addition to viewing who is registered for what events and various information about the people attending their events.

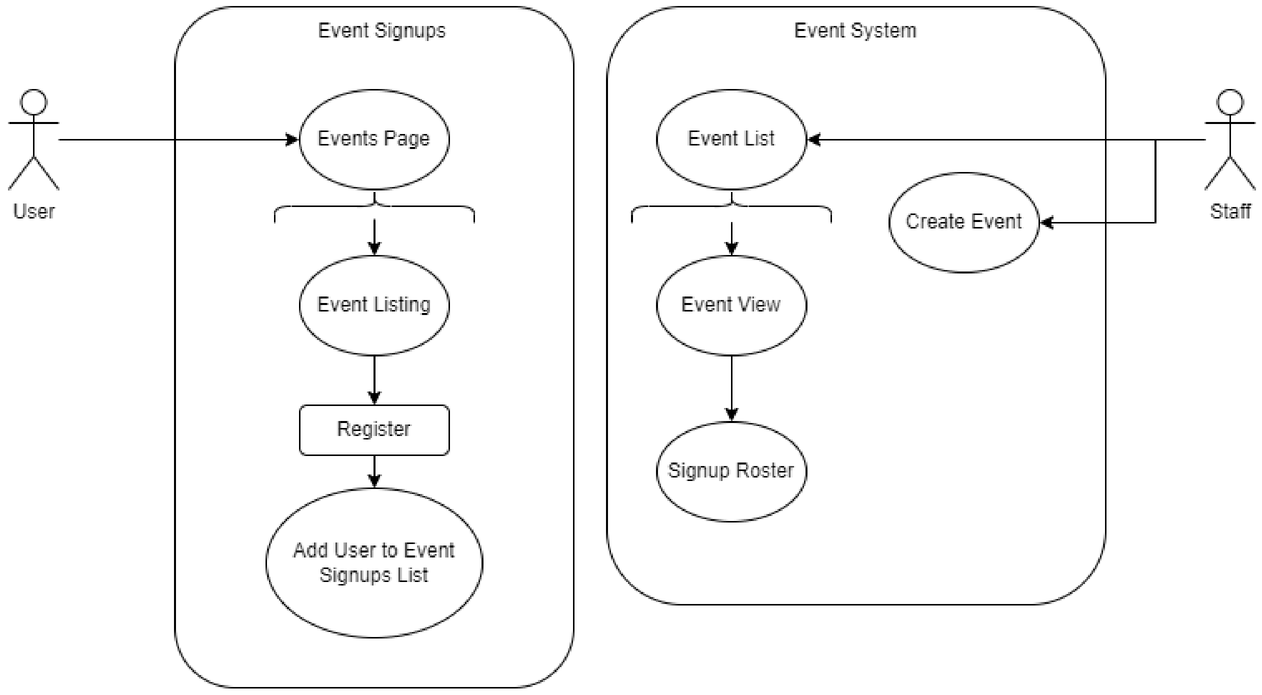
Figure 3.1.1 - Authentication & User System



3.2 Events

Events are created by the staff for the users to view and interact with. The staff can create an event on the events page on the staff portal. Once they create the event, the staff are able to edit, and/or delete the event. When there are events created into the database, the front end will fetch from the API to pull the Events in the database. After it pulls them onto the front end. The user can then view the events, as well as register, or unregister from events.

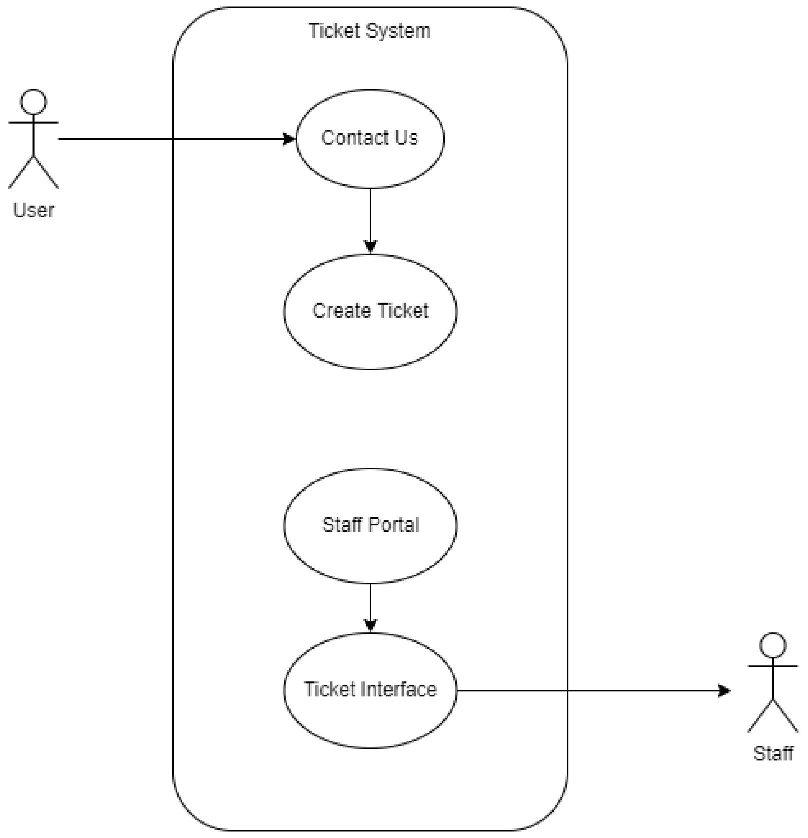
Figure 3.2.1 - Event System



3.3 Tickets

The Ticket System is essential to supporting and assisting users who encounter problems, issues, or simply have questions they would like answers to. Through the use of a Contact Form on the website, users may post questions to event managers and have their questions appear on the staff portal. When submitting data via the Contact Form, a new ticket is created. This ticket, performing in a similar manner to how tickets work in other enterprise applications, will allow Staff to review and respond to said tickets and provide assistance to the requesting user.

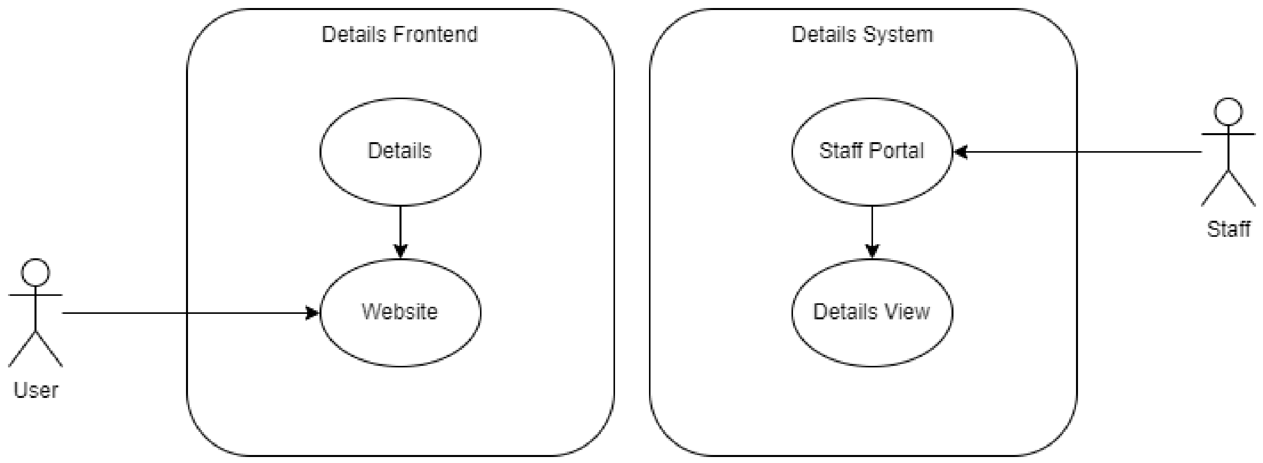
Figure 3.3.1 - Ticket System



3.4 Details

The Details system is intended to make editing data on the website easier for Staff who likely do not have a technological background or experience in website development. Nearly all textual content on the main pages of the website is dynamic and pulls their content from the Details API. This allows Staff to edit what appears on the website right from the staff portal.

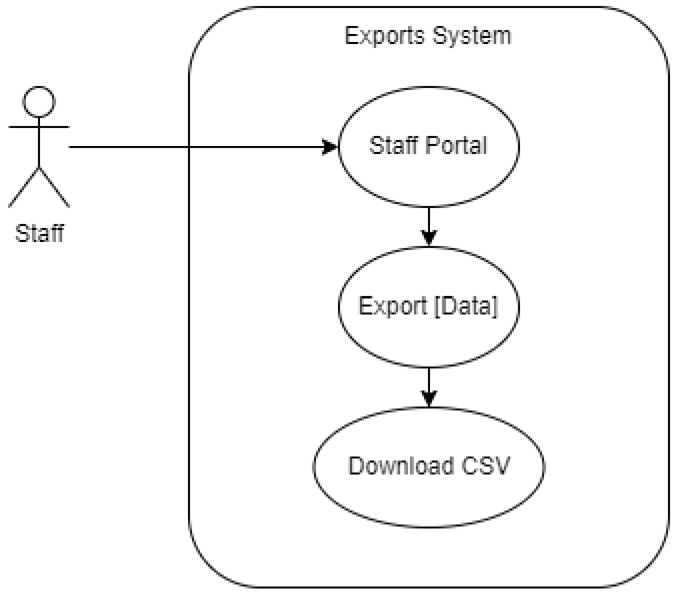
Figure 3.4.1 - Details System



3.5 Exports

Data exports are an essential part of nearly any system, especially this one. Many features are implemented directly in the interface, but not everything can be implemented the right way or include the necessary features for ever-changing use cases. To resolve these issues, data exports provide an easy way to download large portions of data in the database as Comma Separated Value files, also known as CSVs. This allows Staff to take nearly any data in the database and export it to a spreadsheet, which in turn can be used for a variety of unplanned use cases. Being able to pull bulk data from the database is essential to any system.

