CONNECTED to RESEARCH

OREGON STATE UNIVERSITY

CS463 — SPRING 2016

SENIOR DESIGN PROJECT

10 June 2016

# Final Project Report Report & Documentation

*Authors:*
David WINKLER
Chris NGUYEN
Benny ZHAO

*Instructor:*
D. Kevin MCGRATH
*TA:*
Jon DODGE

**Abstract**

Research scientists need to stay up-to-date on funding opportunities to continue their research. This is a time consuming process which can be streamlined by developing a system to automatically present a list of funding opportunities. Connected to Research pulls data from multiple sources of funding and adds them to a database. Information is pulled from that database and presented to users in an understandable interface. Overall, the goal is to make researchers' lives easier by reducing times spent searching for funding.

CONTENTS

# I. Introduction

## A. Our Clients

Research scientists spend a lot of time finding information related to their research. They need to collaborate with other researchers to share their expertise, consult on difficult issues, and network. They also need to stay abreast of developments in their field so that they do not end up reinventing the wheel. And perhaps most importantly, they need to search for funding opportunities so that they can continue their research. With these considerations in mind, a team of research scientists at the Pacific Northwest National Laboratory has proposed a new web application, titled Connected to Research. Our project was requested by researchers at Pacific Northwest National Laboratory. During their careers as research scientists, they spend an excessive amount of time searching for research opportunities to continue their research.

The project was requested to make the researchers' work easier, by removing the necessity of searching for funding opportunities. Connected to Research connects funding opportunities to researchers so they don't have to.

Connected to research makes researchers' lives easier and more productive. By removing the necessity of searching, they have much more time to do the research they have chosen. They also have more time to apply to grants, which means they will hopefully gain more funding for their projects. This is the primary goal for PNNL.

Specifically, our clients were Carolyn Cramer, Nick Cramer, and Geoffrey Elliott. All three of them are research scientists at PNNL, who have a personal interest in our project's success. More then once during our client meetings, they would mention that they had recently searched through funding opportunities. They hope to present our project to PNNL to gain funding for further development sometime in June.

## B. Our Team

Our team was made up of David Winkler, Chris Nguyen, and Benny Zhao.

We all performed versatile roles to complete any tasks necessary. No one person performed in only one category, and no part of the project was completed by only one person. Most of the work was done during group meetings, where we all collaborated on the work we wanted to do that day.

In general, David spent a lot of time developing the database and filling it with test data. David defined most of the database tables and their relations to other tables. SQL queries used in some scripts were developed by him. David created the scripts for matching funding opportunities for user1, and the script for downloading funding agency images. He also wrote a lot of PHP scripts, and created the parser for the FedBizOpps XML files.

As for Benny Zhao, he started off with figuring out how the front-end worked with the back-end from the previous team's work. After learning how the structure of the website worked as a whole, he created the front-end interface of the funding web page. The process of development started from creating the filter column to the funding opportunities column and to the deadlines column. There was debugging involved with the style of the web interface to creating new user-friendly interactions. Then with the use of AngularJS, he could automate and populate the funding opportunities from the database David worked on. From time to time, he also worked on testing the interaction of the web application by running test SQL queries on the database side of the project.

Chris Nguyen spent half of his time on the front-end and the other half of his time on the back-end. He started by learning PHP to create scripts as well as help with the back-end and assisting his teammates on whatever they were stuck on. Then he started learning AngularJS and HTML to help with the front-end. He helped implement features of the web application. Chris created a script for the Grants.gov parser and helped debug PHP scripts as well as HTML and AngularJS files.

## C. Meetings, Communication, and Project Evolution

At the start of Winter term, our client decided that they would like to have bi-weekly meetings with our group. The first meeting we had was on January 11th, where our client wanted to discuss the deliverables that our group is required to complete for this quarter and any questions we may have pertaining to the project. During the meeting, we discussed about the upcoming deliverable deadlines and how to best plan for them.

During the second meeting that happened on January 25th, we discussed our current progress on the project, questions to clarify the design of the interface, and goals that still need to be reached. In the meeting, we explained how our parser works to our client and what we have so far on planning the interface design. The client informed us that they wanted a funding webpage that is able to sort the funding opportunities based on certain filter criteria. They also gave us general criteria of what filters we should create for the UI based on what our parser inputs into our database. What the client would like on the display of the website has definitely changed due to some constraints of what information we can grab from either Grants.gov or FedBizOpps.

Our client wanted us to create a filter for the funding opportunities which will split the view into three parts. One part is the left side where all of the filter buttons will be located, the center where it will display the funding opportunities based on the filter, and the right side where all of the funding opportunities' deadlines will be located.

In order to implement the filter, we had to create a similar design to the previous team in order to keep the design consistent. At first there was a challenge in making the filter display itself. But as we went through the code, we found there to be typos and one of the typos was the bug to why we could not see our filter. Once we got the filter column to show up. We had to implement the SQL queries step-by-step because if a SQL query didn't work properly, we would not be able to see our output. After we got a framework of the filter column, we needed to make it more tailored to what the user would want to see. Therefore we used nested arrays within our filter in order to be as detail as possible so that the user does not have to spend much time scrolling through the funding opportunities.

We continued meeting with our clients to get their feedback on how our project was progressing. They gave us multiple suggestions to improve the site. Most of their suggestions were cosmetic, related to how the data is displayed. One recommendation is to add a picture of the agency offering the opportunity to be shown in the corner of the opportunity data. They also want to be able to select one filter criteria from each section of the filter column. So for example, they could click 'FedBizOpps' to filter all FedBizOpps opportunities, then click on 'Department of Veterans Affairs' to filter all FedBizOpps opportunities that are offered by the Department of Veterans Affairs.

They also had other requests, like moving the location of hyperlinks from the 'Source' text to the 'Title' text, and improving the 'Share' button.

After we finished the beta release, our clients started asking us to improve the site design. Their first request was to allow users to select multiple filter categories from the filter panel on the left side of the webpage. Their next request was to embed images in the funding item titles, showing the agency offering the opportunity. Another request was to implement a peeking feature, where users could see some of the publications from the publications page without actually navigating to that page, and vice-versa for the funding page from publications.

## II. Project Requirements

### A. Original Requirements Document

Our original requirements document is listed here for convenience.

At the beginning, we separated our project into 3 major categories of work. The first category that needed to be completed was the one involving the database. Before we did anything, we needed to define the set of funding sources we would support. After meeting with our clients, we determined that the most effective selection would be to take information from FedBizOpps.gov and Grants.gov. With the combined data from both of these sources, we would get enough data to make our website functional.

To implement this feature, our clients gave us a simple, open-ended set of requirements. We divided the requirements into three categories according to the order in which we thought they needed to be done. The following paragraphs are a description of our requirements.

First, we need to define the sources of funding that will provide the data for our database. We need to define the information that they store to define the database tables. We need to define the database and a set of parsing scripts to fill them. We need to define how other Connected to Research components will need to interact with the funding component. We need to decide how best to parse information from the funding sources and write the scripts. Then we need to implement database accessor functions using the client's preferred language.

Second, we need to work on the system for interacting with user interests. Our clients did not have many requests for this part, so the requirements are sparse. Essentially, we must have some mechanism mapping the funding opportunities to user interests.

Third, we need a workable user interface. This may be the most important part because this project is still a proof-of-concept that will hopefully bloom into a fully-featured web application. We need to develop methods to display funding opportunities to the user, which will be done similar to the previous team's work. We will need to allow images embedded in the opportunity description. We will need to allow users to bookmark opportunities and display them in a chronological list. This will be important for users who need to be reminded of approaching deadlines. Otherwise, they would need to manually manage their own notification systems. Users will also need to 'dog-ear' opportunities separately from the bookmarks. This is for the case where researchers want to make one large sweep to find any opportunity that might apply to them, and then choose which to actually apply to through bookmarking. Users will also want to share opportunities with other users, a feature which will likely tie-in with the 'People' component of Connected to Research, where users can connect to other researchers through the web app.

During one of our meetings with the client, they informed us that they did not want us to create a system to match users with funding opportunities based on their interests. Instead, they will use their own tools. We will not need to process the user interests, but we will still keep functionality for users to record their interests.

CONNECTED to RESEARCH

# Funding Opportunities Component Requirements Document

Benny Zhao, Chris Nguyen, David Winkler
Team Majestic Turtles

# Abstract

This document contains the software requirements for the funding component of the web application *Connected to Research*. This project is our senior design project for the course CS461 at Oregon State University.

These requirements were developed with collaboration from our clients at Pacific Northwest National Laboratory. The funding component seeks to present researchers with funding opportunities based on their interests through a database populated with reliable sources of funding.

# Contents

# 1. Introduction

## I. Purpose

Research scientists need to stay up-to-date on funding opportunities to continue their research. This is a time consuming process which can be streamlined by developing a system to automatically present a list of funding opportunities.

## II. Scope

The component will pull data from multiple sources of funding and add them to a database. Information will be pulled from that database and presented to users in an understandable interface. Users can save opportunities and interact with them.

## III. Overview

This document serves as a list of the requirements pertaining to the funding component of the web application.

# 2. Description

## I. Function and purpose

There is going to be a function that takes the user's interest as input then outputs the opportunities that corresponds to their interests. The function will store the interests of the user into a list and compare them to a list of categories such as chemistry. After finding all of the corresponding categories and stored into a list, compare them to all opportunities that pertains to the categories then output the results onto the user's interface. The results can be stored into the user's opportunity table, a chronological list of opportunities.

The purpose of this function is to provide the user with opportunities that have some kind of correlation to them based off of their interests.

## II. Limitations, Questions and Issues

One limitation is not having a fully functioning product for our component to interface with. This will require forethought from our team to account for future needs for the funding component. For example, changes to the interests tagging system.

# 3. List of tasks to complete

I. Define a database to store funding information

a. Investigate the metadata from FedBizOpps.gov and Grants.gov to determine how best to extract data from them.
b. Define a list of funding sources
c. Research those sources and define common attributes to be stored
d. Determine how other components of *Connected to Research* will interact with the funding component
e. Define the structure of the database and develop a database schema
f. Research the best method of pulling data from funding sources
g. There may need to be multiple ways of pulling information
h. Implement the database using the client's preferred database system
i. Implement the database accessor functions using the client's preferred language (Because the component will need to be integrated with other components)

II. Develop methods for interacting with the user interests
a. Research the interests system of *Connected to Research* to determine how to apply interests to funding opportunities
b. Develop methods to search through the database and retrieve results related to the associated interest

III. Develop a workable user interface
a. Develop methods for displaying funding opportunities to the user, as shown in mockups by the previous team
b. Allow display of images embedded in the opportunity description
c. Allow users to bookmark opportunities and display them in a chronological list
d. Allow users to 'dog-ear' opportunities for later review
e. Allow users to share opportunities with other researchers through the site and through email

# 4. Results

The end result is to link researchers to funding opportunities with a functional user interface that collects reliable funding resources.

At presentation, we will present a working application. We will be able to retrieve data and present it in the user interface.

# 5. References

*Final Capstone Book*, stored in the documents section of this site, describes the overall usage of the site. It broadly outlines the requirements of the funding component.

*Carolyn Cramer MLIS Capstone Abstract* describes the motivation for the application.

https://github.com/tstevens10182/connected-to-research/tree/master/img/mockup

The previous Capstone group created mockups of the site design. It includes plans for the site as a whole and the funding component in particular.

