Team 5 Long Project README

Table of Contents

Approximate hours worked

We worked on our long project individually for about 8 hours each week, in addition to a 2 hour weekly team meeting. In total, approximately 50 hours were spent on this project.

Where your MySQL connection code is (so we can modify it if necessary)

The connection with MySQL is done within Drupal.

Link to your project: https://users.cs.duke.edu/~ch311/oncology-web/

Link to Github repository: https://github.com/cs-190/t5-jomo.git

Usage information for each file and directory

Open the oncology-web folder and you will find the following directories:

1. misc: Drupal's default ajax.js (& other scripts), icons, images, and style sheets
2. modules: Drupal's default modules, extended by custom modules in "sites"
3. scripts: Drupal's default scripts

4. themes: Drupal's core themes; custom themes we used are in the sites/all/themes directory
5. sites: this is where all of our own content is stored, including
    a. 3D viewer libraries
    b. custom modules we installed
    c. the custom theme we are using, called Nexus

Walkthrough

Precision Computing for Precision Medicine (SPARSE) was developed as a response to Duke Radiation Oncology and Duke Computer Science's need to be able to share their research with the academic world in a simple, yet elegant format. Our project's goal was to create a website that members of the faculty who are not experienced with Computer Science are able to navigate and manage important content. As a result, we mapped the layout of the website and began to identify several core functionalities.

Before working on the content of the website, we had to select a proper theme for the website: one which presented the academic research done by Duke Radiation Oncology and Duke Computer Science in an aesthetically pleasing, sleek manner, which is convenient to navigate for the public and manipulate by Duke faculty. As a result, we picked the Nexus theme: a professional yet minimalistic theme which focuses on displaying the content in a streamlined fashion to users. Nexus provided us with a prime selection for block placement on the website, seemingly designed for displaying information or products to the public. Most importantly however, researchers often have a lot of windows and programs open simultaneously, so the responsive design of the Nexus theme was the most important determinant for choosing our theme, in order to ensure that the content can be viewed properly by both faculty and guests.

First, we had to create three different kinds of user accounts: guest, authenticated users, and administrators. Although we want the general public to be able to view the research which has been done at Duke, only Duke faculty members ought to be able to upload content, and only certain people should be administrators for the site. As a result, we successfully implemented a login functionality so that only certain users have content creation and site manipulation permissions: this was crucial to begin working on the site because it provided us with the secure platform from which we could build the rest of the application.

Next, we had to create five major pages: a homepage, a people page, a projects page, a publications page, and a contact page. On the home page, we successfully implemented the slideshow displaying a couple of featured news articles, followed by a banner displaying the mission statement, leading to a news feed displaying the most recent news, ending with a footer displaying the three of our names in addition to a copyright.

There were several features we had to implements between the people page, the projects page, and the publications page. We have implemented a commenting feature on projects and publications pages, where people can either comment on individual pages, or comment as a response to other people's comments. Most importantly, we had to provide an easy way for researchers to populate the new content to appear on these pages: researchers now only have to click "add new content," and select the type of content they are trying to publish: Person, Publication, or Project. Through the utilization of the Biblio module, we have streamlined the uploading process for faculty to only have to enter a PubMed ID or a DOI in order to add a publication. After filling out the form to create the content, it will automatically appear on the relevant page. In order to make sure that content is automatically updated throughout the site, we spent a lot of time designing the relations for the website. As a result, all relations work just by updating the project references on a specific publication: we do not need a publications field on the projects or people content types, only a project field on the publications type.

Further, we had to implement institutional filtering on the people and publications page: the ability for users to view content either from only Duke Radiation Oncology or Duke Computer Science. On the people's page, we have a list of the faculty members involved in research along with the three of our profiles listed. On the project's page, we have a list of projects that SPARSE is working on. On the publication's page, we have a list of publications from SPARSE in addition to a filter for users to be able to sort them based on a variety of factors. In addition, we provide for an easy method for users to be able to filter publications by author or keyword: the drop down menu for publications allows users to view all publications sorted by either qualifier.

