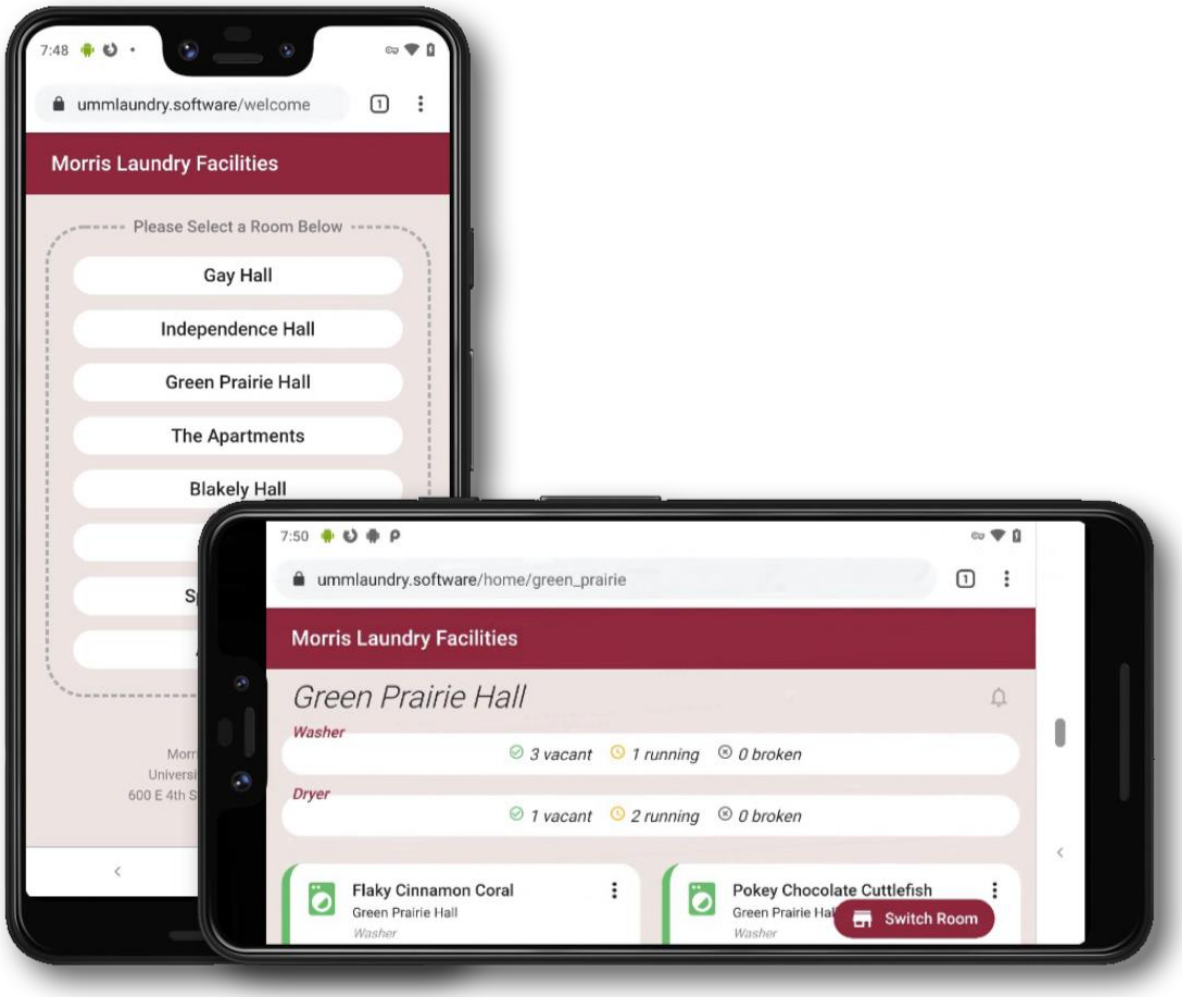
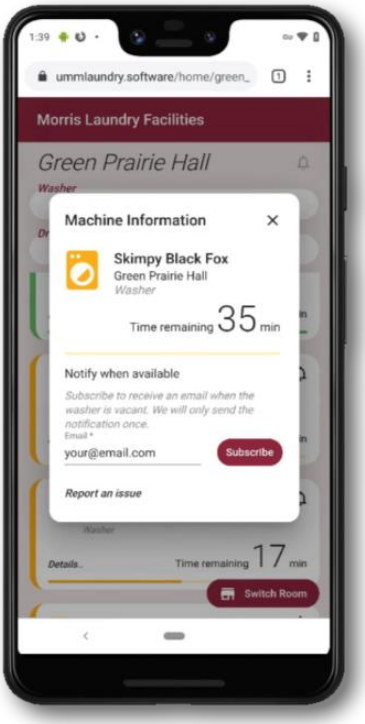
UMMLAUNDRY .SOFTWARE