## 6. Signatures

Carolyn Cramer

Nick Cramer

Geoff Elliot

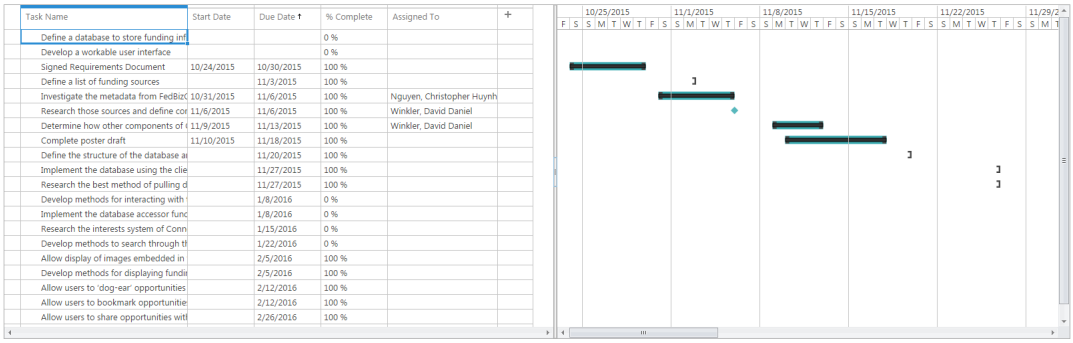



Fig. 1: Our original Gantt chart

### B. Gantt Chart

Although we forgot about the Gantt chart sometime during Winter, we have included our Gantt chart here to illustrate our project's early planning stage. Although we ignored our timetable, our project progressed in a very similar fashion to what we had planned.

We began with investigating the funding sources and the data that would need to be stored. Once we had decided what to get and where to get it, we defined the database tables and the XML parsers that filled them. We developed the user interface alongside the database and parsers to check our data. We essentially copied the Publications component's html, templates, and JavaScript and changed elements of the design as necessary.

When it came time to implement the interest matching system our clients had asked us to do, they told us that they would handle that part, because they have their own pattern matching system. So, a large part of our work was taken care of, and we started work on extra things our clients asked for.

### C. Requirements Changes

| Number | Requirement | What happened to it | Comments |
|---|---|---|---|
| 1.a | Investigate the metadata from FedBizOpps.gov and Grants.gov to determine how best to extract data from them. | After investigating those funding sources, we learned they publish their opportunities in XML files. We decided to write XML parsers to populate our database. | |
| 1.b | Define a list of funding sources | Our clients told us to focus on Grants.gov and FedBizOpps.gov, so we did not spend much time researching alternatives. | |
| 1.c | Research those sources and define common attributes to be stored | We created a list of attributes that would need to be stored, and translated those into a database design. | |
| 1.d | Determine how other components of Connected to Research will interact with the funding component | Other components can interact with the Funding component through a set of scripts we've created. | Each component has a different functionality intended by our client. Depending on the job of the other component, another groups' work could find our scripts helpful or not. |
| 1.e | Define the structure of the database and develop a database schema | After defining the attributes to be stored, we translated them into a database design. | |
| 1.f | Research the best method of pulling data from funding sources | After finding the funding sources' XML sources,we chose to use those. | |

| | | | |
|---|---|---|---|
| 1.g | There may need to be multiple ways of pulling information | There are weekly and nightly files from FedBizOpps, and we chose to focus more on the weekly file. | As forGrants.govXML files, they contained all their funding opportunities. |
| 1.h | Implement the database using the client's preferred database system | The previous team's implementation of the Publications component used-MySQL, so we extended their work. | |
| 1.i | Implement the database accessor functions using the client's preferred language (Because the component will need to be integrated with other components) | The client asked us to use PHP on the back-end, and JavaScript on the front-end. | |
| 2.a | Research the interests system of Connected to Research to determine how to apply interests to funding opportunities | As mentioned, the client asked us not to do this. | |
| 2.b | Develop methods to search through the database and retrieve results related to the associated interest | Same as above. | |
| 3.a | Develop methods for displaying funding opportunities to the user, as shown in mockups by the previous team | Due to data deficiency, we couldn't match their mockups exactly. However, we got close. | See the Appendix for image comparison between the mockup and our end result. TODO: Add appendix number? |
| 3.b | Allow display of images embedded in the opportunity description | We completed this task. | |
| 3.c | Allow users to bookmark opportunities and display them in a chronological list | We completed this task. | |
| 3.d | Allow users to dog-ear opportunities for later review | We completed this task. | |
| 3.e | Allow users to share opportunities with other researchers through the site and through email | We completed this task. | |

### *D. Original Design Document*

Not much changed between our design document and our implementation. The most obvious difference is in the interest tagging system. While we were prepared to tackle that challenge, we asked our clients how we should go about it. However, they told us that they didn't want us to do that, because they had their own, more sophisticated pattern matching system, and they didn't want us wasting our time on our own.

The user interface also changed slightly. We removed the ability to sort the funding opportunities, and instead list all opportunities in chronological order based on expiration date. We also implemented the ability to toggle multiple search categories.

We also didn't implement a page for the user's preferences and options, because we didn't need to work on preferences or interests.

The database and back-end remain mostly the same as our design document describes it. The ERD also matches our final implementation.


# Personalized Information Portal for Research Scientists Software Design Document

Sponsors:

**Carolyn Cramer**, **Nick Cramer**, and **Geoffrey Elliott**

of **Pacific Northwest National Laboratory**

Presented by Team 29, *The Majestic Turtles:*

**David Winkler**, **Benny Zhao**, and **Christopher Nguyen**

# Contents

# 1. INTRODUCTION

## 1.1 Purpose

This software design document describes the functionality and application design of the funding component of the Connected to Research web application. The web application is intended for those who need funding opportunities for research. Researchers shall use the funding component of the Connected to Research web app to discover new funding opportunities. This web application shall make researchers' search for funding simpler.

This software is being designed for researchers at the Pacific Northwest National Laboratory, but the final web application is intended to be used by researchers in any fields, anywhere.

## 1.2 Scope

This software design document shall provide details on three parts of the web application. It shall be composed of the front-end, the database management system, and the back-end database. It shall include details on how the front end interacts with the back-end database through the use of the database management system.

Researchers shall have the ability to interact with the web application. They shall be able to search for funding opportunities that pertains to them based on their interests and preferences. They shall also be able to view their funding opportunities as well as save them within the database.

## 1.3 Context

Connected to Research is a web application that aims to connect researchers to information relevant to their interests. This includes other researchers, publications, and funding opportunities.

The funding component of Connected to Research is just a portion of the final web application. It serves to interact with the user, where the majority of them will be researchers. The component itself constantly updates itself with more up-to-date funding opportunities and then stores the information into the database. The system then narrows the search based on researchers' input of their interests and preferences. The system will store funding opportunities of different types, and present researchers with results relevant to their interests and preferences. This is how the system aims to simplify their search for funding.

Users shall be able to save the funding opportunities they have already viewed as to optimize search efficiency with respect to the researchers' time. This information can later be used for analysis, planning and/or improvement of the software design.

The database will act as the model in a model-view-controller (MVC) setting, where it stores the necessary information and updates on a scheduled basis. Whereas the front-end portion will serve as the view, displaying relevant information pertaining to the researcher's criteria. Lastly, the back-end is the controller, manipulating the incoming data the user inputs into the system and storing it into the database.

## 1.4 Summary

This document refers to the funding component of Connected to Research. It includes the application design which shall detail the architectural design of the three parts of the web application. These three parts include the front end, the database management system, and the back-end database. The web application will be designed similar to that of a Model-View-Controller (MVC).

This document shall also include the data design of the database which includes all of the database tables and database attributes. Along with the details of the database tables and database attributes.

This document shall also include the design viewpoints of the application design.

# 2. DEFINITIONS

**Back-end**

The databases and data processing components of the system. It launches programs in response to the front-end's requests and operations.

**Component**

Component refers to a self-contained portion of the Connected to Research web App. It provides separate functionality that is related to (but not dependent on) other components of the application.

**Controller**

Represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

**Database**

An organized collection of data. It is the collection of schemas, tables, queries, reports, views, and other objects.

**Database Attributes**

A single data item related to a database object.

**Database Management System**

Software that handles the storage, retrieval, and updating of data in a computer system.

**Database Table**

A collection of related data held in a structured format within a database.

**FedBizOpps.gov**

A funding aggregator created and maintained by the US government. It contains all federal procurement opportunities

**Front-end**

The view or display of information to the user for interacting with the system.

**Funding Component**

Component of Connected to Research web app that connects researchers to funding opportunities.

**Grants.gov**

A unified website for interaction between grant applicants and the U.S. Federal agencies that manage grant funds.

**Model**

The representation of the logical structure of data in a software application. This object model does not contain any information about the user interface.

**MVC**

A software architectural pattern which separates an application into three main components: the model, the view, and the controller.

**PHP**

A server-side scripting language designed for web development, but also used as a general-purpose programming language.

**View**

A collection of classes representing the elements in the user interface. Everything that the user can see and respond to on the screen is part of the view.

**Web App**

In computing, a web application or web app is a client-server software application in which the client (or user interface) runs in a web browser.

**XML**

Extensible Markup Language. XML is a markup language that is designed to allow structured data to be stored in files. Many websites publish data in XML formatted files. XML can be *well-formed* if it conforms to the XML standard. XML can be *valid* if it conforms to a schema.

# 3. APPLICATION DESIGN

## 3.1 Introduction

The funding component of the Connected to Research web app will be made up of three interrelated design entities. There will be a front-end presented to users, a database for storing all information, and a back-end to populate the database and connect it to the front-end. The web app will allow users to register for an account. At some point, users will enter their areas of interest and the web app will record them. Their interests will be recorded in the database under a table related to user interests. The user interests will be used to match funding opportunities to users. The user and interest tagging system is not the primary focus of the funding component. However, they will need to be implemented to support the implementation and testing of the funding component.

The front-end of the component refers to the visible user interface that users will interact with. The front end will include a search and browse page. The search page will allow users to find any and all funding opportunities related to their search terms. The browse page will present users with opportunities selected by the component based on their interests.

The database refers to the data storage mechanism used for the user, interest, and funding data. The database will be populated by the back-end scripts that will pull data from funding information sources. For

the funding component, the database will contain data relating to funding sources, user information, and interests. Other components will maintain separate tables in the database.

The back-end refers to the scripts necessary to implement features in the front-end and database. The front-end and database will need functionality implemented in the back-end. Funding opportunities will be periodically updated from online sources. New opportunities will be added, and interests will be associated to those opportunities. Users whose interests match those associated with a funding opportunity will be notified of the opportunity.

## 3.2 Front-End

The design of the front-end will consist of using the Bootstrap web framework as a design foundation. Where the framework already contains pre-built components for better efficiency of web development and usability for the user/developer.

In terms of designing the layout for the user interface, there is already a proto-type design created by our sponsor. But our main focus is on the completion of the functionality of the funding component. Therefore, the front-end portion will be the lowest priority. Furthermore, the approach to a comfortable user interface will be simple enough for researchers to utilize, thus eliminating time required to learn the user interface or searching for capabilities on finding funding.

The layout design for the funding component will consist of three main columns. The first column (left-most column) contains filters for **agencies**, **funding**, and **sort**. The capability of this column is to let the user (researcher) be able to filter funding opportunity based on which **agency** it is originated from, the amount of **funding** available, and **sorting** by either date the funding started, time length or duration of funding, and PNNL projects associated with funding.

The second column (center column) will contain the funding opportunities based on filters and interests of the user. Displayed within each funding opportunity is the title/name of the funding, the agency originated from, and the current amount of PNNL projects associated with the particular agency. It also includes the PNNL contact information, concept paper due date, invitation for full proposal date, and the proposal due date. Lastly a description of the funding opportunity will be within the same bootstrap container.

In the third column (right-most column) will be research events and deadlines previously, currently, and upcoming. The user will be able to click on the hyperlinks of the deadline to navigate to that funding opportunity.

Within the user's account, there will be an option for the user to input their interest. Where the interest will be used for narrowing the search of funding opportunities that pertain to the user. The interest table will store a list of strings that will correspond to a list of strings in the funding opportunity table.

Each funding opportunity has an associated category. For example, Grants.gov has CFDA numbers and funding opportunity categories. Meanwhile, FedBizOpps.gov has NAICS codes and classification codes. Both of these correspond to different interested parties. We need some way to map these to more general interests, like 'Biology'. To do this, we will make use of an intermediate table, called Interests. The Interests table will have a Category, which is the general interest. It will have a String field, which is a list of strings of NAICS and classification codes, CFDA numbers, and any other descriptors that may come along with new sources.

