VTHSPCwebsite

Final Report
CS4626 Multimedia, Hypertext, and Information Access
Virginia Tech, Blacksburg VA 24061

Client: Dr. Godmar Back
Professor: Dr. Edward A. Fox

Jakob Ashworth
Josh Martin

5/9/2023

Table of Contents

Table of Figures

Table of Tables

1 Executive Summary

Virginia Tech hosts a yearly high school coding competition. This contest is held online and contains information on current and past events. The website is dated, which makes it unappealing and hard to navigate. Our client, Dr. Godmar Back, tasked us with the goal to create a new website, with the new website meeting a few requirements. The first requirement is for a new modern design that is both easy to navigate and appealing. The second is the preservation of already existing content of the website and facilitating the addition of new content for each new competition. The client also included a few stretch goals including ease of update, search engine optimization, and allowing users to create profiles and register for competitions online.

We have completed the first requirement of a new contemporary design and the second by preserving the already existing content and making it easy to add new content. We have improved the ease of update, making the update time less than thirty seconds. Lastly, we have greatly improved the website's search engine optimization score. The future work for the website would be on the stretch goals we were unable to achieve. This would mainly be about the user profiles. This includes dynamic content like maintaining user profiles and signing up for yearly contests. It also includes keeping track of statistics of the contests.

Anyone could theoretically use the website by visiting https://vthspc.cs.vt.edu/, but the main users would be those interested in the Virginia Tech High School Programming Contest. Users would be able to see detailed information about the current and past contests. This information ranges from organization details like when and where the contest is held to rules, regulations, and FAQs.

We learned a great deal during the course of this project. We learned about Markdown, which is a lightweight markup language making it easy to write

and edit website pages. We also learned about Docursaurus which is used to build, deploy, and maintain websites. This allowed us to use Markdown to develop.

Our team's main goal for this project was to not just improve the look and feel of the website but to also attract the new generation of coders. We believe that what we completed for this project would help those interested in coding actually take the first step. This high school programming contest is a great way to get high school students involved in the first place, but with a better and modern design, even more people could be attracted.

2 Introduction

To approach creating a new website from scratch, a clear problem and motivation is needed. From there, our team can craft a general approach that will satisfy the requirements of the new website given by the client.

2.1 Client

Our client, Dr. Godmar Back, is an Associate Professor at Virginia Tech, and the hoster and maintainer of the current VTHSPC website. After hosting the contest for nearly a decade, Dr. Back is no longer satisfied with the state of the current website, and wishes to see it updated.

2.2 Problem

As of now, the VTHSPC website is no longer meeting the needs of the client. Firstly, the website looks and feels outdated, which is not ideal at this point in time. With it being 2023, most websites have a modern look and feel, making them easily usable by the user. Secondly, the website is hard to update year after year with new information, which is not good when the high school contest is held yearly.

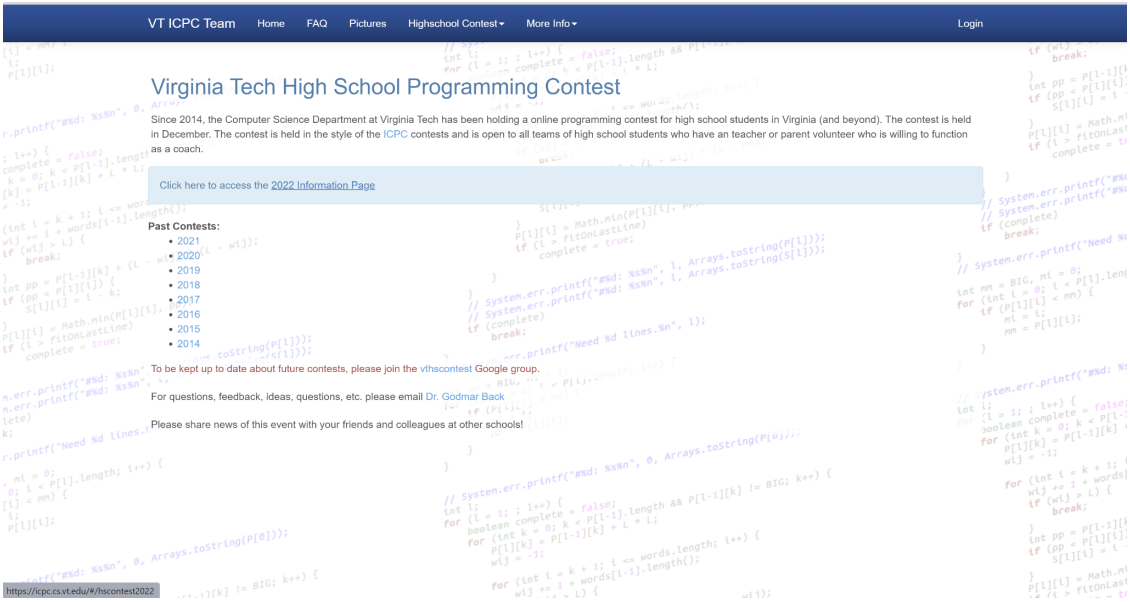


Figure 1: Home page of current website

As seen in Figure 1, the home page is not very appealing to the average user who will come across the website. Due to the unappealing look, users could become disinterested in the content of the website, and subsequently not want to participate in the contest. The home page of a website should draw in users, making them want to stay on the site and learn more about the topic.

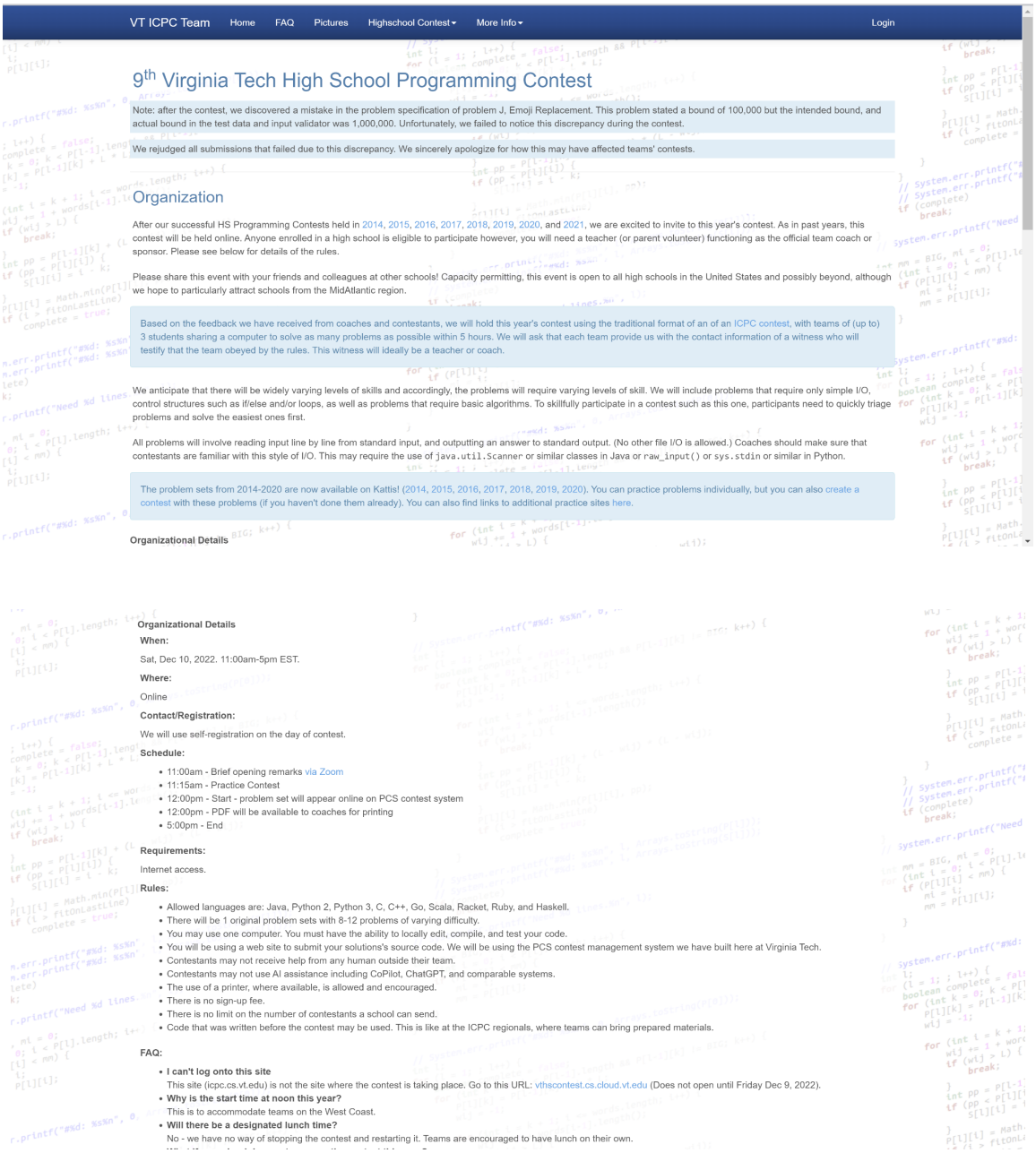


Figure 2: Year page of current website

There are year pages on the current website that each display the information of one prior competition. Each one of these pages (see Figure 2) has the same design as the home page, with the listing of a lot of information about the contest. Due to the large listing of information, the page looks cluttered, which can cause users to become confused and unwilling to read the information.

