Andrew Bond
Nathan Kendrick
Steven McKim

PrepLab

Multimedia, Hypertext, and Information Access - CS 4264 - Fox
Virginia Tech, Blacksburg, Virginia 24061
Darren Maczka
May 8, 2019

Table of Contents

Table of Figures

Abstract

PrepLab is a proposed online repository to aid instructors in the remixing and reuse of instructional materials (e.g. syllabi, assignments, learning activities, facilitated discussions, etc.). To facilitate easy adoption it was designed to be as simple as possible for content creators to upload existing material. This project set the constraints of what content may and may not be easily included. When uploading content, users shall select what creative commons license they wish to apply (the system may impose a minimum license) and confirm that they are the original content creator and/or own the rights to any material they are submitting. Instructions to content creators for structuring new material is also included. Upon initial upload of an item the results of the parser/indexer shall be displayed to the user who is given the opportunity to confirm/edit the results.

This project is dedicated to working specifically on the Graphical User Interface (GUI) for the PrepLab. React.js, a front end javascript framework, was chosen as the design method upon which the front end of PrepLab is built. Other design elements added to the website include Bootstrap as well as traditional CSS libraries. Amazon Web Service (AWS) is used to host the website until a more suitable replacement can be determined when the other sections of this project are completed.

As the backend pertains to a different part of the project, the database holding the instructional materials that will populate the site will be covered in future projects related to PrepLab. The other section that is not covered in this part of PrepLab will include parsing and indexing of documents as well as including an API in order to increase functionality and search capabilities.

Introduction

Currently at Virginia Tech, educators must rely on Canvas or departmental websites to host any notes, discussions, or activities to which students or graduate teaching assistants (GTAs) would have access to. These environments are not the most ideal for simple sharing between educators who could potentially reuse or remix this content for their own benefit when teaching the same or similar courses. The use of Canvas, as well as other websites used in higher education, also generally limit GTAs to the same information as the students in the class rather than all of the information for the entire course or beyond.

The goal of PrepLab is to make scholarly information and material more readily available for use by professors as well as GTAs and even UTAs. This portion of the project is focused on the front-end development of this website; this means it is more about the usability and user experience than back-end functionality.

The simplistic designs are meant to be visually appealing, while also making the website as usable as possible. The main goal of this part of the PrepLab project specifically is for PrepLab to be intuitive and easy for professors and teaching assistants to navigate through and obtain the materials or information that they are looking for.

This is accomplished through the main functionality of the website, which is advanced and detailed searching of materials. Users will have the ability to utilize many different filters in order to find material tailored to their needs. Some of these filters include topic, related major, college, author, class size, and file type as well as others and possibly even more that can be added in future iterations.

Other functionality that is included in this project is the ability for users to upload their own, original material that they have the rights to and they wish to share with other users. These files may be edited later and also deleted. Users can also leave reviews and helpful comments on other material that has been posted.

All of this functionality paired with its styling is important for how quickly PrepLab gets adopted and used as a standard for searching for class material at Virginia Tech, and possibly other colleges in the future. How all of this was accomplished, how to add onto this project, how to use PrepLab as a tool, as well as problems encountered will be discussed in further detail.

Requirements

The PrepLab application has requirements such that the user can share their content as easy as possible. The major requirement for this portion of the app was that a front end version of the site could be accessed by both web and mobile web browsers. Because of this, the client had specified that the app be developed in React.js for simple portability between the two.

Many aspects of the site had to keep the future backend and API in mind so there were requirements set in order to ease the development of the other two parts of the PrepLab system. The first such requirement was that it needed to be able to parse JSON files from the format that was set by the client (this format can be found in the developer's manual). This implementation also had to be as simple as possible for the any consecutive team working on PrepLab to quickly and easily be able to attach the API. This was taken care of by keeping code for connecting to the API separate from most of the front end code and held in as few places as possible; having the code base be modular.

Various pieces of information were required to be shown on the site for each of the hosted files. Some of that information includes: title, author, date created/last modified, description of the resource, topic of the material uploaded, size of the course, related material, and format/type of file. A comment section for each of the files was also needed in order to allow different users to converse and review a file.

These resources needed to be easily accessible for the end user. The best way to accomplish this was to allow users to search for files via various parameters. Some search parameters may not be required for the average user on every query though, so having both a basic and advanced search needed to be implemented.

Users also had to be able to upload their files to the site via an easily accessible form. The form was required to be only accessible for signed in users, so other forms for signing in and registering were also required to exist. Each user was also required to have a profile to display all of the files that they have uploaded along with other information that they wish to display, including a profile picture and short biography.

Design

Inspiration for the design of the PrepLab website was derived mainly from three different sources: Github, a sample PrepLab site that was formatted as a blog, and the National Science Digital Library (NSDL). Github being a popular site as a code repository influenced our layout for the aesthetics on each page. Features like the advanced search and basic search format found on the NSDL website provided ideas for how we could present the most important information to the user without having to redirect them to a different page.

Due to the purpose of the website, being able to search the application for a specific type or set of notes must be simple for the end user. As a result, the search bar is a prominent feature of the PrepLab. This is the reason it is featured in the center of the home page so that everyone visiting the site will see it prominently display immediately upon navigating to the site.

Advanced search options to narrow the search results is a feature that allows users to make more specific searches than the basic search on the home page. These advanced features have been integrated into the front end of the website on the advanced search (Explore) page. Users can search for files with parameters for classes, teachers, sizes of classes for which the materials were used, and other information that would provide further constraints.

Spreading information so that it can be useful to other educators and students is another prominent focus of the website, so different areas were added so users may be exposed to files they may not have thought to search for originally. These were implemented in the design of the website through the recently uploaded files on the home page and with the recommended files on the file pages. The intent is that something will catch the eye of the user and they can integrate the ideas from similar or new material into their own.

There is also a need for a user to register and login to PrepLab so that they are able to upload material, edit or delete their material, comment on other users' resources, as well as have an editable profile.In order to accomplish this for this portion of the project's purposes, an easy way for users to sign up and log in was added. This is included in the top right corner of every page when a user is not currently logged in. When a user is logged in, the options to login and register are replaced with a logout option as well as an upload file option.

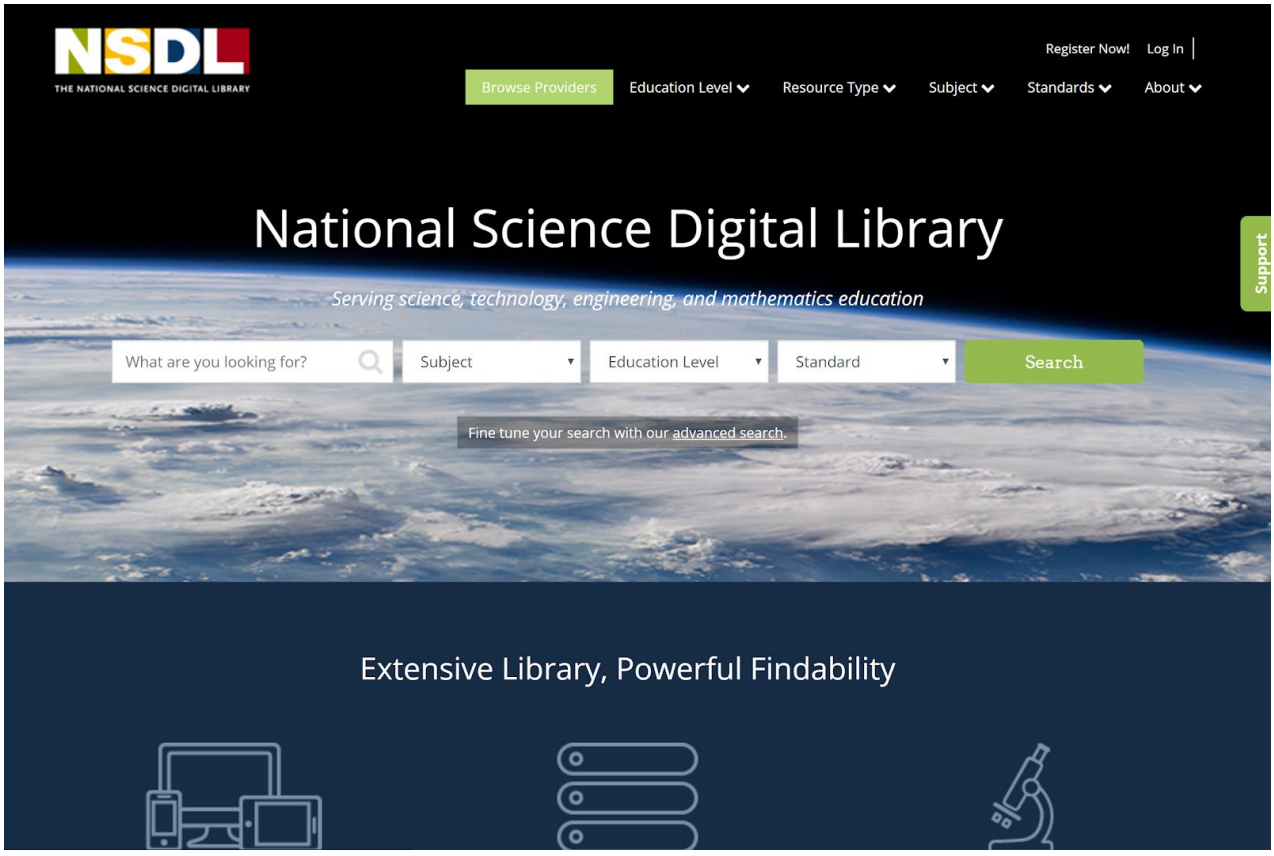

Figure 1. National Science Digital Library Home Page

This is the home page for the NSDL website that was used for inspiration as to how PrepLab should be styled and designed. This site has a very similar goal as PrepLab as the main functionality is searching through academic material by many different parameters. It also includes an advanced search option as PrepLab does. This site also has a way for users to register in order to gain more functionality included in the web application. NSDL is an overall aesthetically pleasing site with great user design that PrepLab attempts to mimic.

Figure 2. Sample PrepLab Blog Home Page

This is a sample page of what the client had developed as a rough estimate for what PrepLab should be that was created using WordPress. Due to it being hosted on Wordpress, it does not have most of the functionality implemented in this project and does not have good portability to mobile platforms. Also, being structured as a blog style due to being created in WordPress means it does not have the same versatility in styling that this project accomplishes. The color scheme as well as some of the images from this version of the site have been migrated to the new site.