## 3.3 Database

The database will be composed of tables to store the funding opportunities. There will be a table for each source of funding defined, because they may contain different fields that will need to be kept track of. For example, FedBizOpps.gov and Grants.gov feature different classification codes for funding opportunities, such as NAICS codes and CFDA numbers, which may overlap and need to be kept distinct. To avoid having unnecessary fields duplicated in each funding source table, there will be a parent table from which

the source table are derived. The parent table will include the shared fields between the funding sources, such as opportunity number, title, post date, and response date.

The database will be populated by the back-end using scheduled scripts. The front-end will communicate with the back-end to service user requests for information. The front-end will access a list of funding opportunities related to a user's interests using a back-end service routine. The back-end will use the user's interests, obtained from the user's corresponding table, to access up to five interest entries.

### 3.4 Back-End

For the back-end, there will be XML files downloaded daily or weekly from funding sources: FedBizOpps.gov and Grants.gov. that will be parsed into the database. A script will be implemented to pull data from these funding sources and into the database. Within this script will be a PHP XML parser that takes in a XML file as input. The PHP XML parser will match all of the elements within the XML file to corresponding attributes within the database. Then it will perform SQL insertion with PostgreSQL's *pg_query()* function to insert the elements' text content to its corresponding database tables.

Both Grants.gov and FedBizOpps.gov publish periodic XML files that we can parse. The XML files from Grants.gov are both well-formed and valid. They also make available a description of the XML files and the data contained in them. We can use this document to map data from the XML file to the database. The XML published by FedBizOpps.gov comes in the form of both weekly and nightly XML dumps. The weekly file is well-formed and valid, but will probably update too infrequently for our purposes. It is also a very large file. The weekly dump as of writing is over 1 GB. The nightly dumps are smaller, but are not well-formed. However, they are still simple to parse. Using these two XML sources, we should be able to pull all the data we need for our database.

## 4. DATA DESIGN

### 4.1 Data Description

The ERD diagram below displays the general overview of the data design. Currently, we plan to support two funding sources. These will be FedBizOpps.gov and Grants.gov. With our design, it will be simple to add support for more funding sources. Any source that may need to be supported in the future will support, at minimum, the fields contained in the Funding Opportunity table. This is not an unreasonable assumption, as any opportunity that does not include that data does not have enough data to describe the opportunity adequately.

The tables derived from the general Funding Opportunity table will contain any additional information specific to that funding source that is not contained within the Funding Opportunity table. The Funding Opportunity will contain a field storing the interests associated with a funding opportunity. There may be any number of interests associated with a given opportunity, for example, their may be an interdisciplinary opportunity for biologists and chemists.

There will be a table for storing each user's information. Each user will be able to list up to five interests related to funding opportunities. The interests in the user table will be related to a category in the interests table. For example, a user might list their interest as 'Toxicology'. Then their interest will map to an entry in the Interest table with a 'Toxicologist' category. The String entry will be a list of substring, each of which corresponds to a, interest string in the Funding Opportunity.

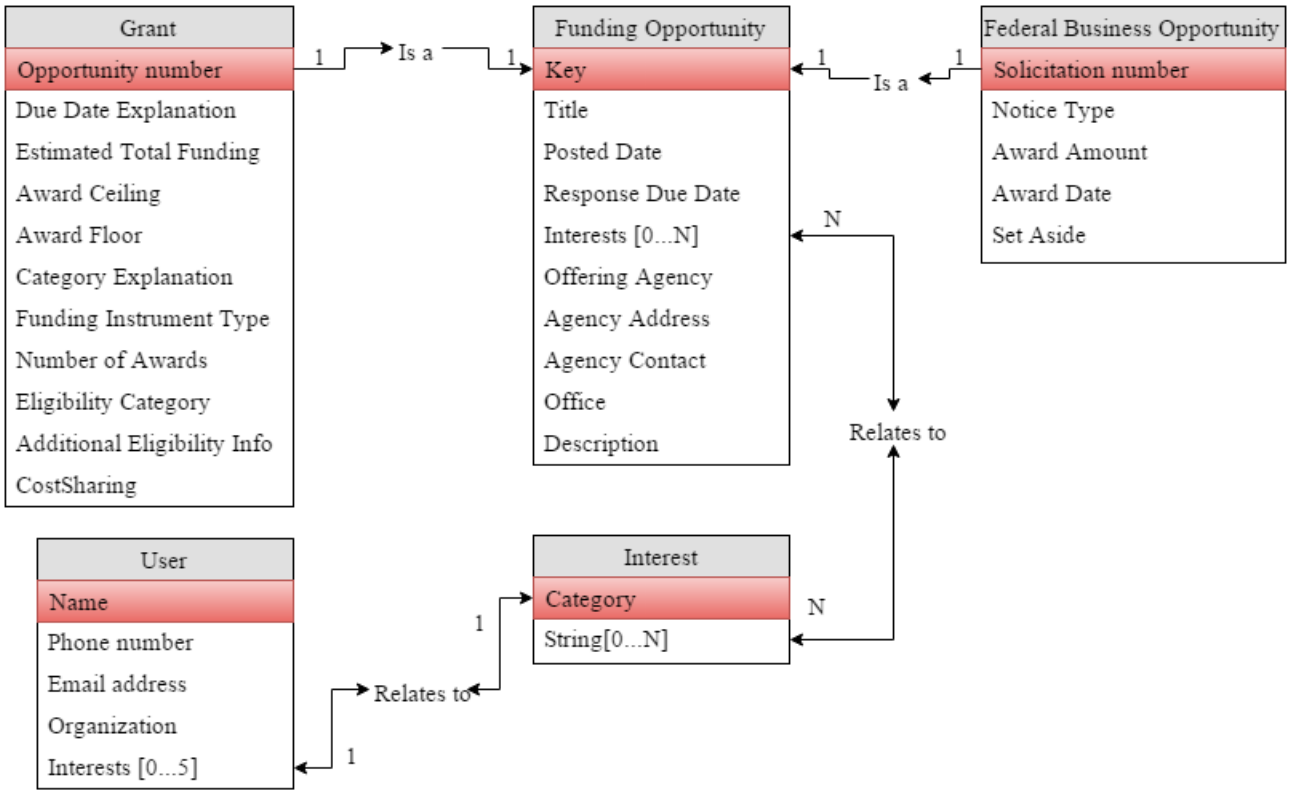



**Figure 1 - ERD Diagram: Data Design**

## 4.2 Data Dictionary

Below is a dictionary of the terms used in the data design diagram.

**Grant**: A funding opportunity from Grants.gov.

**Opportunity Number**: A unique identifier for each grant offered by Grants.gov.

**Due Date Explanation:** Further elaboration on the due date.

**Estimated Total Funding:** An estimate of the total funding available.

**Award Ceiling:** Maximum funding available for specific funding opportunity.

**Award Floor:** Minimum funding available for specific funding opportunity.

**Category Explanation:** Explanation of the funding category.

**Funding Instrument Type:** The type of funding agreement opportunity uses.

**Number of Awards:** Total number

**Eligibility Category:** Category of applicant eligible for this opportunity.

**Additional Eligibility Info:** Additional requirement criteria for the funding opportunity.

**Cost Sharing:** Whether or not funding will be shared.

**Funding Opportunity:** A funding opportunity from FedBizOpps.gov

**Key:** A unique identification number of the funding opportunity.

**Title:** The headlines of the funding opportunity.

**Posted Date:** The date when the funding opportunity is first advertised.

**Response Due Date:** The deadline date for the funding opportunity.

**Offering Agency:** The name of the agency proposing the funding opportunity.

**Agency Address:** The address of the agency proposing the funding opportunity.

**Agency Contact:** The contact information of the agency proposing the funding opportunity.

**Office:** The location of the agency proposing the funding opportunity.

**Interests**: A string of category information which maps to a category in the interests table.

**Description:** The description of the funding opportunity.

**Federal Business Opportunity:** A funding opportunity from FedBizOpps.gov

**Solicitation Number:** A government number that identifies a certain request for quote.

**Notice Type:** The type of notice that the funding pertains to (award notice, combined synopsis/solicitation).

**Award Amount:** The amount of available award.

**Award Date:** The date the award is posted.

**Set Aside:** The proportion of funds reserved for a particular service.

**Interest:** Entity that holds all interests of users and connects that with funding opportunity.

**Category:** Interest category (e.g. biology, toxicology, etc.)

**String:** A list of strings that might appear in funding opportunity.

**User:** Entity that holds the user's information.

**Name:** Name of the user.

**Phone Number:** Phone Number of the user.

**Email Address:** Email Address of the user.

**Organization:** Organization the user belongs to.

**Interests:** User's research interests.

### *E. Original Tech Review Document*

For pulling the data from the funding sources, we were going to use the FBOpen API, developed by 18f. Instead, we chose to use the XML parsing approach. We chose not to use FBOpen because we didn't want to depend on an outside source for our project. XML parsing is simple enough that we could make our own system that works well for our purposes.

For the database, we chose to use MySQL instead of PostgreSQL, because the previous Publications component of the project was already using MySQL.

To develop the user interface, we continued to use bootstrap, because the rest of the project uses bootstrap. We also continued using PHP and JavaScript, because the rest of the project also used those languages.

-

# CONNECTED to RESEARCH

Benny Zhao, Chris Nguyen, David Winkler

Team 29, *The Majestic Turtles*

-


## Abstract

Our goal is to create a system that improves the way researchers research funding opportunities. the system will pull data from funding sources, add opportunity data to a database, and present that data to users. The system is being developed as part of the *Connected to Research* project, with collaboration from researchers at Pacific Northwest National Laboratory.

We have identified 4 areas where we will have a choice between multiple technologies:

1. A way to pull data from funding opportunity sources.
2. A database system to store that data.
3. A way to use the database to present information to interested parties.
4. An agreed-upon language to increase maintainability.

This document will describe each of those parts and the technology our team can use to complete them.

-

## Methods of Retrieving Data (Task I.a or I.f):

**HTML Method**: To extract opportunity data, we could search through the website's HTML files. This would involve using web scraping. We would need to implement a script that searches through the HTML document and extracts data from the website using tags. This would involve iterating through each opportunity on the website, reading through the HTML file until we reach an identified tag, and then associating the data from that field and inserting it into our database. For example, on FedBizOps.gov, the solicitation number of an opportunity is listed in a div as follows:

*<div id="dnf_class_values_procurement_notice__solicitation_number__widget"...*

This approach has many flaws. First of all, it would be very inefficient. We would need to load many webpages and generate many requests. The script we develop would also need to read through whole HTML files and match patterns. If the website we pull data from ever decides to change their website format, we would need to update our script. Clearly, this option should only be our last resort.

*Pros*: All of the information available to the site users will be accessible through our script.

*Cons*: It is very inefficient and susceptible to change. We have no guarantee that the HTML design will stay the same for long. The scripts will need to load many webpages and read through them all to extract the relevant data. This will involve many requests and a lot of processing. We will need to study the HTML layout of the site and investigate ways of iterating through the website using our chosen programming language.

**XML Method**: Another method would be reading through the XML file of each resource. Both FedBizOps.gov and grants.gov publish XML documents that contain information about funding opportunities. Using these files, we can use a script that automatically downloads the file and extracts data from it, pushing the formatted data to our database. For example, the solicitation number of an opportunity is listed in an XML file as follows:

<SOLNBR> ...

This approach is similar to the HTML method. However, this method is much preferred. Using this method we will have access to all available information in a clear and legible interface. Since the XML file is designed to contain only relevant information, we can pull data from it significantly faster than we could from any HTML page. The XML document data tags will likely not change for a long time, and if they do, such a change is likely to be announced. Unlike a change to the website HTML form. This way, we will be able to predict and prepare for changes before they occur.

As for getting the document, there are two types of XML documents. There is a weekly document, which contains all of the current funding opportunities, and a nightly document, which contains new and updated opportunities. If we use the weekly file, the one file will take longer to parse, but will likely be faster than parsing nightly files. However, the data gleaned from it will become out-of-date faster. If we use the nightly files, our data will be more accurate, but the files will need to be parsed daily.

Grants.gov publishes their entire database as a daily XML file. We could choose to parse the file daily, or on some other schedule.

