Career Day Application: Final Report

Byron Jones
Tracy Karol
Erica Manzer
Anna Nienhaus
Joseph Zeng

Colorado School of Mines
Computer Science Field Session 2018
Advisor: Dr. Jeffrey Paone

Introduction

During this field session, we worked with the Colorado School of Mines Career Center and the Colorado School of Mines IT department--Computing, Communications, and Information Technology (CCIT)--to develop a mobile-friendly web application. This web application was designed for use by students in order to help them prepare for the Mines Career Day, and to help them navigate their way through Career Fair. Currently, the Career Center hands out program guide booklets including information about companies that will be attending Career Day. This app digitalizes the information in this book.

The website has two main functions: allow students to search for companies they may wish to talk to at Career Fair in the Companies tab and find where these companies' booths will be in the Maps tab. The Maps tab will include different map PDFs for each room at the Career Fair.

On the Companies page, students are able to view the list of all companies who will be present at Career Fair. They may then filter this list of companies by what majors and degree level or year they are hiring. Students may also use the search bar to look for a specific company.

Screenshots of the different pages on the site have been included in the appendix so that the reader may see how the pages discussed appear online.

Requirements

At the beginning of the project session, we spoke to our client about what they expected us to design during the six week field session period. Here we describe the vision and expectations for the project.

High-Level Vision

In planning this project, each stage was defined in different "phases." Phase 1 entailed any features or functionality that should be implemented in the 6 weeks of field session that must be complete in order to meet our Definition of Done. While we did not work on Phase 2 and beyond, it was important that we recognized these features to come so that we could design the website to be able to incorporate these functionalities later.

Upon the completion of Phase 1, users can see a list of all companies at the Career Fair. This list has a sort feature that can narrow by major and year. On the companies list, each entry has a link to that company's page. This holds information such as the company's booth number, website, and description. In addition, users can view PDF maps of the booths at the Career Fair on the Maps page. The booth numbers on the companies list and the Company pages are links to the PDF map that can be used to find that company's location at the fair. The company information is automatically updated from a web service connecting to the Career Center's SQL database.

Features that will be completed in Phase 2 and beyond may include but are not limited to embedded advertisements, company videos added to individual company pages, an interactive booth map where a booth on the map links to the page of the company that will have that booth, the ability to filter companies by industry, and a way to gather statistical data of user interactions.

Functional Requirements

The mobile-friendly website must allow users to search for and view information of the companies who will be attending Career Day. Users should be able to view all the companies that will be present and which majors and years they are hiring. Additionally, each company should have its own page showing their booth location, website link, and brief description. Finally, the booth maps should be available to view on the website. The website will have the following pages:

- **Home Page** (Figure A1) - The home page includes information about Career Day (date and time, locations, etc.) as well as links to other Mines websites that contain career information. These links were provided from the client.
- **Company List Page** (Companies Page, Figure A2) - This page includes a list of companies and their booth numbers. It allows users to search companies by whom they are hiring (year and major) and by industry. Industry searching was not implemented in Phase 1, but a placeholder in the code allows for this to be added later.
- **Individual Company Pages** (Company Page, Figure A3) - This page includes the company name, booth number (acts as a link to the booth map page), a link to the company's website, and a brief description of the company. It also includes a table indicating which majors the company is hiring and which years they are hiring for each major.
- **Booth Map Page** (Figure A4) - This page contains links to each of the different booth maps, which are hosted on the individual map pages.
- **Individual Map Pages** (Figure A5) - This page displays the map of a particular location.

The app communicates with the CCIT SQL database that currently exists for Career Day. The database is controlled and updated by administrators at CCIT and our app receives this information for its displays.

Non-Functional Requirements

- Visual Studio Community for IDE
- ASP.NET application framework
- C# used for back-end logic
- SQL database (preexisting)
- WCF REST web service for communication with SQL database (provided by CCIT)
- Bootstrap library for styling
- Razor markup language used for views

System Architecture

Here we describe the architecture of our project, both in the front end and back end. We also describe how all involved entities interact with each other to make the program run.

Technical Description

We developed the project as an ASP.NET application using the Visual Studio Community IDE and adhered to the Model-View-Controller (MVC) architecture. This means we separated our code into three components: the Model, which handles everything involving data; the View, which handles everything involving user interface; and the Controller, which handles the communication between the two. Refer to Figure 1 for a visual representation of the MVC architecture. A template for this type of application was provided through a package included in Visual Studio and it served as our starting point for the website. The views were coded in the markup language Razor provided in the template and Bootstrap was used for styling. The controller and model logic were written in C#.

Figure 2 displays interactions and relationships between different entities involved with the project. In Figure 2, we modeled the relationships between our app, the web server API, CSM's web server, and the users. The app will be hosted on the Mines server and will receive views from the web server API. Our app sends requests to receive data from the database with the web service API acting as a intermediary.