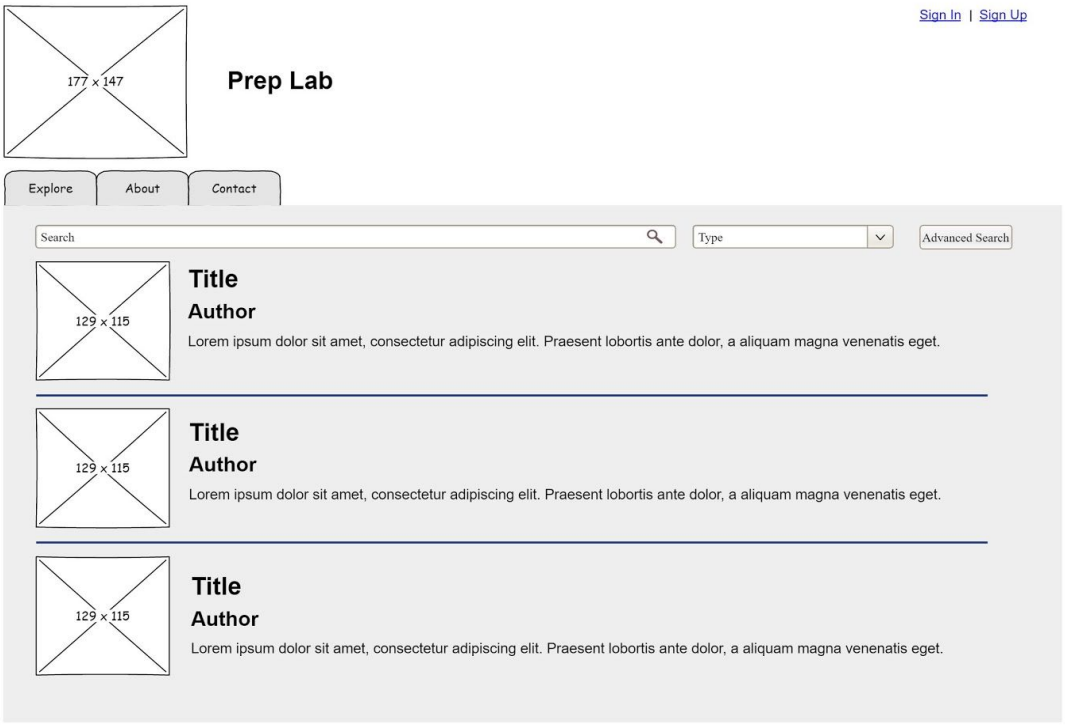

Figure 3. Homepage Wireframe

Figure 3 is the wireframe for the original design of the home page. In the top left of the screen the PrepLab logo along with the name of the site are present. This feature is carried throughout every page with the intent that it will redirect users back to the home page when clicked. Registration, logging in, and logging out of the site are presented in links on the top right of the home page. A block separates the search bar and recently posted resources from the header of the page to draw more focus on the main functionality of the site. The different tabs are listed below the logo in the header, and will stick to the top of the page so the user is always able to navigate to any tab available (different when logged in versus not logged in) at any time with ease. More tabs were added in later iterations.

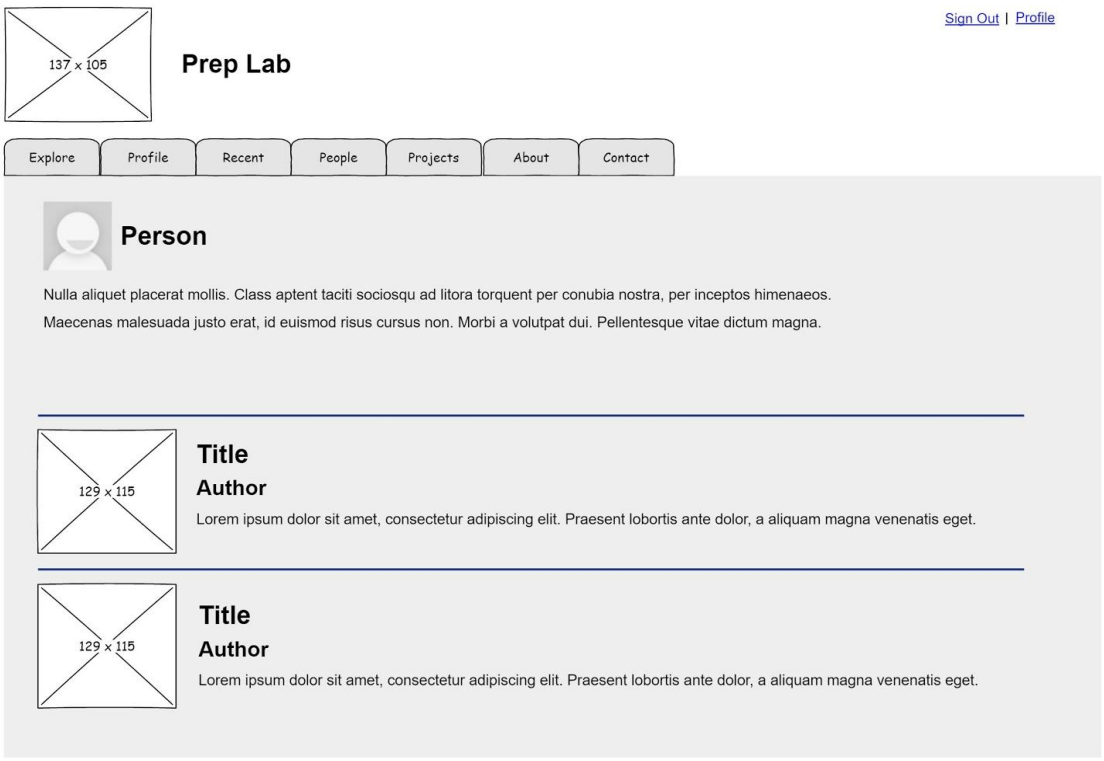

Figure 4. Profile Page Wireframe

Figure 4 is the profile page of a potential user. On this page users have an area for a profile picture, their username, and a short biography about themselves. Below the users information is a list of files that this user recently added to the site. The tabs above the user information allow for navigation back to the home page, a search page, other projects/users they follow, or a page displaying all of the repositories that they have made.

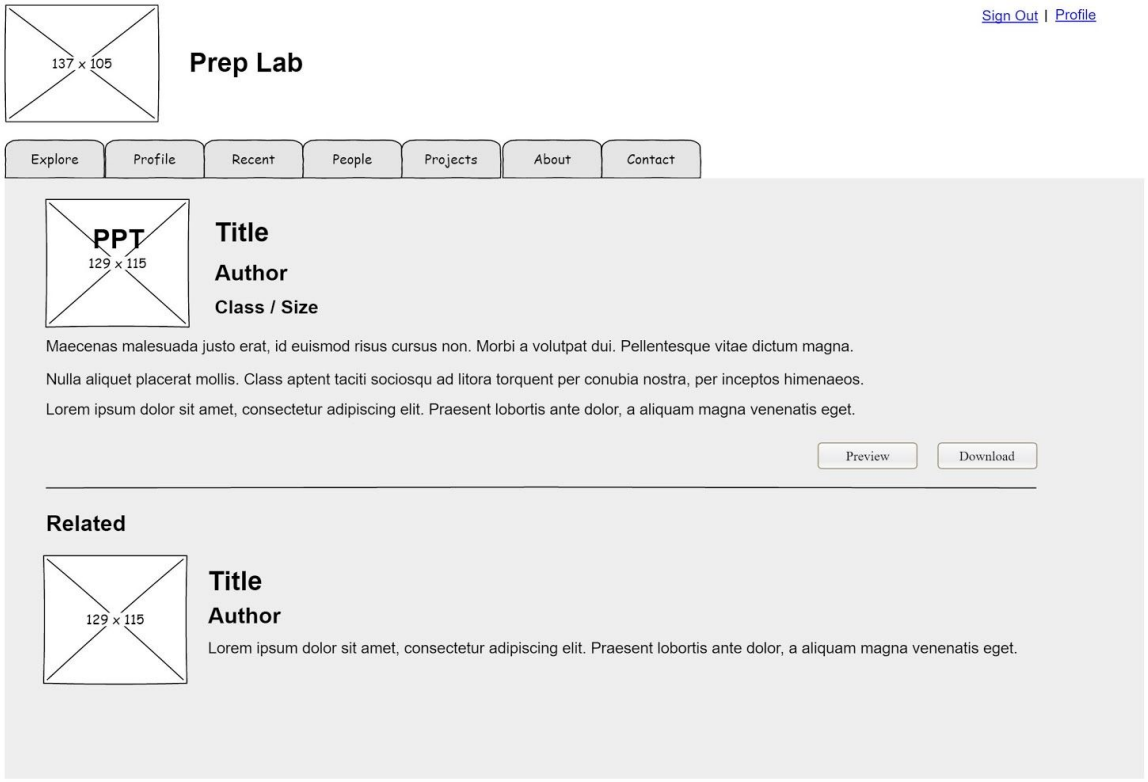

Figure 5. File Page Wireframe

A file page of a specific resource holds more detailed information about a particular file submission including more than simply the title, author, description, and file type of the resource. Additional information included, depending upon availability, is permissions for the file, the class for which the file was made, the size of the class for which the file was used, tags that identify the file, date created and last modified, and any additional notes the author would like to accompany the file.

Implementation

The project was developed using Atom as the choice IDE and utilized Git and GitHub for version control through an Atom plugin connecting the two. After the website was designed between the requirements and wireframes which created the idea of the basic layout and functionality that PrepLab should be, development for the site started with static web pages to generalize this design. Bootstrap and additional CSS then contributed to the look and feel of the website by making it have the feel of a more finished product.

After the static web pages had been developed, basic functionality, such as linking all of the pages together through routing, started to be implemented. This included breaking smaller elements out into React components so that they could be reused easily and there would be much less duplication of code as well as keep the code base modular. This also contributed to the consistency that the web application has.

Once this code was cleaned up and connected for a smooth flow between pages, additional functionality was added. This included a pseudo-login to hide elements of the application behind. Also, a way to comment on files, and a pseudo-searching functionality of the sample material to be connected through actual API and database calls in the other sections of PrepLab. Requirements such as being able to edit material a user has uploaded and editing their profile was implemented here as well.

After the main functionality was completed, actualizing the mobile friendly version of the design began. While the design implemented at the time functioned when accessed from a mobile device, it did not look finished, as the formatting was not entirely compatible with the smaller screen size. Here, time was taken to ensure that PrepLab had a similar usability level when accessed from a mobile device as it does from a computer. Most of this was simply utilizing bootstrap alongside simple conditional programming in the CSS files when the screen size was smaller than a certain width.

After the application was deemed usable with good user experience on mobile devices, progress then began on making this web application able to be connected to a database at in a future iteration. This included adding hooks where the information displayed on pages could eventually be connected to a database once one is made. It also included some base API functionality as well as calls to a JSON file, which was created for testing purposes, that can be used in the future for other developers to gain access to metadata from the back-end once developed.