-

*Pros*: XML files are designed for this. Both PHP and Javascript have supported methods for parsing an XML file. If necessary, there are many open source examples that we can adapt for our purposes. The XML files are an officially supported method to pull data from our funding sources. The fields in the XML files are not likely to change, even if the website design changes.

*Cons*: we will need to develop and maintain the scripts ourselves. We will need to inspect the XML files to determine what fields to extract. The XML parsing scripts will need to be scheduled, and we will need to decide when they should be run to keep our database up-to-date. We will need to determine which XML files to parse, and at what times.

**FBOpen Method**: 18f is a US government agency dedicated to making digital products for government organizations. One of their products is called FBOpen, an open-source project that allows users to search for funding opportunity data. Though it's named for FedBizOpps, it also allows users to search Grants.gov, with plans to incorporate more sources in the future.

Using this API, we can make use of an official API created by the government to access government data. We can use this API to avoid duplicating code and reinventing the wheel, and to improve the maintainability of the system.

The FBOpen implementation is open source and can be modified to suit our needs. According to its website, it is a "RESTful search API, hosted by the awesome api.data.gov, and backed by the search index server, Solr." The API can be used to query formatted data for all opportunities, or to retrieve information for a particular opportunity. If we choose to, we could store only the opportunity numbers and tags in our database, and retrieve all other information directly from FBOpen. That would give users the most up-to-date information about the funding opportunities.

*Pros*: FBOpen is developed by a department of the US government. FBOpen queries the FedBizOps and Grants.gov sources directly. Using FBOpen, our database might only need to store opportunity IDs, and associated data can be obtained through FBOpen. The project claims it will add support for more funding sources. FBOpen is free and open source, so it can be modified to suit our needs.

*Cons*: it might be better suited for individual queries, not to populate our whole database. We are limited to 10,000 queries per day, though we are encouraged to request more. Development is ongoing, and some features are not yet supported. Other features may change in the future. Our scripts may need to be updated when the FBOpen API changes.

**Our Selection**: We plan to use the FBOpen API. We will consider the XML parsing option as a way to populate our database initially. Parsing XML will probably be faster and more straightforward than querying every entry through FBOpen. Afterwards, we will keep our database up-to-date using the FBOpen API. Using this approach, we will be able to reduce the size of our database while also improving reliability. There may be a speed penalty due to the time it takes to query the FBOpen API. We believe that the penalty will be offset by the storage saved and increased efficiency of the component.

-

## Determine the database system to use (Task I.e):

MySQL: This is an open-source relational database management system. The software's source code was developed and distributed by a Swedish company called MySQL AB under the terms of GNU General Public License. MySQL is now owned by Oracle Corporation. This was the first database management system that we were going to use. The reason for this is because we were all familiar with MySQL and it is open source. MySQL is supported on most operating systems such as Windows, Max OS X, and Linux. We will not be using MySQL due to our client's request of PostgreSQL.

*Pros*: MySQL is well known, so it is simple to manage our data. Scaling the project with MySQL would not be a problem due to simplicity which can lead to more functionality and/or automation testing. Doing simple read heavy operation is an ideal use of MySQL compared to other database management systems due to simplicity of MySQL calls rather than doing the same operation with complex calls that would slow down performance. MySQL would be useful in cases where there is a continuation of the project. In the most recent versions of MySQL, there are JSON features that can be used. This is a benefit because this project may require JSON support. MySQL also can be run from using the command line.

*Cons*: A disadvantage of MySQL would be that it does not provide as much features or potential functionality of PostgreSQL. The main disadvantage of MySQL is that it does not scale well with the project's performance as the project scales in size and functionality. This project may potentially deal with a huge volume of data where there may be numerous operations waiting to be processed. The concern is with poor performance as the project demands greater and greater volume of data and operations.

**Microsoft SQL Server**: This is a relational database management system developed by Microsoft. The most recent release being Microsoft SQL Server 2014 version 12, and ongoing with Microsoft SQL Server 2016 version 13 being released in 2016.

*Pros*: Microsoft SQL Server is known to have high performance since its 2012 product due to its support for performance optimization. Manageability is relatively simple to perform giving us flexibility. It has the ability of scale with the project's database as well as having a hybrid of a disaster recovery and backup tools.

*Cons*: Microsoft SQL Server is not open source which means we must buy a license in order to use it and it is expensive. A big problem with this database system is that it is not portable which means it can not be used in any operating system but Windows. Another disadvantage is that Microsoft SQL Server is graphical user interface (GUI) based. The reason why being completely GUI based is a disadvantage is because it does not perform well with low-bandwidth or high-latency connections. External language binding is difficult with Microsoft SQL Server as it may have different requirements such as creating classes to store the data you are querying or install extra drivers.

**PostgreSQL**: This is an object-relational database management system. PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors. This is free and open source software with a permissive free-software license. Our sponsor requires our team to use PostgreSQL for the database hosting. A great reason to use PostgreSQL is because it supports standard SQL and our sponsor is familiar with it.

-

*Pros*: PostgreSQL supports advanced SQL functionality as well as support for a variety of data types, such as arrays and user defined types. PostgreSQL also does performance optimization, like Microsoft SQL Server. PostgreSQL can be driven completely from the command line, like MySQL. A big advantage of PostgreSQL compared to MySQL and Microsoft SQL Server is its external language binding. Its external API, "libpq" is easily implemented to connect and be used from programming environments. PostgreSQL provides absolute data integrity and reliability which is crucial for this project in protecting users' data and information.

*Cons:* With PostgreSQL, due to having more complex functionality than MySQL, may potentially be less performant. An example of this would be simple read operation where MySQL could use a simple SQL command to do the work while PostgreSQL uses a more complex SQL command to do the same operation. This is the reason why systems that utilizes fast read operation should not use PostgreSQL.

**Our Selection:** We will be using PostgreSQL. Our clients chose PostgreSQL for us. They probably chose to go with PostgreSQL because of its extended functionality and the simplicity of connecting to it from programming environments.  Another reason is because PostgreSQL provides absolute data integrity and reliability which is needed for our project. Lastly PostgreSQL was requested to be implemented by the client.

-

## Different way to present the data (Task III):

**Bootstrap Method**: Our client recommends our team to use the Bootstrap framework as a way to display the aggregated funding resources to the user. As a team, we have decided that this will be our main method to implementing a comfortable front-end user interface for researchers to search for funding opportunities based on their interests. A strong reason for our team to use Bootstrap is because it is a popular and well-developed framework, yielding a sufficient amount of documentation if we require clarifications.

This approach is feasible because there is enough documentation and there is simple bootstrap snippets of code already implemented from the previous senior design group. Our tasks is just to develop more from the last group and essentially add-on more specific features.

*Pros*: Has responsive design for all devices. Has a grid layout for organizing the display of different resources on a single webpage. Has pre-processor for SASS and LESS.

*Cons*: Only supports limited amount of browsers compared to Foundation (Bootstrap supports only Firefox, Chrome, Safari, and IE8+).

Road-Map:

- Create a bookmarking feature for user, if they are interested in saving a funding opportunity.
    - Be able to save the bookmarks possibly in a database table so that it could send sorted information back to the user in chronological view.
- Create and test different displaying options that the user could possibly interact with.
- Create a share option, if user would like to share funding opportunity (My Research Circle, Researcher, or via Email Address)

**Foundation Method**: Foundation is an front-end framework made by the company ZURB and is our alternative. Most likely, we won't be using Foundation as our main way to present data since our client would like us to use Bootstrap to embellish the front-end. Also the fact that there is already small snippets of existing Bootstrap code from the senior capstone group before us, it is more advantageous to continue developing using the same framework. Another reason we picked Foundation as an alternative is due to it being the second most popular and well-developed framework beside Bootstrap. Also large companies utilize Foundation, which gives the framework itself more credibility.

This approach would also be feasible since it is one of the top web frameworks next to Bootstrap being the most popular. But our focus is moreover on the Bootstrap framework. As to why this framework is chosen as our alternative is due to the sufficient amount of documentation it has, it's credibility that the company has been around for a while and it supports more specific mobile devices.

*Pros*: Also has a grid layout template for organizing different sections of a webpage. Supports Chrome, Firefox, Safari, IE9+, iOS, Android, Windows Phone 7+.

*Cons*: Only has pre-processor for SASS compared to Bootstrap.

**Zimit Method**: A second alternative to Bootstrap we found was Zimit, which is also a front-end framework that is open source. The reason to picking Zimit as our second alternative is for it's capability in building prototype HTML5 websites with responsive features. Zimit also is light-weight, where it can

-

compile and minify files to smaller sizes for faster loading. It also has a unified style that can assist in creating a light-weight user friendly interface. Plus the framework supports the more popular web browsers such as Chrome, Firefox, Opera, Safari and Internet Explorer.

This approach is not as feasible as the two mentioned above due to the fact that it is fairly new compared to Bootstrap and Foundation. But it is still a good alternative if we choose to write code for testing and prototyping in HTML5. The documentation is fairly limited since this is a new open-source project.

*Pros*: Light-weight, based on modular and scalable framework. Has design skeletons and a suite of stylized components.

*Cons*: Only has pre-processor for LESS and not SASS compared to Bootstrap.

**Our Selection**: We plan on using Bootstrap as our way to present the data to the user. The reason behind to using Bootstrap is mainly due to the popularity and sufficient level of documentation it can provide to our team. Also our client pushes us to continue developing on the existing Bootstrap code from the previous group. Nonetheless, it seems that Bootstrap supports SASS and LESS pre-processors , plus it has a well responsive design.

-

## Determine the languages to use (I.i):

**PHP**: PHP is a server side scripting language for delivering dynamic content in HTML pages. PHP has the advantage of having a very large codebase and many tutorials. Many open source projects are written in PHP. PHP is very widely used and supported by every major web browser and operating system.

PHP executes on the server, not in the client like JavaScript. PHP can be used to generate HTML files that service requests, but cannot be used to create interactive web pages. For that, JavaScript is necessary. Because of this limitation, PHP is usually used for server-side scripting, while JavaScript is used for interactive elements in HTML pages.

PHP was initially developed in 1995, and has become widely adopted. Due to its age and popularity, it has grown to include a wide variety of functionalities. PHP can be used to generate HTML pages, or it can be used to run scripts to process data. We can use PHP to parse the XML documents mentioned above. PHP also allows either object oriented or procedural programming. PHP can parse an XML file and insert the resulting data into a database.PHP is also compatible with Postgres, our chosen database system.

*Pros*: PHP has a long history. It can be used in many places. It has support for many database systems. It can be used to parse XML files or generate HTML pages. It can be used on most web browsers. It has a large codebase with many supported features.

*Cons*: It can't be used to create interactive web pages. PHP is designed for server-side web development, not client-side development. PHP is less often used for creating dynamic web pages than it used to be.

**JavaScript**: This is a high level, dynamic, untyped, and interpreted programming language most commonly used for the frontend. JavaScript is one of the three essential technologies of the World Wide Web (WWW) content production where a majority of websites utilize and support it. This is the programming language that all of the interactions with the user interface will be programmed in. We will be using JavaScript for the frontend.

*Pros*: Applications developed by JavaScript are reliably responsive. Task processes complete almost instantly, which helps with memory management since they don't have to be processed in the website's server and get sent back to the end user, which consumes bandwidth. JavaScript utilizes JSON which is used and seen as an object. The benefit of JSON is that it is more compact than XML and can easily be loaded into JavaScript for use. JSON's processing speed is fast and does not need parsing in comparison to XML since everything inside of it is seen as an object and can be called and used like a variable. A big advantage with using JavaScript is that there are many third party add-ons that extends JavaScript's functionality.

*Cons*: Programming on the client side can potentially be difficult due to the possible concurrency issues. Since this project is based on users searching the website which uses the database to obtain information, the frontend and backend must be able to work together synchronously. There are security issues with JavaScript. The issue is once JavaScript snippets are appended onto the web page they execute immediately on the client servers which can potentially be used to exploit the user's system. Rendering JavaScript may vary depending on the layout engine. This is a small issue and is becoming a smaller issue with the latest versions of JavaScript implementing a universal standard for rendering.