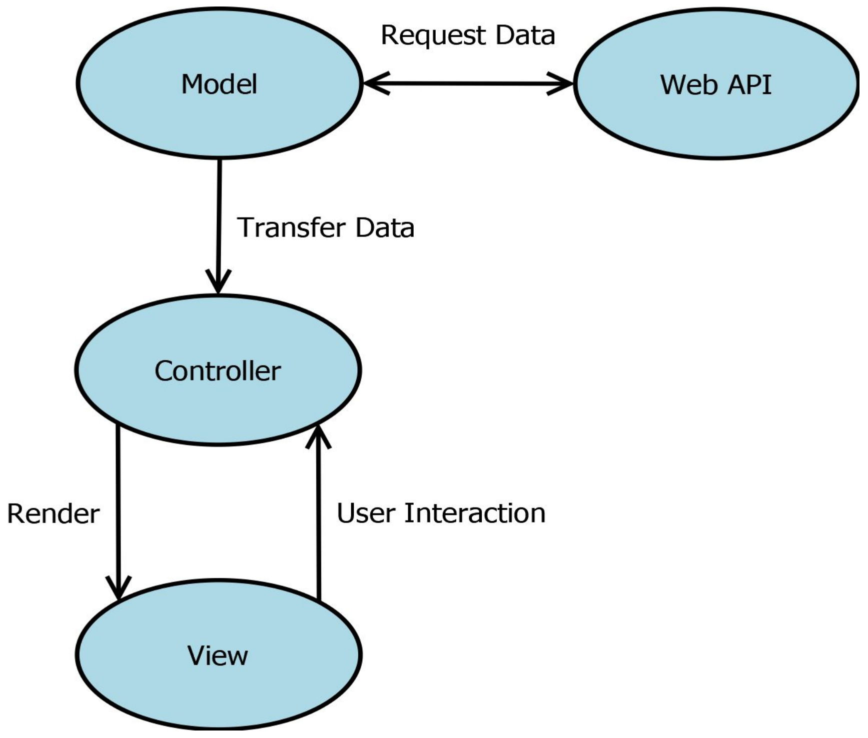


Figure 1: A model of how our application fits into the MVC architecture.

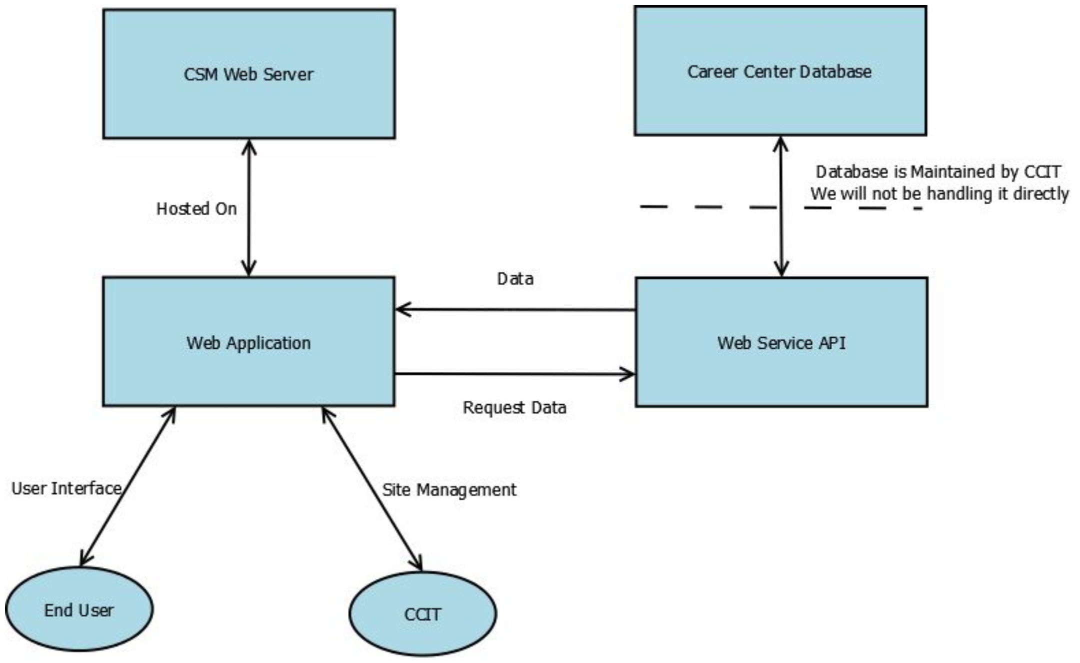


Figure 2: A model of the different interactions between project entities.

User Interface

Our user interface consists of various, easy-to-navigate pages. A list of pages and features and their functionalities are given below:

- **Navigation Bar** - The navigation bar appears at the top of each screen and allows users to navigate between pages. The features of this bar are as follows:
    - Home button - returns the user to the home screen
    - Company List button - takes the user to the company list page
    - Booth Maps button - sends the user to the page that contains the booth maps
- **Home Page** (Figure A1) - The home page is the page the user first sees upon authentication and will give them information about Career Day.
- **Company List Page** (Figure A2) - Upon arriving at this page, all companies are displayed with their booth number. Each company name is a link that the user can follow to the individual company page. The headers on the table have the functionality to sort alphabetically by company or by booth number. A search bar allows users to search for specific companies. Dropdown menus allow the user to filter companies by the following criteria.
    - Major - Shows only companies that are looking to hire the specified major.
    - Degree/Year - Shows only companies that are looking to hire students of the given degree or year.
    - Industry - Will show only companies of the specified industry. This feature is not part of Phase 1, but there is a placeholder in the code for when this can be implemented.
- **Company Pages** (Figure A3) - Each company has its own page displaying the company name, booth number, description, and website link, as well as a table describing who they're hiring. The booth number also links to the map containing that booth.
- **Booth Map Page** (Figure A4) - This page has links that the user can use to go to individual map pages.
- **Individual Map Pages** (Figure A5) - This page allows the user to view a specific map.

Technical Design

Framework

Using the ASP.NET framework for this application was a new experience for everyone on the team. It is an open-source server-side web application framework that handles all of the background communication between the code and the browser. Visual Studio has ASP.NET packages that set up a nice template website when a project is first created. This template comes equipped with Bootstrap styled pages and a file structure that is based on MVC architecture. We were given three .cshtml files under the Views directory and a controller file written in C# with the template. We were able to edit the existing cshtml files and add more so that we had one corresponding to each of the pages for our website and alternate cshtml files for mobile displays.

For all of our Controller logic, we utilized the HomeController.cs file from the template. Our framework has a URL template in the format: '{controller}/{action}?{id}'. This enables the functions in the specified controller file to be called and given arguments through the URL. As an example, to bring up the Map page that corresponds to a company's booth number, the URL would look like this: 'Home/Map?id=2554' where the company has an id of 2554 in the database. When this URL is received, the application knows that it needs to respond to the browser by going into the HomeController file and executing the Map function using the id for the argument.