Finally, on the contact page, we had to set up a form with a drop down menu with two choices, to send a message in the form of an email to either website research feedback or website developer feedback. Based on the user's selection, the form sends the email to a different address.

All in all, we have created a product which allows Duke faculty to properly manage their own website in a presentable fashion to the academic community: we hope that SPARSE will be the platform for further academic dialogue and debate.

Technical details

User registration & user profile set-up (under My Account)

- registration process was customized under Configuration/People/Account Settings
- added person contact form feature so that users can contact each other while keeping email address private
- customized email verification and notifications
- added new user fields, including full name, about me, institution affiliation, projects, publications, LinkedIn profile link
- User profile page displays profile picture as well as the relevant information

Adding news & showing up on homepage feed

- added new fields in the Article content type
- added new block on homepage that automatically displays news articles chronologically

Homepage slideshow

- changed under Appearance/Theme Setting/Front Page Slideshow
- allows the web admin to choose 3 featured news articles

Who's online (homepage right sidebar, only viewable by admin)

- shows the users that are currently logged in to the website
- edited visibility so that only admin can view this block

Administrator dashboard

- under Dashboard (left-most option on toolbar), web admin can view relevant information about their website all in one view, including recent content, active forum topics, recent polls, etc.

Administrator toolbar (hovering shows all options; enables fast access to any administrative resource on the website)

- used the Administration Menu module to change the admin toolbar from click-based to hover-based
- now when an admin hovers over the button, all options automatically display for quick navigation

3D image viewer

- installed modules such as 3DHOP, JSC3D, and Thingiview.js to enable the viewing of various 3D files, including .obj, .stl, .ctm, .3ds, etc.

Because we used Drupal as our CMS to create SPARSE, most of the work done on the website was done through a manipulation of structures and views. In order to create content types for projects, publications, and people, we created new content types: person, project, and publication. Blocks in Drupal are various “placing points” for each website in SPARSE: it provides for a template to display content on the website. We set up the physical layout of SPARSE through utilizing the Views module, discussed later, in addition to manipulating what in the given blocks. Because we utilized the Nexus theme, it came equipped with a core set of blocks which were convenient for setting up SPARSE. Every page of content is dubbed as a “node” in Drupal, the node number is the unique identifier for content. In order to implement the majority of functionality on the website, we downloaded and used various “modules,” plugins which extend the Drupal core. Modules are a collection of files containing some functionality, primarily written in PHP; a module is no different than an independently created PHP file which can drive multiple functionalities. For example, the implementation of the Views module enabled us to manipulate what content is displayed on a page, block, and more. Views effective creates database queries based on settings made in the Views user interface. The Taxonomy module is a Drupal core module which allows you to create “Vocabularies” that include “Terms”; taxonomical structures allowed us as developers to create content specific filtering and labeling metrics. Similar manipulations with modules ended up extending to many others which are discussed further in detail.

In the implementation of the people, projects, and publications pages, the "Views" module was installed and used, in conjunction with several other modules. What this module does is display all content of a specific content type, in a certain display format (page, block, content pane, etc.). So in order to display all people, projects, and publications on a certain page, these three content types had to first be created. In creating a content type, you must specify the input fields for that type. For the publications type, the "Biblio" module was installed and used, but renamed as "Publications" in order to be more user-friendly. In addition to this module's pre-installed fields, a custom field was added to it in order to create and manage the publication's relations with its corresponding projects, and affiliations (more on this later). For the "Project" content type, several fields that should be present across all projects were included. This includes things such as title, summary, image, research topics, and a url to a webpage. In order to keep the "People" type, simple, some basic fields were included like title, body, image. On the Projects and People types, there are fields for specifying whether that content is affiliated to the computer science department or the radiation oncology department, Note that these two types do not have any fields for specifying related publications. Similarly, there are no fields for specifying the affiliations of a publication.