Below is a list of figures that displays all of the final implementation of most of the pages and modals that were created for this portion of the project with descriptions about each one.

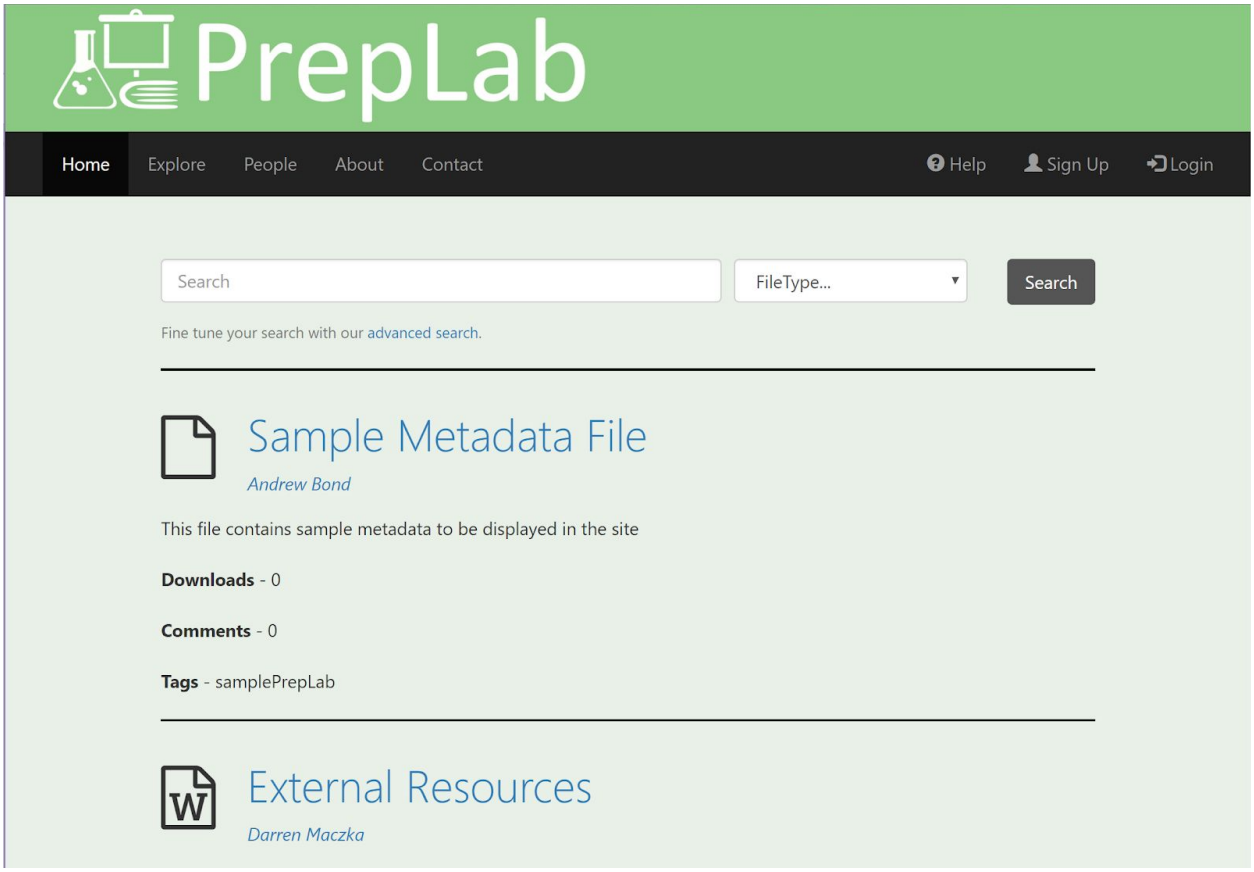

Figure 6. Home Page Prior to Login

Figure 6 is an image from the home page of the website. Here we can see the basic search functionality of the application. Below the search bar is a link to the advanced search page and a few sample files can also be seen below that. Scrolling through the page will show more files that are available to view. This is the view before a user has logged in or registered, as you can see the signup and login icons in the right side of the toolbar. On the left are all of the tabs available to a user who is not logged in.

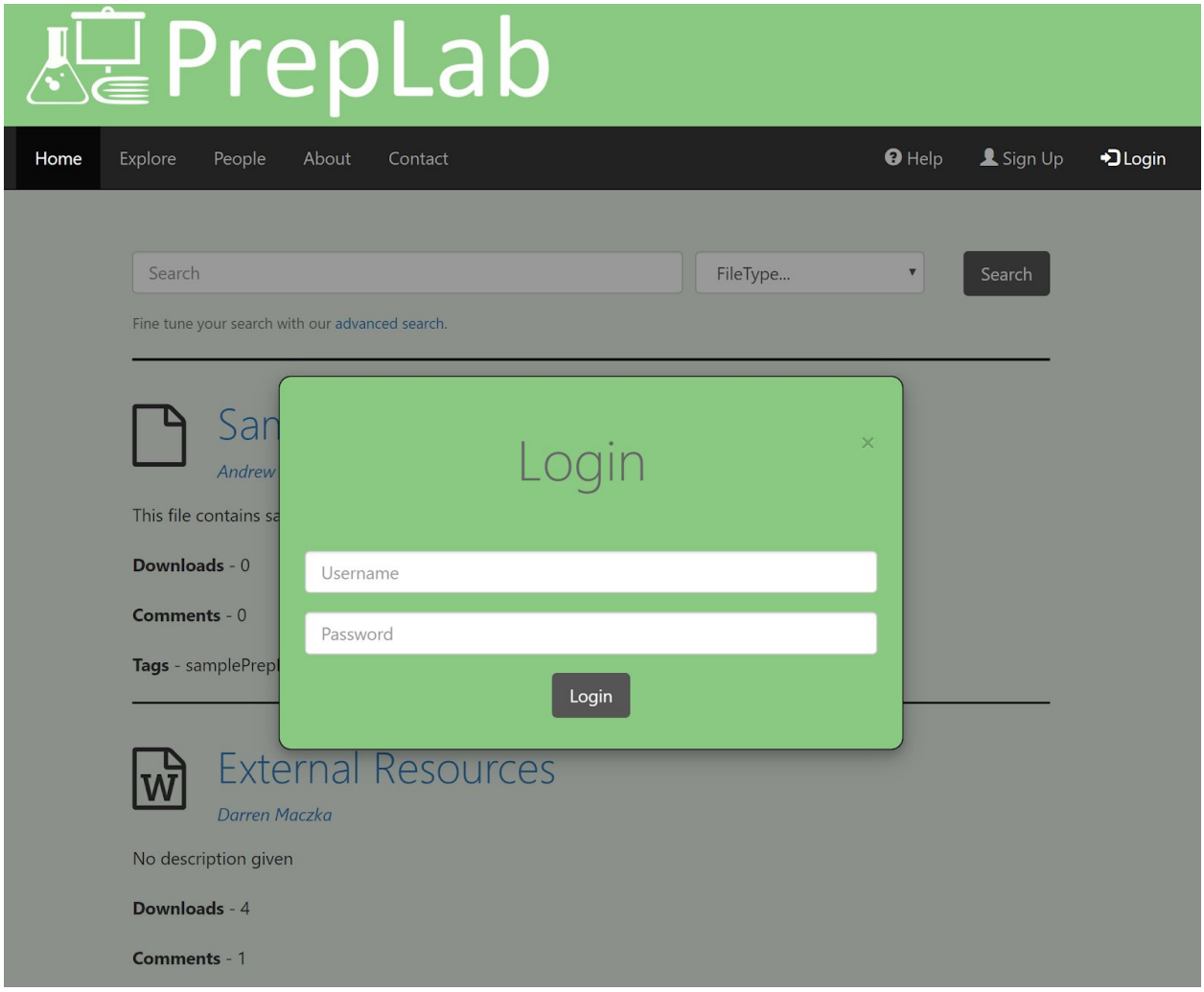

Figure 7. Login Modal

This is what the user sees after clicking the Login icon. This modal is a form that allows a user to sign into PrepLab and have access to more functionality. The modal also displays error messages if the user enters an incorrect login.

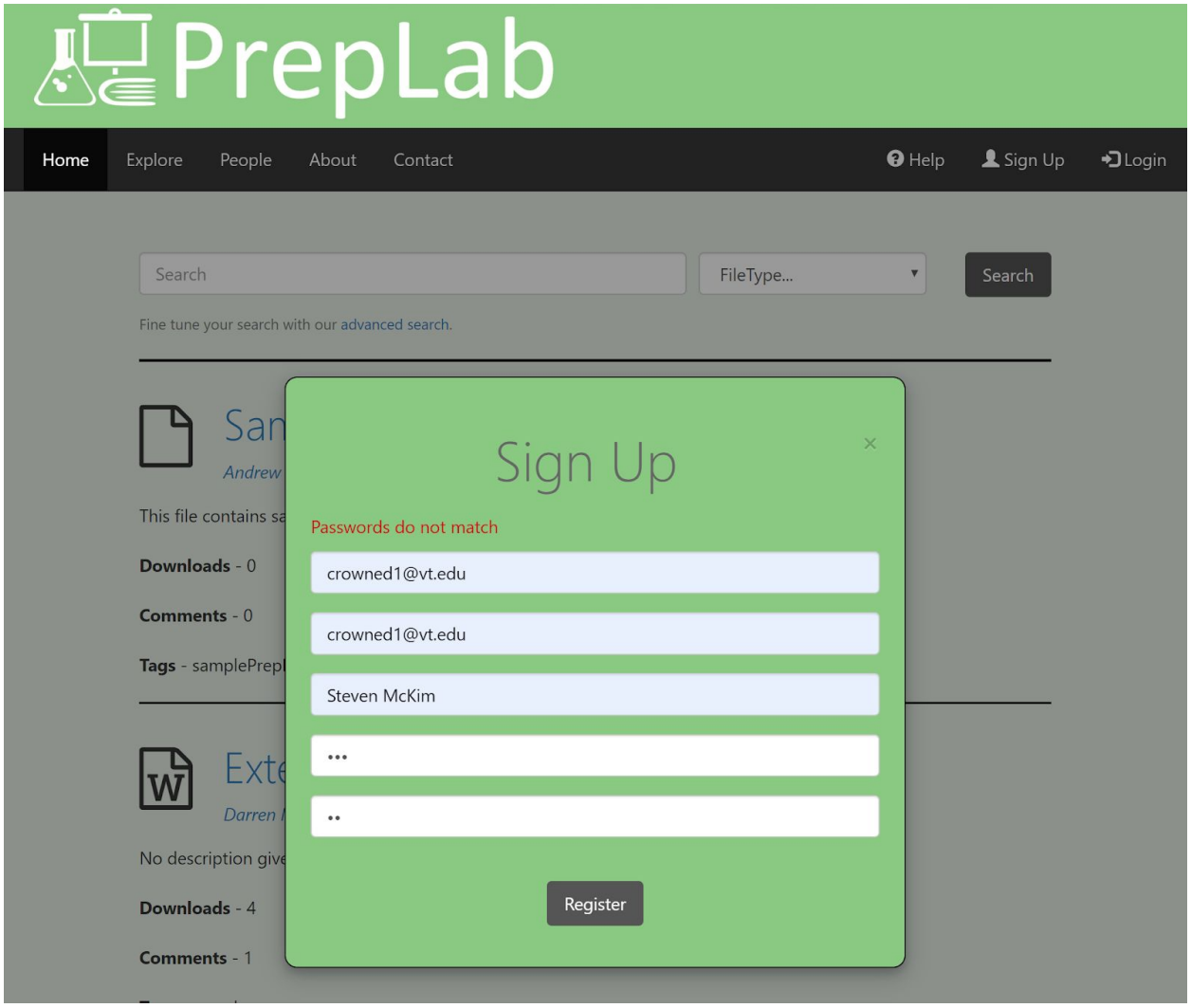

Figure 8. Sign Up Modal With Validation Error

This is what the user sees after clicking the Sign Up icon. This modal is a form that allows a user to register for PrepLab and have access to more functionality. The modal also validates the form and displays error messages to the user when these validations fail, as shown here.