The website we will create will have a modern design, which will be determined by the client's review of our designs. The new website will also contain all the information from the previous website, and be easily updatable, without having to edit the HTML code (since the previous website requires editing HTML code).

2.3 Motivation

Our motivation for this project is quite simple; we wish to make the website up-to-date, easily navigable, and developer friendly (i.e., updates / maintenance are easy to carry out). This will make more users want to use, stay on, and learn more about our website.

Due to our team being composed of Computer Science students, we understand the importance of more people entering our field, and wish to see people of younger age become more engaged in the field. We think that coding contests are a great way to achieve these goals. If we are able to create a website that will attract and engage more people, hopefully more people will participate in the contest. This will get more high school students involved in the Computer Science field, which everyone will benefit from.

2.4 General Approach

Our general approach for this project will take roughly four steps. These steps will include:

1. Decide on software and tools to use
2. Design and implement the new website
3. Fully test the website
4. Attempt stretch goals

First, before writing any new code, we need to decide on which software and tools to use when creating the website. This will include the development

frameworks, design packages, and website hosting. Second, we will design and implement the website. We will use online design tools to create layout ideas for our pages, and then code the pages based off of our designs. We will attempt to follow several principles of rich web design [1]. Third, we will test our newly created website, ensuring that it is completely usable and meets the requirements. Last, time permitting, we can attempt stretch goals, which are specified in the next section.

One way we can visualize our approach is through its methodology, which involves services that will be offered to specific users by our project results. The in-depth overview of the methodology can be found in Appendix 12.1.

3 Requirements

Our client Dr. Back has tasked us with two main requirements to be completed. The first is a redesign of the current VTHSPC website. He wants it to be more of a modern design but not overly futuristic or “flashy”. The other requirement is the preservation and transformation of the already existing content on the website. This includes facilitating the addition of new content for each new competition. With these two main goals completed, Dr. Back would consider this project a success.

The client also specified additional stretch goals for this project. One is improving the ease of updating. This means that any minor changes or updates to the website should take less than thirty seconds. The second stretch goal is having the website be search engine optimized. Search Engine Optimization, or SEO for short, is the process of making your website better for search engines. Improving SEO would improve the website’s visibility on search engines and thus increase web traffic. The last stretch goal that we completed was that the website would require only a static asset webserver.

Dr. Back also included a few more stretch goals that we were unable to complete. These goals concern the dynamic portion of the website. He would like the functionality to include maintaining user profiles and registering for yearly contests. We go more into detail about these in Section 9.4 (Future Work).

4 Design

4.1 Used Software

While creating the website, we will need to use specific software to accomplish our set requirements. We decided on two different packages: Docusaurus [3] and Node.js [4].

Firstly, Docusaurus is a static-site generator that can be used to create any kind of website (personal, product, blog, marketing, etc.). While taking advantage of the power of React, an open-source JavaScript framework and library, it is able to build a single-page application with fast client-side navigation. With its various documentation features, we intend to create a well-documented website that creates a pleasant user experience.

Secondly, Node.js will be used to assist with our hosting process. It is an event-driven JavaScript runtime, and it is designed to build scalable network connections. With this and NPM (command-line tools to install packages and dependencies), we can take the first steps to hosting our website. Once the website is built, it will then be deployed and hosted on the VT server, where it will be accessible to the public.

4.2 Front-End Design

In order to get an idea of how we wanted to create the layout and user interface of the new website, we decided to use an online wireframe editor to craft a few ideas of pages. The specific editor we used was Balsamiq [2]. Figures 3-5 show the final planned designs for the pages that we would eventually code.

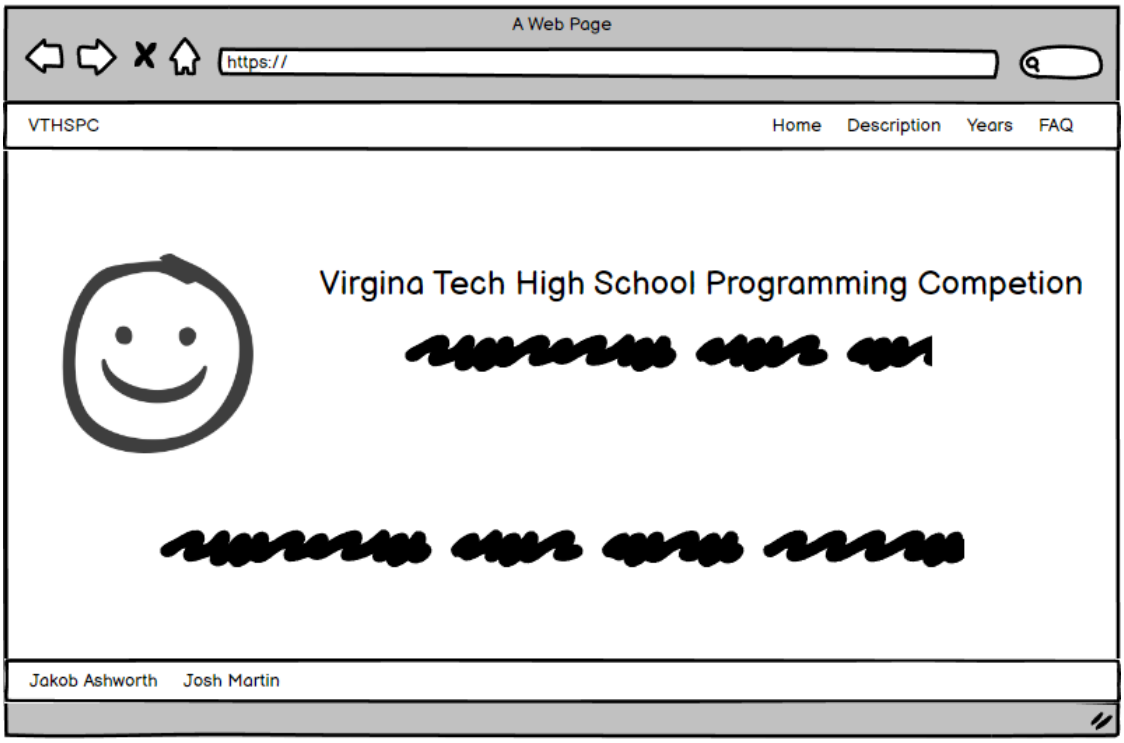


Figure 3: Planned home page

As shown in Figure 3, we designed the home page to have a simple design that will draw the user in, but not overburden them with too much information. We plan to include a Virginia Tech theme, the title of the website, and minor information about what the contest is about. By keeping a simple design, it will appear more modern.

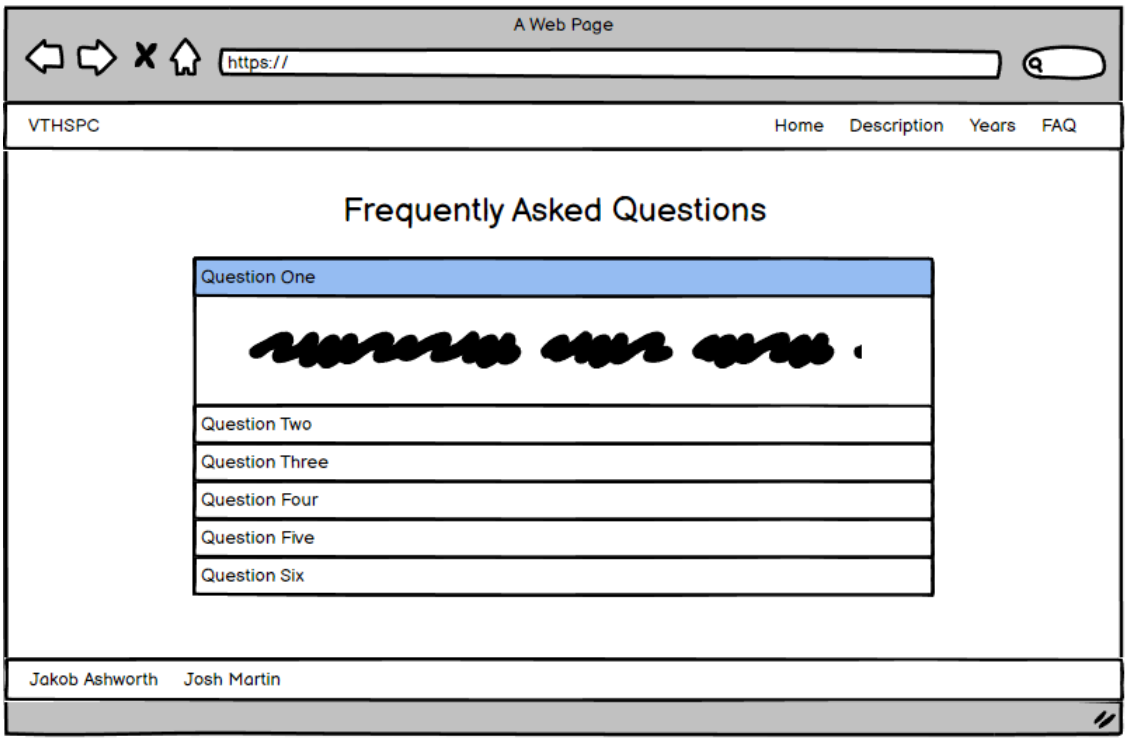


Figure 4: Planned FAQ page

Each year's contest page on the old website contained the same listed FAQ, so we decided to create a separate page on the new website (see Figure 4) that will hold that information. It will contain a dropdown type list with each question and answer.