The HomeController file has functions to render each page of our application. The names of the functions correspond to the names of the .cshtml files in our Views directory. The ASP.NET framework sets up a path so that when these functions return a View, it simply looks in the Views directory and returns the View with the matching name. Most of the pages require being given background information to know how to build itself. One such page is the Companies page (see Figure 3); since there is a filtering capability, there are many combinations of companies that are shown on the page depending on the parameters that the user specifies. To handle this problem, the function in the HomeController that brings up this page takes many arguments (Major, Level, Search) and is able to narrow down the list of companies accordingly. The function will make a call to the Model code to receive all of the companies in the database, then it will pass it to a helper function called Filter that takes the filtering parameters and returns a new list containing only the qualifying companies. This new list is then handed to the View file and is displayed on the page.

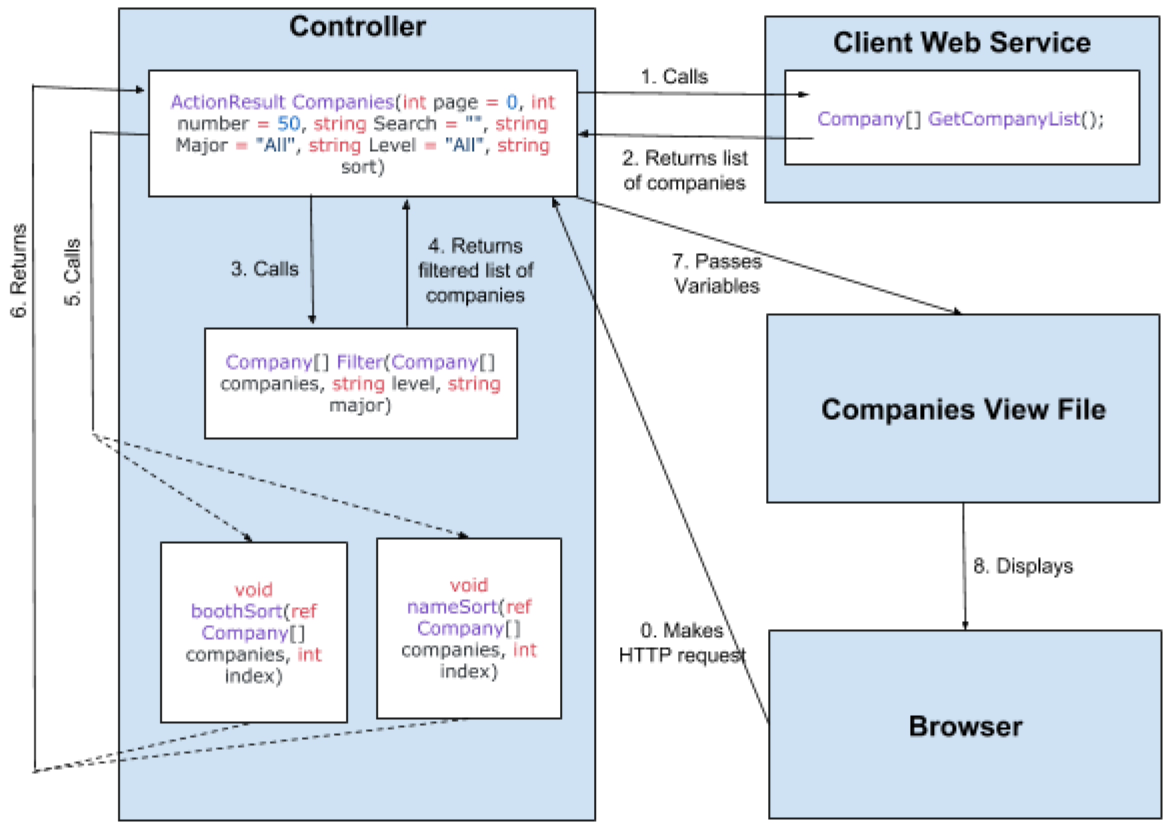

Controller
ActionResult Companies(int page = 0, int number = 50, string Search = "", string Major = "All", string Level = "All", string sort)
Client Web Service
Company[] GetCompanyList();
1. Calls
2. Returns list of companies
3. Calls
4. Returns filtered list of companies
Company[] Filter(Company[] companies, string level, string major)
5. Calls
6. Returns
void boothSort(ref Company[] companies, int index)
void nameSort(ref Company[] companies, int index)
7. Passes Variables
Companies View File
8. Displays
0. Makes HTTP request
Browser

Figure 3: Diagram displaying how the Companies page builds itself using filter variables. Arrows represent steps, labeled 0-8. Dotted lines indicate that either one function or the other will be called, but not both.

Getting the Data

In order to communicate with the Career Day database, we were provided a WCF web service API from the CCIT department. The service was deployed on the CSM webapps-test server and to integrate it with our application, we just needed to add calls to it within our code. To do this, we used a servicemodel metadata utility tool to auto-generate the service model code to be used with our program. This generated code makes up the bulk of our Model. It is a client API consisting of a list of classes in C# that know how to build themselves based on what is received from the database. Within this file, there are four important classes called Company, WantsToMeetWith, Major, and iCareerDayServiceClient (as shown in Figure 4). The iCareerDayServiceClient class has three methods that are used in our application called GetCompanyList() which returns all existing companies, GetCompanyById(int id) which returns information about a specific company, and GetMajorList() which returns all existing majors. These functions make calls to the web service and receive a JSON string of data in response. These strings are then deserialized into their corresponding objects, so that the HomeController receives the data as either Company objects or Major objects.

In the HomeController file, whenever information from the database is needed, an instance of iCareerDayServiceClient is instantiated, one of its three methods is called, and then the client is closed. Following the example from the previous section, when the Companies controller function needs to retrieve the list of companies from the database, the client is opened and the GetCompanyList() method is called to receive the array of Company objects. The auto-generated client API does all of the work involved with converting the raw data into the objects we need, so this made it very easy for us to integrate the web service with our application.