Figure 9. Profile Page

Here is what a logged in user sees after visiting their profile page. Notice that some of the icons on the top right of the page in the toolbar have changed and also that this (Profile) is an additional tab on the left. This is different from the prior screenshot as the user has logged in and now has the ability to add files as well as view their profile. They also have the ability to log out whenever, as the navbar is sticky and will always remain on the top of the web application as well stay no matter which page the user is on. As for the page itself: it displays a profile picture as well as the user's name in the forefront with a short biography below. There is also an option to edit the logged in user's profile by way of button in the top right (this will not appear when viewing other users). At the bottom are then a list of all of the files that this user has submitted to PrepLab that can be clicked on to access the detailed file view.

Figure 10. Edit Profile Modal

This is what the user sees after clicking edit profile. It allows the user to change their email, name, and/password when completed. The form also offers validations as previous forms did.

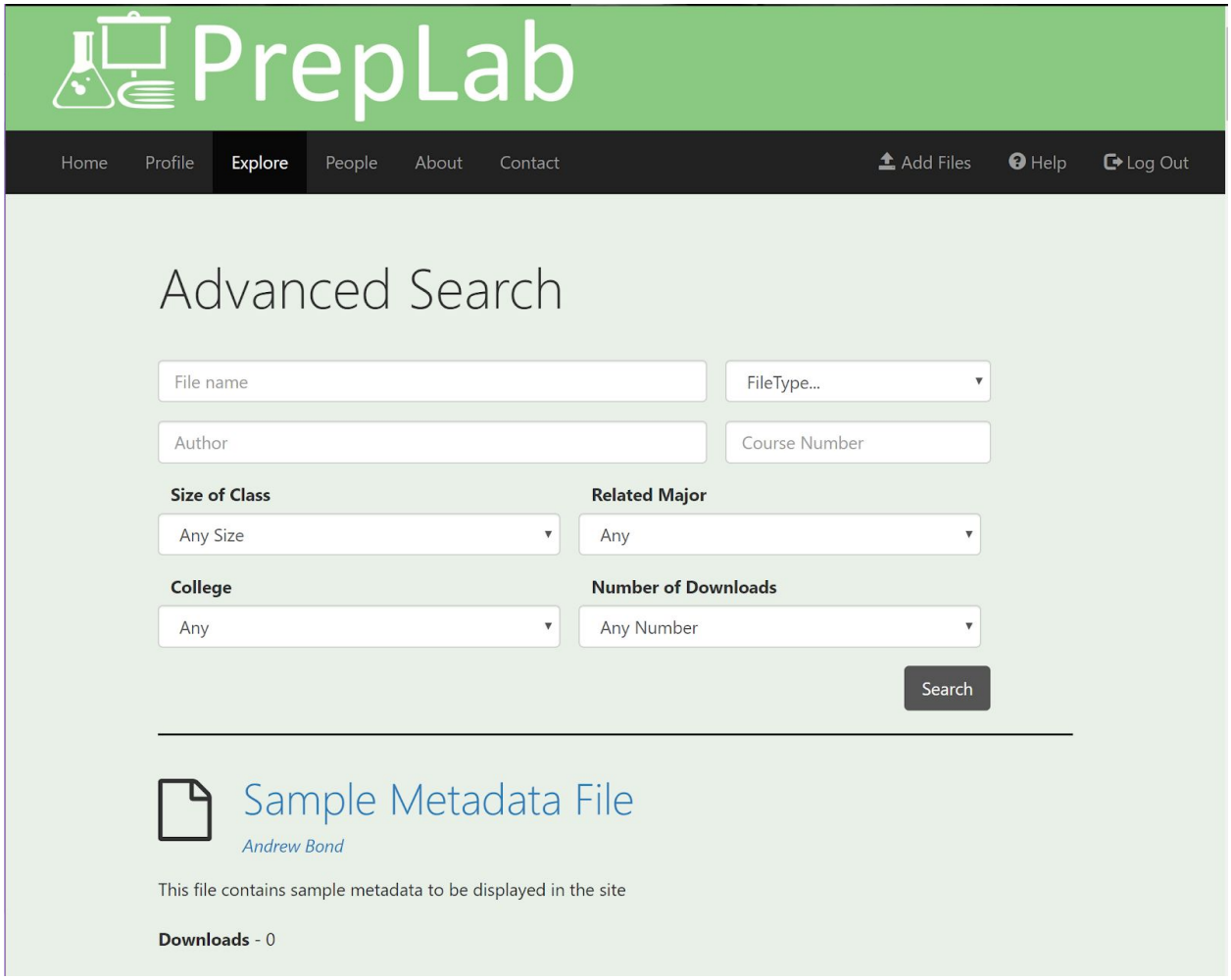

Figure 11. Advanced Search Page

Figure 11 shows the advanced searching page. It is similar to the home page, however it has more searching parameters available that allow users to narrow down the specifications of the resources they are looking for.

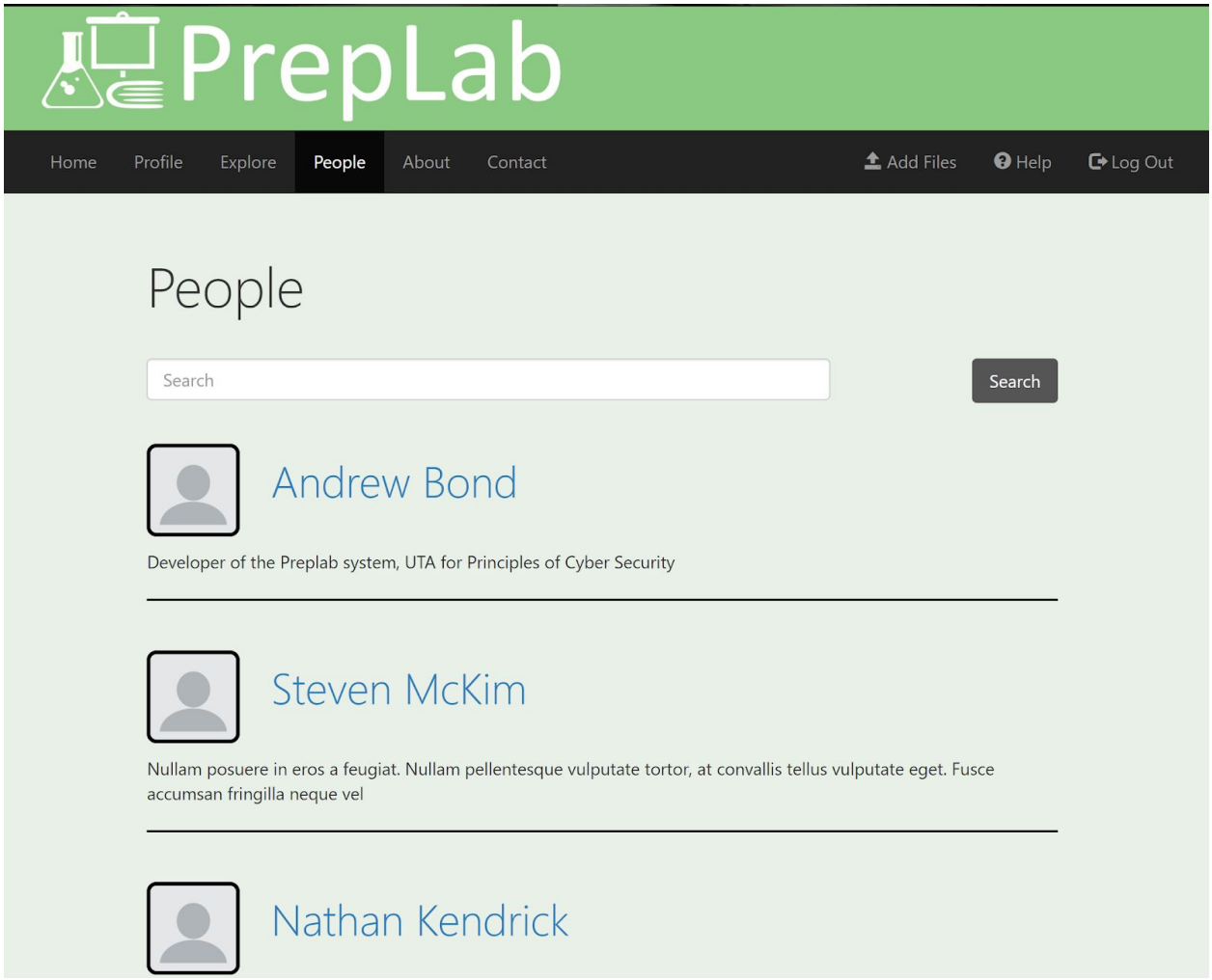

Figure 12. People Page

In figure 12, the user is able to see all of the PrepLab users. They may also search for specific users if desired. The list displays users' profile pictures, names, and biographies. When a name is clicked on, the user will be directed to that user's static (non-editable) profile page.

Figure 13. About Page

Figure 13 is the static page that describes in some detail what PrepLab is and how it came to exist.