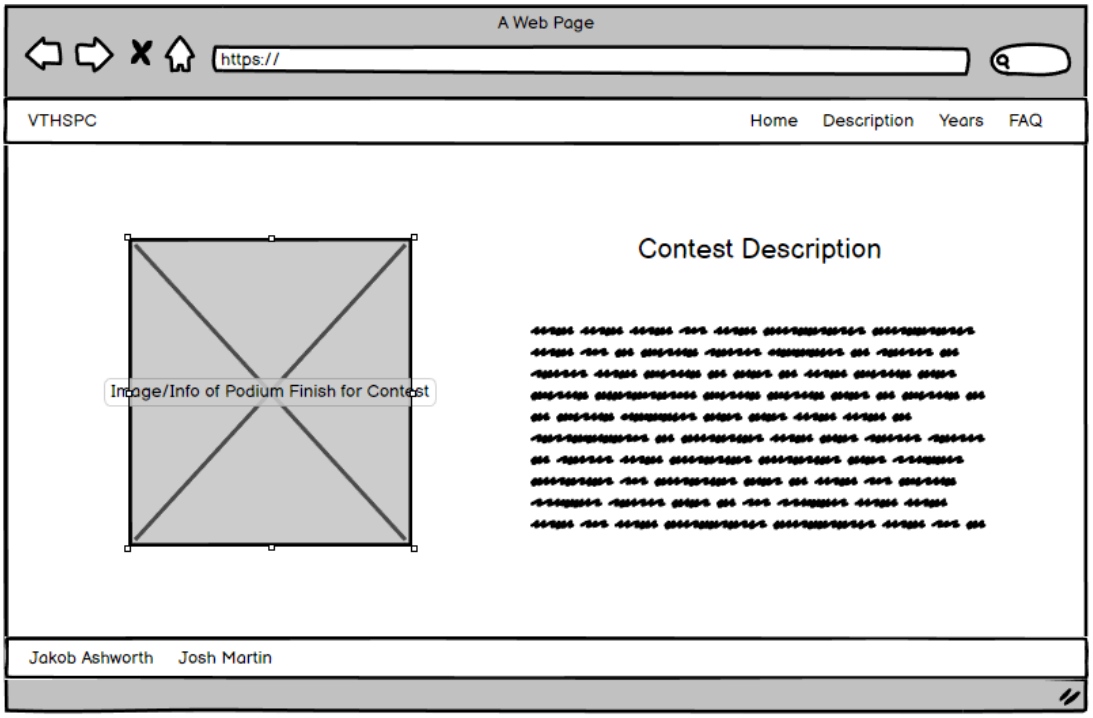


Figure 5: Planned Yearly Contest Page

The yearly contest page (see Figure 5) will contain most of the core information describing the contest for a given year. We plan to transfer most of the data from the old website to the new one, keeping the crucial information. We will also try to not clutter the information, instead using modern design techniques to convey it. There will be a section on this page about the winners from that year's contest.

5 Implementation

After going through the process of designing the entire project (i.e., selecting software to be used and web page designs), we can begin implementing the actual website. This will consist of three steps:

1. Downloading the Correct Software
2. Implementing/Coding the Web Pages
3. Hosting the Website

To begin the implementation process, we must download the necessary software that we will be using to structure the website. This mainly consists of Docusaurus, which requires the correct version of Node.js to be installed on the machine [3]. To install Node.js (and NPM), it takes a few Linux terminal commands [4]:

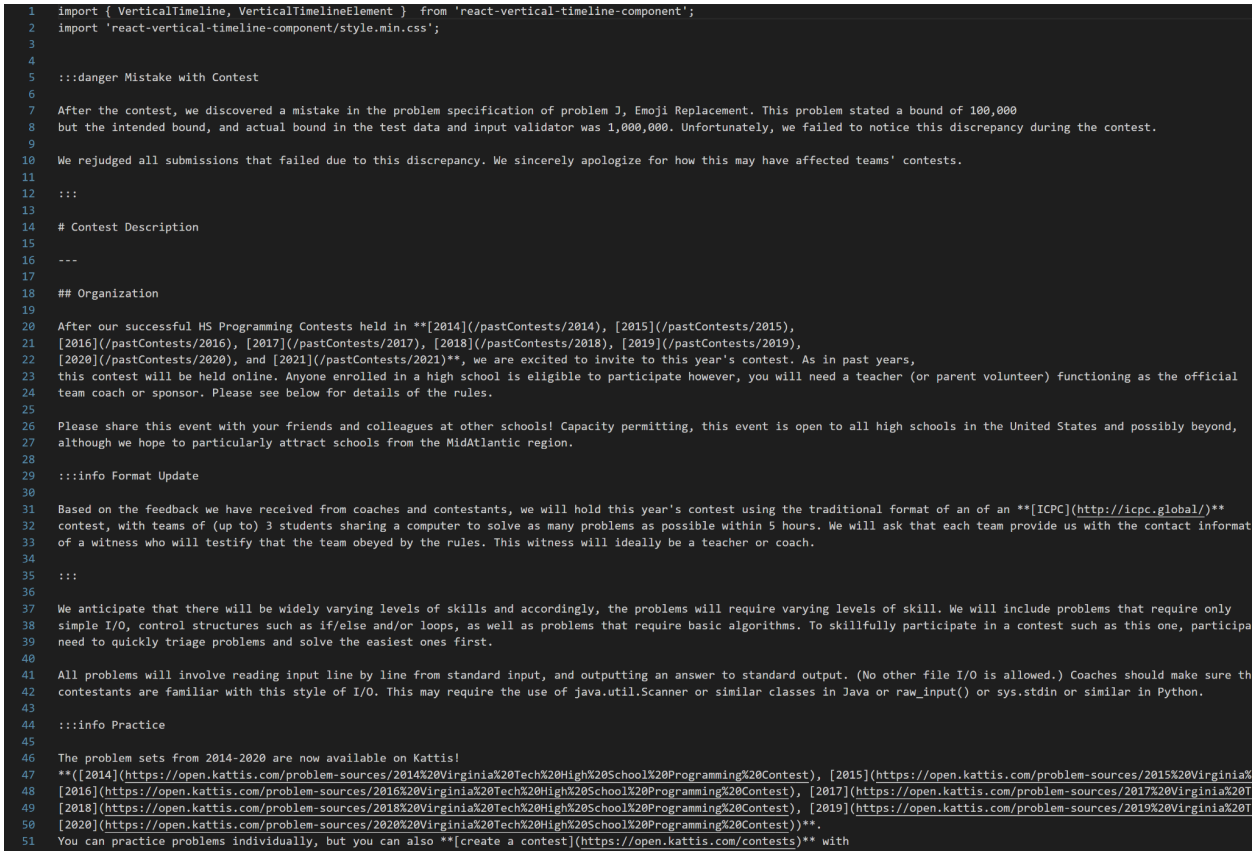
```
sudo apt install curl
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo npm install -g n
```

After installing these, we downloaded the wanted Docusaurus package in a directory of our choosing:

```
npx create-docusaurus@latest my-website classic
```

This command created a new directory to house the files for the site, and these files were scaffolded to ease implementation for developers. We chose the ‘classic’ template as it contains all the necessary features we needed. From here, we were able to make the necessary edits, additions, and removals to the available files.