Any data needed for the website that may change but does not come from the web service API is gathered from configuration files. These are JSON files that hold information such as which PDFs to display and times and dates. The setup of the classes built from the configuration files can be seen in the UML diagram in Figure 4.

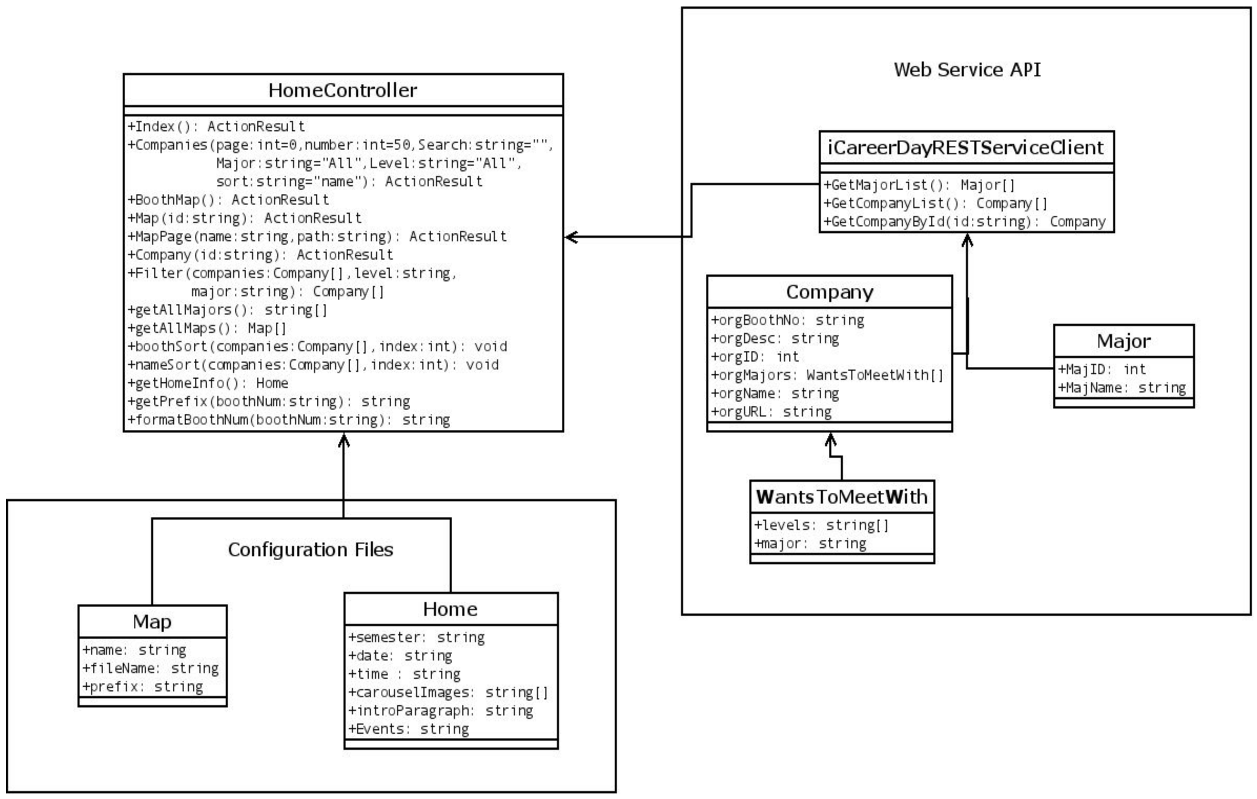

Web Service API
HomeController
+Index(): ActionResult
+Companies(page:int=0,number:int=50,Search:string="",
Major:string="All",Level:string="All",
sort:string="name"): ActionResult
+BoothMap(): ActionResult
+Map(id:string): ActionResult
+MapPage(name:string,path:string): ActionResult
+Company(id:string): ActionResult
+Filter(companies:Company[],level:string,
major:string): Company[]
+getAllMajors(): string[]
+getAllMaps(): Map[]
+boothSort(companies:Company[],index:int): void
+nameSort(companies:Company[],index:int): void
+getHomeInfo(): Home
+getPrefix(boothNum:string): string
+formatBoothNum(boothNum:string): string
iCareerDayRESTServiceClient
+GetMajorList(): Major[]
+GetCompanyList(): Company[]
+GetCompanyById(id:string): Company
Company
+orgBoothNo: string
+orgDesc: string
+orgID: int
+orgMajors: WantsToMeetWith[]
+orgName: string
+orgURL: string
Major
+MajID: int
+MajName: string
WantsToMeetWith
+levels: string[]
+major: string
Configuration Files
Map
+name: string
+fileName: string
+prefix: string
Home
+semester: string
+date: string
+time : string
+carouselImages: string[]
+introParagraph: string
+Events: string

Figure 4: UML diagram describing the classes used in the application.

Decisions

Throughout the course of the project, our team had to make several decisions on how best to implement the web application. Our main goal when facing each of these decisions was to make the application easy for CCIT to maintain and edit on their existing system architecture. CCIT's ability to maintain the web application was facilitated by the team's decisions of languages, to utilize an MVC architecture, how to have the app communicate with the existing Career Day database, and how to setup the configuration files.

Languages Used

The languages used for the website where C#, Razor (markup language), CSS, and Bootstrap in an ASP.NET framework. We developed the project using Visual Studio 2017 Community. This IDE provided a template for the website upon creating the project, which we were able to build off of. These languages and the IDE were selected due to constraints on the servers the website will be deployed to. CCIT informed the team which languages would be used at the start of the project. We had to meet CCIT's language requirements so they can easily maintain and deploy the site.

MVC Architecture

In setting up our code, we decided to use the MVC architecture. This was the structure our client preferred we use. It was beneficial to organize the code this way because it kept different components of the application separate. Thus, it allows for referring to chunks of code in different places and helps maintain a "Don't repeat yourself" or DRY mentality within our code.