Figure 3.5.1 - Exports System

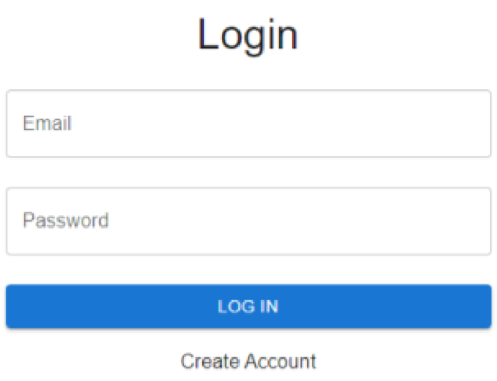


4.0 Design

4.1 Users

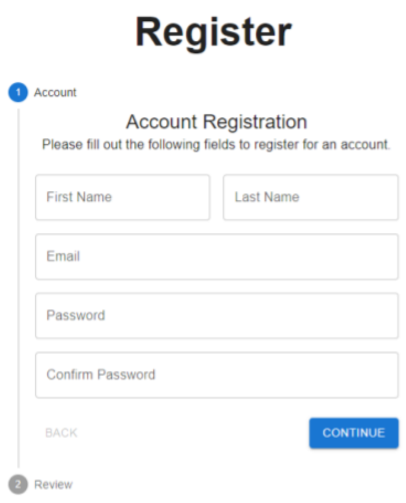
Logging into accounts requires a seamless interface. As with most login pages, the system's login page is quite simple with the username and password fields, along with a submit button.

Figure 4.1.1 - Login Form



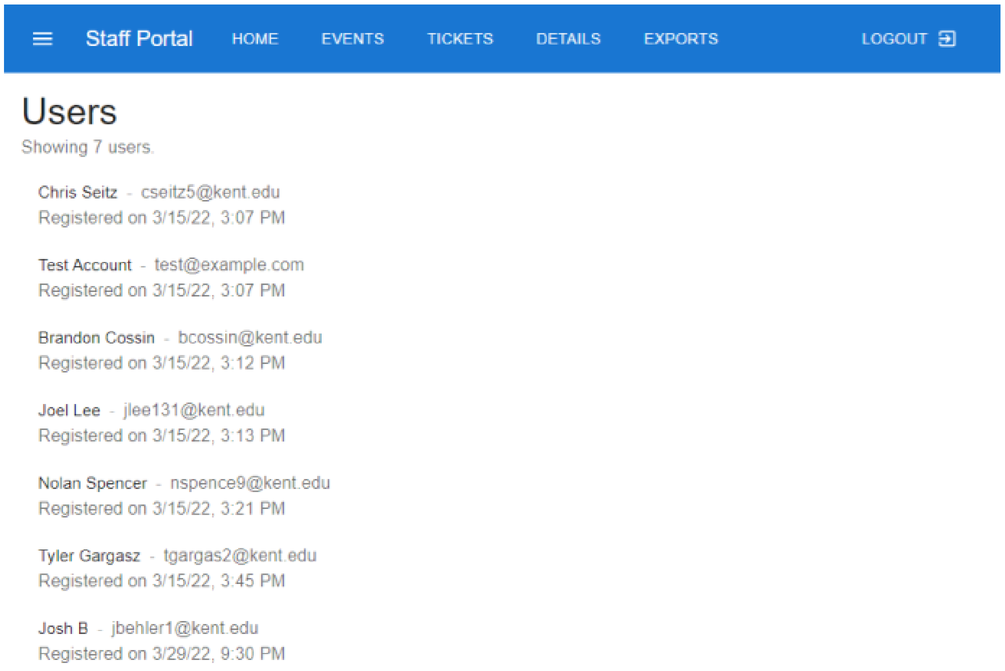
Meanwhile, for account registration, a nicely designed interface utilizing a stepper is critical due to the nature of the system. It is important for this system to be capable of having additional registration information added to the signup process, such as asking users their food preferences or allergies. Due to this, the use of a stepper ensures that there isn't too much information on the screen at once and that the registration process is nicely sliced into various portions that make it especially user-friendly on mobile devices.

Figure 4.1.2 - Register Form



The User Roster shows a basic list of users who are located in the database. Each entry shows essential information such as the name, email, and registration time of the user. Clicking any row in this list brings up the User View, which is detailed next.

Figure 4.1.3 - User Roster



Users

Showing 7 users.

Chris Seitz - cseitz5@kent.edu
Registered on 3/15/22, 3:07 PM

Test Account - test@example.com
Registered on 3/15/22, 3:07 PM

Brandon Cossin - bcossin@kent.edu
Registered on 3/15/22, 3:12 PM

Joel Lee - jlee131@kent.edu
Registered on 3/15/22, 3:13 PM

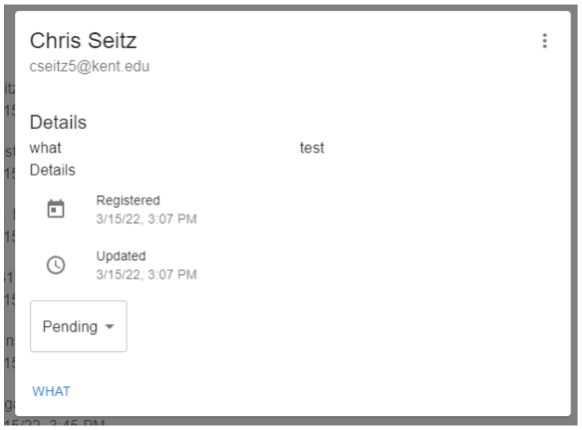
Nolan Spencer - nspence9@kent.edu
Registered on 3/15/22, 3:21 PM

Tyler Gargasz - tgargas2@kent.edu
Registered on 3/15/22, 3:45 PM

Josh B - jbehler1@kent.edu
Registered on 3/29/22, 9:30 PM

The User View is where all fields for the user are viewable and editable. Important features such as changing a user's role or approving them are located here.

Figure 4.1.4 - User View



Chris Seitz
cseitz5@kent.edu

Details

what	test

Details

Registered
3/15/22, 3:07 PM

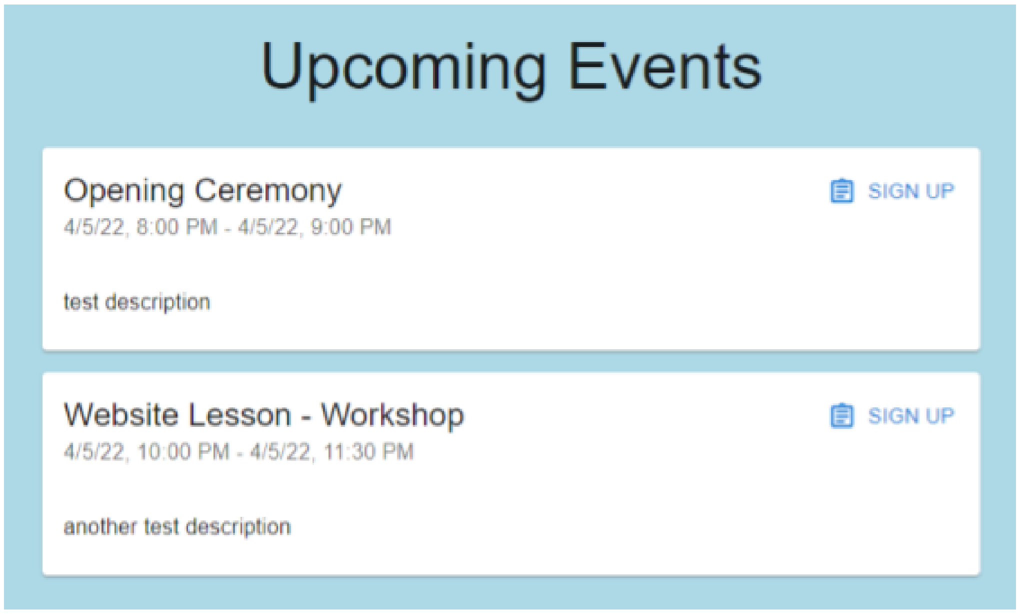
Updated
3/15/22, 3:07 PM

4.2 Events

Events must have a simple and intuitive interface given their critical role in the entire system.

The event page is a list of event cards. On these cards, it displays the name of the event, the date and times of the event, a description of the event, as well as a multi-function button that handles event signups and deregistration. If you are logged into the system you can sign up for events, otherwise, the signup button will redirect you to the registration page.

Figure 4.2.1 - Events Page



Upcoming Events

Opening Ceremony
4/5/22, 8:00 PM - 4/5/22, 9:00 PM

test description

Website Lesson - Workshop
4/5/22, 10:00 PM - 4/5/22, 11:30 PM

another test description

On the staff end, the list of events draws many parallels from the user list detailed in the previous section. Clicking on any event allows you to open the Event View for that event. Additionally, one can click the Create Event button to easily bring up the same View in creation mode, allowing you to create events.