Our primary focus was the creation of each individual web page, which required us to make a Markdown (MDX) file for each [5]. This was a fairly simple process, as writing in Markdown is straightforward, and we had prior experience with it.



```
import { VerticalTimeline, VerticalTimelineElement }  from 'react-vertical-timeline-component';
import 'react-vertical-timeline-component/style.min.css';


:::danger Mistake with Contest

After the contest, we discovered a mistake in the problem specification of problem J, Emoji Replacement. This problem stated a bound of 100,000
but the intended bound, and actual bound in the test data and input validator was 1,000,000. Unfortunately, we failed to notice this discrepancy during the contest.

We rejudged all submissions that failed due to this discrepancy. We sincerely apologize for how this may have affected teams' contests.

:::

# Contest Description

---

## Organization

After our successful HS Programming Contests held in **[2014](/pastContests/2014), [2015](/pastContests/2015),
[2016](/pastContests/2016), [2017](/pastContests/2017), [2018](/pastContests/2018), [2019](/pastContests/2019),
[2020](/pastContests/2020), and [2021](/pastContests/2021)**, we are excited to invite to this year's contest. As in past years,
this contest will be held online. Anyone enrolled in a high school is eligible to participate however, you will need a teacher (or parent volunteer) functioning as the official
team coach or sponsor. Please see below for details of the rules.

Please share this event with your friends and colleagues at other schools! Capacity permitting, this event is open to all high schools in the United States and possibly beyond,
although we hope to particularly attract schools from the MidAtlantic region.

:::info Format Update

Based on the feedback we have received from coaches and contestants, we will hold this year's contest using the traditional format of an of an **[ICPC](http://icpc.global/)**
contest, with teams of (up to) 3 students sharing a computer to solve as many problems as possible within 5 hours. We will ask that each team provide us with the contact informat
of a witness who will testify that the team obeyed by the rules. This witness will ideally be a teacher or coach.

:::

We anticipate that there will be widely varying levels of skills and accordingly, the problems will require varying levels of skill. We will include problems that require only
simple I/O, control structures such as if/else and/or loops, as well as problems that require basic algorithms. To skillfully participate in a contest such as this one, participa
need to quickly triage problems and solve the easiest ones first.

All problems will involve reading input line by line from standard input, and outputting an answer to standard output. (No other file I/O is allowed.) Coaches should make sure th
contestants are familiar with this style of I/O. This may require the use of java.util.Scanner or similar classes in Java or raw_input() or sys.stdin or similar in Python.

:::info Practice

The problem sets from 2014-2020 are now available on Kattis!
**([2014](https://open.kattis.com/problem-sources/2014%20Virginia%20Tech%20High%20School%20Programming%20Contest), [2015](https://open.kattis.com/problem-sources/2015%20Virginia%
[2016](https://open.kattis.com/problem-sources/2016%20Virginia%20Tech%20High%20School%20Programming%20Contest), [2017](https://open.kattis.com/problem-sources/2017%20Virginia%20T
[2018](https://open.kattis.com/problem-sources/2018%20Virginia%20Tech%20High%20School%20Programming%20Contest), [2019](https://open.kattis.com/problem-sources/2019%20Virginia%20T
[2020](https://open.kattis.com/problem-sources/2020%20Virginia%20Tech%20High%20School%20Programming%20Contest))**.
You can practice problems individually, but you can also **[create a contest](https://open.kattis.com/contests)** with
```

Figure 6: Content Description MDX

In Figure 6 you can see an example of an MDX file, written in Markdown. The syntax is fairly simple, with certain syntax corresponding to syntax in HTML. For example, a single ‘#’ in Markdown is equivalent to an ‘h1’ in HTML, and a simple paragraph is equivalent to ‘p’. Since it is an MDX file, we are able to incorporate React components into the page. On the content description page, we import and use a vertical timeline element, which is a React component.

During this development process, we could preview our changes by running the website on a local development server. We would do this by using a terminal, changing to the correct directory, and running the command ‘npm run start’. This will create a browser window at http://localhost:3000. This allowed us to correctly implement the web pages by seeing live updates.

Once we completed writing and editing the necessary files, we moved on to the hosting process. This began with having a meeting with our client, where he created a server on the official Virginia Tech network for us to use and host our website on. When we had access to the server, we moved the content from our local machines to the server’s machine.

Since Docusaurus is a modern static website generator, we had to build the website to create the static contents that will be viewed. This includes creating (translating) the necessary HTML files from our MDX files. Running the ‘npm run build’ generates all the static content and houses it in a ‘/build’ directory. We moved this directory to the correct location so it was viewable on the web server.

Once this was all complete, our website was usable at https://vthspc.cs.vt.edu/.

6 Testing

Every change that we made to our website was tested to make sure that there were no bugs or errors in the code. Most of our testing comes from the front end design. Since a new modern look and feel of the website is a main priority, we put most of the effort into testing that. This included creating multiple designs of the website, receiving feedback from our classmates, and discussing with our client what they think.

6.1 User Feedback

One of our best ways of testing was through classmate feedback. We were able to send out a Google form to our classmates to receive feedback on our website design. We asked multiple questions on how they rate the design, how easy it is to access certain information, how easy it is to navigate through the website, and asked for an overall judgment on the website. The feedback we received rated our new design and use of the website highly, especially compared to the old design. Figure 7 shows some of the (rate out of 10) questions we used. We asked them to rate the design and usability of the new website design and also what they think should be improved upon. Figure 8 shows test case questions we asked. So we asked them to find certain information on the website and rate how easy it was to find it.

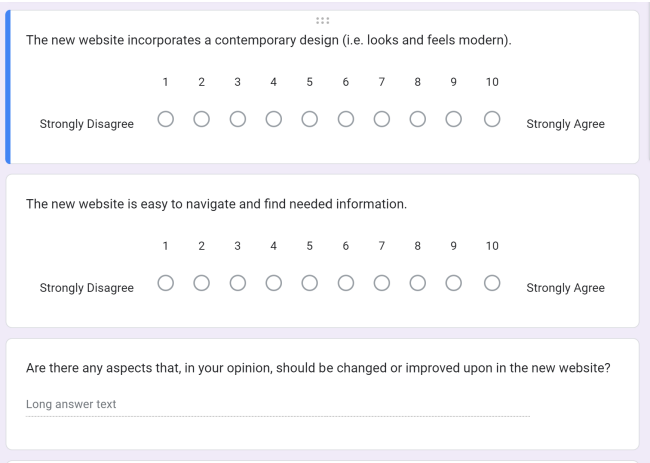


Figure 7: Rate out of 10 questions

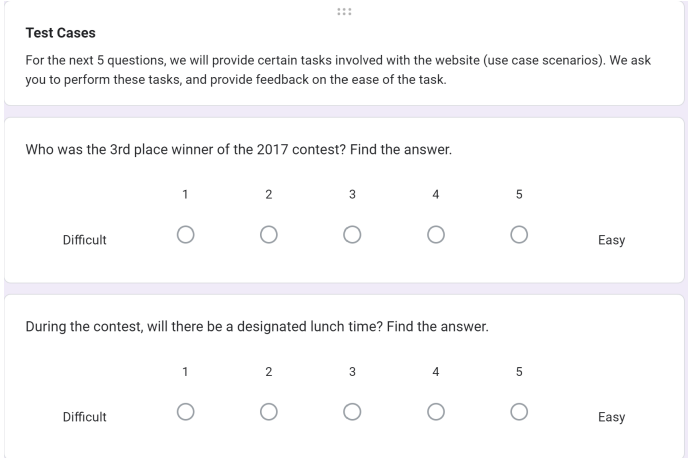


Figure 8: Test cases questions

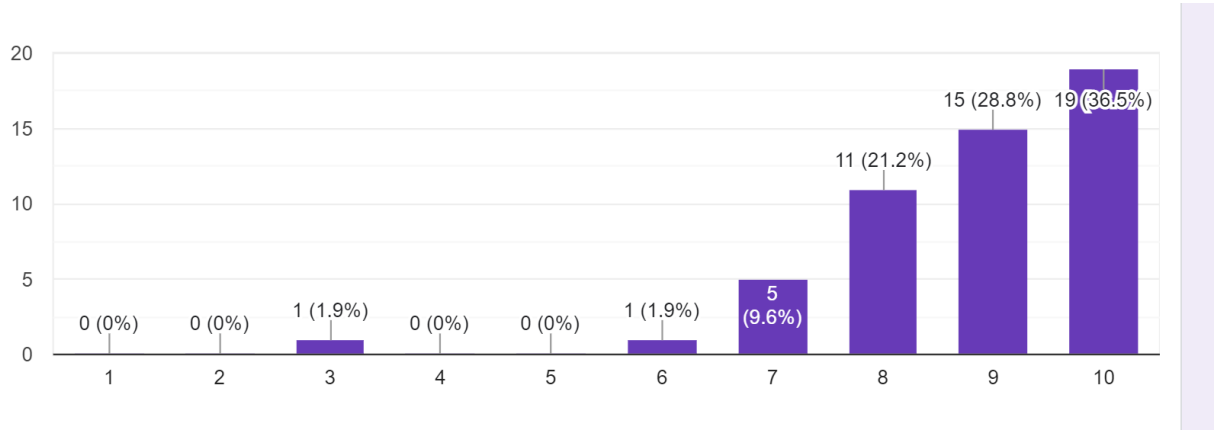


Figure 9: Response from rate out of 10 questions

As shown in Figure 9, our feedback from the rate out of 10 questions about the design and navigation of the new website design is pretty good. The average was an 8.8 out of 10, based on 52 responses.

6.2 Other Testing

Besides testing regarding the main requirements we also had to test the stretch goals that we ended up completing. The first goal we were able to

complete was the ease of update. This meant that the time to make a change and update it on the website was under thirty seconds.

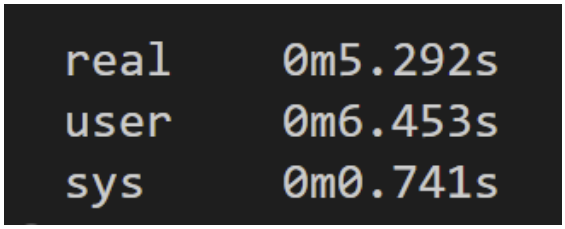


Figure 10: Website update time

Figure 10 shows the update time being about 5 seconds, which is significantly faster than the previous update time of 3 minutes our client was having. This change is mostly due to our use of Docusaurus and it being more streamlined.