-

**AngularJS**: AngularJS is also known to be based off of JavaScript, but more importantly it is a JavaScript framework that uses directives to extend HTML and binds data with the use of expressions. AngularJS is known to be one of the more popular front-end frameworks used with HTML. It has a large community and is backed by Google, which gives the framework itself credibility. The advantages to using AngularJS is that it helps categorize the application into model-view-controller (MVC) components and the most notable feature is it's two-way data binding.

*Pros:* The top advantage of AngularJS is it's two-way data binding, where any changes on the view will be immediately reflected on the model and vice versa. This feature provides an instant projection of the model, where testing is simpler because it is easy to test the controller alone without the view dependency. Other pros to using AngularJS are it is fast in development once familiar with, good with applications that deal heavily with interaction of client side code and less code is needed for the same results.

*Cons:* The downside to AngularJS is that the basics are simple, but the learning curve gets harder. It is also hard to debug scopes within AngularJS and the directives can be hard to use.

**Our Selection:** We will be using PHP on the backend, and JavaScript on the front end. This choice was made for us by our clients, but we would likely have made the same decision. PHP is well-suited for server-side scripting, and JavaScript is well-suited for interactive web pages. We can use both in their separate spheres of operation to create an efficient and attractive web site. PHP and JavaScript are also compatible with each other. AngularJS was used by the previous capstone group, and we will be extending their codebase. Therefore we will likely have to make use of AngularJS in our code.

# CONNECTED to RESEARCH

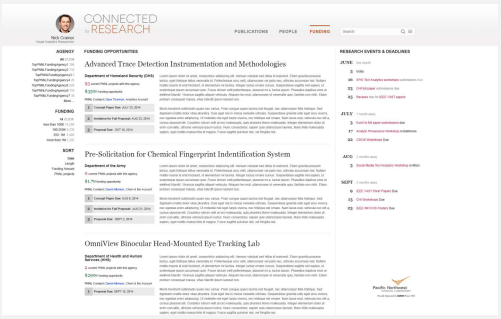
## Motivation

Research scientists need to keep up-to-date with advancements through reading publications, writing proposals and obtain funding for their work, and finding experts in their field to collaborate with.

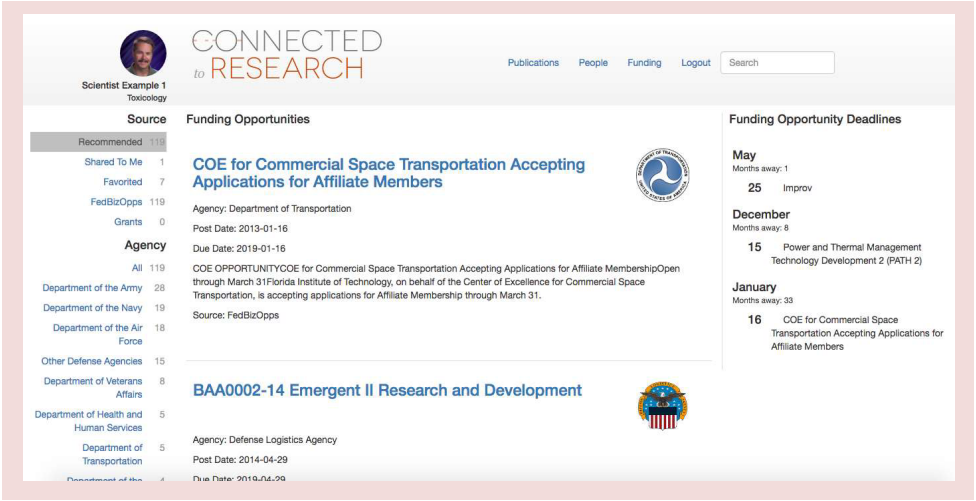
It is time consuming to comb through all of the different online information sources for publications and funding and it can also be challenging to find other experts in their field to collaborate with. Our goal is to help mitigate these problems.

Connected to Research is a web app that presents researchers with relevant publications, people, and funding opportunities with very little effort. It does this by storing a user's interests and utilizes them to make recommendations.

Connected to Research will promote collaboration and research by presenting a unified web interface for researchers, so they will no longer need to manage multiple websites and search manually.




*Project sponsor's mockup for funding component*




## Project Overview

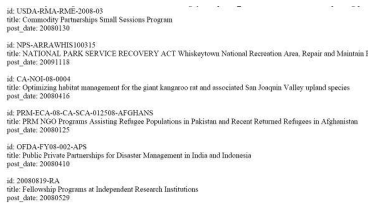
Connected to Research aims to make researchers' lives easier by connecting them to relevant publications, funding opportunities, and experts in their field with very little effort.

Our research team worked on the funding component of the Connected to Research tool which will:

- Gather funding data from Grants.gov and FedBizOpps.gov
- Recommend opportunities based on researchers' interests
- Let researchers spend more time researching
- Allow the researchers to facet and filter through different funding opportunities.

## Solution Implemented

- Researchers no longer need to search in various sources for funding opportunities, the system pushes these opportunities to the user based on their interests.
- AngularJS to create the visuals and interactions on the screen.
- PHP to obtain data from websites (XML) and store it into a database.
- MySQL to manage data and get information from the database to the user.
- PHP scripts to return JSON encoded data about funding opportunities.




*Output from Grants.gov XML parser*

The Majestic Turtles (Team 29)

TEAM MEMBERS
David Winkler (winkleda@oregonstate.edu)
Benny Zhao (zhaobe@oregonstate.edu)
Chris Nguyen (nguyenc5@oregonstate.edu)

PROJECT ADVISOR
Kevin McGrath (dmcgrath@eecs.oregonstate.edu)
Jon Dodge (dodgej@oregonstate.edu)

PROJECT SPONSOR
Carolyn Cramer (carolyn.cramer@pnnl.gov)
Nick Cramer (nick.cramer@pnnl.gov)
Geoffrey Elliot (geoff.elliott@pnnl.gov)

## Deliverables

- An automated system to gather funding opportunity information and publication venue information.
- A database to house funding opportunity and publication venue information updates gathered by the automated system.
- A web user interface that extends an existing Connected to Research UI which displays funding opportunity and publication venue information.

Pacific Northwest NATIONAL LABORATORY

Oregon State UNIVERSITY

# IV. Blog Posts

During the first quarter, we made regular blog posts. During the second quarter and beyond, we often forgot to log our work.

## A. October Posts

*1) Tuesday (October 6, 2015):*

**What We Did:** On this day, CS 461's lecture ended earlier than usual because each group had found their group members. Our group decided to find a place to meet right after and as our meeting progressed, we contacted our client via email for an official meeting with them. Our client was busy on that week's Thursday, which was October 8th. Instead, we decided to meet on Tuesday of the following week, October 13.

**How We Did It:** We met after class and got to greet each other as to learn each member's background in Computer Science. We initiated contact by sending an email in an attempt to learn each other's schedules and make a meeting. We learned that their schedules favored meeting at 4pm, Tuesdays and Thursdays.

**Problems We Encountered:** During our initial team meeting, we learned that none of us were very confident in our web development skills. However, after discussing it, we discovered that our skills and interests complement each other well. We decided that each of us would focus on a component of the final design, and improve our skills as necessary to aid our other team members. We would collaborate and specialize as necessary to complete our goals.

*2) Tuesday (October 13, 2015):*

**What We Did:** This was our first official group meeting with our client. Since our clients are stationed in Washington, our only way of communication was either by phone, email or skype. For this meeting, our method of communication was through skype so that our clients can get a recognition of our face to names. During the meeting, we each introduced ourselves by answering a set of questions provided by our clients. The clients wanted to get a better view of us as developers and discover our skills. We also discussed what requirements the clients desired and how we could meet them.

We also met with our teaching assistant, Jon Dodge. We didn't have much to report, but he gave a summary on what he expected from us.

**How We Did It:** We emailed our client last week to schedule a meeting with them officially and we kept in contact with each other via email. Our client had sent us a Skype Invitation via email prior to the meeting and one of our team members used his laptop as the main communication channel. Since it would be confusing and separated if we each used our own laptop to talk with our client.

**Problems We Encountered:** Shortly after meeting with the clients, we discussed the technologies we were asked to use, and how to implement them. We agreed that pulling data from the funding sources is the primary source of difficulty in our project.

Initially, the client asked us to look at the metadata related to the funding source websites. Our initial investigation involved looking at the HTML itself and identifying the HTML tags that correspond to the entries we were interested in. After we had done this, we developed our initial design for extracting data. We didn't pursue this approach for very long, however.

*3) Saturday (October 24, 2015):*

**What We Did:** We started outlining the requirements document after our meeting with our teaching assistant Jon Dodge earlier in the week on Tuesday. We used our earlier meeting with the client to brainstorm rough requirements. The clients emailed us some work from the previous team, and we used that for further insight into what they wanted. We scheduled a meeting with our clients for the next Tuesday.

**How We Did It:** Our clients required us to use four technologies in our implementation. The first type of technology required was methods of retrieving data in order to pull data from funding opportunities sources. The second type of technology required was database systems to store data from either the user or the funding opportunities sources. The third type of technology required was methods of the presenting the data to display funding opportunities for the user to view. The fourth and last type of technology required were computer languages used for manipulating aggregated data.

**Problems We Encountered:** After our meeting, we decided that our first idea (HTML scraping) would be very inefficient. This approach would involve figuring out some way to iterate through all of the pages of each funding source website. Then, we would need to download the raw HTML of the website and parse through it, searching for tags corresponding to the fields we identify. If we had implemented this approach, we could have likely run into problems later on if the funding source had decided to redesign their website in any way. It also would have been very slow, with a large amount of overhead for the data gained. We would need to download a large amount of HTML files. We would have to parse through a lot of irrelevant

HTML tags until we find one that corresponds to a tracked field in our database. This would be very time- and space-inefficient. We unanimously decided that this was the wrong approach to take for the design.

*4) Tuesday (October 27, 2015):*

**What We Did:** For this day, we had our second group meeting with our client via Skype regarding what they want for the requirements document. During the meeting, we asked our clients some questions to clarify the concept that they had in mind, so that we know what requirements to include. We met to discuss the implementation of our data parsing component. During the meeting, we took notes on the technologies and specific requirements they had. After the meeting, they emailed us a bulleted list of requirements.

Our clients also outlined the specific technologies we would need to use. They asked us to use PHP for the back-end scripting, JavaScript for the front-end, and PostgreSQL for the database. We later used this information of our Tech Document.

We translated the bulleted list the client gave us into a set of requirements, and broke them down into smaller, simpler instructions. We also added more requirements based on what they had told us during our meetings but had glossed over in their list. We felt it was better to add these to our requirements instead of leaving them in the air as to whether we would implement them or not. We didn't want to disappoint our client.

**How We Did It:** Similar to our first Skype meeting with our client, they had sent us an invitation where we could join them by just having one member click the invitation and log into Skype. Prior to the meeting, we had already brainstormed some questions for our client that required clarification.

Our client asked us to use certain technologies, but we decided not to list the technologies in our requirements document. Our client told us we could be flexible in where to use them, and didn't seem dead-set on us using them. They accepted that we left it off the document and signed it as-is. Later, we mentioned the technology requirements in our Technology review assignment.

**Problems We Encountered:** During our meeting, our clients gave us a small set of requirements that we needed to think about more thoroughly. We needed to work with them to discover more intricate requirements and break their relatively vague requirements into a concrete set of instructions for us to implement.

The client wanted us to use PostgreSQL, but we are not sure how to set up a development environment that we all can access that will allow us to test with a PostgreSQL database. We know that we have access to our personal MySQL servers through the university, but we are not sure what sort of effort will be required for converting between MySQL and PostgreSQL. We ultimately decided to test our code locally. When we get to implementation, we will need to discuss specifics of how to accomplish this.

*5) Thursday (October 29, 2015):*

**What We Did:** We met on this day to discuss and create the timeline we expect to complete during the whole year of the senior design course. In order to create the timeline, we used a Gantt Chart to organize the different tasks that we had thought of. The Gantt Chart we created is included on our SharePoint site, so we could keep track of the tasks we have completed so far. We also improved the formatting and completed the requirements document in order to meet the due date.

