Sprint 3 Report

Katie Rose Lansing, Sebastiano Grace Riverso, Cameron Alexander Schmidt

Improvements to Design

We continued to improve our Figma designs and added more to the content pages following the end of Sprint 2. One improvement we made for the homepage was to have the quick navigation section fixed while scrolling so that our homepage is divided into two rows with three sections each. We added more examples of content pages to finalize what elements would be common across multiple pages. Additionally, we spent time deciding on how we would organize the content of these pages to be less overwhelming for the reader, while still conveying the necessary information.

We found that a header, subheader, button, and text block were all commonly reused elements for each page, so these were the main components we focused on reproducing when it came time to create our website.

Figma Homepage
Figma Content Pages

After reaching a satisfactory point with the Figma prototype, we moved on to creating our Next.JS website.

Updated Menu Organization

In our previous designs, we had an idea of what our menu would look like but hadn't considered the exact categorization of each page of our menu. In this sprint, we finalized the pages within each menu. Many changes we made were to consolidate similar information on different pages together into one page. We reduced the number of total categories by two. To better organize our lengthy list of categories, we continued with our design from the last menu iteration by dividing the pages for the categories of Events and Family Resources into two sections.

For exact details of our menu, see the following doc: AACMS Menu

Next.JS Prototype

Setup:

The following steps are required to set up our Next.JS website to be hosted locally:

1. Clone the repository or pull the code from GitHub: https://github.com/sebriverso/HCI-Blue-Team
2. Navigate to the /aacms_web_page dir in the terminal
3. Run the following commands:
   $ npm install
   $ npm run dev
4. Open your browser and navigate to http://localhost:3000
5. (OPTIONAL) using dev tools to set the screen to mobile size (only will work for homepage)

Web Version:

We created the web prototype using Next.JS, incorporating the languages of CSS, HTML, and TypeScript and began by implementing the homepage to have something to work off of as we worked on fabricating the components we would need for the rest of the site. The homepage closely resembles our design

Modular Design:

We made a conscious effort to make all components as generalizable as possible, considering how many content pages we had to populate. Because of this, you can see in our Components folder the many different components used for headers, lists, buttons, etc. This makes it extremely easy to add more pages and content to the website, while also allowing us to make website-wide design changes by editing one file. This aligns with current web standards that look to reuse components for a more streamlined building experience.

This modularity was particularly useful for our content pages. We originally had the idea to combine a header and text into one component which worked well in some instances, but realized we needed more flexibility, so further created three more components: a header, subheader, ordered list, and text component. These components worked well to populate each page with consistent-looking content.

Pain Points:

Submenus Appear on Hover:

Creating the submenus proved to be difficult, the onMouseEnter and onMouseLeave listeners were somewhat difficult to find out about. Additionally, some tinkering was needed to make sure that boolean values for showing the submenus stayed true when the onMouseLeave listener for the menu button was triggered but the mouse was hovering over the submenu. This took a bit of time to make sure that movements could be fluid and that hovering over any space on the Submenu kept it open.

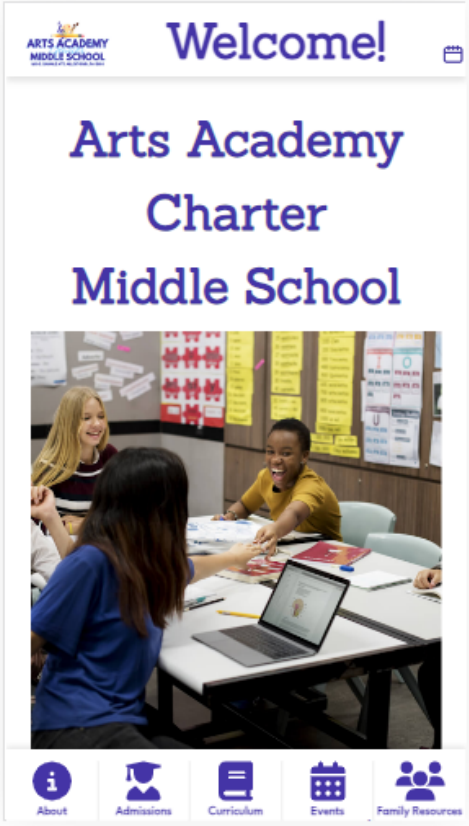
Sticky Components:

With a lot of our quick navigation components, it is important that they stay in the same areas when scrolling while not overlapping with headers and footers. This was hard to implement at first, but through setting fixed positions and z-index layering we