The next stretch goal that we had to test was our Search Engine Optimization (SEO). The way we tested this is by using a tool called PageSpeed Insights by Google [5]. It is a great tool that rates a website's performance, accessibility, best practices, and SEO. This made it easier on us to actually see numerical improvements and what else we needed to work on. It gave our website very impressive results in each category with a 100% in performance, 98% in accessibility, 92% in best practices, and most importantly a 100% in SEO.

7 Users' Manual

Users will first have to visit our website at https://vthspc.cs.vt.edu/ . Once they reach the website, they would be greeted with the home page shown in Figure 11.

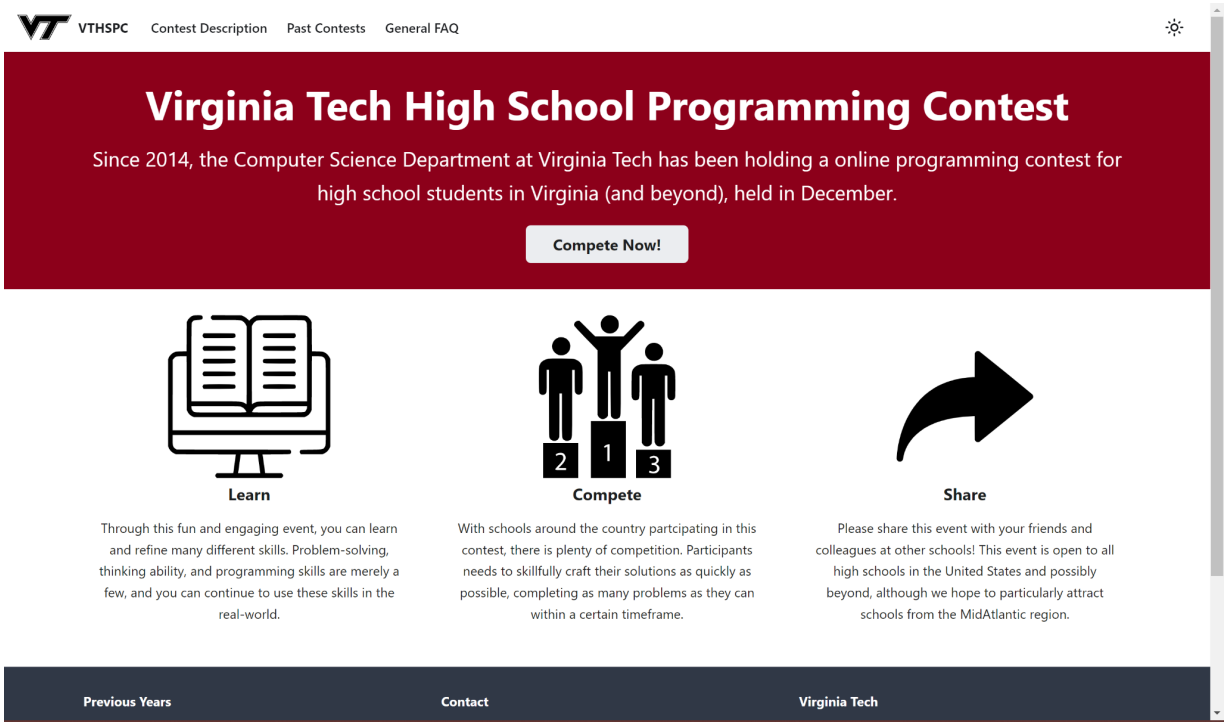


Figure 11: Website Home page

The home page acts as a way for users to understand what the Virginia Tech High School Programming Contest is all about. The user will see how long this contest has been going on, when the contest happens, how to compete, and many other details. The top navigation bar includes pages about the current contest description, past contest descriptions, and the general FAQ list.

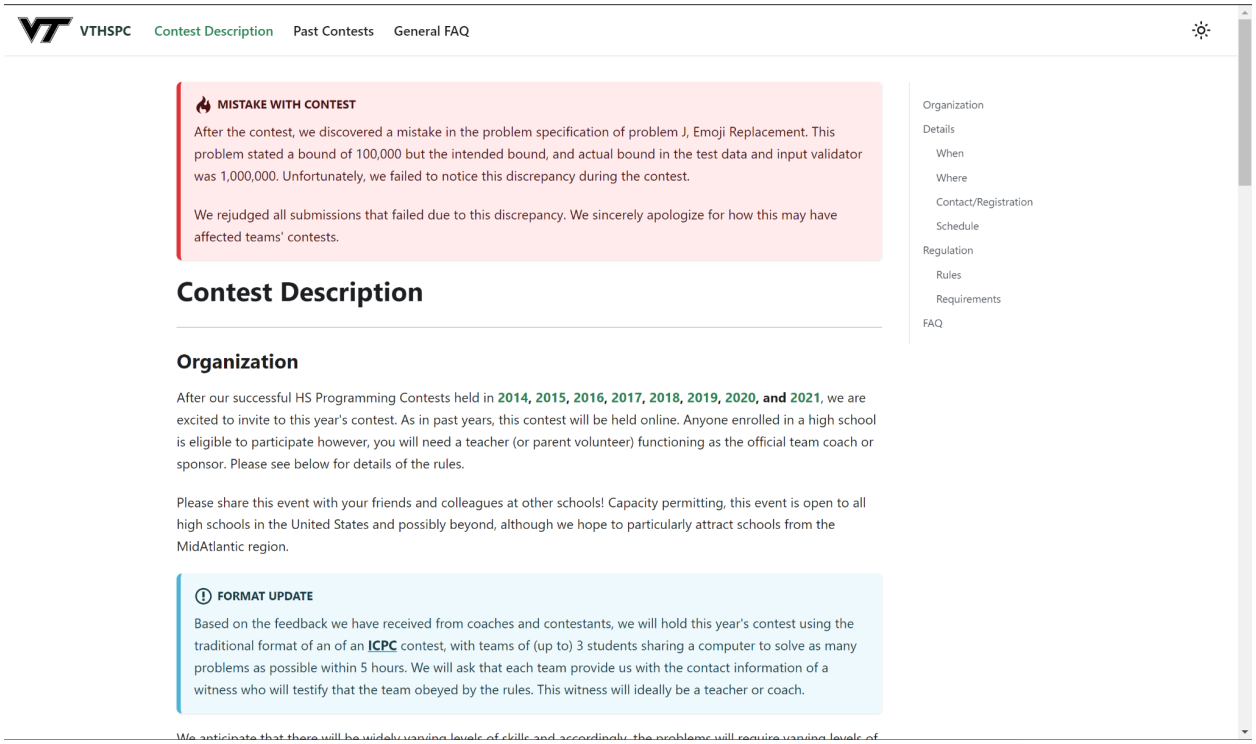


Figure 12: Current Contest Description

Clicking on the Contest Description link on the navigation bar brings you to the page shown in Figure 12. This page has all of the information needed about the current contest. On the right side is the table of contents for the page. There is the Organization section which has the organization details of the contest. Then there is the Details section shown in Figure 13. This gives information about when and where the contest is, how to register, and the schedule of the contest. Next is the Regulations section shown in Figure 14. This section lists the rules and requirements for the contest. Lastly is the FAQ section (see bottom of Figure 14) that lists the most frequently asked questions and links them to the FAQ page on the navigation bar.