**How We Did It:** In order to create our timeline we used SharePoint to create a Gantt Chart by listing the tasks that we would need to follow through based on what our requirements document stated. The deadlines for completing each task was more of an estimate which we thought would be feasible. It was not meant to be a concrete requirement.

**Problems We Encountered:** As part of our ongoing research, we came across other methods of pulling metadata from the sources. A helpful Google search found us the full FedBizOpps.gov XML dump. After looking into it further, we found more information about the XML files published by Grants.gov and FedBizOpps.gov, thanks to 18f. 18f is a US government agency that makes digital products for government organizations. They created their own scripts for parsing the XML data published by both our targeted funding sources. 18f's XML parser is named FBOpen, and is released under the public domain. This is good for us, because it means we can adopt their code and adapt it for our specific uses when necessary.

*6) Friday (October 30, 2015):*

**What We Did:** To summarize this week's progress, we focused a lot on completing the requirements document and getting it approved and signed by our client. So we planned the work that would need to be done. We met up earlier this Tuesday with our client. It is safe to say that we turned in our final requirements document approved and signed by our client and our group.

**How We Did It:** For us to finish and complete the requirements document on time, we had to be on topic each time we had a meeting because our schedules differed and didn't allow us to meet as much per week. By being productive for the week of October 26, we each delegated different portions of the requirements document to each member. Then as we added more details for the document, each of us had proofread over each other's writing and from time to time, we had to make changes. It was more efficient for our team to use this methodology because we could ask each other to elaborate on what they wanted to say and at the same time critique each other's work. By questioning each member's idea, we were able to see different viewpoints and potential improvement on our methods.

**Problems We Encountered:** We were not sure how much information to include on the requirements document. We spent a lot of time looking it over as a group, and asked the client if they were satisfied. They were, and signed it.

### B. November Posts

*1) Friday (November 6, 2015):*

**What We Did:** This blog post is to highlight the week's earlier Tuesday, where we had our weekly meeting with our Teacher Assistant to clarify some questions we were unsure of. On the same Tuesday, we sat down together as a group to brainstorm how we would like to present the different types of technologies we will use for the project. We also delegated responsibilities for each member of the team to the completion of the document. Similar to the requirements document, each member volunteered to finish one category of technology.

**How We Did It:** Our method of brainstorming the different technologies that may be utilized in the project was to research alternative technologies that would help reach our main goals. We defined our main goals as the different technological methods that we would implement in order to complete the funding component. These technological methods were defined in the technology review document.

**Problems We Encountered:** While researching the FBOpen technology, we discovered that it is implemented with JavaScript. However, our client has asked us to use PHP on the backend. Because the XML parsing is a back-end process, this would conflict with our client's request. After discussing it, we decided it might be better to implement the back-end XML parsing scripts ourselves, using the FBOpen implementation as a reference for the structure of the XML documents we are using.

*2) Tuesday (November 10, 2015):*

**What We Did:** On this day, we had finished our Technology Review document by reviewing what we already had and adding more information. As a group, we also discussed the possible technologies and what the feasibility would be like if we were to use them for our project. Feasibility is an important part of the project because we want to be able to meet our deadlines and be able to complete most of the functionality of the funding component. We do not want to waste time looking into technologies that we may or may not use.

**How We Did It:** Our process to complete the technology review document was to research the alternative methods to compare and contrast it with our main technology methods. During the research, we would summarize our findings and determine the best solution for the project.

**Problems We Encountered:** In researching the XML parsing, we decided as a group that we would look into implementing it ourselves. This will involve more work on our end, but ultimately more maintainable. Instead of relying on outside tool that we may need to configure to suit our purposes, we can design our own tailor-made scripts. This will give us greater control over, and understanding of, our system. Implementing it ourselves also means that we (and likely our successors) will not need to worry about updating the FBOpen scripts and integrating the changes into our code.

*3) Friday (November 13, 2015):*

**What We Did:** For this week's summary, we fleshed out our plan's for the website's design with additional details. We also researched the different approaches we could use to implement the website. Also we have become familiar with the different technologies that we will be using to construct the funding component. Then we discussed the design of the database furthermore and created a document to show the data we believe should be stored in the database. We have also identified the similarities between the data that would be stored.

We had done research on the XML documents published by the funding sources before, and created three sheets within it to describe the data contained for each funding source. We listed the fields associated with each source, any notes, and related fields between them. Then, we created a super sheet for the shared attributes between them. We used this to create an overview for our database tables.

**How We Did It:** Within the group meetings, we had discussions regarding how we would like the database to interact with the display of information on the website and how the back-end manipulates the many files of data. We would discuss what each of us thinks on the approach and then we would state alternatives and how we could improve the design.

*4) Monday (November 16, 2015):*

**What We Did:** On November 16th, we had finished the rough draft expo poster for our group.

**How We Did It:** We utilized the Power-point template that was provided to us in order to keep consistency with the branding guidelines for OSU. To start the poster, we laid out a skeleton of what we wanted and where on the poster. Then we filled in the sections with relevant information while following guidelines provided to us. We had to think as a team, what the audience would like to see on presented on the poster and what type of audiences we might get at the engineering expo. Therefore, revisions were made in order to refine our rough draft. There is most likely going to be more revisions as the year goes on.

**Problems We Encountered:** There was more information that we wish we could have included on the poster, but since space was limited, we only picked what was relevant and important to the audience. We also could not include much information about implementation, which made describing the final product difficult.

*5) Friday (November 20, 2015):*

**What We Did:** The weekly update for November 16th week was we met to discuss our poster, our elevator pitch and any upcoming assignments. On Monday, we finished our poster and posted it on our SharePoint along with the website's

documents. We were unsure as to how much we needed on our poster but we gave it a good overview of the project. Then we started on the script of our elevator pitch and went through a few different approaches in order to memorize the contents. Lastly, we discussed the upcoming assignments and when we can meet in order to complete them. Our minds needed to be collective on one design that will work.

**How We Did It:** Since the poster process is already explained above, we can now explain our process for our elevator pitch. To start off, we went through our previous documents to see where we could highlight some of the important points in our project. Next we created a skeleton of our elevator pitch. Then we filled it with contents and decided to read it out loud with a timer. We were definitely over the 30 second time limit. So we broke up the elevator pitch into three different sections, according to the number of members in our group. As we progressed, we kept rearranging the sentences without changing the meaning of what we would like to convey. After many iterations and trials, we came to a concluding elevator pitch that was within the 30 second time limit.

**Problems We Encountered:** Some problems we encountered were during the different trials of the elevator pitch, sometimes the team members would mess up and sometimes we would forget our parts. It was a frustrating process, but nonetheless to solve our issues, we had each member type out the format of how they usually speak. Only changing the sentence formatting and yet keeping the meaning the same.

*6) Friday (November 27, 2015):*

**What We Did:** This week was short but we started outlining the design document and planned to keep working on it by setting up upcoming meetings. We decided to meet during dead week in order to complete the progress report as well as continuing on the completion of the design document. Due to thanksgiving, we couldn't meet in person or online.

**How We Did It:** Since this week we couldn't meet up as much as the other weeks, we tried to work as much as possible on the design document and then possibly add more remotely during the Thanksgiving break. Before the break, we had a skeleton for how we would like to layout the design document sections. Then we started to fill in information we already knew into sections we were familiar with.

**Problems We Encountered:** At first it was hard to know what information we needed to include in the design document, but after some researching we learned what was essential to add to the document while keeping with the IEEE format. Also, asking the Teacher Assistant for clarifications of what we should include was very helpful.

### *C. December Posts*

*1) Wednesday (December 2, 2015):*

**What We Did:** Our final fall term meeting with our Teacher Assistant was yesterday (Tuesday, December 1). Then we met up afterwards to complete more of the design document and we got a majority of it completed. By the end of the day, we had completed our design document.

**How We Did It:** For the meeting on Tuesday, we continued developing on what we already had for our design document. As we each proofread what each member had wrote, it jogged better improvements on our document. We applied the newer changes and continued to research what our document was missing. As for Wednesday's meeting, most of the workflow was the same as Tuesday but we were closer to completing the design document itself. When we completed the document, we sent the final copy to our Teacher Assistant, as to double check what we were still missing.

**Problems We Encountered:** Our major problems was in not understanding the IEEE format required for a Software Design Document. We each read through the document, and weren't sure whether we needed to include a bulleted list of every design viewpoint, or if we needed to simply address them at some point in a document of our own layout. We looked online for examples of other design documents, and found that it was better to include general sections, and having many complex UML diagrams was unnecessary. Instead, we included one ER diagram to describe the layout of our database, which we feel is part of our design most likely to confuse readers. After we finished, we emailed our Teaching Assistant to get his opinion on the document, and after getting his approval, we saved the document as a PDF to our documents section of the SharePoint site.

During the creation of our document, we were not sure how to represent the database. We had thought about the interest tagging system, but had not designed a database table to represent them. We discussed a couple approaches, but there were confusing difficulties that we weren't sure how to work around. They are described in the Design Document. There are different fields used to categorize opportunities between Grants.gov and FedBizOpps.gov. These can be CFDA numbers, categories, Classification codes, and NAICS codes. Each of these could be associated with a different interest held by a user. We would need some way of storing all of those possible values with each interest. The design we came up with was to store a list of strings for each interest code/category/etc.

*2) Friday (December 4, 2015):*

**What We Did:** This is our weekly update for week 10. We met up more this week in order to finish the last two documents for our project. As we completed the design document, we immediately started to work on the progress report. By Friday, we had already completed both documents.

**How We Did It:** We were more productive during each meeting for this week, more than usual because we wanted to meet the upcoming deadlines and not fall behind. As we sent out our final copy of the design document to our Teacher Assistant, we started to work on collecting the blogs together and describing more in-depth of what had happened. We also looked into what we could possibly include in our progress report as to be more detailed on our development.

### *D. January Posts*

*1) Thursday (January 7, 2016) :* **What We Did:** Started on the project and tried to get connection.php to work with our own database. We are not sure how to use the scripts running with our database.

**How We Did It:** We tried different methods to jumpstart on the project.

**Problems We Encountered:** We were not sure how to get the scripts to run with our own database.

*2) Tuesday (January 12, 2016):*

**What We Did:** We continued where we left off from the last meeting. We also started to work on the XML parsing, XML downloading, and debugging Javascript errors.

**How We Did It:** We delegated the work among the three of us so that we all wouldn't be stuck on one single problem.

**Problems We Encountered:** There was a ng-repeat duplicates Javascript error that we found and was trying to fix it so that we could display the front-end displays.

*3) Monday (January 18, 2016) David's Work :*

**What He Did:** He finished the XML download scripts and updated the database creation script.

**How He Did It:** There are PHP functions to download the weekly and nightly FBO files, and the Grants.gov full XML. The FBO weekly file and Grants file are large and take a long time to download. As for the database creation script, it creates tables for storing the funding opportunity data. The tables are as described in the Design Document, in the documents section.

**Problems He Encountered:** There were some irregularities in the XML files that he had to discover on his own.

*4) Tuesday (January 19, 2016) Chris's Work :*

**What He Did:** Completed and tested a basic XML parser that parses elements in the XML file.

**How He Did It:** The elements in the XML parser are placed into their respected grant objects which are structures. Then all of the objects are appended onto an array. The code is easily scalable for more elements if needed. The array is to make the implementation for our SQL insertion easier.

*5) Tuesday (January 19, 2016) Benny's Work :*

**What He Did:** Tried to get rid of the AngularJS error of ng-repeat duplicates.

**How He Did It:** Tested the track by method in the ng-repeat directive to see if it debugs the error, but no viable result. Tried to email client to see if they are getting any thing to show up on their screen.

**Problems He Encountered:** Still getting the same error and no display on publications page.

*6) Monday (January 25, 2016) David's Work :*

**What He Did:** Finished FBO nightly parser.

**How He Did It:** Since the nightly files are not XML, he could not parse them as normal XML like the weekly FBO and daily Grants.gov XML files. Up to this point, the only notice types that are added to the database are pre solicitation, sources sought, combined synopsis/solicitation, and special notice.