Figure 4.2.2 - Events List

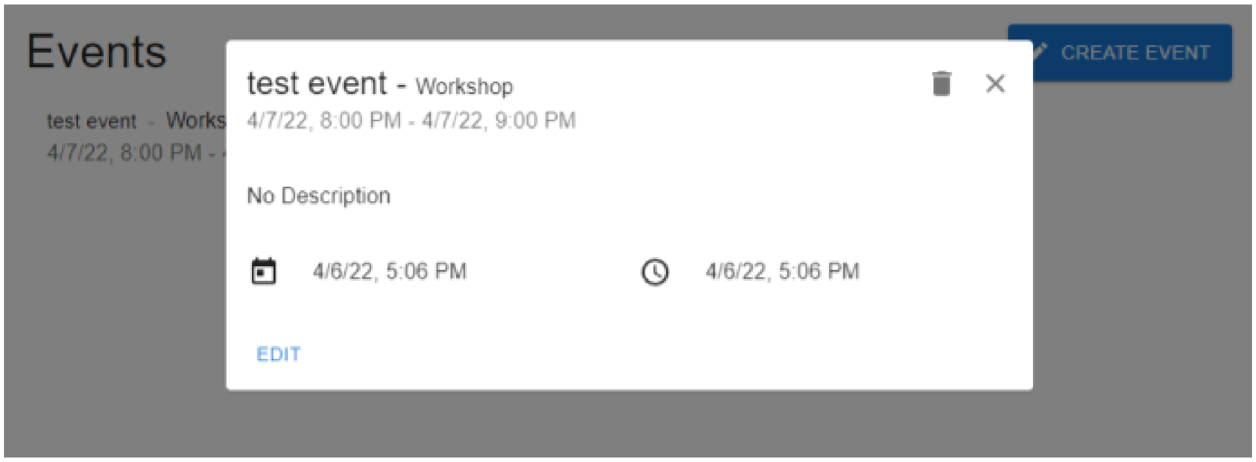
Events

CREATE EVENT

test event - Workshop
4/7/22, 8:00 PM - 4/7/22, 9:00 PM

The Event View is an essential part of the event management system, as it allows you to view events and create/edit their contents. The Event View is initially just in read-only mode, but pressing an Edit button allows you to swap into edit mode. When making changes, one can then click either the Save or Discard button to respectively save and discard their changes.

Figure 4.2.3 - Viewing an event on the Staff Portal

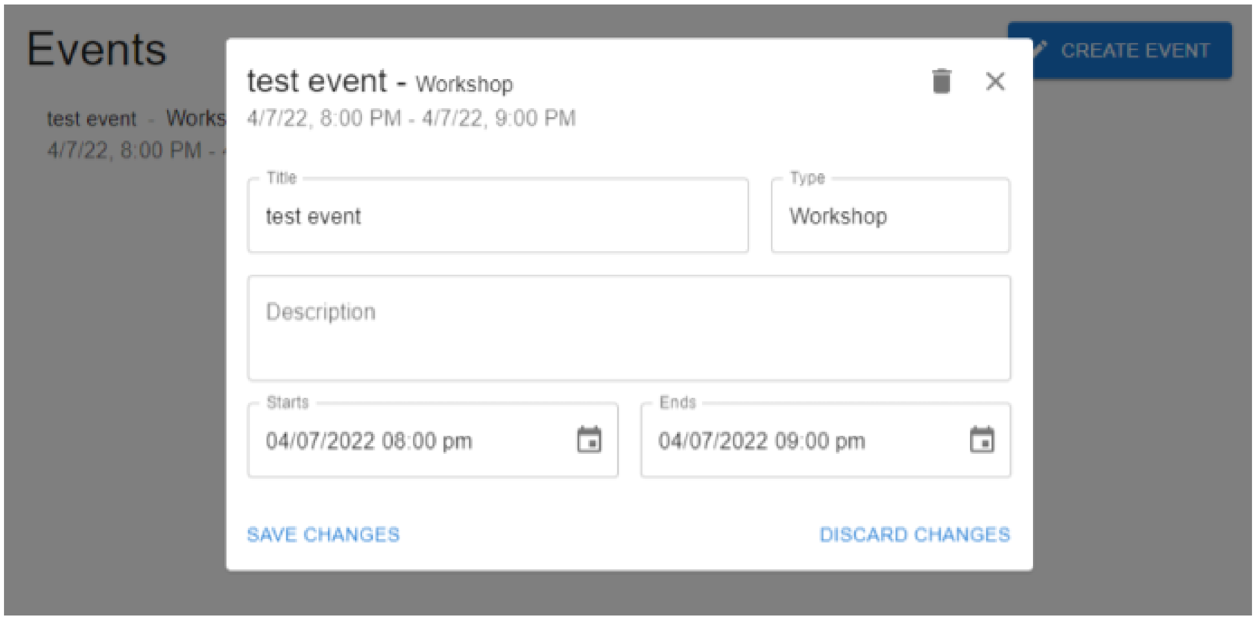


test event - Workshop
4/7/22, 8:00 PM - 4/7/22, 9:00 PM

No Description

4/6/22, 5:06 PM   4/6/22, 5:06 PM

Figure 4.2.4 - Editing an event on the Staff Portal



test event - Workshop
4/7/22, 8:00 PM - 4/7/22, 9:00 PM

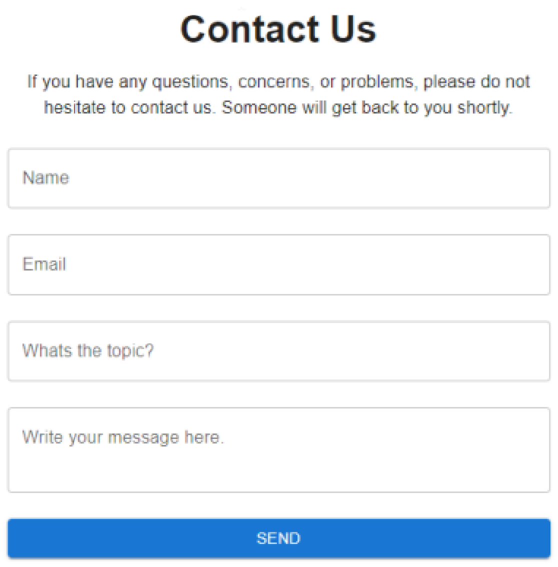
Title: test event
Type: Workshop
Description
Starts: 04/07/2022 08:00 pm
Ends: 04/07/2022 09:00 pm

4.3 Tickets

The Ticket system is an important part of nearly any user-friendly application. As the main facilitator of communication between users and administrators, the ticket system must be easy to use and very clear on what questions are being asked and how to respond to them.

The contact page shows a nice interface that makes it incredibly easy to ask any question and have it be sent to the staff portal as a ticket.

Figure 4.3.1 - Contact Form

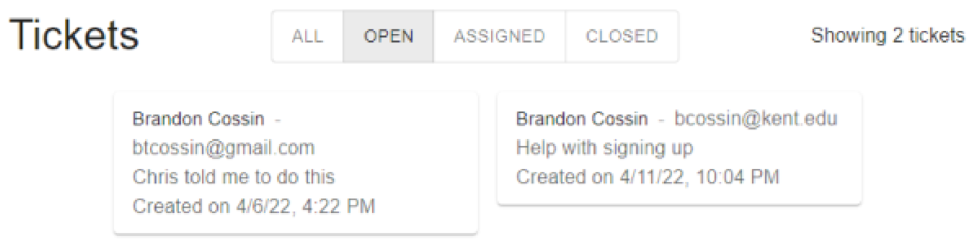


Contact Us

If you have any questions, concerns, or problems, please do not hesitate to contact us. Someone will get back to you shortly.

Once a ticket is created, it appears via the Ticket List on the staff portal. Each entry in the list shows basic information about the ticket, such as the name and email of the submitter and the subject of the ticket. Clicking on a ticket opens the Ticket View for it, where details and the status of the ticket can be changed. Additionally, the list of tickets can be filtered by ticket status with the selector listed at the top of the interface.

Figure 4.3.2 - Ticket List



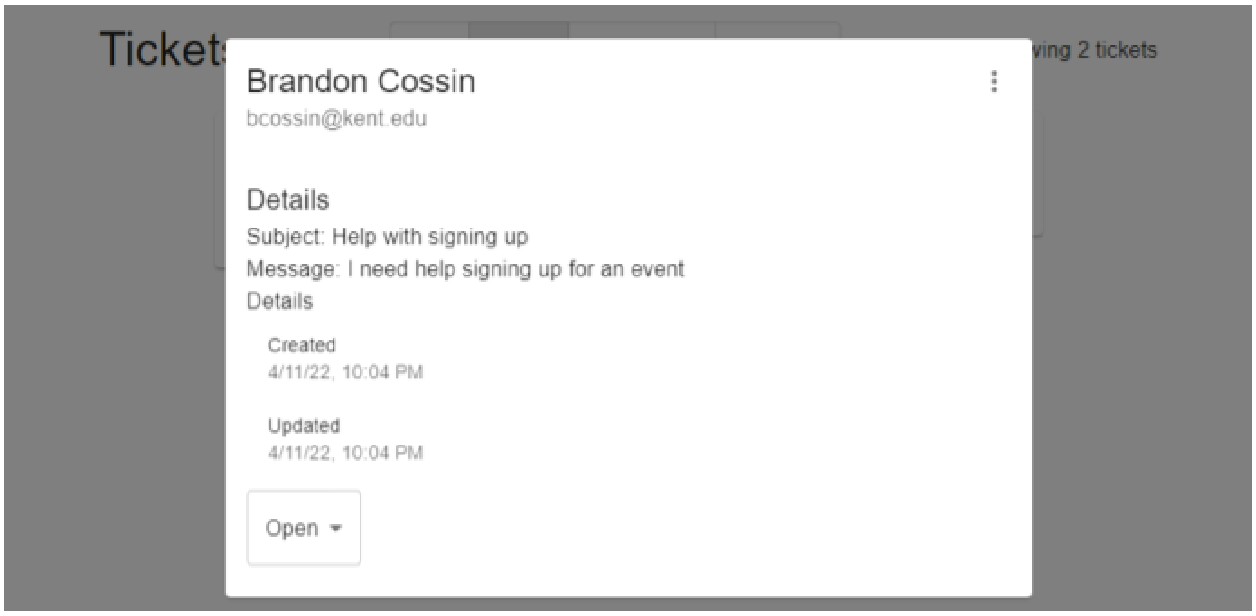
Tickets

Showing 2 tickets

Brandon Cossin - btcossin@gmail.com
Chris told me to do this
Created on 4/6/22, 4:22 PM