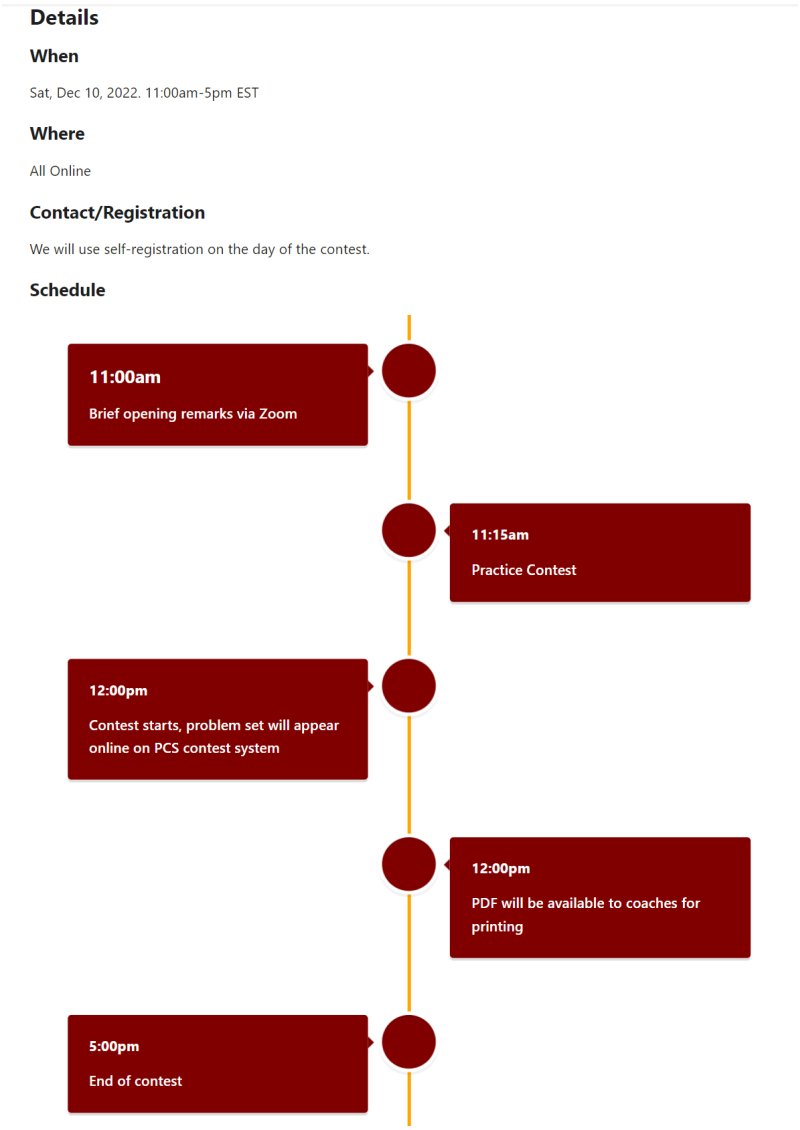
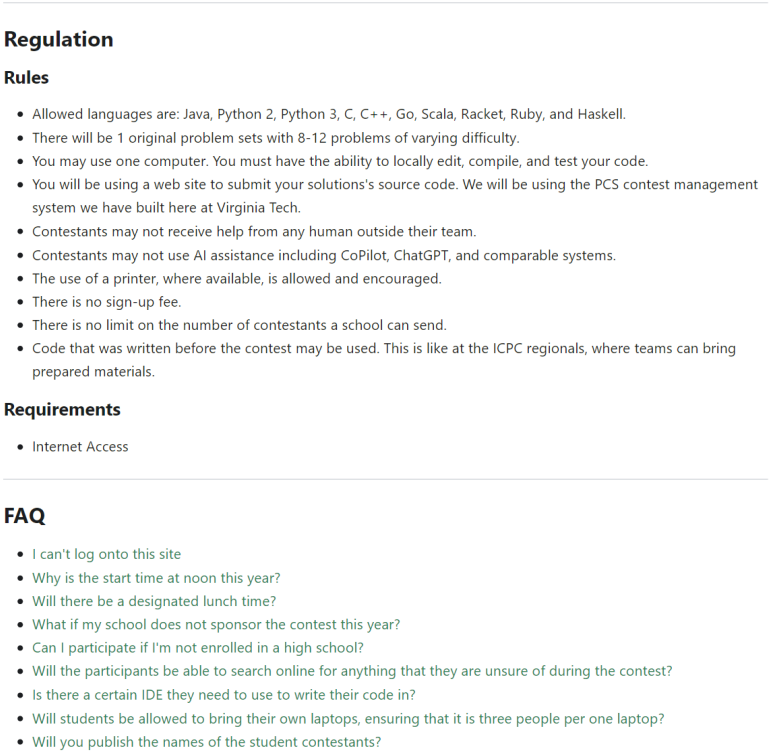


Figure 13: Details Section



Regulation

Rules

- Allowed languages are: Java, Python 2, Python 3, C, C++, Go, Scala, Racket, Ruby, and Haskell.
- There will be 1 original problem sets with 8-12 problems of varying difficulty.
- You may use one computer. You must have the ability to locally edit, compile, and test your code.
- You will be using a web site to submit your solutions's source code. We will be using the PCS contest management system we have built here at Virginia Tech.
- Contestants may not receive help from any human outside their team.
- Contestants may not use AI assistance including CoPilot, ChatGPT, and comparable systems.
- The use of a printer, where available, is allowed and encouraged.
- There is no sign-up fee.
- There is no limit on the number of contestants a school can send.
- Code that was written before the contest may be used. This is like at the ICPC regionals, where teams can bring prepared materials.

Requirements

- Internet Access

FAQ

- I can't log onto this site
- Why is the start time at noon this year?
- Will there be a designated lunch time?
- What if my school does not sponsor the contest this year?
- Can I participate if I'm not enrolled in a high school?
- Will the participants be able to search online for anything that they are unsure of during the contest?
- Is there a certain IDE they need to use to write their code in?
- Will students be allowed to bring their own laptops, ensuring that it is three people per one laptop?
- Will you publish the names of the student contestants?

Figure 14: Regulation Section

Clicking on the Past Contests link on the navigation bar brings you to a page that gives a quick overview of the VTHSPC and lists the past contests on the left hand side. This is shown in Figure 15. Every Past Contest page has the same layout as the current contest description shown in Figure 12. This makes it easy to navigate through contest years.

Figure 15: Past Contests Page

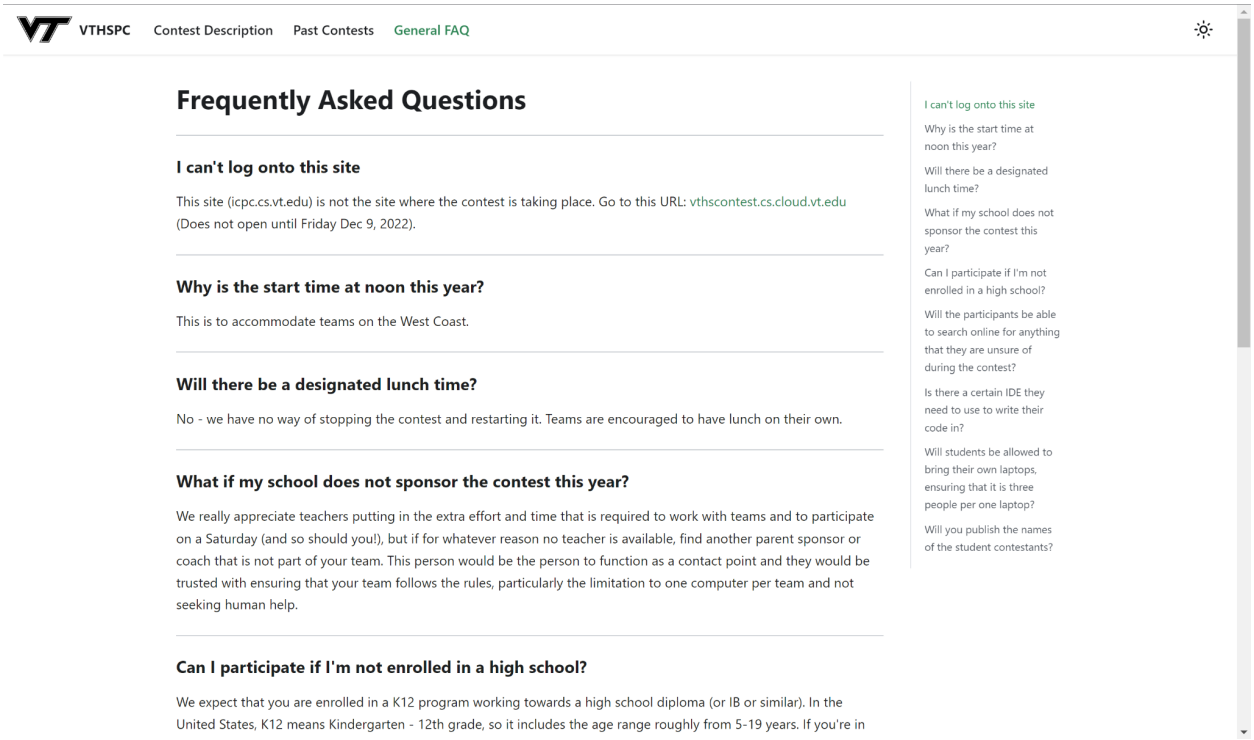


Figure 16: General FAQ

Figure 16 shows the General FAQ section of the website. This is shown when the user clicks on the General FAQ link in the navigation bar. This page lists out the most frequently asked questions and provides the answers for them. Each question is also listed on the right side table of contents for easy

navigation. Also each question is able to be deep linked. This means that it is possible to provide a link to navigate to a specific question on the page.

8 Developers' Manual

8.1 Front-End

The front-end of the website was written using Markdown (MDX), which is then enhanced by React and JavaScript [6]. The files are organized in a simple directory structure (see Figure 17), with separate folders that contain files of the same group and use.

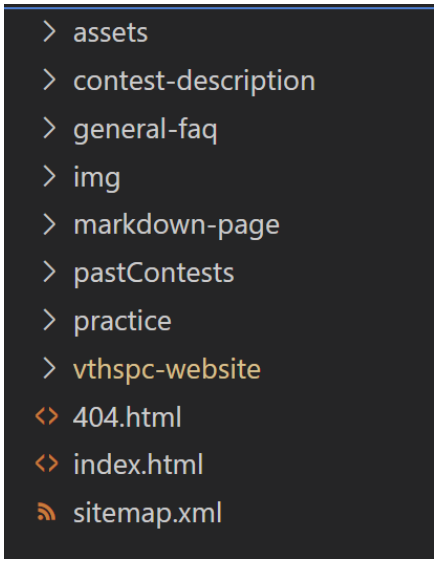


Figure 17: Base directory structure

In the base directory of the website, we can see there are several folders, some miscellaneous files, and an index.html file. The index.html file contains the home page data, while the folders contain other HTML pages, assets, pictures, and so on. These were created through the build process that was run in the vthspc-website subdirectory. This subdirectory contains everything that is used and edited by the developer.