*7) Monday (January 25, 2016) Client Meeting :*

**What We Did:** We met with the client to give them an update on the progress we've made. We told them the work we've done and what we expect to do soon. They also gave more detail on how they expected our design to look, including filter options and information display. We also discussed the types of notice that should be taken from FedBizOpps.

*8) Thursday (January 28, 2016) Benny's Work :*

**What He Did:** Created a funding.html for the user to view the funding opportunities and a funding.js file to manage the funding module, controllers and directives. Also added a filter.tpl.html to format the filtering column on the main HTML page.

**How He Did It:** Created a simple format of funding.html using bootstrap and simple Javascript file for manipulating funding controllers. Linked the index.html to the funding.html page and vice versa.

**Problems He Encountered:** There is an implementation error in the filter-funding.php file that will be fixed.

*9) Friday (January 29, 2016) Chris's Work :*

**What He Did:** Started the basis of the parser to work for Grants.gov files. Ideally, will create a test for larger XML files from Grants.gov. Started working on the SQL insertion part of the parser.

**How He Did It:** Created a foundation of idea of how the Grants parser will work with small XML data.

**Problems He Encountered:** Small kinks still needed to be worked out in the code for full functionality.

### *E. February Posts*

*1) Thursday (February 4, 2016) David's Work :*

**What He Did:** Finished the FBO nightly file parser which adds records that aren't in the database and updates for AMDCSS and MOD notice types. Also finished the FBO weekly file parser which adds and updates records in our database.

**How He Did It:** The parsers read through all of the records in the XML file one by one, adding each record or updating existing records. The assumption here is that the weekly file takes precedence over anything else. So if a record in the XML file contradicts a record in our database, our record will be overwritten.

**Problems He Encountered:** Some of the records are odd and don't have solution numbers. While others have strange date formats, and the descriptions have HTML tags in them. The script rejects records with no solution numbers and ignores improper dates.

*2) Friday (February 5, 2016) Benny's Work :*

**What He Did:** Still working on filter-funding.php script at this point. Also worked on funding.js to get the main funding opportunities column started.

**How He Did It:** Created a template view for the main column with bootstrap and certain tags and fields are subject to change depending on changes made to filter-funding.php script. Created a new directive and a controller for funding items to appear in the main column.

**Problems He Encountered:** There are still some potential bugs in filter-funding.php script that will need to be worked out.

*3) Monday (February 8, 2016) Chris's Work :*

**What He Did:** Finished the insertion of opportunities from Grants.gov.

**How He Did It:** Used SQL insertion method to insert the opportunities into the database.

**Problems He Encountered:** There are some minor problems that need to be fixed regards to memory and timeout limit.

*4) Monday (February 15, 2016) (Post Midterm) Client Meeting :*

**What We Did:** Today we met with our client and discussed what we already have for the Alpha level of the project. Our client is satisfied with what we already have and we have discussed what to add to the Beta level. We have already finished the Alpha level of the project by this point.

*5) Friday (February 19, 2016) Chris's Work :*

**What He Did:** Fixed all shortcomings of the Grants parser. Currently learning how to use AngularJS more proficiently to work on the deadlines column.

**How He Did It:** He converted the parser's design to stream the data by iterating the structures rather than holding all of the input data into a buffer.

*6) Friday (February 19, 2016) Benny's Work :*

**What He Did:** Worked on the funding deadlines column.

**How He Did It:** Added a funding-deadlines.php file to select the funding deadlines that pertain to the specific user and will need to fix the calculating of the time of deadline. Added a insert_funding_deadlines.php file to insert the funding deadlines into the new table called ctr_user_fund_link. Added a template for displaying the funding deadlines called funding-deadlines.tpl.html. Added a new CSS style for the funding deadlines column. Added a new table to the database_creation script for holding funding deadlines.

**Problems He Encountered:** There are some bugs in inserting the funding deadlines that will be addressed.

*7) Friday (February 26, 2016) Benny's Work :*

**What He Did:** Worked on completing the add button feature that sends opportunity to the funding deadlines column when the user clicks on it. Refactored the SQL queries in the filter column so that it displays the count that is from the ctr_user_fund_link table. Also the funding items main column is working properly now. Started to look into the multi-toggle filtering as the client requested.

**How He Did It:** Used components from bootstrap to add a button and AngularJS to bind that button with an insert SQL statement to add the specific funding opportunity to the deadlines column.

**Problems He Encountered:** There is only adding to the deadlines column, potentially might want to have a remove feature in the deadlines column.

### *F. March Posts*

*1) Saturday (March 5, 2016) Benny's Work :*

**What He Did:** Worked on fixing the type checking for the filter column groups Agency, Notice, Post Date, and Due Date. In the end we chose to utilize a second controller for the sharing feature so that it can be independent by each funding opportunity.

**How He Did It:** Tested the share funding button and then refined it accordingly. Tinkered with the sharing feature for quite a while, until we found that the functionality wasn't independent by each funding opportunity.

**Problems He Encountered:** At first we ran into the problem that the sharing feature was doing the same behavior for all opportunities and not unique enough.

*2) Saturday (March 12, 2016) Benny's Work :*

**What He Did:** Finished up poster requirements with group for this term. Also worked on finishing up final video and final report. Also worked on getting the dog ear favorite to be feature complete.

**How He Did It:** Completed parts of the poster and rendered the final video for group.

### *G. April Posts*

*1) Wednesday (April 6, 2016) Benny's Work :*

**What He Did:** Met up with client for the second time of the term and discussed the upcoming plans and extra features. For development, added a peak feature that displays the top 3 recommended fundings for the index.html and funding.html.

**How He Did It:** Utilized some CSS and bootstrap to create a similar peak feature to the last group. The design of the peak feature requires to split the ng-app of funding.html and index.html, so that each page shows the other's top 3 items.

**Problems He Encountered:** Had a hard time finding a way to use two ng-apps in one HTML file.

*2) Saturday (April 16, 2016) Benny's Work :*

**What He Did:** Met up with group to solve the bug of funding opportunities not displaying due to implementation of the new feature of multi-toggling.

**How He Did It:** Added console logs in Javascript file to find and figure out what the error is and how a fix can be applied.

**Problems He Encountered:** Still trying to figure out why multi-toggling feature is not working.

*3) Friday (April 22, 2016) Chris's Work :*

**What He Did:** Met up with client and group to discuss current progress and demonstrating multi-toggle filter and other enhancements. Received feedback on our poster from our client where they would like changes to be made. Our group revised the poster, made the changes necessary, and sent it back for approval by client.

**How He Did It:** Revised the poster with group. Started working on getting agency logo images for each funding opportunity.

*4) Friday (April 22, 2016) David's Work :*

**What He Did:** Added a script so developer can add images for agency logos to funding opportunities.

**How He Did It:** Created an img_agency directory and an agency_add_image.php file to insert the images using an image URL and agency name. Last Wednesday, modified the funding-items.php script to add the image file name to the JSON returned when queried.

*5) Sunday (April 24, 2016) Benny's Work :*

**What He Did:** Created a front-end for displaying the agency images.

**How He Did It:** Grabbed the image queried by David's funding-items.php script to display the image with the help of AngularJS.

**Problems He Encountered:** At first the images didn't display, but after a while of figuring out with Chris, we utilized ng-show from AngularJS to complete the agency image display.

*6) Friday (April 29, 2016) Benny's Work :*

**What He Did:** Slowed down on development because most of the client's requests are fulfilled and started to work on the poster to get approval from T.A. and client.

**How He Did It:** Revised poster even more for the preparation of Expo. Also worked on midterm report and video.

### *H. May Posts*

*1) Thursday (May 26, 2016) Benny's Work :*

**What He Did:** Added a script to populate user 2 with funding opportunities pertaining to the user in preparation for Engineering Expo.

**How He Did It:** Created a replication of David's script, researched and added NACIS codes for chemist (user 2) to the new script. Ran the script to populate user 2.

## V. Final Poster


Fig. 2: The funding page, showing filter and an opportunity

## VI. Technical Instructions

### A. Visual Structure

The website currently displays three columns, each with a designated role. The left-most column acts as a sort-by filter, where the user may select which funding opportunities he or she would like to view. In the middle column is the main view of the funding opportunities, whose display is based on the left-side column filter. Lastly, the right-most column is planned to be a collection of the research events and deadlines currently and upcoming. All of these are contained in the funding.html file. Figure 2 shows the filter on the left of the page, and an opportunity on the right. The top of the image shows the navbar, with the logged-in user's name and profile picture.

Next, we have a funding.js file that basically acts as a link between the Ajax PHP scripts and tpl.html files. Within the funding.js file are AngularJS directives and controllers that manipulate what the user views in funding.html. The directives that are currently in the JavaScript file are for the navigation bar, the filtering column, the funding item column, and the deadlines column. The directives are for creating custom html tags that can grab information from the .tpl.html files. Then we have the AngularJS controllers that hold JavaScript functions and grabs functionality from the Ajax PHP files in the Ajax directory.

### B. Technical Structure

The funding component of Connected to Research is made up of three interrelated design entities. There is a front-end interface for users to interact with, a database storing all of the data related to funding opportunities, and a back-end suite of scripts to connect the front and back-ends. The web app allows users to create an account, be linked to funding opportunities, and interact with the opportunities by favoriting, sharing, and tracking them.

The front-end of the application will be what users see and use to interact with the funding opportunities. It includes a browsing page with a multi-toggling filter panel. The togglable filter panel will allow users to intelligently refine the opportunities they see. They can toggle based on offering agency or offer type.

The database is where the funding opportunity data is stored after being parsed out of the funding source's published data. It is populated by scripts existing in the back-end.

The back-end is responsible for populating the database and retrieving data for the front-end. The funding opportunities are pulled from XML files made available by the funding opportunity sources and inserted into the database. The back-end also includes scripts to retrieve data from the database to serve the front-end.

### C. Installation & Use

*1) Installation:* First, you'll need to git clone the repo somewhere internet-accessible. We used the public_html directories on the schools servers. Those are accessible at http://web.engr.oregonstate.edu/ ONIDUSERNAME. You will also need to set up a MySQL server. We used the database provided by the school, though ONID. To access it as a student, go to onid.orst.edu, log in to ONID, and click 'Web Database' in the left pane. Note the database name and password it generates for you. Next, modify the file 'connection.php' in the project's 'scripts' directory. Modify the dbhost, dbname, and dbuser variables. Create a new file named 'password.php', and use it only to create a variable named dbpass, your database password. Don't push that file to github.

After that, the site should be ready. If you used the school servers to host the site, be sure to reset your web directory from your TEACH page, https://secure.engr.oregonstate.edu:8000/teach.php. Click the link 'Personal Web Tools' under 'Account Tools', and when the page loads, click 'Reset Web Directory'. If you don't reset the web directory, you will get a 403 forbidden error. According to engineering college support staff, "The way php runs in user home directories, relies on the files being owned solely by the user and assigned to the upg (user personal group)." After cloning from git, the files will probably not belong to you, so the files will not be accessible.

If the project has been set up properly, you can now load a blank white page. To make things appear, you will need to add data to the database.

There are no special hardware, operating system, or runtime requirements needed to run the software. To run the web application, only a working computer with a functional web browser is needed.

To host the software, users will need a server to host a website and MySQL server. Specific hardware and operating system choices can be left to the user.

*2) Use:* There are multiple scripts included in the project's scripts directory. They are described in more detail in the documentation section.

First, run database_creation.php to create the necessary database tables. This script should only need to be used once. If new tables are created, they should be created through this file, and not through a direct SQL query.

After the tables have been created, populate them with the scripts from the parsing directory. Use parse_fbo_weekly.php and parse_grants.php to parse both sources of funding opportunities. These will download two large XML files from the sources, so make sure there are no space limits. They also take a long time to run, so you may need to increase the PHP execution time limit. Our scripts increase the limit to one hour.

Use recommend_to_user.php to connect users to funding opportunities. This script takes three HTML GET variables:

- Type — either article or call, for the publications component, or funding for the funding component.
- Users — a comma-separated list of users to be connected to the funding.
- Ids — a comma-separated list of funding opportunity ids that exist in the database.