Database API

The web service API was provided to us by Bryan Siebuhr, the manager of application systems at CCIT, because security measures restrict us in what data we can receive for the application. Our application makes calls to this service within the controller file and receives a list of objects containing information about companies attending Career Day.

Configuration Files

The client asked us to include configuration files in order to avoid hard-coding data that will change from semester to semester. There were already configuration files in the template code; however, we opted not to use them. These .config files contains a lot of unnecessary information our client would have to look through in order to find the correct place to edit. Because our client wished for the editing of the website to be as easy as possible, the .config files would not be optimal for her.

Instead, we decided to include two JSON files to act as configuration files that only included the settings specific to the client. One of these files controls information and photos on the homepage while the other controls map PDFs that will be uploaded to the website. Using JSON files works well for our purposes because only information that our client will be editing are included, and it is well-labeled that it will be easy to understand.

Results

The goal of this project was to create a web application for Career Day that would give students easier access to the information found in the hard copy of the Career Day program guide. This application pulls from the Career Day database in real time as the page is loaded and it currently runs on Google Chrome, Microsoft Edge, and Firefox. The website is also mobile friendly.

Performance Testing and Results

For this project, our client did not require any unit testing. While we did not do any unit testing with our website, we did quite a bit of practical testing, as well as an alpha test demo (described in the next section).

Our practical testing involved an in-depth exploration of the website each time we implemented a new feature. This meant that each time we added a new functionality, we would try it out from all angles, and click all links that could possibly be associated to ensure that the addition of these features did not cause errors elsewhere. Most of the time, we had multiple team members pull the new changes and investigate.

In addition to checking for anything that might yield an error message, we also investigated all new features to ensure that they would work the way they were intended to. For example, once the filter feature was added to our list of companies, we tested this by applying some filters, then checking several companies that appeared to make sure that they did indeed match the filter criteria. For further checking, we selected filter criteria that we knew would bring up few results and looked to see if the resulting companies list was as we expected.

Outside of testing specific features, many team members also took time to play around on the website without looking for anything specific. We used the site in a way we expected an end user may use it. Doing this, we found a few bugs that were then corrected.

Alpha Testing and Results

During Week 3, we performed an Alpha test with the students from the Physics field session. During this test, we demonstrated the alpha version of our website for these students, explaining its features, and then allowed them to explore the site themselves on their laptops or mobile phones. After this, 28 students filled out a survey to let us know what they thought of the app.

In the first question, we asked the students to answer four questions on a scale of one to five. The questions, listed in order, were: how likely are you to use this app in planning for career day, how likely are you to use this app during the career fair, how easy is it to navigate the site, and how aesthetically pleasing do you find the site? The results from the 28 students who took the survey, shown in Figure 5, were quite good. The first three questions scored over 3.75 on average, with the last question scoring just above a three. This means that students find the app useful and easy to use, though only averagely attractive.

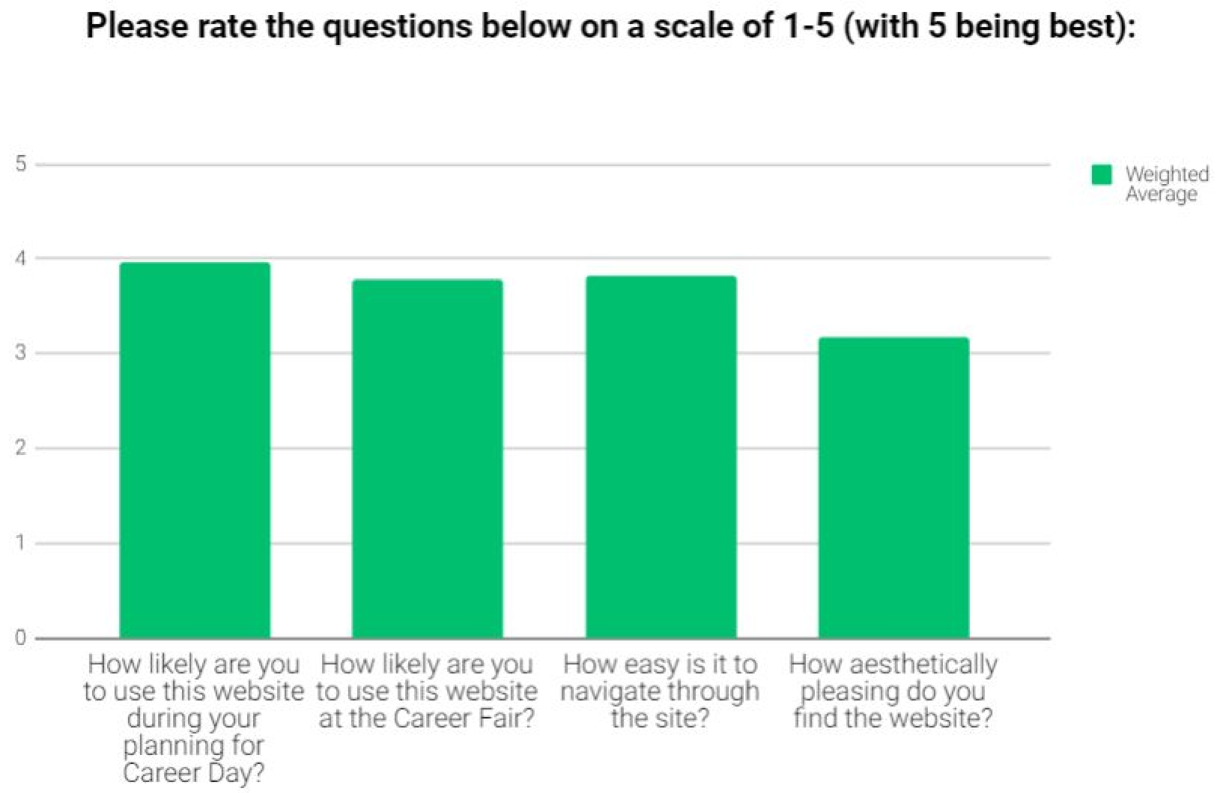


Figure 5: Bar graph displaying survey results for the first four survey questions

In Questions 2 and 3, we asked if the filtering feature for the Companies page and the ability to change results per page, respectively, were helpful. 92.86% of students said yes to Question 2, and 75% to Question 3. This indicates that the features we added to our site were practical and useful.