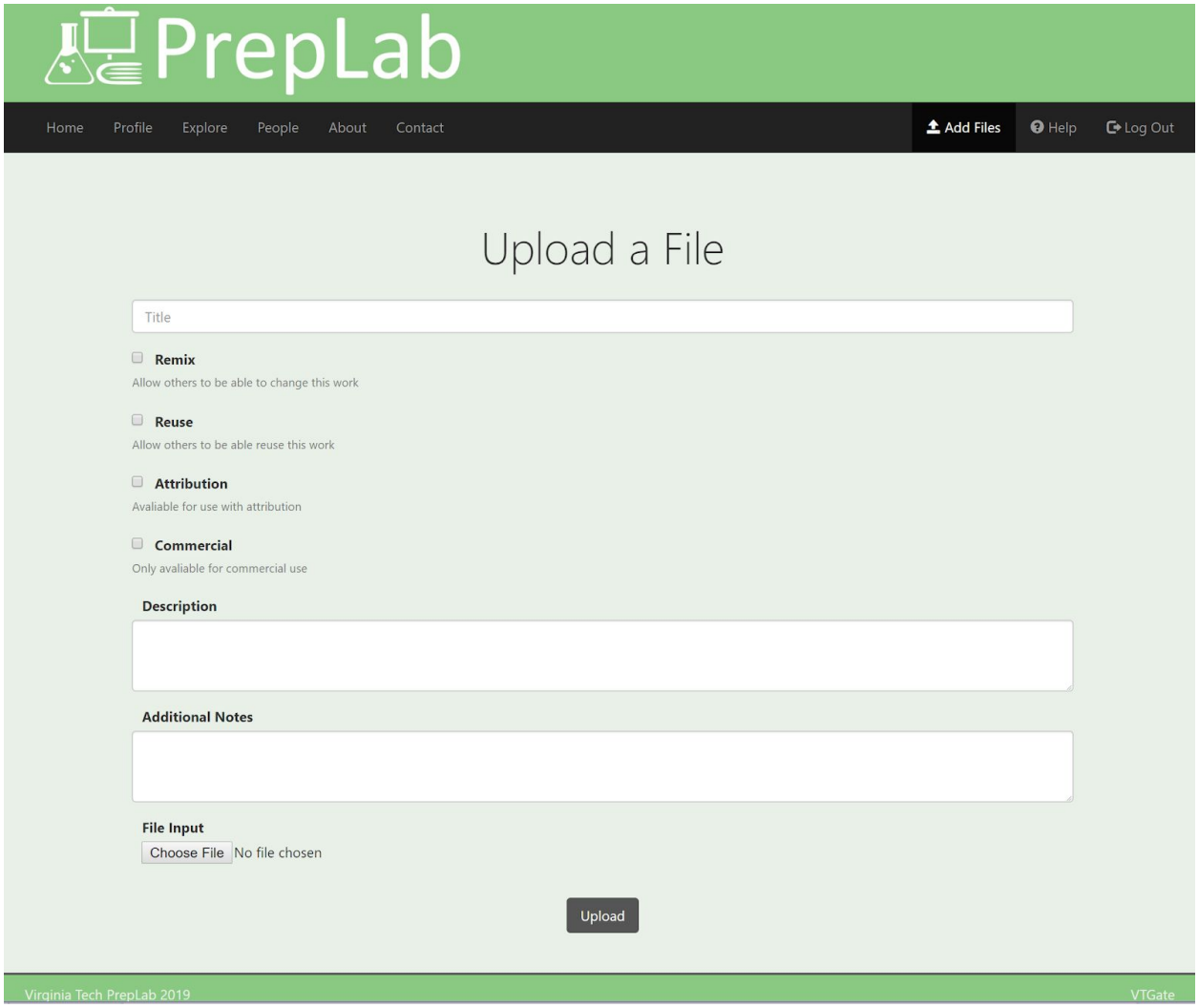

Figure 14. Add File Form

When a logged in user clicks the Add Files icon in the navbar, they are directed to this form (shrunk in size to display all; since it was smaller, the footer, which is on the bottom of every page when scrolling that far, is also visible). Here the user can add all of the information and rights for a file that they upload as a resource. (The edit file functionality - discussed later - looks nearly identical other than the header and there will be no confirmation screen afterwards).

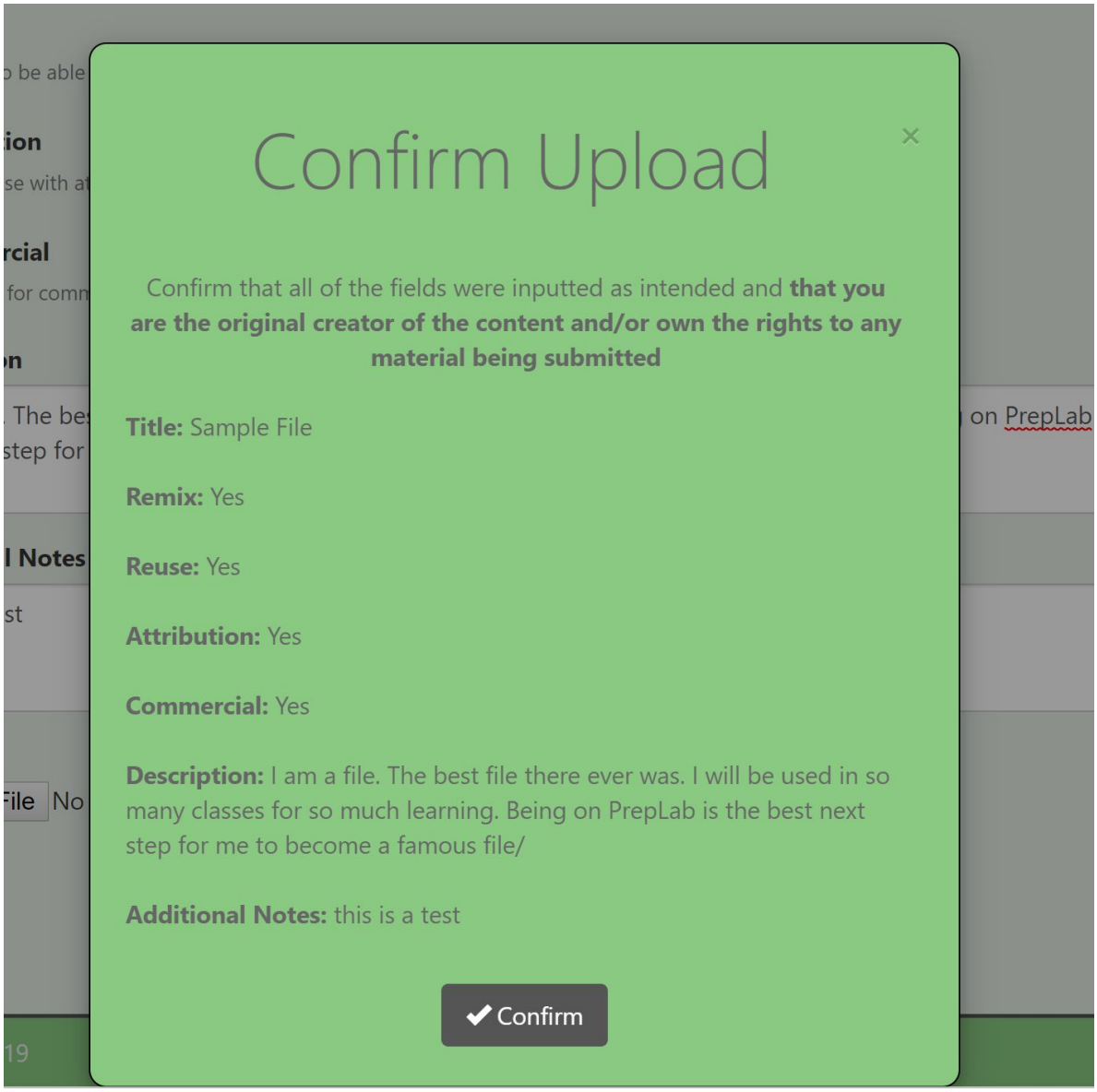

Figure 15. Confirm Upload Modal

This form appears after the user attempts to upload a file. It verifies that the user has the rights to the content or is the original creator of the material, as well as gives the user the ability to see the information they inputted and verify that it is correct.