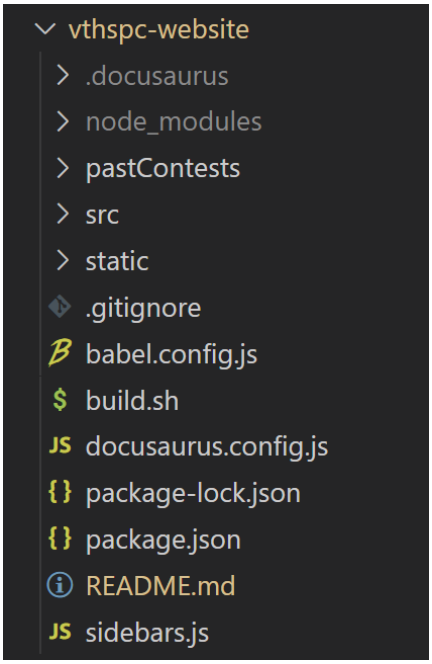


Figure 18: Development folder

The development folder (see Figure 18) contains all of the elements that are involved with the development of the website. From top to bottom of this subdirectory, the .docusaurus and node_modules have no relevance to the developer, and do not need to be edited at any time. These are involved with the build process that Docusaurus specifically implements.

Figure 19: Past Contests folder

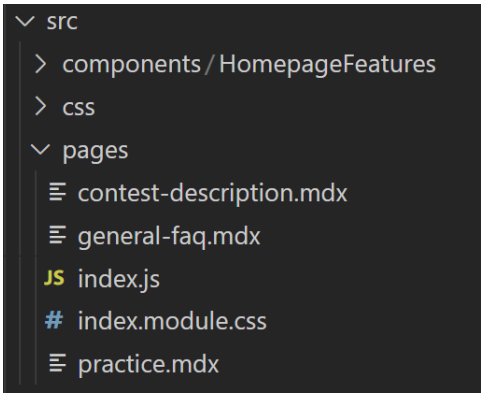


Figure 20: src folder

From there, the past contests and src folder contain most of the Markdown (MDX) files that represent the pages of the site. For past contests (Figure 19), there is a specific year folder for their related MDX file and related data, with the infoPage.md being the base page for all the past contests.

The src folder (Figure 20) houses most of the data for the separate pages. The components and CSS folder hold most of the CSS information for the site, with components being related to just the home page while CSS relates to the entire website. The pages folder contains the data related to the separate pages, that being the Contest Description, General FAQ, and Practice pages.

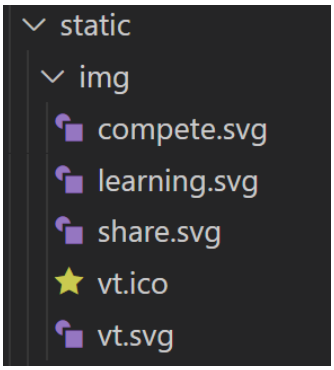


Figure 21: Static folder

As stated by its name, the static folder (Figure 21) contains all of the objects that will remain static on the website server. Specifically, the only objects we are housing are the images used throughout the website (i.e., logo and pictures).

The rest of the files in the development folder (Figure 18) are the JavaScript, JSON, and bash files that are used in the build process. Specifically, the routine docusaurus.config.js creates the data on the navigation bar and footer, while package.json and package-lock.json explicitly identify packages being used on the website. The build.sh file is a bash script created by us to

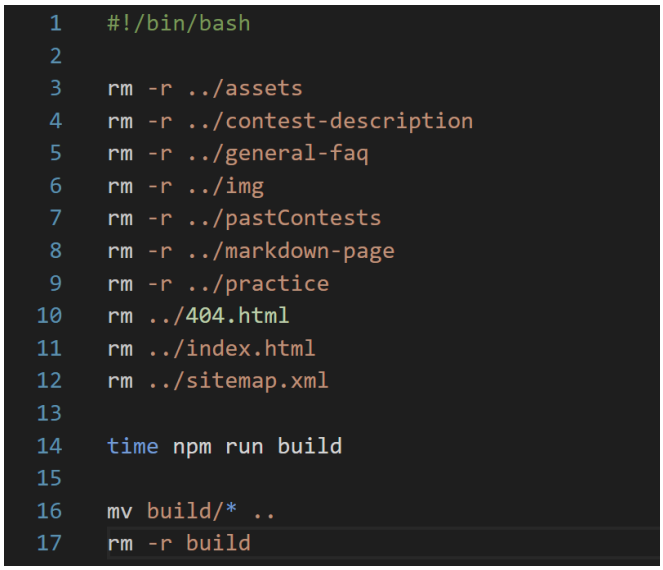
make the build process easier for the client, as will be explained in the next section.

With this simple structure, the client and future developers can easily navigate to which files need to be updated and changed. They can also create new files if needed, and store them in the correct locations.

8.2 Hosting

The process to host the website is a fairly simple quick process. Since we are employing Node.js and NPM, it only takes a few command line calls to get the updated content onto the website. The website is hosted on the VT Server.

In order to make the process even simpler, we created a bash script named build.sh, that is held in the development folder (Figure 22). When it is run, it builds the website's contents, and next moves the built content to the correct directory, where the new content will then be available on the website.



```
#!/bin/bash

rm -r ../assets
rm -r ../contest-description
rm -r ../general-faq
rm -r ../img
rm -r ../pastContests
rm -r ../markdown-page
rm -r ../practice
rm ../404.html
rm ../index.html
rm ../sitemap.xml

time npm run build

mv build/* ..
rm -r build
```

Figure 22: build.sh

Specifically, the first several lines of build.sh (Figure 22) removes all the previous content from the web server directory to make room for the edited,

new content. Then, it runs the ‘npm start build’ to build the new content, which creates a build directory in our development folder. Lastly, the script moves this newly created build directory to the web server directory, thus putting the edited content on the web server. The website is then available with up-to-date content.

If a developer wanted to edit content on a webpage or add a new page, follow these steps:

1. Find and edit the specific MDX file or create new MDX file
2. Save file and make sure connections for page are correct in JS files
3. Run build.sh

9 Lessons Learned

9.1 Timeline

In order to stay on task and complete the project on time, the schedule shown in Table 1 was prepared during the project launch. Work was completed on schedule.

Date	Completed Tasks
2/14	● Presentation 1 completed ● Project plan completed ● Base for website created
2/28	● Website design plan completed ● Began work on individual website pages ○ Home ○ Contest page ○ FAQ page ○ Year pages ● Considerable progress on the interim report
3/15	● Presentation 2 completed ● Website pages demonstrate considerable progress ● Start hosting of website ● Trying to implement SEO
3/31	● Front-end pages of website are completed ● Basic functionality is done ● Testing in process
4/15	● Work towards stretch goals (e.g., user accounts, …) ● Final Presentation progress ● Final Report considerable progress
4/28	● Project Completion ● Fully functional website is accessible and running

Table 1: Timeline

9.2 Problems

The only major problem we faced was the first approach we took implementing the website. We took the approach of creating the HTML, CSS, and JavaScript completely ourselves, implementing aspects of Bootstrap 5, a popular framework for creating websites [7]. Throughout this process, we created nearly a complete website, which we now call Version 1.