At first, I tried using the relation module to manage the relationships between the three content types. This required creating a relation type, and explicitly adding each node to the relation. However, after doing this, the page started to show duplicate projects when it had multiple related publications. Similarly, duplicate publications were created when it had multiple authors and keywords (which is all of the time). Also, this method was not ideal for the researchers to update with. Next, I tried using the "corresponding node/entity references" modules. This means that you need to have a "publications" field on a project and a "project" field on a publication, but you would only need to update one field on one type, and the other field would be updated automatically. This is easier to maintain than the relation module, but after testing, I realized it did not work 100% of the time. Next, I tried using taxonomy terms. While this did work, I decided it would be a little too hard for the researchers to maintain, so I looked for other options.

In the end, to display the related publications of a project, and the related authors/ keywords of a publication, separate views were created and a contextual filter of a content id had to be placed, and configured, onto them. This means that the fields that the view displays will be filtered based off of content id, meaning for only a specific node. The view that is used to display related publications also needed to have a "content referencing" relationship, in order to show the citations of the publications that reference the specified project. This means that only one content type (publications) needs to hold a reference. After doing this, the next step was to find a way to display a view inside of a view. This was done by installing the "views field view" module, and the "entity views attachment" module. This allowed us to use a view as if it was a field, and display it inside of a view. After doing this, however, it was displayed in list format, and the text was too big. To fix this, I went into the style.css file, and manually made it display inline, and made the text smaller.

For the affiliations, I had almost as hard of a time. At first, I tried using the "flag", "button field", and "rules" modules. This would require creating two flags on each content type, one for a computer science affiliation, and one for a radiation oncology affiliation. Also, a block that displays two buttons, one for each affiliation, was created and displayed on the sidebar of every page. Next, I tried using rules to make it so that whenever a specific button was pressed, the page filters based off of the corresponding affiliation. However, a multitude of problems ensued. First of all, the up to date relation module turned out to have a bug. To fix this, I uninstalled it and installed an older version Then, it turned out that in order to use rules with the views module, I had to install the "views bulk operations" and "views field rules" module. After doing so, it took a while for me to realize that doing this would in fact filter the results as needed, but it would not display it on the specified view. In the end, it turned out to be a little easier than I thought. I had to add two term references to each type, one for each affiliation, and add them as filter criteria. I installed the "better exposed filters" module in order to be able to configure these filter criteria better, and display them more appropriately.

There were a lot more steps in between that I did not mention because it would take too long to explain and you probably wouldn't understand the drupal lingo, but I included all the major components/obstacles I faced. Throughout working on the site, there have been several errors/warnings that appeared that required us to search on the drupal forums for users who have relevant issues, and usually necessitated going into a file of a specific module and changing a few lines of code (usually just 2-3).

With that being said, the resources used were almost exclusively the drupal documentation, and the drupal forums.

Names, emails, and NetIDs

Oscar Hong

- ch311@duke.edu
- ch311

Srikar Pyda

- email: sp285@duke.edu
- NetID: sp285

Mohab Gabal

- email mg250@duke.edu
- NetID: mg250

Team number and name: Team 5 - jomo

Attributions (who worked on what)

Oscar Hong

- User registration & user profile set-up (under My Account)
- Adding news & showing up on homepage feed
- Logo design
- Homepage slideshow
- Adding project and publication content
- Who's online (homepage right sidebar, only viewable by admin)
- Administrator dashboard

- Administrator toolbar (hovering shows all options; enables fast access to any administrative resource on the website)
- 3D image viewer
- Elevator pitch & slideshow

Srikar Pyda

- Search functionality+assigning search weights
- Mission Statement
- Contact form
- Front page slide-show
- URL shortening (switched from displaying node numbers to a more representative description of the page)
- User settings/privacy
- Implementation of menu block module
- Offensive comment flagging
- Calendar functionality (still being worked on)
- People page (adding People content)
- Uploading content in general
- Taxonomy terminology utilization

Mohab Gabal

- Creating and configuring content types
- Structure of People, Projects, Publications pages
- Adding useful modules
- Publication creation via DOI & PubMed lookup
- Displaying related keywords and authors of publication
- Filtering based off keywords/authors
- Managing and displaying relations (Allowed for automatic populating of data)
- Implementing institutional filtering
- Implementing content filtering

Sources (a full list of sources) with links, which should also appear inline in the code