Questions 4 and 5 were open ended. We asked if there were additional filters that students would like to see, and if they had other recommendations for the site. From this, we found that several students would like the ability to filter companies by industry, location, and what kind of job they are offering. While it is not currently possible to implement these filters since the information required is not currently in the Career Day database, it is something to consider for the future.

Overall, we received good feedback that indicated our app is on track to be useful for the users. We also received several good suggestions for features to implement in the future.

Features Not Implemented

At the beginning of the project, we discussed with our client the possibility of tracking usage statistics on the site to see who is using the site, and what they are doing while using it. However, this would have required using tracking cookies which is a privacy issue as well as permission from CCIT. Since we never got word from CCIT on whether or not this would be permissible, this feature was never implemented. However, this feature could still possibly be implemented in future work.

Work for Later Phases

Currently, the web application does not filter companies based on industry or other suggested criteria because there is no entry for this data in the Career Center's database. Once this data is added, then the app can query that information and store it within the company objects. In addition, another entry's addition to the database would allow it to store the logo of each company, and then add the logo to the individual company page. Another potential feature would be to make the booth maps interactive so that if the user clicks on a booth, it takes them to the company home page.

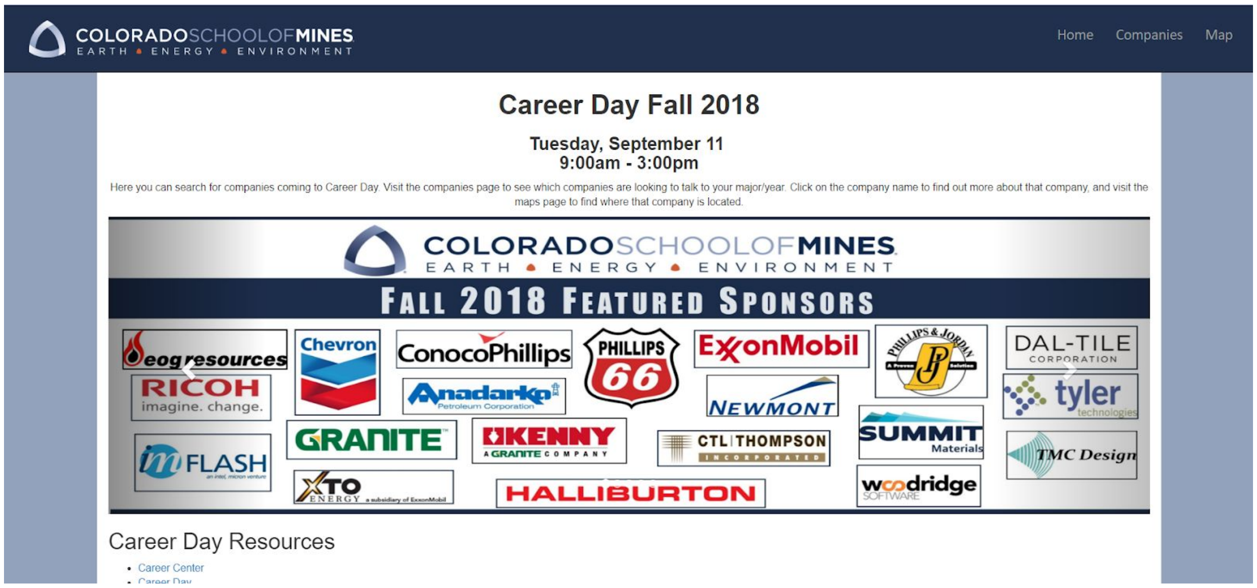
Lessons Learned

- We originally had issues testing our application on mobile as there was no way for us to host the website on a server without going through CCIT. However, we were able to use developer tools in Google Chrome that allowed us to test the the mobile version in a timely manner. By going to the Google Chrome menu and selecting "More Tools," then "Developer Tools," we were able to access a page that allowed us to switch to a mobile view. Using this, we did not have to purchase an expensive emulator or rely on limited-time trials. We learned that Google Chrome has additional tools that were extremely helpful for our project and could be useful for future development projects.

- While pair programming was quite helpful, we also had success programming as a group with the entire team working at one machine. When we ran into a large or difficult problem while programming, we would connect one computer to a projector so that all team members could see the working code and offer suggestions. By doing this, we were able to efficiently solve problems with the code and work through design issues. However, we would not recommend doing this for an entire project, since it is more effective to divide up smaller problems so that we could tackle more issues at once.

Appendix

This appendix includes screenshots of the website so that the reader may better understand the site's layout and what information is included on each page.


COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT
Home
Companies
Map

Career Day Fall 2018

Tuesday, September 11
9:00am - 3:00pm

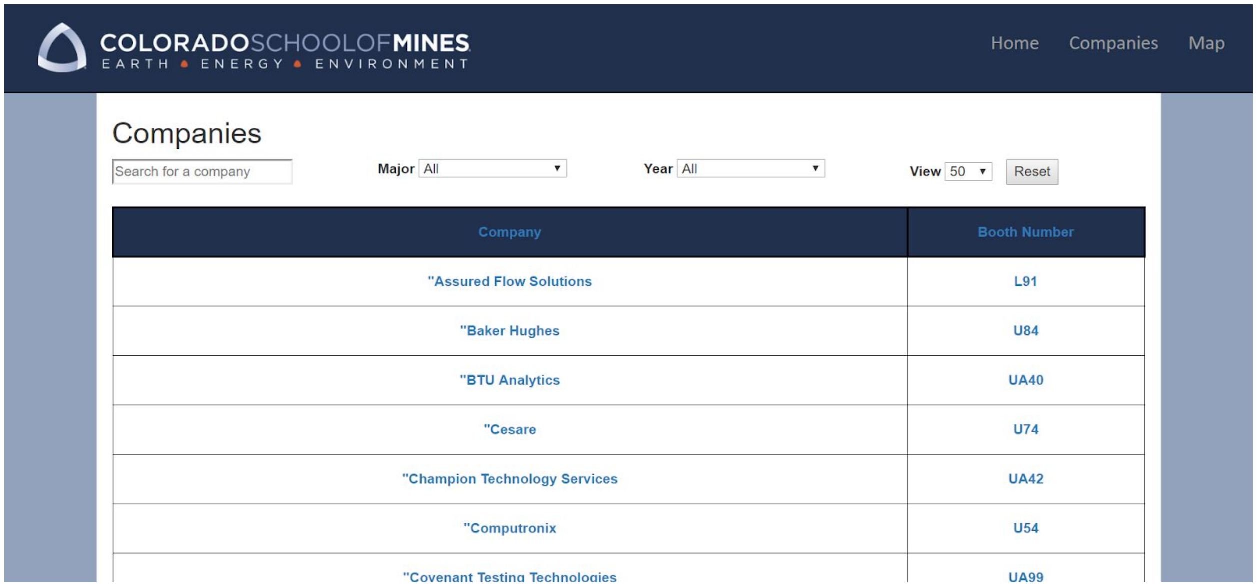
Here you can search for companies coming to Career Day. Visit the companies page to see which companies are looking to talk to your major/year. Click on the company name to find out more about that company, and visit the maps page to find where that company is located.

COLORADOSCHOOLOFMINES
EARTH • ENERGY • ENVIRONMENT
FALL 2018 FEATURED SPONSORS

Career Day Resources

- Career Center
- Career Day

Figure A1: A view of the top portion of the home page. This is what the user would see upon navigating to the website.

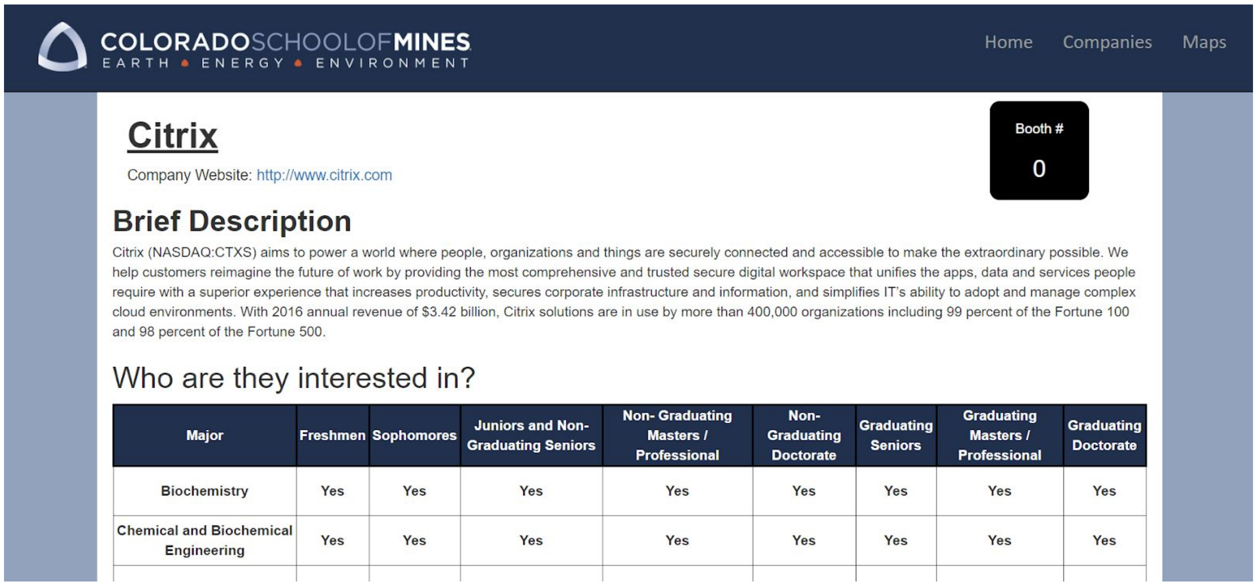


Companies

Search for a company · Major All · Year All · View 50 · Reset

Company	Booth Number
"Assured Flow Solutions	L91
"Baker Hughes	U84
"BTU Analytics	UA40
"Cesare	U74
"Champion Technology Services	UA42
"Computronix	U54
"Covenant Testing Technologies	UA99

Figure A2: A view of the top portion of the Companies page. The user can search for companies using the search bar and dropdowns at the top of the page. The page is accessed by clicking on the "Companies" tab on the navigation bar.