Brandon Cossin - bcossin@kent.edu
Help with signing up
Created on 4/11/22, 10:04 PM

Figure 4.3.3 - Ticket View



Brandon Cossin
bcossin@kent.edu

Details
Subject: Help with signing up
Message: I need help signing up for an event
Details

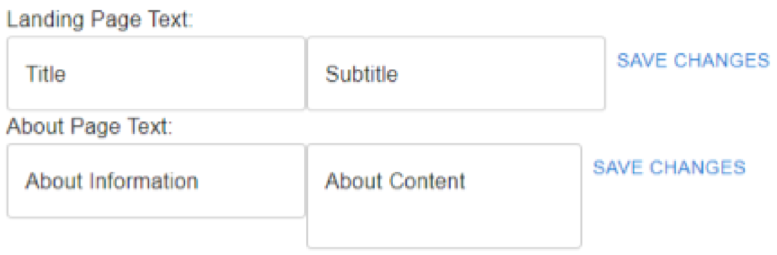
Created
4/11/22, 10:04 PM

Updated
4/11/22, 10:04 PM

4.4 Details

The Details interface is quite simple, as it is just various inputs for basic text fields. This interface only needs to provide a way to change basic information inside the database such that it can appear on the website. Each section has a corresponding save button which uploads the changes to the database via the details API.

Figure 4.4.1 - Details View



Landing Page Text:
Title | Subtitle | SAVE CHANGES
About Page Text:
About Information | About Content | SAVE CHANGES

Figure 4.4.2 - Homepage with content from the Details API

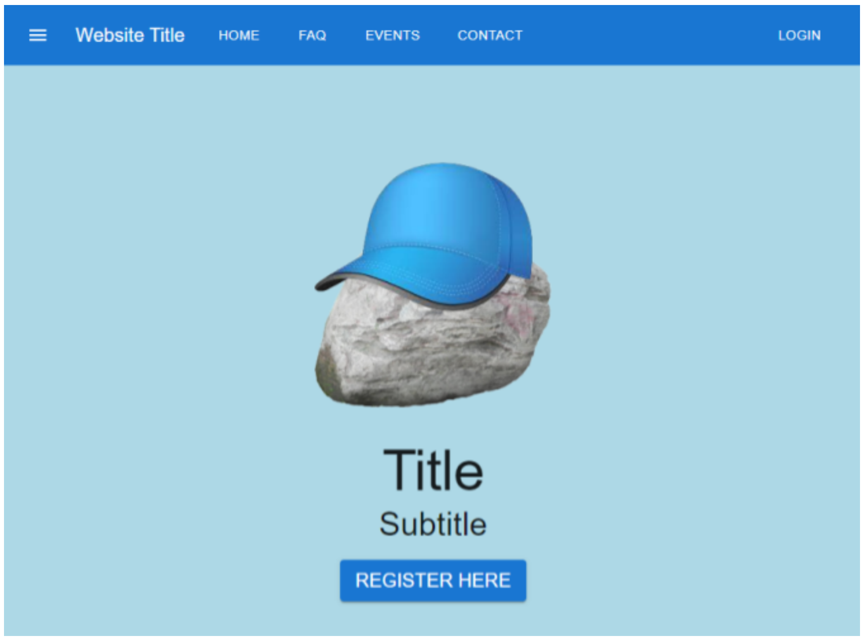


Figure 4.4.3 - About Section of the Homepage

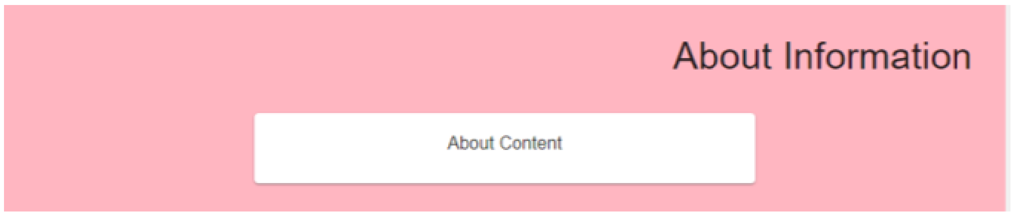
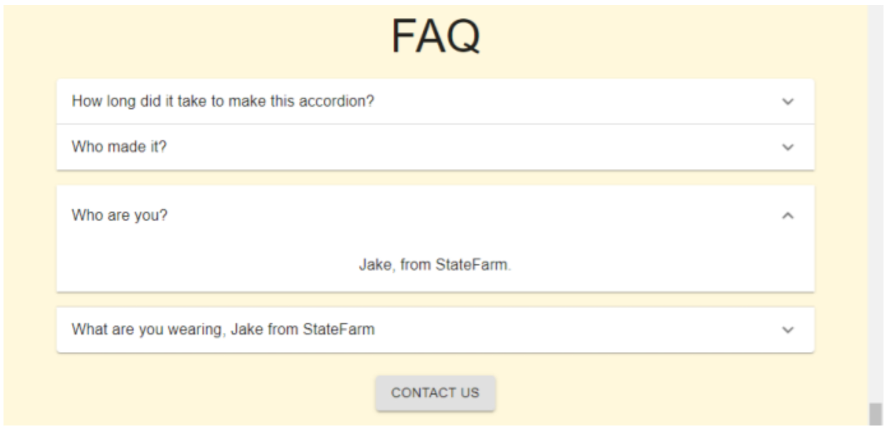


Figure 4.4.4 - Frequently Asked Questions Section


FAQ

How long did it take to make this accordion?

Who made it?

Who are you?

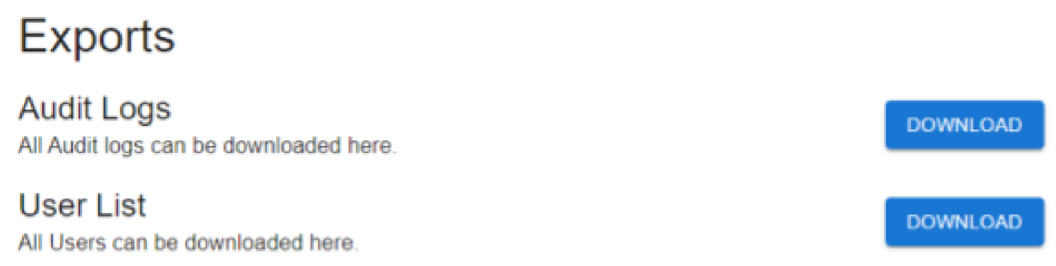
Jake, from StateFarm.

What are you wearing, Jake from StateFarm

4.5 Exports

The data exports page shows the various exportable datasets in a basic name-description-download format. Each entry shows the name of the dataset, a brief description of the dataset, and a download button that downloads the CSV file.

Figure 4.5.1 - Exports Page



Exports

Audit Logs

All Audit logs can be downloaded here.

User List

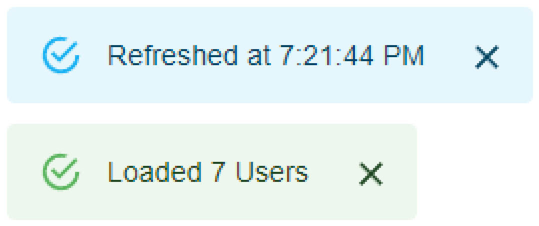
All Users can be downloaded here.

4.6 Alerts

An essential part of any interface is providing feedback to users of the application. Through this desire, the Alert interface was created.

Alerts come in various forms, success, error, info, warning, and more. These alerts are hooked into the main functionality of the system and show in the bottom left corner of the screen.

Figure 4.6.1 - Alerts shown by the User List



Refreshed at 7:21:44 PM

Loaded 7 Users

Exception handling is detailed more in its corresponding section.

5.0 Architecture

5.1 Overview

The project utilizes a variety of third-party systems to facilitate the rapid prototyping and development of our application. There are many features that we intend to implement, and it is critical that our project is easy to maintain, almost self-documenting, and abstracts the process of running our system by delegating complicated operations to third parties who specialize in such tasks. Third-Party systems are the backbone of our application and our project's development. These dependencies are detailed in the next section of this document.

5.2 Third-Party Systems

React

React version 17 is utilized to make our application modular and to abstract complicated re-rendering logic. Through the use of React, we can easily isolate the logic of our application from our application's interface and yet also nicely integrate it with ease, while also providing a uniform way to develop components of our application and use them elsewhere.

Material UI

Material UI version 5 is utilized alongside React as a User Interface Component Library. Because our application has many features yet is constrained to a short timeframe, the use of Material UI provides dozens of pre-made components with fully implemented features. Rather than coding the individual inputs for our application, one may just utilize Material UI's component library. Through this, we can maintain a high level of quality for our application's interface yet at a fraction of the effort, time commitment, and cost.