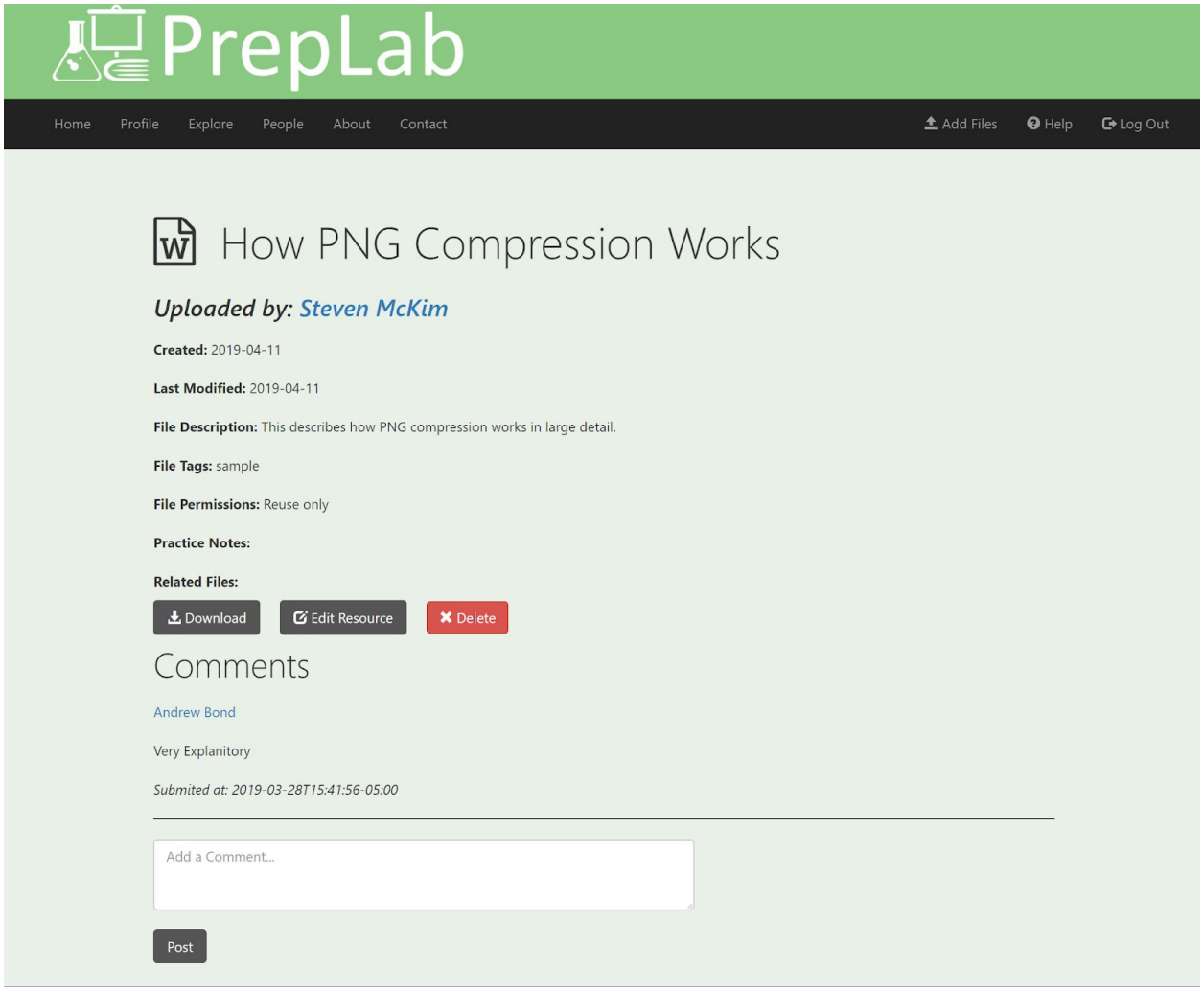

Figure 16. Specific File Page

Figure 16 is an example of a specific file page (zoomed out to see all features) that has been navigated to by the user after searching for it or simply finding it and clicking on it. It displays the file type icon next to the name with who uploaded it underneath. The name can be clicked to navigate to that user's profile if the current user wishes to see more resources that they have uploaded. Below this is more detailed information about the file that will be pulled from the database once that is attached to PrepLab. Underneath this, any user will be able to download the file if the user who uploaded it had set the permissions to allow such. Next to that are the options to either edit or delete the resource, which are only visible if the logged on user is the creator of the resource. The edit resource form looks the same as the add file form but in a modal like the edit profile modal. Below these options is a comments section that shows all of the comments that have been written about this file along with who wrote them (which links to their profile) as well as when they submitted it. After all of the comments is a small form that allows the current user to leave a comment on this file.

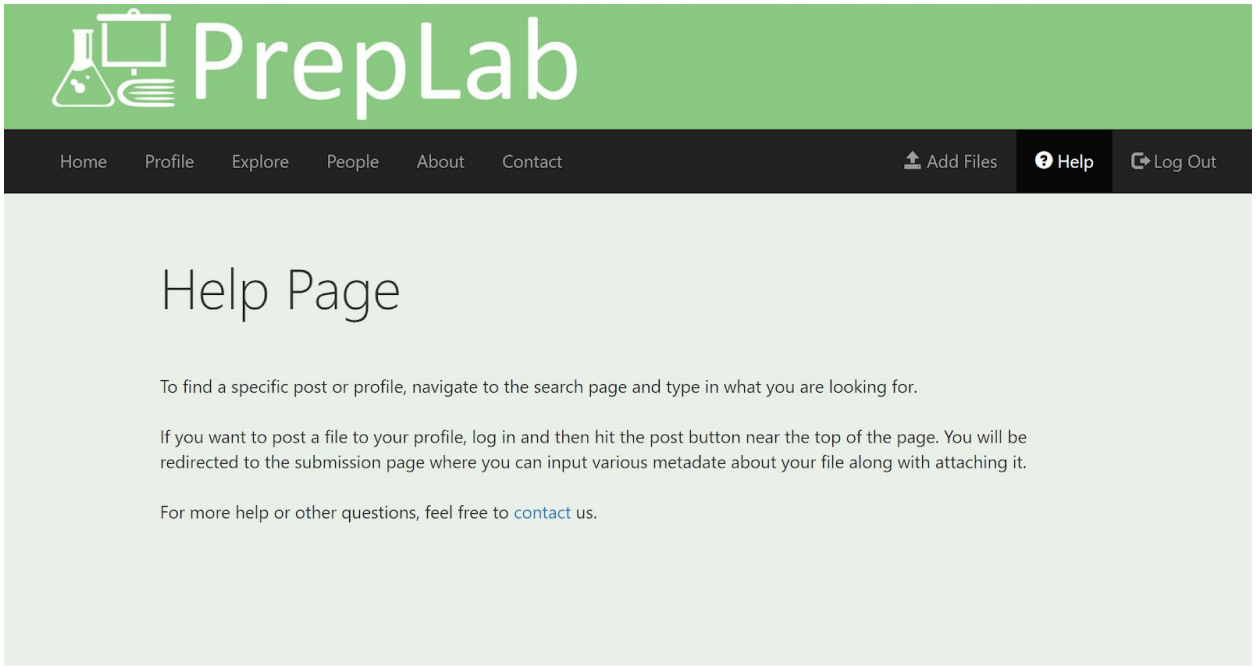

Figure 17. Help Page

This is the page that is displayed when a user clicks the help icon on the right of the navbar. It gives helpful information and also directs them to the contact page, where our clients (the PrepLab team) is displayed with contact information (not displayed in this paper). In future iterations of this project, more information will be added as well as some help tutorials with gifs, pictures, and/or videos with walkthroughs.

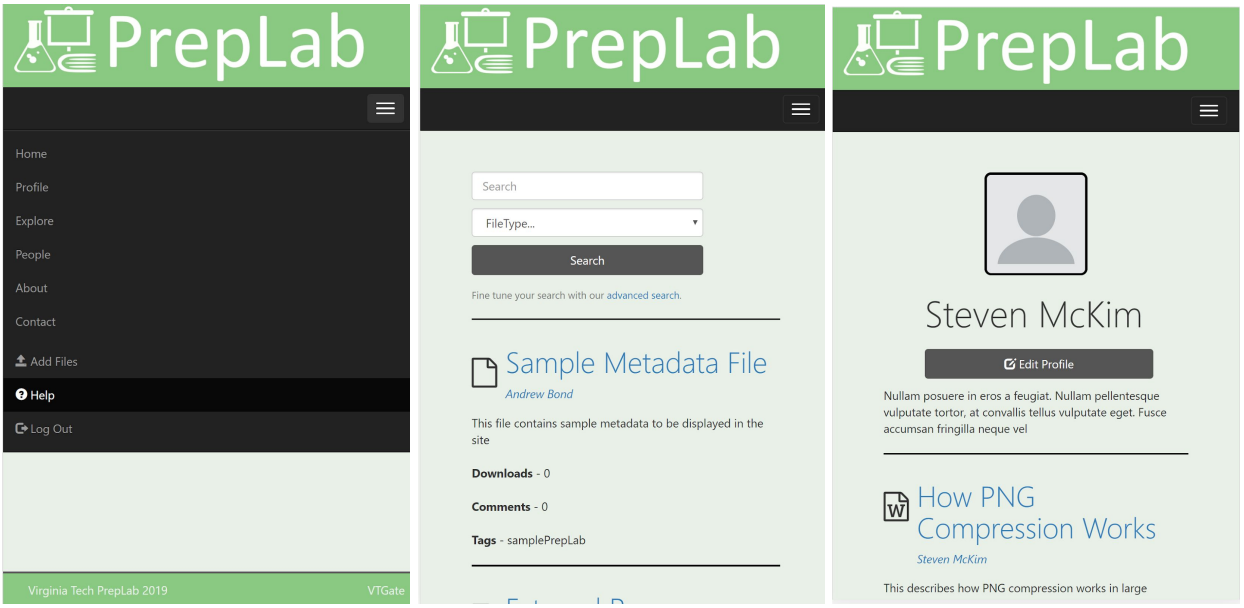

Figure 18, 19, 20. Mobile Version

Figures 18,19, and 20 are three examples of what PrepLab looks like on a mobile device. The image on the left is what the page looks like when the menu is opened into a navigation list that has the same options as the navbar in the web version of the site. The right two have the navigation menu closed in order to see the content of the application. The middle screenshot is the home page, which is the same as before other than the formatting. The last screenshot is the mobile version of the editable profile page, also the same as the web version other than formatting.

Testing/Evaluation/Assessment

Testing of the system was done with GTA and UTA students for various courses in a variety of majors. Due to the backend not being developed, only basic functionality and ease of use can be tested.

When TA's were testing the site, they were given a list of tasks to complete. Each task was assigned a difficulty rating to determine how difficult it was to accomplish the task along with a comment section for the tester to give comments on particularly confusing parts of the site. We also had a developer with them to count the number of times they had large issues along with timing how long each task took. Below are the tasks that are given in the survey:

Task 1:	Finding "CHEP Presentation" file
Task 2:	Logging in using a given login
Task 3:	Viewing Own Profile
Task 4:	Viewing a specific user's profile
Task 5:	Submitting a file with specific options
Task 6:	Comment on "CHEP Presentation" file
Task 7:	Delete "How PNG Compression Works" file
Task 8:	Edit "Example PNG File" file to have specific options
Task 9:	Logout
Task 10:	Register an account to PrepLab

Additional notes and qualitative data about the aesthetics and layout were collected using a survey that the users have an opportunity to add to after each task and add general comments at the end of completing all tasks. From these results, changes were made to the application to cater toward the intended users' preferences.

After reviewing the comments and suggestions from the user testing, we concluded that the site had overall great usability. A couple areas of the site were tweaked, however, when bugs were found or when small amounts of confusion occurred. After doing this, the TA's were consulted again to check that the changes helped clear up confusion or did, in fact, fix the bugs.

Evaluation was also conducted with the clients. Demoing an earlier version of the web app, the clients provided feedback regarding the file pages. Specifically they wanted a comment section so that discussion about the resource could take place without having to reach out to the author, as well as letting the community see what other users thought about said resource.

Other changes due to the client response include adding the number of downloads for each file as well as adding pseudo-login functionality in order to be able to hide some features behind it. Being able to mimic the backend was important because it was more attractive for the clients when trying to recruit others for the project at VTGrate.

User's Manual

Upon reaching the PrepLab website, users will have the option to sign up or sign in, figures 7 and 8, with small links located in the top right of of any page. When these icons are clicked in the navbar, it allows the user to fill out forms that are validated for correctness before they allow a user to login or sign up.

Once a user has already logged into the system the can log out of that account with a link located directly right of the profile button, which will take them to their profile page, figure 9. If users wish, they can use the search bar, located prominently in the middle of the home page, to search for a specific file. Below the search bar, different recently uploaded files along with some accompanying information will appear depending on what was searched for or if they are just getting to the site.

When looking through the search page, figure 11, the user can look more specifically at each file. Each specific file page, figure 16 holds all of the different information the PrepLab system stores along with a comment section that a logged in user can write in. They can also view all of the comments written about this particular resource, seeing information such as the comment itself, the writer of the comment (which the user can click on to go to that users profile page) and a timestamp of when the comment was written. If the user wants to download the file, they can click the download button to save the resource to their system as long as the author of the resource had set the file permissions to allow it.