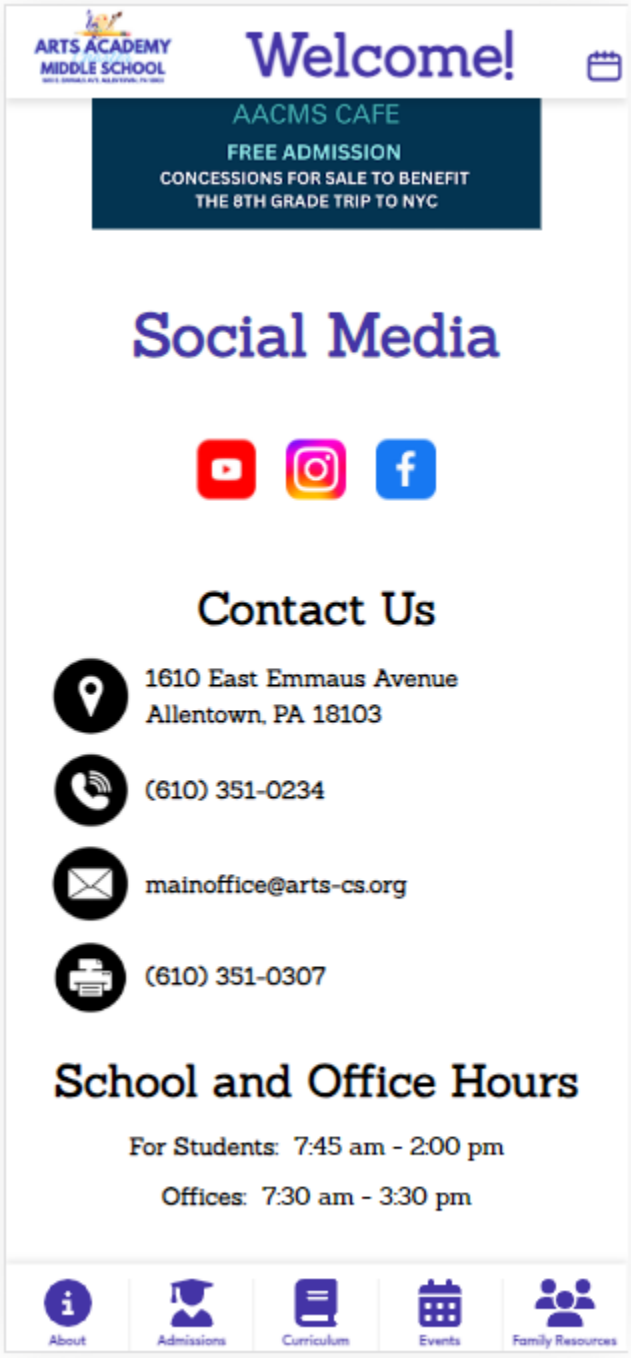
were able to accomplish this. This makes our website feel much more professional and visually appealing by reducing whitespace.

Mobile Version:

Upon meeting with Professor Smith, we were blatantly met with the fact that our design did not incorporate mobile compatibility. Our design did not scale to small screen widths and some of the components (including menus and dropdown navigation menus) would not be able to be adapted to these mobile devices, making one design for all screen sizes impossible. To address this we started to work on a mobile version of the website with adjusted layouts to make mobile user-friendly and mimic real-world apps. Cam adapted these designs following Google's guidelines for Android app development and layouts, meaning our designs follow protocols and standards that are industry standard in mobile development. Currently, these designs are work in progress and need to be built upon but a prototype homescreen was created as shown below. The website currently implements functionality to detect mobile devices and adjusts the layouts accordingly.



Adjusted header content incorporated on homepage:



Future Improvements

Web Content:

Many of the pages from the original site still need to be added to the new prototype after refining and paring down some of their content. Additionally, there is still some work to be done on the header as two of the navigation buttons need to have a second column added to them and destinations need to be solidified so navigation works fully. We will need to refactor the homepage to allow the Quick Navigation section to properly stick and follow the user's scrolling, and add events to the calendar component.

More customizability with the content page text would also be preferable, but this is limited by the functionality of parameters passed through Next.JS to each component. We found that it's easy to pass strings and lists, but not blocks of HTML making it hard to modify text to be bold, italicized, etc. in the components. Most times, we then resorted to rewriting our HTML so finding a workaround for this problem would be ideal.

Lastly, we'd like to give the website more character by adding more images and graphics into the content pages themselves, so finding a way to integrate these would be another improvement.

Mobile Layouts:

The mobile layout needs refinement, especially the header and footer. The header needs to be redesigned to make the schedule icon larger and placed in a better position while the footer click navigation functionality still needs to be implemented.

The content pages also need to be adapted to be mobile friendly, but once we have a template this should go quickly and we should be able to populate these pages.