Alternatively, there are scripts insert_recommendations.php and insert_recommendations_user2.php, which will automatically connect the accounts user1 and user2 with funding opportunities. To use this, run the insert_users.php script to create the test users 1-3.

After the user has been connected to their funding and publications data, there should be opportunities appearing on the funding page and publications page. The funding page will not have any agency images to appear next to funding opportunities. To add them, use the script agency_add_image.php, which takes these HTML GET variables:

- Image — A URL for an agency image
- Agency — An agency name that exists in the database

After images have been added to the site, they should appear next to the appropriate funding opportunities.

## VII. Documentation

### A. Scripts Directory

*1) agency_add_image.php:* This script downloads an image file from a URL, and adds it to the database to map an agency to an image. It takes these HTML GET variables:

- Image — A URL for an agency image
- Agency — An agency name that exists in the database

*2) article_likes.php:* This defines a PHP function, get_likes(), which returns a count of all articles a user has liked. It's parameters are a user id and mysqli object.

*3) connection.php:* This script is used at the top of all PHP files that need to use the MySQL server. It connects to the server named in the variables and uses a file named password.php, which stores the server password.

*4) create_endnote.php:* This script creates an endnote for the publications component.

*5) database_creation.php:* This script creates all the database tables necessary for the project to function. User tables, funding tables, etc. Doesn't take any arguments, and returns success or failure for each table.

*6) delete_funding_deadlines.php:* This script removes a funding opportunity from the currently logged-in user's list of funding opportunity deadlines, the list that appears in the right panel of the funding page. It takes one HTML GET variable:

- id — the ID of the funding opportunity to remove.

*7) favorite_article.php:* This script adds a favorite record to the database for the currently logged-in user. It takes one HTML GET variable:

- id — the article ID to be favorited.

*8) favorite_funding.php:* Same as favorite_article.php, but for funding opportunities. Takes a HTML GET variable, which is the funding ID to favorite.

- id — the ID of the funding opportunity to favorite.

*9) generate_citation.php:* This script creates an string for an MLA citation of a given article. It takes one HTML GET variable:

- id — the ID of the article to cite.

*10) get_users.php:* This script returns all of the users that exist in the database, encoded in JSON.

*11) get_users_funding.php:* This script gets the users for the funding components 'share' feature.

*12) insert_events_and_deadlines.php:* This script adds a link between a user and a research item, which is used for the publication component. The events and deadlines are listed on the right panel on the publications page. It takes one HTML GET variables:

- id — The research ID

*13) insert_funding_deadlines.php:* Similar to insert_events_and_deadlines.php, this script links users to funding opportunity deadlines to be displayed in the right panel of the funding page. It takes on HTML GET variables:

- id — the funding opportunity ID

*14) insert_recommendation.php, insert_recommendations_user2.php:* These two scripts connect our test researchers to some funding opportunities, based on an array of NAICS codes. It takes no arguments, but requires either 'user1' or 'user2' to exist.

*15) insert_users.php:* This is another test script that adds test users user1, user2, and user3 to the database.

*16) insert_xml.php:* This is the script the publications component uses to populate its articles database. It takes a raw string of XML, and parses it. It takes two HTML GET variables.

- XML — The string of XML
- type — The type of XML

*17) login.php:* This handles the site's login functionality. It takes two HTML POST variables.

- email — the user's email.
- password — the user's password.

*18) logout.php:* This handles the site's logout functionality. It takes no variables.

*19) password.php:* This is a file that we used to store the database password. We kept this in a separate, untracked file so we wouldn't push it to github.

*20) recommend_to_user.php:* Adds a record to the table to connect a researcher to an article, call to research, or funding opportunity. This is the script used to create the "connections" in Connected to Research. It takes three HTML GET variables.

- type — either article (for publications), call (for a call-to-research), or funding (for a funding opportunity).
- users — a comma-separated list of user IDs to be connected to the items.
- id — A comma-separated list of item IDs to be connected to the users.

*21) register.php:* Used to register a new user profile for the site. Takes four HTML POST variables.

- name_f — the user's first name.
- name_l — the user's last name.
- email — the user's email address.
- password — the user's chosen password.

*22) share.php:* Used to share an article from the currently logged-in user to another user. Unlike the items added through recommend_to_user.php, these show up under a 'shared' table, instead of 'recommendations'. It takes three HTML GET variables.

- type — either article or funding.
- id — the funding or article ID to be shared.
- users — a comma-separated list of users to share the item with.

*23) view_favorite_fundings.php:* Queries the database to get all of the favorited funding opportunities of the currently logged-in user. Takes no arguments.

*24) view_favorites.php:* Used to get the favorited articles of the currently logged-in user., similar to view_favorite_fundings.php. Also takes no arguments.

*25) view_share_fundings.php:* Similar to vew_favorite_fundings.php, but instead gets all items shared with the current user.

*26) XML_download.php:* This script contains functions for the XML parsers to use to download the XML files. It contains three PHP functions that take no arguments. One downloads the FedBizOpps nightly XML, another gets the weekly FedBizOpps XML, and the third gets the Grants weekly XML.

### B. AJAX Directory

*1) filter-funding.php:* This file is for the funding component. It uses mySQL statements to select the count of each filter category, put them in arrays and display it using AngularJS with the help of keys named groupItem, amount, and filterName.

*2) filter.php:* This file is for the publications component. It uses mySQL statements to select the count of each filter category, put them in arrays and display it using AngularJS with the help of keys named groupItem, amount, and filterName.

*3) funding-deadlines.php:* This file uses a mySQL statement to get the funding deadlines stored in ctr_funding_base. Then it calculates the funding deadlines by converting the raw date data in the table and storing it into an array, where it then returns the converted date to JSON.

*4) funding-items.php:* This file depends on what the user chooses in the multi-toggling filters. Depending on which filters are chosen will utilize certain mySQL statements for selecting information from the table. The queried data is then stored in an associative array and encoded into JSON.

*5) nav-bar-fundings.php:* This file is called from funding.js to use a mySQL statement to select the top 3 funding opportunities from ctr_funding_base and display it using the peak feature in the top navigation.

*6) nav-bar-publications.php:* This file is called from app.js to use a mySQL statement to select the top 3 publications from ctr_article and display it using the peak feature in the top navigation.

*7) participation.php:* This file is used for app.js to select the call for participations in the publications component.

*8) publication-items.php:* This file uses a GET variable to grab the type of article the user has selected from the filter. Then uses a switch statement to query the data based on the filter selection. The data is put into an associative array and encoded as JSON.

*9) research-events-deadlines.php:* This file uses a mySQL statement to get the research and event deadlines for the publications component. Then it calculates the funding deadlines by converting the raw date data in the table and storing it into an array, where it then returns the converted date to JSON.

*10) user-nav-bar-info.php:* A file that uses a mySQL query to select the user's information from the ctr_user table based on the SESSION variable of email.

### C. JS Directory

*1) angular.min.js:* This is the minified version of angular.js. The file angular.js is the framework of AngularJS.

*2) app.js:* This file is used as the controller of the publications component. It pulls data based on users' requests then displays the data onto the view and updates the data. The file pulls data by executing one or more PHP files. It connects directives for all of the template html files that are used for the publications component to the main view of the publications component.

*3) App.min.js:* This is the minified version of app.js.

*4) bootstrap.js:* This file is a framework for developing responsive, mobile first projects on the web.

*5) Funding.js:* This file is used as the controller of the funding component. It pulls data based on users' requests then displays the data onto the view and updates the data. The file pulls data by executing one or more PHP files. It connects directives for all of the template html files that are used for the funding component to the main view of the funding component.

*6) login.js:* This file contains the implementation of user registering and login. If the user registers a user, register.php will be invoked via login.js or if the user is logging in, login.php will be invoked via login.js.

*7) login.min.js:* This is the minified version of login.js.

*8) npm.js:* This file is the package manager for Javascript.

*9) npm.min.js:* This is the minified version of npm.js.

*10) ui-bootstrap-tpls-0.12.0.min.js:* This file allows Bootstrap to use directives from AngularJS.

## VIII. Lessons Learned

### A. Benny Zhao

The technical information I learned from this project was the process of developing a web application from a low fidelity to a high fidelity proof of concept. I also learned how to use AngularJS to bind data from the server side and client side. Also with the help of my teammates, I gained valuable knowledge of XML parsing with the use of PHP scripts.

The non-technical information I have learned is how to format a poster adhering to branding guidelines and presenting a project at an expo environment.

From this project I have learned about the documentation process of a large scale project, how to meet the clients' initial needs, and the process of meeting deadlines. I also learned about the adaptive process it takes to fit the clients' changing requirements.

As for project management, I have learned how delegation of group tasks is important for meeting deadlines in a timely manner. Plus communication is crucial for conveying design implementations correctly and the success of the project. I have

learned how to work in a development team environment by collaborating ideas and discussing proof of concepts with group members. Also that group members can have different interpretations of clients' needs and this can be confusing during development. If I could do it all over differently, I would have tried to a design profile page for the end-user to keep track of their own information and perhaps try a more efficient way of implementing data queries for the project.

### *B. Chris Nguyen*

I learned how to work well with front- and back-end web development. PHP and AngularJS were new languages to me and now I am decently fluent in PHP and nearly fluent in AngularJS. I still have to search online for documentation on HTTP for implementations that are not basic.

I learned how to communicate well with my teammates and how we can collaborate easily without a need for a leader. We each had distinct strengths for this project, but we all learned from each other. I learned how you can explain broken code to another person and can potentially find your bug or error.

What I have learned from project work is that you have to stay consistent with the project in order to make progress. I learned how to get my work done in time along with my teammates in order to demo the work to our client. What I have learned the most from this project is working on documentation and the process of working with a client.

For project management, we give each person tasks to complete and the expected date or deadline to get it done by. What I learned from this is that if we stay consistent as in completing our work load on time then the project goes smoothly. There was one occasion that we could not finish a task on time, so we ended up grouping together and worked on it. I found that working together at times could be slow, but if everyone is up to date and figuring out multiple solutions then we could finish our task quickly. Overall, working in teams was beneficial to me since I can see progress in other parts of the project as well as how it was implemented.

If I had to do this project differently, I wouldn't because we spent a good amount of time thinking about the design then implementing it. We all worked hard to get the project done and fulfilled our client's expectations. Our goal was to satisfy our client's needs and we completed that objective.

### *C. David Winkler*

I learned a valuable lesson about teamwork. In a more technical sense, I learned far more about back-end web development. I became almost fluent in SQL syntax, and usually no longer need to consult documentation when programming in PHP or SQL. I learned how to carry on an existing project, learn the source code and design plans, and make meaningful improvements to it. I also learned how to recognize gaps in team knowledge and volunteer to fill the gap.

I also became much more familiar with LaTeX.

Due to the fact that this group had no specific leader, I learned how to communicate effectively with team members to make sure that everyone knew what needed to be done, and who was willing to work on it.

If I had to do this project differently, I'm not sure if I would. I feel that we all did our best and tried to finish our project satisfactorily.

### *D. Reference Material*

Websites that were helpful were stackoverflow.com, jsfiddle.net, php.net, w3schools.com, and youtube.com with stackoverflow.com being the most helpful and youtube.com being the least helpful of the five websites.

With stackoverflow.com, we were able to find many different solutions to many of the conflicts we had with our project which resulted in us fixing almost all the problems with stackoverflow.com. The website jsfiddle.net allowed us to play around with code to see how a feature of the project would be implemented and executed. This helped us fix some small bugs. Some of which were preventing the implementation of the code from executing the way we wanted. Both php.net and w3schools.com provided us with documentation about AngularJS, SQL, and PHP. The last website youtube.com was our last resort to finding any information that we can't find in the first four websites and provided us with tutorials to help us understand concepts in AngularJS, SQL, and PHP.

Anton Igorevich Dovzhik, one of the members of last year's capstone team that worked on our project for the publication component helped us with populating data for the web application. He gave us instructions on how to create users and populate data for that user.