Citrix

Company Website: http://www.citrix.com

Booth #

0

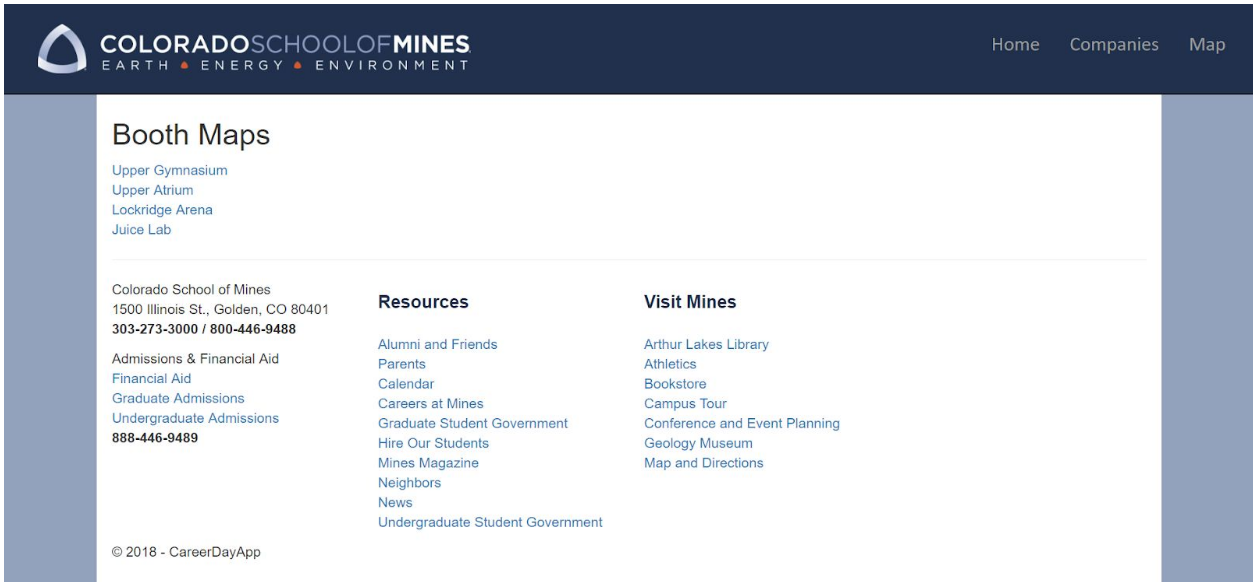
Brief Description

Citrix (NASDAQ:CTXS) aims to power a world where people, organizations and things are securely connected and accessible to make the extraordinary possible. We help customers reimagine the future of work by providing the most comprehensive and trusted secure digital workspace that unifies the apps, data and services people require with a superior experience that increases productivity, secures corporate infrastructure and information, and simplifies IT's ability to adopt and manage complex cloud environments. With 2016 annual revenue of $3.42 billion, Citrix solutions are in use by more than 400,000 organizations including 99 percent of the Fortune 100 and 98 percent of the Fortune 500.

Who are they interested in?

Major	Freshmen	Sophomores	Juniors and Non-Graduating Seniors	Non- Graduating Masters / Professional	Non-Graduating Doctorate	Graduating Seniors	Graduating Masters / Professional	Graduating Doctorate
Biochemistry	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Chemical and Biochemical Engineering	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Figure A3: A view of the top portion of an individual company page. This page can be located by clicking on the company name on the Companies page.



Booth Maps

Upper Gymnasium
Upper Atrium
Lockridge Arena
Juice Lab

Colorado School of Mines
1500 Illinois St., Golden, CO 80401
303-273-3000 / 800-446-9488

Admissions & Financial Aid
Financial Aid
Graduate Admissions
Undergraduate Admissions
888-446-9489

Resources

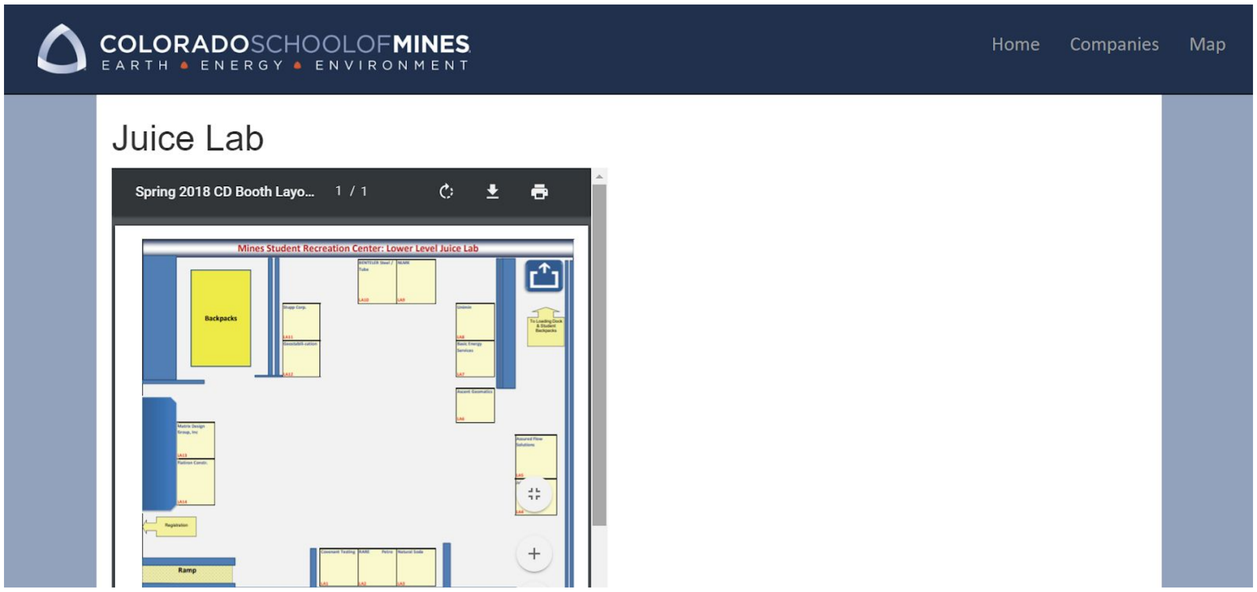
Alumni and Friends
Parents
Calendar
Careers at Mines
Graduate Student Government
Hire Our Students
Mines Magazine
Neighbors
News
Undergraduate Student Government

Visit Mines

Arthur Lakes Library
Athletics
Bookstore
Campus Tour
Conference and Event Planning
Geology Museum
Map and Directions

© 2018 - CareerDayApp

Figure A4: A view of the Booth Maps page, accessed by clicking the "Map" tab on the navigation bar. Each of the listed booth maps links to the individual map page. This view also includes a view of the website footer, which includes links to other various Mines resources and is present on every page.



Juice Lab

Figure A5: A view of the top of an individual map page. This page displays a PDF of a booth map. This page is accessed from one of the links on the Booth Maps page.