Further functionality on the specific file page exists if the logged in user is the author of the resource. In-line with the download button, there will also exist options to edit the file and delete the file. When the edit resource button is clicked, a form will appear allowing the user to change information about the resource. When the delete button is clicked, a confirmation modal appears, confirming that the user would like to remove this resource from PrepLab.

If a user has logged into the site, they will be taken to their profile page which shows some information that they have added to their account including a profile picture and a short biography. Also, if this is the logged in user's profile, they will see an edit profile button, figure 10, on the top right of this page. This button will take them to a form that allows them to change any of their user information. Under their profile, some of the files they have submitted are shown. To find more information on any of the files, simply follow the clickable link on the file name.

If a user is looking to find another specific user, they can find them via the People page, figure 12. When navigating to the People page, one can just click on the link in the top bar. This will show a list of people on the site and is searchable by the search bar at the top of this page.

A user can go to each specific user's profile through the link on their name. You can also get to their page by following links where their name is listed under a file.

Other tabs in the navbar include the about page, figure 13 and the help page, figure 17. These are static pages that a user may navigate to if they wish to learn more about the PrepLab project, contact the PrepLab team, or if they require any assistance when using the application respectively.

Any of the tabs can be reached at any time in the user's navigation of the web app, as the navbar will always remain at the top of the screen. In the mobile version of this application, figures 18, 19 , and 20, the only difference in functionality, other than minor formatting changes, is that the navbar menu becomes collapsable, and in order to navigate through the site, the user must first click the menu button (triple bar on top right) in order to open the navigation menu as a list.

Developer’s Manual

Several commands are required to start working with this version of the code base for PrepLab. After downloading all the files from GitHub, or wherever the repository may be moved to in the future, the following commands need to be ran on the hosting system assuming that the host system is linux. If the host system is not linux, the developer should download Node.JS and use that command prompt. Whichever way this is done, the developer should navigate to the folder ‘prep-lab’ in their terminal from wherever they put that folder in their file system when cloned and run these commands:

```
sudo apt install curl
curl -sL https://deb.nodesource.com/setup_10.x | sudo bash
sudo apt-get install -y nodejs
sudo npm install pm2@latest -g
sudo npm install react-scripts -save
sudo npm install react-router-dom
```

Once these are ran, make sure that everything is properly installed by running “node --version”. If everything is installed properly then run the instance of PrepLab locally by running “npm start” or “pm2 start npm -- start” in the “prep-lab” directory of the terminal. If the app is being ran locally then it can be reached via “localhost:3000/home”. The home page is found under the route ‘/home’, the rest of the site can be accessed through here.

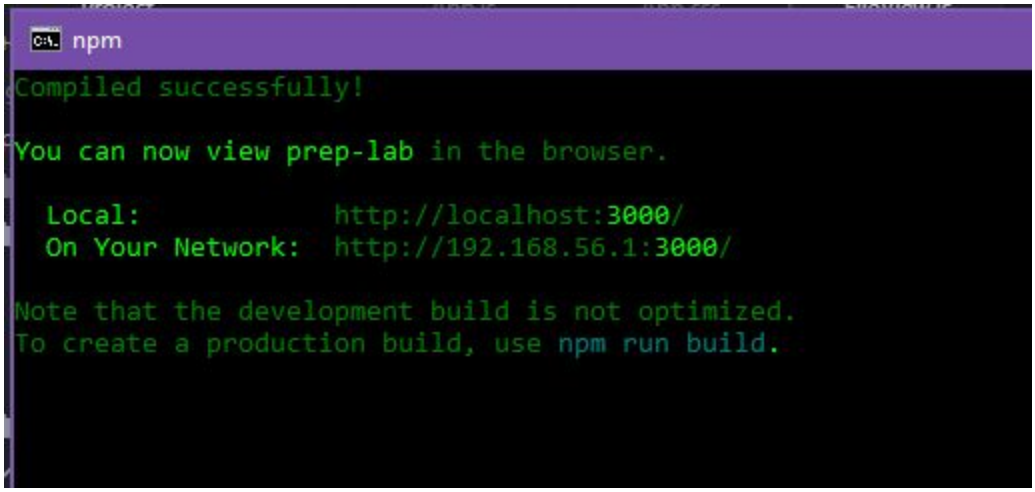

Figure 21. Successful NPM Start

Figure 21 is an example of what the terminal should look like if the repository compiled successfully after npm start is ran. You can see two options of how to reach the now hosted PrepLab site, and npm should automatically open the localhost version in the system's default browser. This is seen in a Node.JS terminal on a Windows 10 machine.

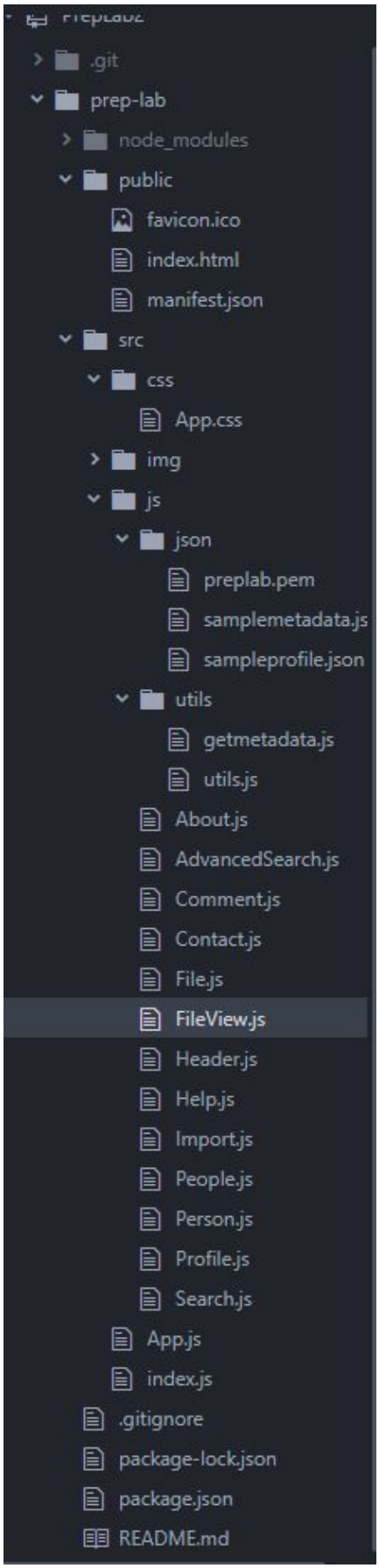

Figure 22. Developed Files

In figure 22 are all of the files that were made for the front end to work. The .git folder contains all of the workings of git and shouldn't need to be modified. The node_modules folder should also never be modified, as this is the folder where all dependencies are installed, such as "react-scripts" and "react-router-dom," which were installed on set-up. Make sure that this folder is ignore in the .gitignore file.

The public folder contains the PrepLab icon, as well as the only HTML page, which is where all of the React elements are rendered. This file should also not ever need to be changed, unless the footer is being edited, as all other components displayed on the site are coming from React. The "manifest.json" file is simply information about the web application that allows browsers to render the icon in the tab. The files in this folder should be the only ones that a user is able to see publically.

Everything in the src folder is where all of the real magic happens. The css folder is self-explanatory, as it contains the CSS file (in the future more can be added) that was created for all of the styling of the front-end of PrepLab. This is also the file where the conditional CSS is held that allows this web app to be mobile friendly. Edit this file with caution, as most classes are used in multiple places. When in doubt, either use existing CSS classes that were created in this section of the project, or create new classes and put them at the bottom, but above the conditional CSS. Remember that since CSS is cascading, properties are overridden by the last one read, so always keep the conditional at the bottom, or some styling may not actually change when viewed on a mobile device, which can cause mobile formatting errors.

The files that are in the "img" folder have been omitted because these are just images that were used for profiles pages and the contact page. There is a javascript file ("getprofileimage.js") in here that allows for getting user profile images.

All files within the “json” directory pertain to connections to the backend or API. Both of the json files (“samplemetadata.json” and “sampleprofile.json”) are what we are expecting to come back from the API after calls contained in “getmetadata.js”, which is located in the utils folder. This file connects the JSON data to the front-end by putting JSON data in a format that React understands, which for our purposes, is simply an array.

The remaining file in the json folder are “preplab.pem”. This is a file that allows developers to access a server that hosts the site so that developers not running the code locally can access the site via mobile or web rather than locally. This file should be replicated when the next development team comes in to continue development on PrepLab by completing the backend and API.

The file named “utils.js” contains basic static functions that are used throughout the web application. This includes functions such as “getFileIcon”, which will return the icon that will show for each file type, and other basic functions that help PrepLab run smoothly and look and feel more intuitive. Putting these types of functions here - static ones that are used in multiple parts of the code - helps keep the code modular and reduces the amount of repeated code greatly.

From here, we see all of the React component files. All of these files correspond to different React components (sometimes whole pages) in order to keep the code base modular. The “About.js,” “Contact.js,” and “Help.js” are all static pages that will display under the header when navigated to. These can be edited by directly changing the HTML code in the render method of those components.

The file named “Header.js” corresponds to the header that remains in the web application no matter which page is navigated to. This files is one of the most important as this component's state stores which tab the user is currently on. It also stores the state of a user being logged in or not (this will eventually be moved out of here, as this exists solely for our pseudo-login) in order to hide or show certain elements in the navbar. This file also contains the code for the login and sign up modals that appear when the user clicks on these icons in the navbar, as well as the validation code that goes along with these modals.

The files named “Search.js” corresponds to the home page, since this is where the basic search is held. It contains basic search functionality as well as a list of all of the files that can be navigated to, as well an the authors of said files where there profiles can be navigated to. The “AdvancedSearch.js” file is very similar, in that it also contains pseudo-functionality for search (not connected to a database, so just parses what’s in the component, as does the basic search) as well as a list of all of the resources in the same format. The list of resources is gotten using a function in the utils files, for both, that allow constraints on which files are displayed.

The “People.js” file is similar to the last two in that it has a basic search functionality as well as a list of smaller components, except here that component is people rather than files. It

contains the search bar as well as a call to the getmetadata.js file in order to get all of the different users of PrepLab, which can all link to their full profiles.

“FileView.js” is the file that displays specific file information. It contains all of the code that displays information about a particular resource, as well as all of the comments on that resource. It also contains a form that has a hook that will eventually be connected to the database of PrepLab, once it gets developed, that allows logged in users to post comments. It also contains the ability to edit or delete a file (again once connected to a database) that is hidden behind, at the moment, a check to see if the logged in user is the author of this file. This is hard coded for now, as there is only 1 login that works. Since these functionalities exist here, this file also contains the code for the modals that allow the user to edit the resource or confirm the deletion of the resource if they are the ones who uploaded it.

“Import.js” is the React file that contains the form to add a resource to PrepLab. This file contains all of the update functions to all of the fields in the form as well as the code for the confirmation modal discussed in previous parts of this paper. The modal gets filled with the information that the user has inputted into the form.