Next.js

Next.js version 12 is utilized to provide a production-ready React development environment. React itself remains just a library to build applications, but it allows the implementation of React and how the project is structured left up to the developer. Next.js facilitates a well-designed environment with fast project compilation, automatic page reloads on changes, proper error handling and debug tools, server-side rendering, and more.

Node.js

Node.js version 16 is utilized as the runtime for any backend logic in our system. Next.js runs off of Node.js, allowing us to use JavaScript and TypeScript to write both website code and backend database code; and even allow interoperability between some functions that can run on both the client and server.

TypeScript

TypeScript version 4.5 is utilized as a substitute for plain JavaScript. All portions of the application are written in JavaScript, so the enhanced tooling and IDE capabilities of TypeScript and the language being a strict superset of JavaScript make it a perfect fit for our project. Thanks to Next.js, TypeScript is inherently supported, thus switching to TypeScript is extremely low-cost yet facilitates very considerable productivity gains through detailed autocompletion and code documentation.

MongoDB

MongoDB Community Edition version 5 is utilized to store all data in our project. MongoDB is a document-oriented NoSQL JSON database, which is a perfect fit for our JavaScript-oriented application of loosely-related data structures.

Mongoose

Rather than interfacing directly with the MongoDB database, Mongoose version 6 is utilized to allow us to clearly define our database schema and perform database operations through object-oriented models. Through mongoose, we can modularize our backend database logic and also provide a clear interface to interact with our data.

5.3 System Architecture

The system is broken down into various services and components that interact with each other through an Application Programming Interface. With the exception of internal services, the API acts as the sole mediator and communication pipeline between any aspect of the system and another aspect of the system.

The following diagrams show the overall flow of data through the system, in addition to various subsystems. Arrows illustrate directional data flow, with each arrow utilizing a handful of isolated API routes for each subsystem.

As an example, the event system utilizes API routes located at `/api/events`.

Figure 5.3.1 - System Overview

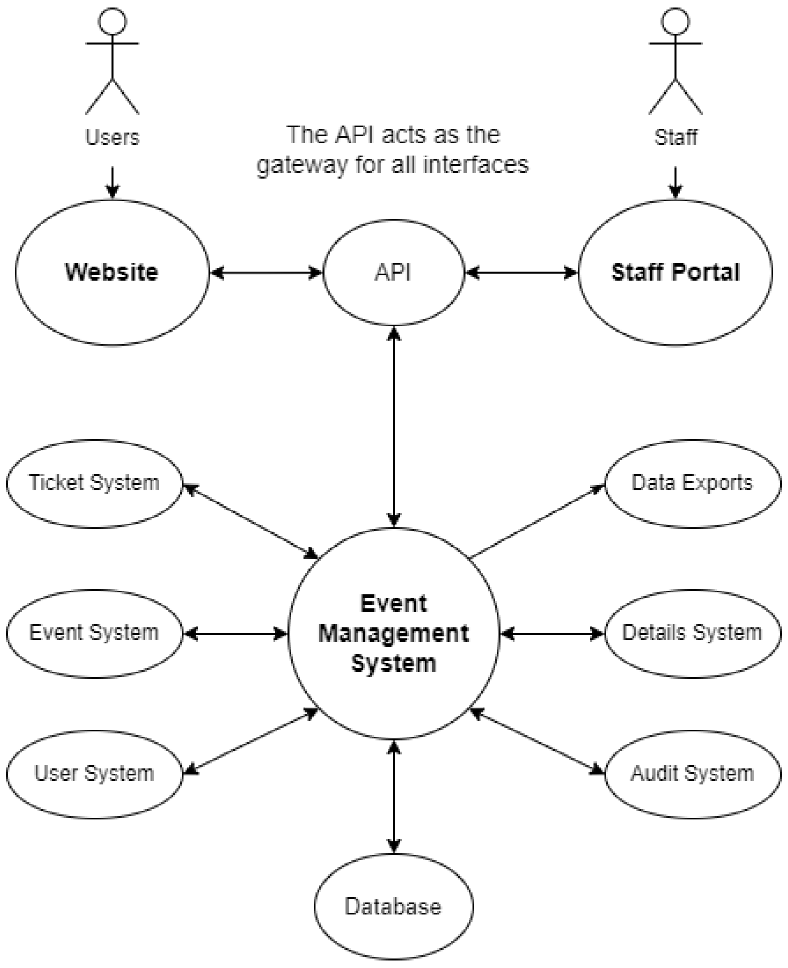


Figure 5.3.2 - Website Context Diagram

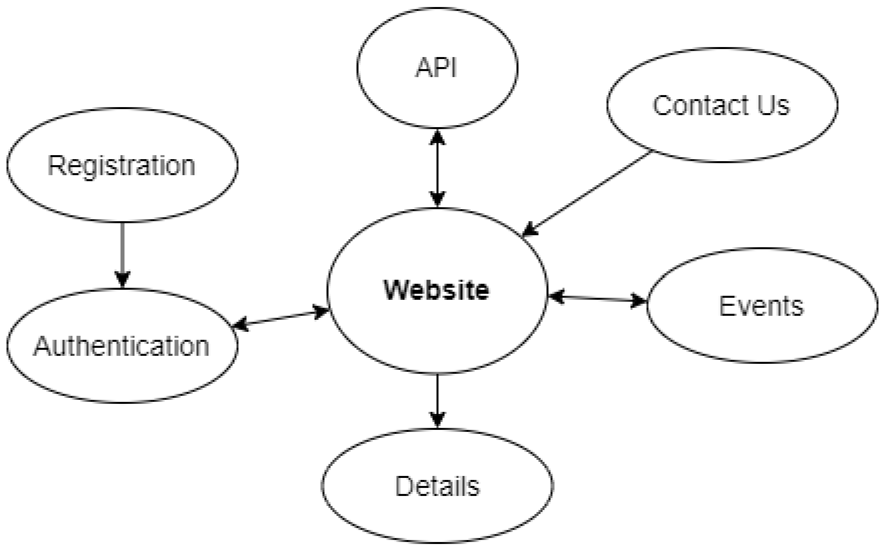
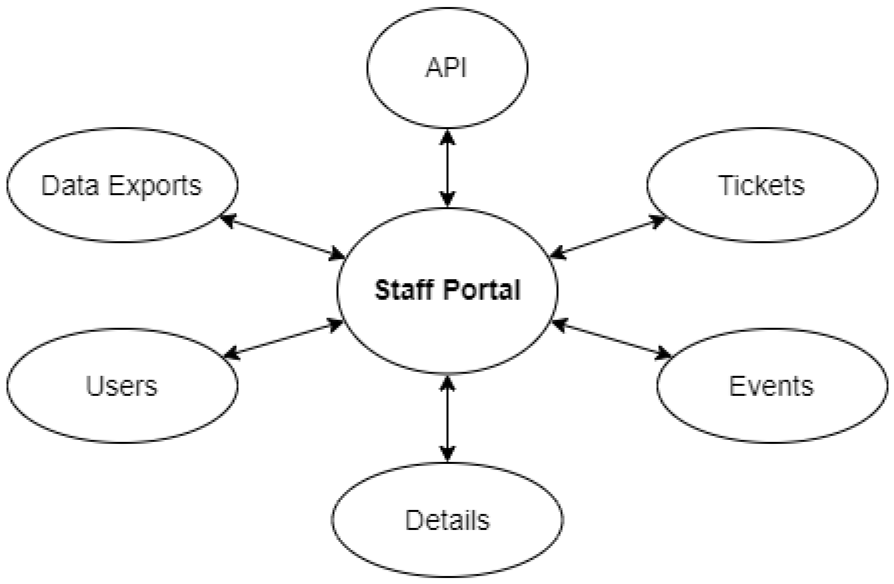