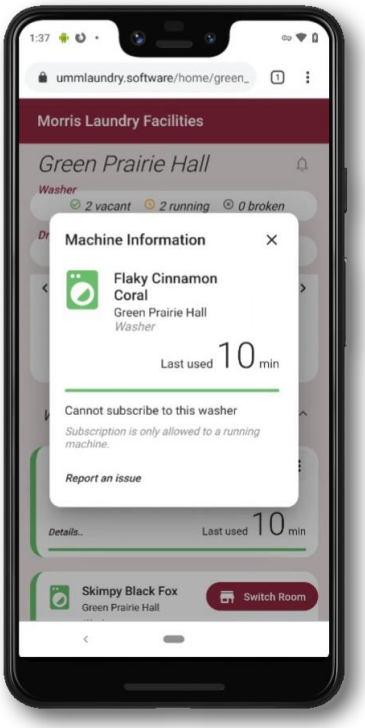
Robert Beane
University of Minnesota Morris
https://github.com/Robert-Beane

What is UMMLaundry?



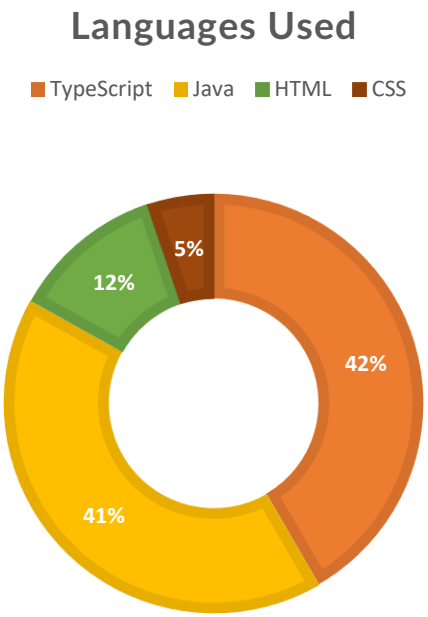
UMMLaundry is a web app for the students of Morris to easily see how many laundry machines are currently being used and if it would be a good time to do laundry. Each hall has its own dedicated page listing all machines. Each machine includes time remaining or the amount of time since its been last used.

There is also a simple status indicator telling the user how many of each machine type are available, running or broken. If a user discovers that a machine is broken, there are buttons within each machine that send the user to a report form with pre-filled in information regarding what machine is broken and where.



If all machines are running, a user can sign up for email notifications for the entire room or their favorite machine. They will then get an email letting them know their machine or any machine is available to be used. The webpage is also compatible with all modern browsers meaning it can be accessed almost anywhere and still function properly.

How we Made UMMLaundry



The final project was a product of four different iterations with four different teams. Tools such as JetBrains IntelliJ IDEA and ZenHub were used throughout the development of the web app. A mix of different languages were used when developing the final product.