The file “Profile.js” is similar to the FileView file, as it shows the detailed information about a specific person, as the FileView does for file. This file also contains the code for the modal that allows the logged in user to edit their profile (this check is also hard coded at the moment since there is only one login). It also contains code that calls files in utils that parse the sampledata for only files written by this user and displays those files under the profile.

The rest of the files in this section of the code base are smaller components that are used on several different pages. The “Comment.js” file contains the Comment component, which contains the information on how to display each specific comment on the FileView page. The “File.js” file contains the File component that does the same for each specific file that gets displayed on the home page, the advanced search page, as well as the profile page. The “Person.js” file is the same for the Person component, which is displayed on the people page. Separating these components out keeps the code base modular and reduces repetition of code. It also allows easy edits of a smaller proportion that changes these objects the same way throughout the entire web application, making it easier to keep the user experience consistent.

Below this section of React files, we have the main two React components, “App.js” and “index.js.” These files are the backbone of the React part of the application, which is the main part of this portion of PrepLab

“App.js” contains all of the routing between pages. At the bottom of the class in the App component, there are three different Routes that contain the three formats of routing options. The first is a base page, which calls upon the Body component to display one of the pages

based on the url path. This component also makes sure that the header is always displayed, which takes in the property of which page is being displayed so that the header styling can inform the user of which page they are on. The next two routes are for specific profiles and files. These routes will display the profile page or file view page respectively, and also pass in the id, of the specific user or file that is being accessed, through the url parameter so that these pages can pull information from that specific object from the database.

The “index.js” file is a simple file that should never need to be changed. The purpose here is to simply display the final HTML after all of the React components have rendered together by sending all of the HTML code that React has compiles to the index.html file that is publicly facing.

From here, we have the .gitignore file which can be edited so that git ignores changes to certain files. There is also some package data, which is comiles if any packages are installed through npm (or yarn or pip or comprable commands) as well as the README file for the project.

The files with the “.json” extension hold data that is read from the API in the following format:

“Samplemetadata.json”

```
id:                 # integer (primary key)
title:              # string
author:             # integer (foreign key)
created:            # ISO-8601 date and time
last-modified:      # ISO-8601 date and time
file-type:          # string
downloads:          # integer
comments:           # a list of comments in the following format (can be changed to be own
                        table in database in future iteration, which would mean this would
                        just be a foreign key for a comment object id):
                    - author:       # integer (foreign key)
                    - comment:      #string
permissions:    # a list of the creative commons permissions associated with the item (this
                    can be changed to all be separate boolean fields for easier use in future).
                        - remix
                        - reuse
                        - attribution
                        - commercial
description:    # string (formatted with Markdown)
tags:           # a list of tags associated with the item, some examples follow, these will be
                    set either by the contributor, or a content moderator
                        - lab
                        - large class
```

```
                                - peer interaction
practice_notes:     # an array of dated text notes, formatted with Markdown
                                - note
                                - YYYY-MM-DD
```

“**Sampleprofile.json”**

```
id:                 # integer (primary key)
name:               # string
profile-image:      # location of file through different call
files:              # This is a list of integers (foreign keys) that represent the id’s of files this
                        user submitted
description:        # string
```

The API for this project will eventually be able to be called and pull data into JSON files in these formats directly from a backend database, both of which are parts of separate projects.

Lessons Learned

Creating a fully formed idea of what the website as a whole would be able to do in future use cases would have helped in developing a thoughtful design for the front end to accommodate. While this was the practice adopted for this project, input from potential users in the UTAs and GTAs could have been collected at an early stage to aid in the designs of the web pages.

Throughout the project, the design and implementation had been rather fluid, changing as the project progressed. Having a few solid deadlines for how far into the implementation of PrepLab the project should be would allow for a more refined design and less rushed development, as it would have allowed for more iterations.

Figuring out how to test a website without full functionality was a challenge. As a result, surveying users was the only method we had of collecting feedback on the design and aesthetics of the front end created for PrepLab.

Adding hooks where a database was also difficult with nothing to connect them to with no way to test if they will work or not. This includes simple calls that the React files could make to get information to display as well as the JSON formatting and API calls. Having even a dummy database would have helped with the testing and implementation of this immensely.

When designing the static web pages, it would have cut down a decent amount of work if it had been broken up into reusable components earlier. When the static pages were being made, there was quite a bit of repetitive code that needed to be changed in several areas even if only one component was being changed.

Future Work

The current website is fully functional however many features could not be finished until the API and backend of the site are finished by other projects, or were simply not priorities to complete fully. The parts that need to be changed so that they work properly are commented within the code in each of the files that require it. The files that require some change in order to work with the API and back-end are listed below:

- samplemetadata.json
  - This file works as the pseudo-backend so that we could show functionality on files. It can be removed once a backend is present.

- sampleprofile.json
  - This file works as the pseudo-backend so that we could show functionality on profiles. It can be removed once a backend is present.

- getmetadata.js
  - This file has a method for reading in data about various files and profiles in the system. It may need to be modified to connect with the API easier in the future.

- utils.js
  - This file helps with getting file icons for various pages. If there are more file types that are permitted, then they should be added here.
  - If there is any static functionality that is used by more than one component, that should be added as a function here.

- AdvancedSearch.js
  - This serves as the advanced search page for finding more specific files on the site. Proper connections to the API need to be made so that when a

back-end is made, it is actually searching the database rather than what is simply displayed on the page.
    - Additional search capabilities should be added here as in the current iteration of PrepLab only checks the title, description, and file type.
    - An increased functionality here would be to display the filters the user has selected in an svg list and allow them to deselct those filters by pressing an ‘x’ next to the fields.
    - Another increased functionality could be to allow users to select multiple options from the dropdowns.
    - If any more fields are added to file objects when the back end is made, they should be added here as an additional way for users to filter resources.
    - A user experience improvement that should be made is to paginate the resources so that not all of them are displayed at all times.
    - When a back-end is connected, more options should also be added to the drop downs, as not all possible options were added in this iteration of PrepLab.

- Comment.js
    - If fields are ever added to comments in the database, edit this file to be able to display any additional fields.

- FileView.js
    - The comment section will require the API to properly post to the backend of the site. The download button also needs to be properly linked to the API once developed.
    - As more fields are added to the file objects, they should be displayed here.
    - The edit resource modal should eventually display the values that the resource currently has and when connected to the database through an API call, only change the fields that are changed in the form.
    - Validations should be added to the edit resource modal in order to make sure the data the user inputs is valid.
    - Additional fields should be added in the future to allow the user to add class size, college, topic, etc to the resource.

        - Any field that is a drop down should be made into a multi-select so that the user has the ability to select multiple topics and such.
    - The confirm delete modal should actually delete the file from the backend once it is submitted by the user.
    - To increase usability, a user should be able to preview the files somehow in PrepLab.

- Header.js
    - This page contains the login modal, which allows a user to login as long as they have registered with the site. This needs proper linkage to the API once it is developed.
        - There is only one hard coded login at the moment.
        - Once logged in, the application should change more than just the state in the header so that the whole application knows whether or not the current user is logged in.
    - This page also contains the sign up modal, which allows a user to register to PrepLab. This needs proper linkage to the API once it is developed.
        - More fields should eventually be added to this form such as biography and profile image.
        - Validations for additional fields should be made
        - Once registered, the application should change more than just the state in the header so that the whole application knows whether or not the current user is logged in.
    - Once the state of being logged in is more global, static url navigation (where the user simply types in '/import' or other in the url) should not allow non-logged in users to get to pages hidden behind login.
        - Should also hide edits, deletes, and comments behind global login as these are currently hard coded in several areas of the code as explained elsewhere.

- Help.js
    - Additional help should be added.
        - Should include walkthroughs that include pictures, gifs, and/or videos so that users are easily be able to accomplish tasks in PrepLab if they did not know how to previously.

- Import.js
    - This will need to be updated to allow for proper submission of files to the API and backend.
    - As more fields are added to the file objects, they should be able to be edited for new resources here.
    - Validations should be added to the form in order to make sure the data the user inputs is valid.
    - Additional fields should be added in the future to allow the user to add class size, college, topic, etc to the new resource.
        - Any field that is a drop down should be made into a multi-select so that the user has the ability to select multiple topics and such.

- People.js
    - A user experience improvement that should be made is to paginate the people list so that not all of them are displayed at all times.
    - The list of users should eventually pull from the table of users in the backend through API.

- Profile.js
    - As more fields are added to the user object, they should be displayed here.
    - The edit profile modal should eventually display the values that the user currently has and when connected to the database through and API call, only change the fields that are changed in the form.
    - Additional fields should be included to the edit profile modal, such as biography and profile image.
    - Additional validations should be added to the edit profile modal in order to make sure the data the user inputs is valid.
    - A user experience improvement that should be made is to paginate the resources so that not all of them are displayed at all times, or format them differently than in the two list views.

- Search.js
    - This is the homepage of the website, it shows various files hosted. The searching mechanism needs to be connected to the API once it is developed.
    - A user experience improvement that should be made is to paginate the resources so that not all of them are displayed at all times.

Any file not listed above should not need to be edited, but if it does become required, check the developer's manual section to see what each file in the code base does. Future development of the API and database was kept in mind to make it as easy as possible for the next projects to continue on from what was developed in this section.

Acknowledgements

The team of Darren Maczka, Britton Hipple, Leanna Ireland, and Sarah Donnelly were indispensable in this project. Without their help and ideas the project would not have gotten up off the ground. We gained valuable feedback for how the site looked and functioned. The ideas that underlie how preplab was designed borrowed heavily from the sample site that they developed.

Another thanks is due to Dr. Fox who lead the course that inspired the team to pick up this project. Without his guidance and support we could not have had as polished of a site or had the full understanding required to build out the site.

References

Github - Build Software Better, Together - 2019
https://github.com/

National Science Digital Library - 2019
https://nsdl.oercommons.org/

PrepLab Sample Page - 2019
https://blogs.lt.vt.edu/preplab/

React Tutorial- 2019 -
https://reactjs.org/tutorial/tutorial.html

Darren Maczka - Client - dmaczka@vt.edu

Leanna Ireland - Client - lireland@vt.edu

Britton Hipple - Client - britthip@vt.edu

Sarah Donnelly - Client - srdonnelly04@vt.edu

Edward Fox - Professor of Computer Science - fox@vt.edu

Appendices

React – A Javascript Library For Building User Interfaces
https://reactjs.org/

W3 Schools CSS - Style Sheet Language for displaying HTML
https://www.w3schools.com/css/

Bootstrap - Toolkit for developing with HTML, CSS, and JS
https://getbootstrap.com/docs/4.3/layout/overview/