Figure 5.3.3 - Staff Portal Context Diagram

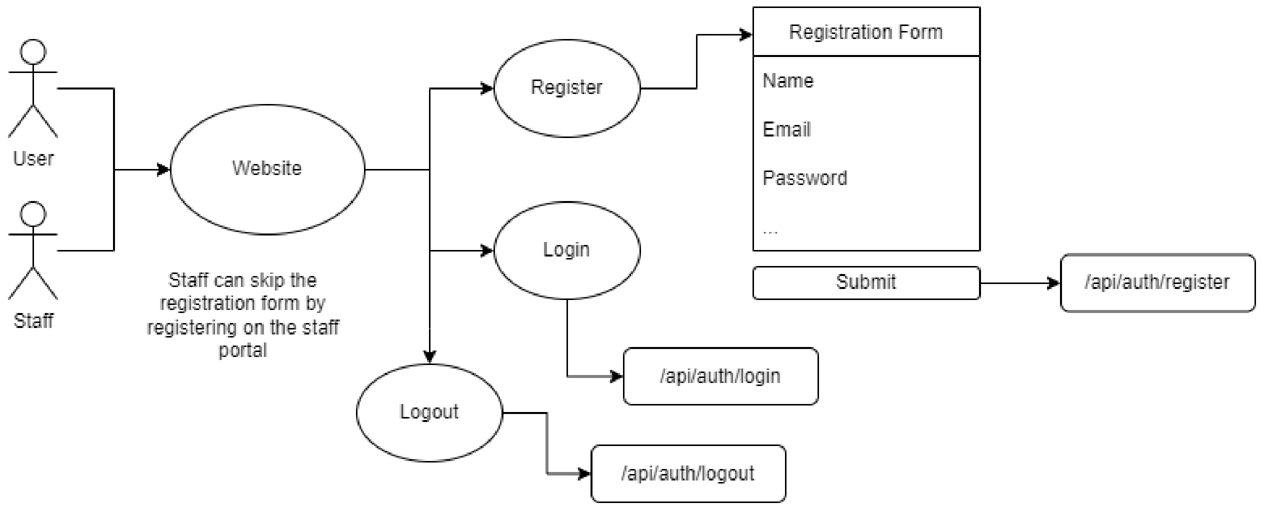


5.4 System Components

Users

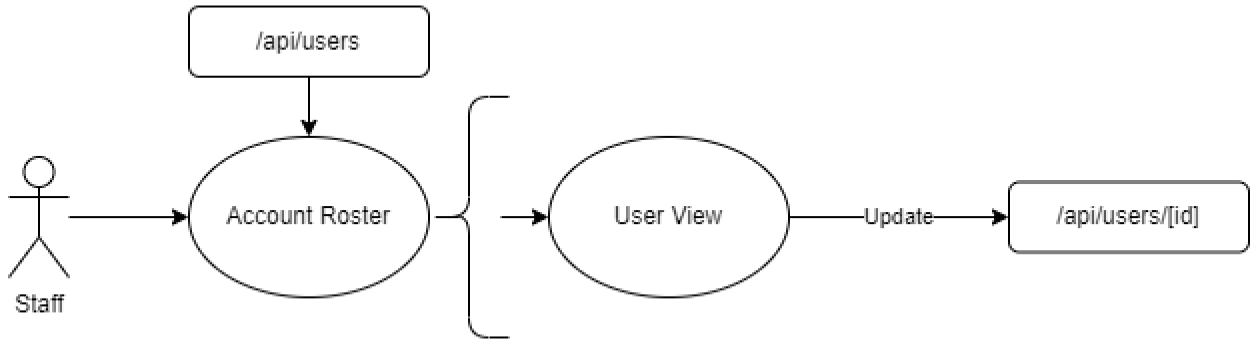
The Authentication process for the system involves basic login and registration. This component utilizes a stepper registration form in order to provide a good user experience for registration, while also providing a simple login interface as well. This component handles all authentication needs for the system.

Figure 5.4.1 - Data Flow for registration, authentication, and logging out.



The Account Roster, also known as the User List, is used to display a list of users from the database. Various filters can be specified, and other components may utilize the user list, such as showing a list of registered users on an event. This component is used to show users and perform operations on them.

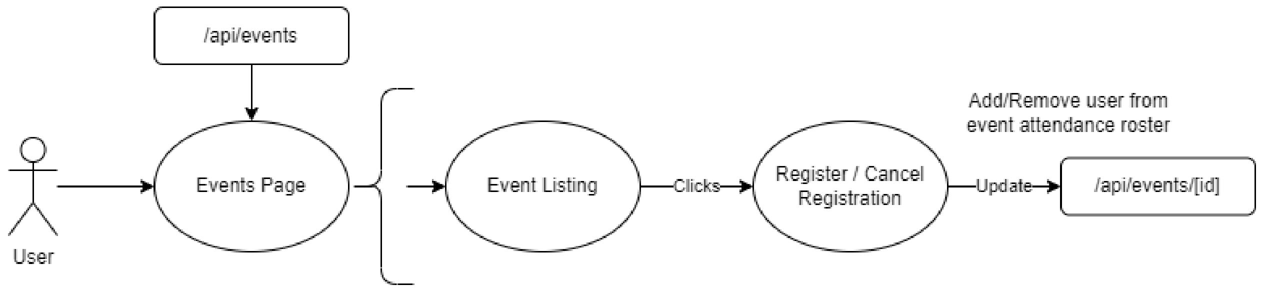
Figure 5.4.2 - Data Flow for displaying the user roster.



Events

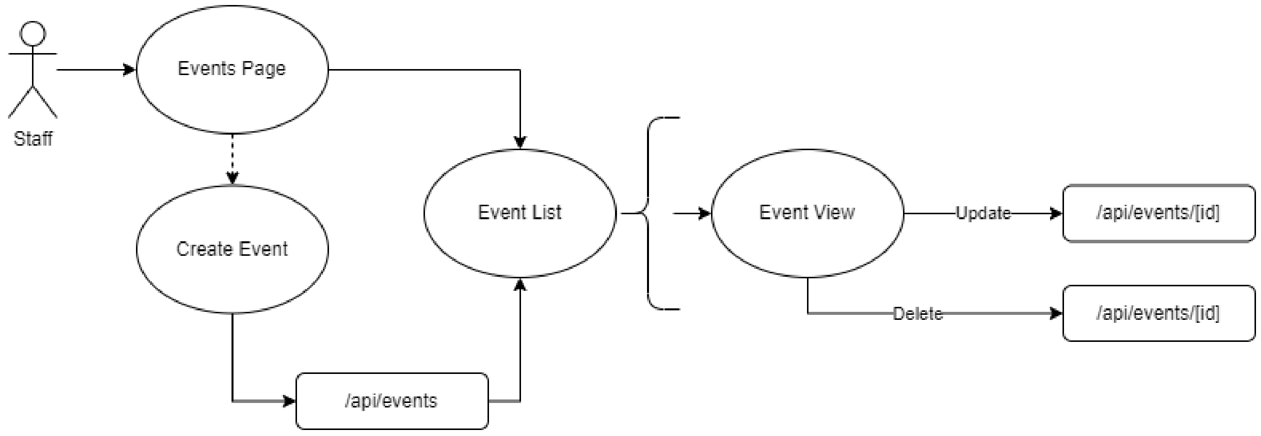
The Events component deals with all data and operations in regards to events. Users can view a list of events on the events page, which is pulled from the API. For these events, they can register (aka RSVP) for events, allowing them to be displayed on the staff portal along with the event itself. This allows Staff members to clearly see who is attending the event and facilitate potential communication with them.

Figure 5.4.2 - Data Flow for how events function on the frontend.


/api/events
User
Events Page
Event Listing
Clicks
Register / Cancel Registration
Add/Remove user from event attendance roster
Update
/api/events/[id]

The Staff Portal allows Staff members to create, view, and edit the various events in the system. A list of all events can be pulled from `/api/events`, with individual events being able to be created and updated from `/api/events/[id]`, with the id being the identifier for the event in the database.

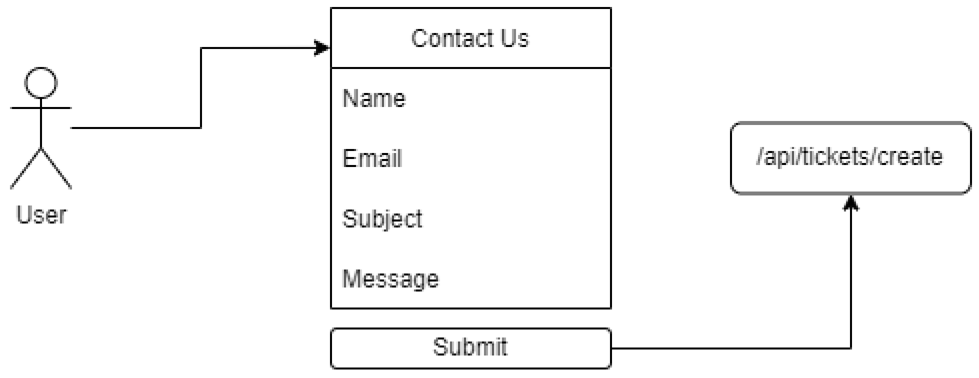
Figure 5.4.3 - Data Flow for how the events portal functions.



Tickets

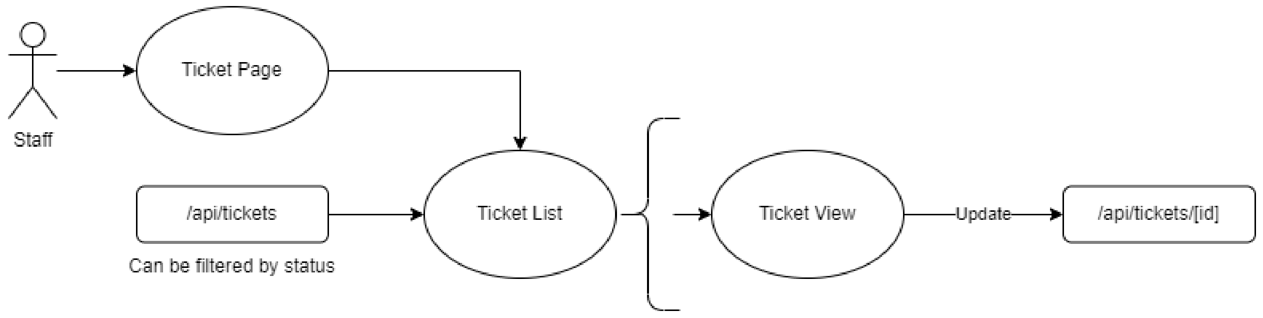
The Contact Component on the frontend is the main entry point for creating tickets that appear on the Staff Portal. Users fill out a basic contact form to create tickets, with the data being posted to an API endpoint and providing useful user feedback to the user informing them of successful or unsuccessful submission.

Figure 5.4.4 - Data Flow for Users creating tickets



On the Staff Portal, those tickets are loaded from the tickets API endpoint and used to populate a Ticket List. Each ticket can then be expanded into the Ticket View, allowing edits to be made to the ticket itself, and having all updates being sent back to the individual ticket API endpoint.