Different Frameworks

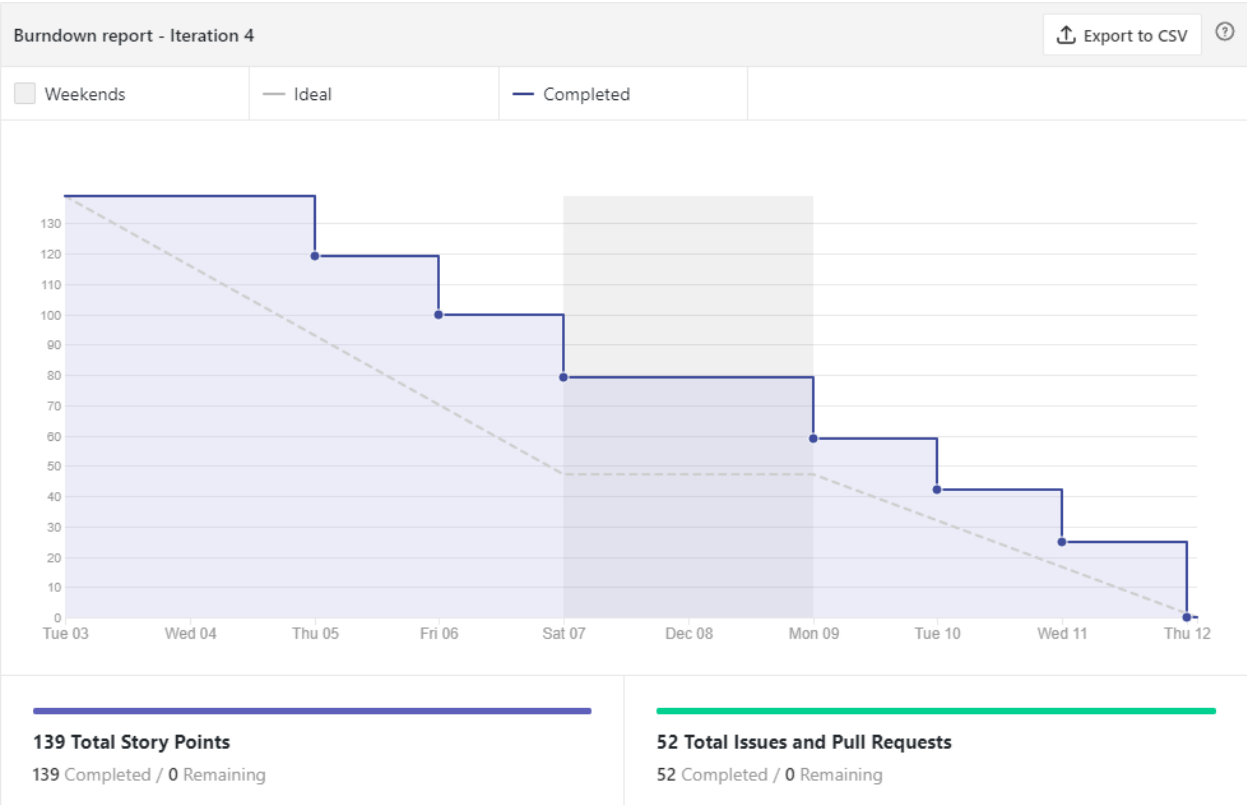
Throughout development, frameworks used stayed the same. Angular 8 was used for all thing's client side and Angular Material for styling and icons. The server was comprised of a Java Spark server with MongoDB serving as our database. Hosting took place through DigitalOcean along with later iterations using Cloudflare to allow HTTPS to be used.

Languages and Tools Used

For the client side, we used Typescript with Angular. We used a large assortment of different HTML and CSS elements throughout the web app. Yarn was used for installing packages. Testing relied on Karma and Jasmine. Third party dependencies such as Chart.JS and ngx-cookie-service were used within the project.

Using Agile Throughout the Iterations

During the course, we actively used an agile approach. At the very start, the class created an inception deck so we could have ideas of what the final project would contain and maybe look like. We had a showcase every two weeks to show off our teams work to the customer and received useful feedback. We also had the customer “shop” the stories we had chosen for that iteration. Throughout every iteration, we used ZenHub through GitHub to help us display our stories and make estimates for each story. At the end of the iteration, we had a completed burndown chart. We also used ZenHub to create issues regarding the iteration so we had an updated list of known issues so we could know what we needed to fix before the iteration deadline.