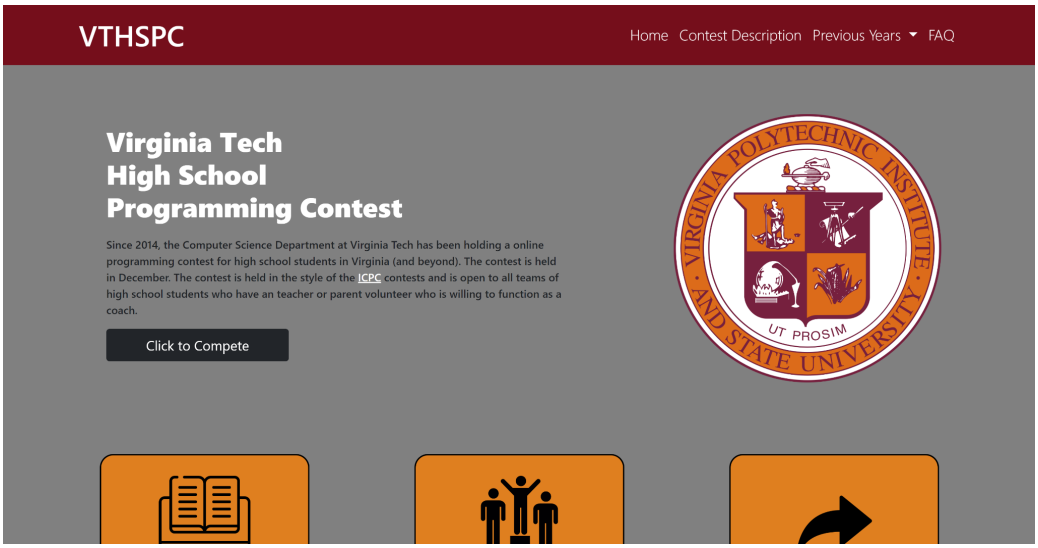


Figure 23: Version 1 Home page

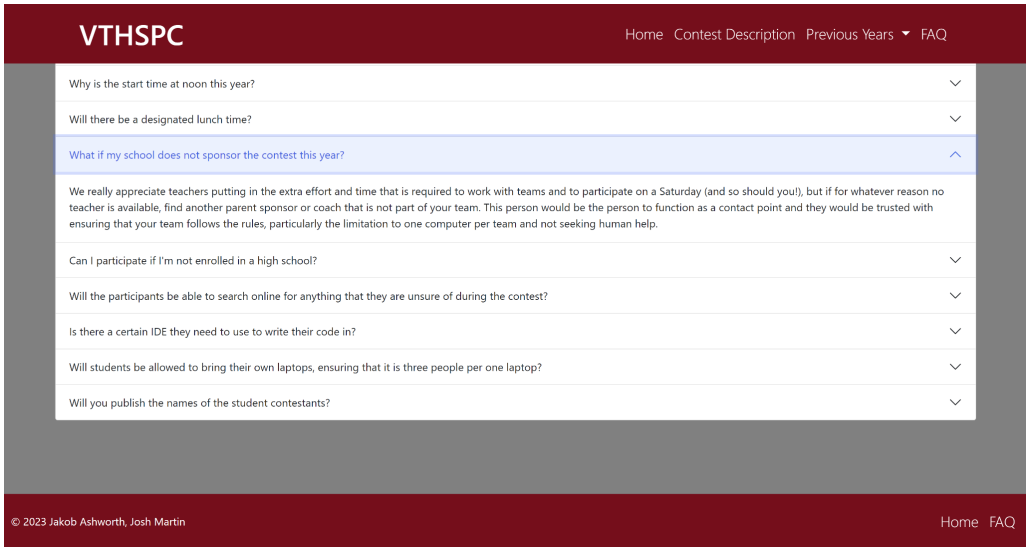


Figure 24: Version 1 FAQ page

Figures 23 and 24 give some example pages from Version 1 of the website. To the team, the website seemed fairly generic and was not meeting the requirements we set forth. Additionally, there was no easy way to incorporate SEO as the pages were written strictly in HTML, and editing the website for minor changes would have been a hassle. It would have involved editing the HTML code directly, which is not an improvement on the previous website.

9.3 Solution

The solution to our major problem was simple, but it required some work. After a meeting with our client, we brainstormed several new approaches for creating the website. One idea, which we eventually went forward with, was Docusaurus. This allowed us to start creating Version 2 of the website from the ground up.

Using Docusaurus to implement the website allows for automatic SEO, and editing the site is much simpler with just having to edit the Markdown (MDX) files. Also, using Docusaurus gave Version 2 a much more sleek and appealing design.

9.4 Future Work

Future work can be found in the stretch goals outlined by the client, as we were not able to complete work on them within this semester. Future work for the website includes:

- User profiles
- Dynamic content - would require API; functionality would involve maintaining user profiles and registration for the yearly contest
- Statistics of Contest - keeping track of past contests via statistics (i.e., winners, problems solved, etc.)

10 Acknowledgements

Godmar Back, Ph.D., Computer Science - Associate Professor at Virginia Tech
Email: gback@vt.edu

Edward A. Fox, Ph.D., Computer Science - Professor at Virginia Tech
Email: fox@vt.edu

11 References

[1] G. Rauch, “7 principles of Rich Web Applications,” *Guillermo Rauch's Blog*, 04-Nov-2014. [Online]. Available: https://rauchg.com/2014/7-principles-of-rich-web-applications. [Accessed: 28-Feb-2023].

[2] Balsamiq, “UI Design Wireframes,” *Balsamiq*, 2023. [Online]. Available: https://balsamiq.com/. [Accessed: 28-Feb-2023].

[3] Docusaurus Team, “Build optimized websites quickly, focus on your content: Docusaurus,” *Docusaurus RSS*, 2023. [Online]. Available: https://docusaurus.io/. [Accessed: 11-Apr-2023].

[4] Node.js Contributors, *Node.js*, 2023. [Online] Available: https://nodejs.org/en (accessed May 2, 2023).

[5] Google Team, “Make your web pages fast on all devices,” *PageSpeed Insights*, 2023. [Online]. Available: https://pagespeed.web.dev/ (accessed May 2, 2023).

[6] Matt Cone, “Basic syntax,” *Markdown Guide*. [Online]. Available: https://www.markdownguide.org/basic-syntax/, 2023. [Accessed: 11-Apr-2023].

[7] Bootstrap, “Bootstrap 5 (v5.2.3),” 22-Nov-2022. [Online]. Available: https://github.com/twbs/bootstrap. [Accessed: 28-Feb-2023].

12 Appendices

Appendix A. Methodology

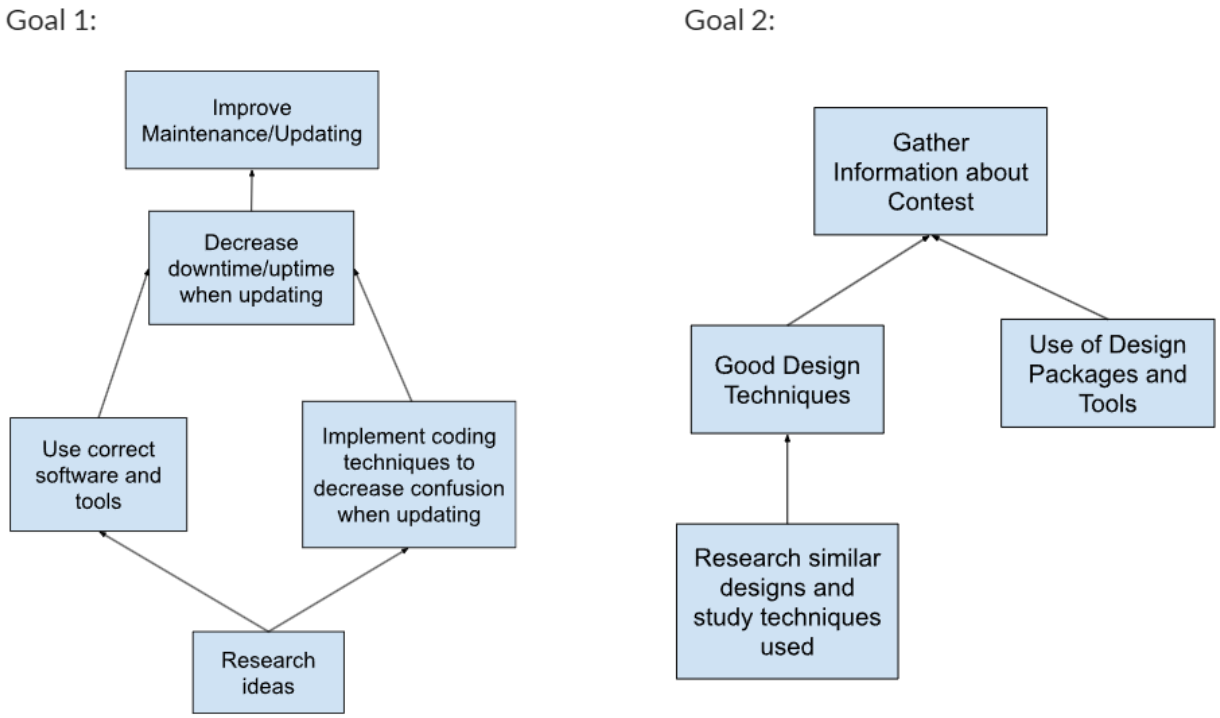


Figure 25: Tasks and subtasks

The flowcharts shown in Figure 25 specify two main tasks for the personas of this project (client and users), broken down into subtasks that will need to be taken care of to complete the task.

Service ID	Service Name	Input file name(s)	Input file Ids (comma-sep)	Output file name	Output fileID	Libraries; Functions; Environments
1A	Content Edits	MDX files of web pages	1	MDX files of web pages	1	Docusaurus, Server
1B	Website Build Process	build.sh	1	Files built from build call	1	Bash, NPM, Linux
1C	Host Environ-ment	N/A	N/A	N/A	N/A	Virginia Tech Official Server
2A	Contest Desrc. Page	N/A	N/A	contents-description.mdx	1	MDX
2B	Past Contests Page	N/A	N/A	/pastContests/(year).mdx	1	MDX
2C	FAQ Page	N/A	N/A	N/A	N/A	MDX

Table 2: Breakdown of services

As we work throughout the project, there are several different services that we must take into consideration when completing the website. In Table 2, each service is specified, with the criteria for each listed.

Then, in order to maintain a proper structure for these tasks and services, we can create two separate workflows. This can be shown by:

- Persona - Client, Task 1 - Increase Ease of Maintenance/Update
    - Workflow 1 = Content Edits + Website Build Process + Host Environment
- Persona - Website Users, Task 2 - Gather Information about Contest
    - Workflow 2 = Contest Description Page + Past Contests Pages + FAQ Page