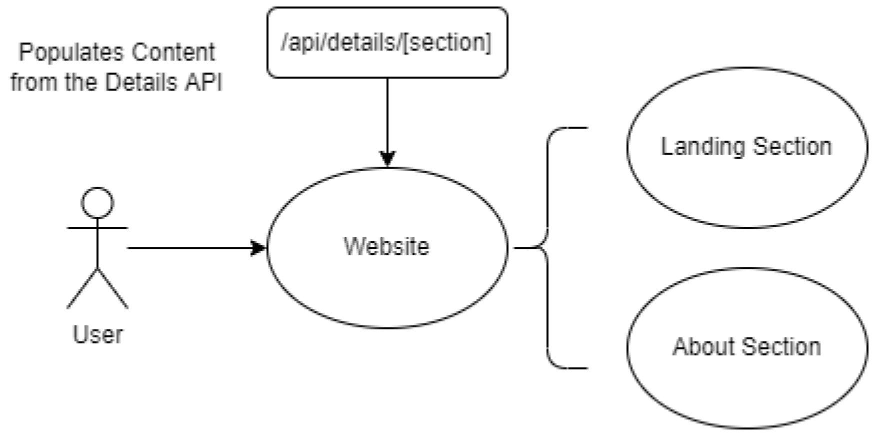
Figure 5.4.5 - Data Flow for Staff interacting with tickets



Details

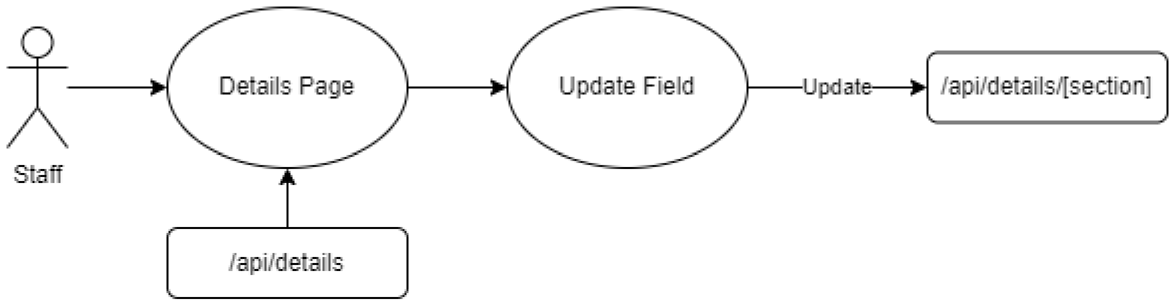
The Details system component is intended to make it easy for Staff to edit the textual content on the frontend without having to delve into the application's code or perform any deployments. This is simple string substitution with data stored in the database, and having said data be accessible and updatable via API endpoints. The website simply pulls this information from the website and inserts it into the corresponding sections of the website.

Figure 5.4.6 - Data Flow of pulling details from the API



On the Staff Portal, these fields appear as simple text inputs that can be edited and saved with the click of a button. Clicking the save button after making changes uploads those changes to the API, and all clients visiting the website receive that update in under a dozen seconds thanks to the built-in auto-refreshing.

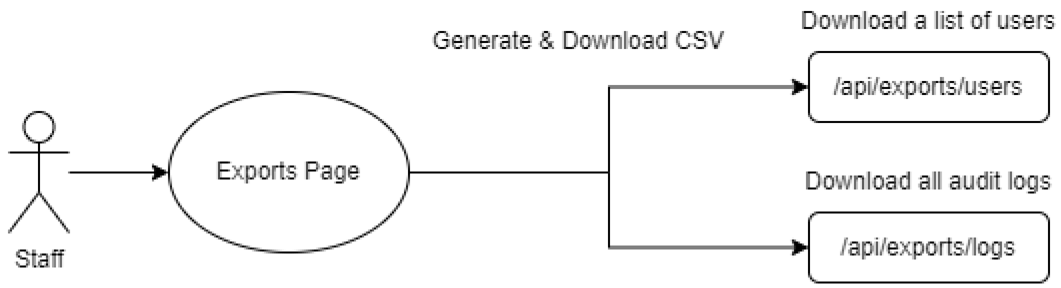
Figure 5.4.7 - Data Flow for editing details.



Exports

Exporting data from the database is essential for nearly any system, thus the inclusion of the Exports system component. This component allows logic to be defined to easily translate an entire database table into a nicely formatted Comma Separated Value file, also known as a CSV. Initially, only two exports are defined; user exports, and audit log exports. The user's endpoint downloads a CSV of all users registered in the system, while the logs endpoint downloads a CSV of all audit logs in the system; effectively allowing Staff to keep track of who made what changes to what objects in the database.

Figure 5.4.8 - Downloading exports from the database



5.5 Hardware Components

Server

In order to run our application, an internet-accessible computer is required to serve the application. Our project is designed to use a minimal amount of resources.

During the development of our application, our project is being deployed to a Droplet from DigitalOcean. This droplet has 1 GB of Memory, 20 GB of SSD internal storage, and 1 “Premium Intel” CPU. Given this, it would not be difficult to host our application on nearly any personal computer as long as one would know how to port forward and make such a computer accessible via the open internet.

It is preferred for the server to run Ubuntu Linux at a version greater than or equal to 18, but the project is unlikely to encounter problems on older versions within reasonable versioning proximity to Ubuntu 18 or other distributions of Linux.

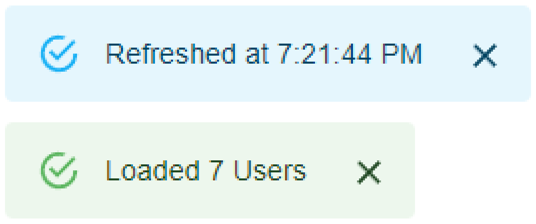
Internet Browser

A device capable of utilizing an internet browser is essential to all users and event managers for our application. Given our project is web-based and all operations are performed via a website, an internet connection and accompanying browser are necessary to utilize our application.

5.6 Exception Handling

All systems need to be designed to properly handle exceptions to ensure high availability, and this system is no different. Thanks to the utilization of a JavaScript-based stack, the possibility of encountering a critical error is slim; and the inclusion of TypeScript's tooling to ensure the project builds assists in finding bugs and critical flaws long before they are deployed into production. Because of this, exception handling is primarily user-focused with various alters being used to provide feedback and status updates to users as they use the application.

Figure 5.6.1 - Alerts shown by the User List


Refreshed at 7:21:44 PM

Loaded 7 Users

Alerts such as these make it easy to provide feedback to users and inform them of failed data loads, errors, and other issues; and even more so when things are done successfully.

However, in the event of a critical system failure, there are multiple failsafes implemented to prevent any significant downtime of the application. To begin, there are dozens of error handling blocks located throughout the code and in error-prone areas to ensure most errors are caught and properly handled. If an error occurs and it is not caught by these catchers, any critical failure in the code that causes the program to exit will also trigger an immediate restart of the program thanks to software such as Docker and PM2, the process management solution used on our production server.

The system is designed to utilize resilient technologies that are very flexible and difficult to break. Overall, one would be hard-pressed to find a way to break the software in an unrecoverable way through normal operation. Even then, all data is stored in simple and portable form factors that allow for easy mitigation of issues and recovery. Even MongoDB is as simple as a drag and drop of the datasets in order to restore the database.

6.0 Quality

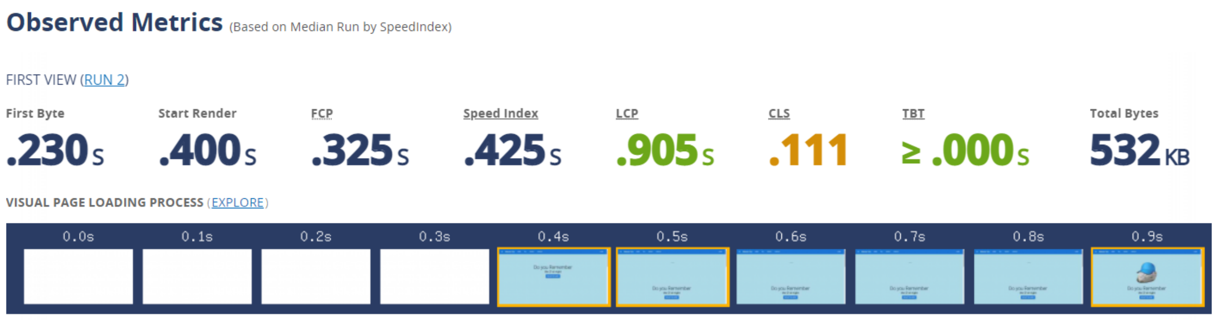
6.1 Performance

The expected capacity of the system assumes there be no more than a thousand users. All technology utilized in this project is capable of handling well in excess of that number, but it is also assumed that any larger events would have their own team behind it and would not be looking to use a system such as this one, given the size and likely backing of an organization of that size.

As a result, the expected performance is not to be as fast as a static webpage due to the project's utilization of dynamic pages and routes, but should still be fast enough that users have no issue using the website on a variety of platforms and internet speeds.

To test performance, WebPageTest.org was utilized to perform benchmarks of the project's deployed website. A highlight of the results is visible, with the full report accessible here. As one can see, it effectively takes 0.9 seconds to finish loading all content on the page, which is remarkable considering none of the data is cached with this first run of the test.