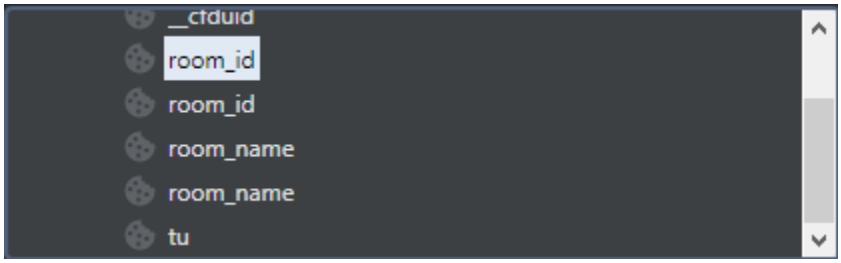


My Personal Contributions

Throughout the iterations, my main focus was with the user experience on the front end through Angular, Material and CSS. Besides the front end, I also was the one that deployed the iteration's project on DigitalOcean.

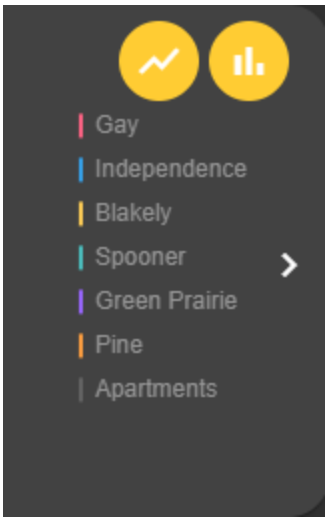
With iteration one, I focused on how machines were displayed. The implementation was very minimal, the only notable things that kind of stayed were the addition of color indicators for the status of the machines. I also worked on basic sorting which sometimes worked and sometimes did not.

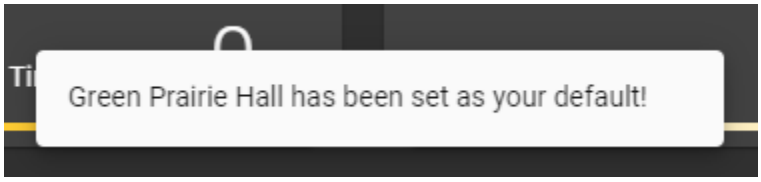
With iteration two, I mainly focused on the addition of a proper dark mode. Dark mode was something that was overlooked and forgotten about with the first iteration, so it was added with this iteration.



Iteration three had the most changes out of all the iterations. My main focus for iteration three was the addition of cookies. We used a dependency called ngx-cookie-service which had the exact functions we needed. When a user set a specific room as their default, cookies were stored which then redirected the user on refresh to the page they set as default. Another use of cookies that I added was the function of being able to set a preference for the graph. I used cookies to remember either bar or line which was then called

within the buildchart() function. Besides my work on cookies, I also worked on some CSS and HTML. I added an angular snackbar to pop up when a user set a room as default to let the user know that they actually set a room as default. Since I added the functionality of switching graph types, I added two buttons that would change the graph type and set the correct cookie. Along with all the client-side changes, I also set the website up with a proper domain and set up HTTPS with Cloudflare.





Our final iteration, iteration four, contained a lot of final adjustments and finalizing what exactly was wanted from the customer. I worked on adding different color progress bars to match up with the status of the machine making it easier for the user to tell the difference. I also did some work on the overall colors of the web app, finalizing exactly what colors and where we wanted the colors to go. I deployed the project in the same way I did for iteration three. However, we had issues with the graph showing data from six hours in the future. Because of this, I had to fiddle with the DigitalOcean droplet to make it so the data is displayed at the correct time. I also worked on finalizing the README and writing documentation to explain certain elements of the project. I also created the pamphlet for our iteration.

Links

GitHub Page

https://github.com/UMM-CSci-3601-F19/iteration-4-rockin-reindeer

GitHub Repository

https://github.com/UMM-CSci-3601-F19/iteration-4-rockin-reindeer.git

Promotional Pamphlet

https://github.com/UMM-CSci-3601-F19/iteration-4-rockin-reindeer/blob/master/Documentation/softDesignBrochure.jpg