Figure 6.1.1 - Performance Test of the Website by WebPageTest.org



Observed Metrics (Based on Median Run by SpeedIndex)

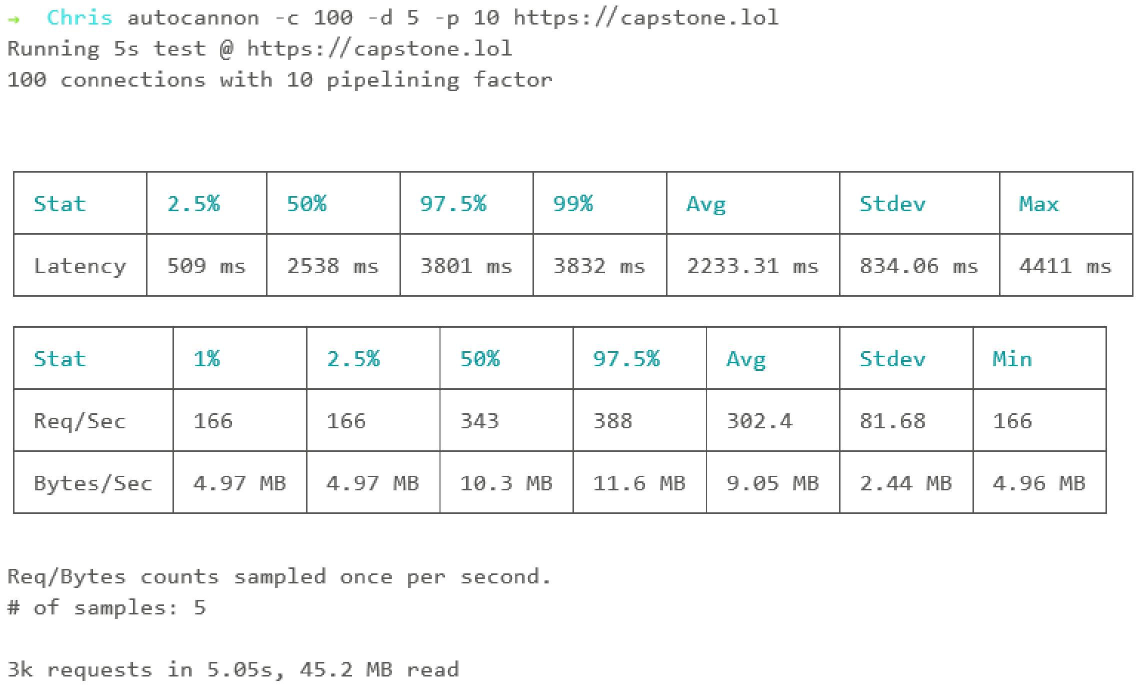
FIRST VIEW (RUN 2)

First Byte	Start Render	FCP	Speed Index	LCP	CLS	TBT	Total Bytes
.230s	.400s	.325s	.425s	.905s	.111	≥ .000s	532KB

VISUAL PAGE LOADING PROCESS (EXPLORE)

In addition to WebPageTest, a tool called AutoCannon is utilized to stress test our webserver via the website and staff portal. AutoCannon sends out hundreds of connections at once to calculate latency, response times, bytes per second, and more. Those results are viewable below, with all requests originating from a modern desktop computer wired into Kent State University's campus internet. The server is hosted on a $10/month DigitalOcean Droplet located in their New York data center.

Figure 6.1.2 - AutoCannon Stress Test of the Website



```
→  Chris autocannon -c 100 -d 5 -p 10 https://capstone.lol
Running 5s test @ https://capstone.lol
100 connections with 10 pipelining factor
```

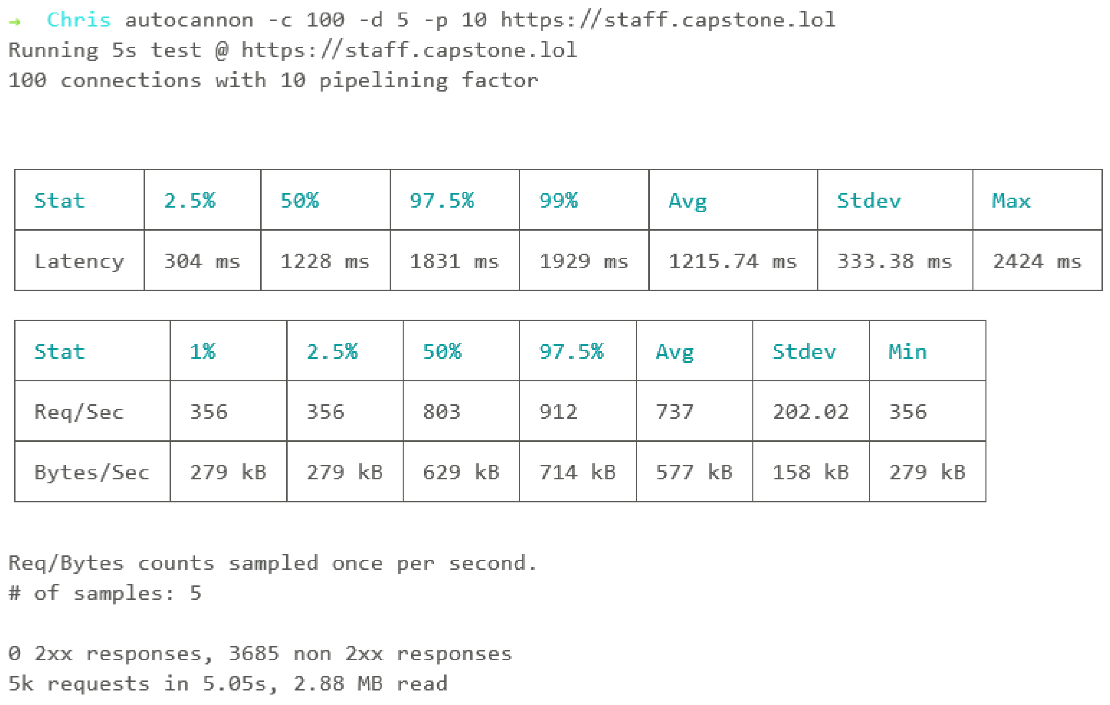
Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	509 ms	2538 ms	3801 ms	3832 ms	2233.31 ms	834.06 ms	4411 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	166	166	343	388	302.4	81.68	166
Bytes/Sec	4.97 MB	4.97 MB	10.3 MB	11.6 MB	9.05 MB	2.44 MB	4.96 MB

```
Req/Bytes counts sampled once per second.
# of samples: 5

3k requests in 5.05s, 45.2 MB read
```

Figure 6.1.3 - AutoCannon Stress Test of the Staff Portal



```
→  Chris autocannon -c 100 -d 5 -p 10 https://staff.capstone.lol
Running 5s test @ https://staff.capstone.lol
100 connections with 10 pipelining factor
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	304 ms	1228 ms	1831 ms	1929 ms	1215.74 ms	333.38 ms	2424 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	356	356	803	912	737	202.02	356
Bytes/Sec	279 kB	279 kB	629 kB	714 kB	577 kB	158 kB	279 kB

```
Req/Bytes counts sampled once per second.
# of samples: 5

0 2xx responses, 3685 non 2xx responses
5k requests in 5.05s, 2.88 MB read
```

6.2 Security

The security of the system is relatively secure for a system of this caliber. Although not a major focus of the project, basic security and protection has been implemented nevertheless. All passwords are hashed and basic web security is implemented to prevent the usual web vulnerabilities from being applicable, such as cross-site-request-forgery. All sessions are implemented with JSON Web Tokens, ensuring easy access to session data by clients yet remaining secure. As expected of a system that includes sensitive data, all API routes that provide access to data are protected via a permission system that is integrated with user roles; ensuring that no one has access to data they are not allowed to access.

6.3 Reliability

The system itself should be quite reliable for anyone using it. Thanks to it being written via the Node.js stack, utilizing the NGINX web server, a MongoDB document database, and TypeScript to find issues during compilation; every point of the software is designed to be resilient to errors and avoid any critical failures during runtime. On top of this, Docker is set up to automatically restart each aspect of the system in the event of a critical failure that ceases execution. With all of this factored in, the system has never encountered a major critical failure during development, providing evidence that the system should be quite reliable in production.

7.0 Conclusion

After months of eliciting requirements, planning, and development; the project is very close to what it was intended to be. All major features have been implemented, although there remains much to improve upon. The project kept true to its motivation of being an improved system that could be a drop-in replacement for Kent Hack Enough, while also remaining simple and configurable enough to be applicable to a variety of use cases and other organizations.

7.1 Future Work

Overall, the system is quite fleshed out and extremely versatile while looking beautiful thanks to the use of Material UI. The development team will be reaching back out to the Kent Hack Enough team to see if this project could be integrated into the hackathon as a replacement for its original and aging backend system. If the team approved, the development will proceed on the system and fine-tune it for use for HacKSU's next event, Kent Hack Enough 2022. Only slight adjustments will be needed, and a few more features will need to be added to the system; yet all the essentials are already present.

Regardless, this system remains part of Open Source and is freely available on the GitHub Repository. Anyone may fork this project and make their own changes to it at their own discretion.

7.2 Acknowledgments

Thanks to Alfred Shaker for being an exceptional stakeholder in this project and guiding the development team throughout the entire process.

Kent Hack Enough's annual event remains the basis for the motivation behind this project. The event and its organizers were incredibly helpful in providing a baseline and feedback for such an event management system.

Lab Instructor Safa Shubbar was incredibly helpful throughout the semester in assisting the team in documenting and detailing the project.

Finally, Dr. Samba's material and lectures provided essential information and guidance on completing this capstone project, especially in maintaining the quality of all materials.

Event Management System

a Capstone Project

GitHub Repository

Live Website

Live Staff Portal