

January 2016

# Workflow Web Application Design

Hui Zheng

*Worcester Polytechnic Institute*

Jiacong S. Xu

*Worcester Polytechnic Institute*

Follow this and additional works at: <https://digitalcommons.wpi.edu/mqp-all>

---

## Repository Citation

Zheng, H., & Xu, J. S. (2016). *Workflow Web Application Design*. Retrieved from <https://digitalcommons.wpi.edu/mqp-all/4106>

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact [digitalwpi@wpi.edu](mailto:digitalwpi@wpi.edu).



# WORKFLOW WEB APP DESIGN

A Major Qualifying Project for Angelo, Gordon & Co., New York City

Submitted to the faculty of the  
WORCESTER POLYTECHNIC INSTITUTE  
In partial fulfillment of the requirements of the  
Degree of Bachelor of Science

By

**Jiacong Sophie Xu**, Industrial Engineering  
**Hui Zheng**, Computer Science

Project Sponsor: Angelo, Gordon & Co. (Project ID: *KMS-AAVF*)

**Submitted to:**

On-site Liaisons: Scott Burton, Managing Director  
Cindy Aguilar, IT Department Administrator

Project Advisors: Arthur Gerstenfeld, School of Business  
Kevin Sweeney, School of Business, Wall Street Project Center Director  
Michael Ciaraldi, Department of Computer Science  
Jon Abraham, Department of Mathematical Sciences

**Submitted on:**

December 18th, 2015

245 Park Ave p. +1.212.692.2000  
New York, NY 10167 f. +1.212.867.9328

information@angelogordon.com  
<https://www.angelogordon.com/default.aspx>

# Abstract

---

We worked with Angelo Gordon & Co. IT Department to optimize both the candidate-searching and on-boarding processes. The goal of this project is to design a business tracking system that deals with position seeking, candidate selecting and on-boarding procedures. Our ultimate goals are candidate data storage and extensive future use.

We created a web application called Angelo Gordon Hire Direct (AGHD). It is considered as a workflow web app to initiate candidate seeking for consulting positions. It can also exchange and store vendor and candidate information, schedule for interviews and make final candidate selections. This web app also provides managers and administrators quicker overviews with detailed information.

The end goal is set for documentation purposes. AGHD should provide proofs when being reviewed by the Legal Department for purposes of being just, fair and open. Along with the functionalities and goals, having their own hiring system for keeping track of all documents, standardizing the process, reducing the workload, and storing information for future references.

# Acknowledgements

---



We would like to express our appreciation to Worcester Polytechnic Institute (WPI) and the Interdisciplinary and Global Students Division (IGSD) for making arrangements for us to join the Wall Street Project Center. Given the opportunity for working with the firm, our team had a really grateful experience in finishing the project. We would like to thank Professor Kevin Sweeney, Professor Arthur Gerstenfeld, Professor Jon Abraham and Professor Michael Ciaraldi for advising our project and providing us with guidance and encouragement each week. This project would not have been possible without their effort. We would like to express our gratitude to all of the academic advisors for this opportunity.



We would like to thank Angelo Gordon & Co (AG) for providing us with a wonderful working environment and the resources needed to construct our project. Thanks to our sponsor Scott Burton, and Cindy Aguilar for overseeing our project as advisors during our time at AG, and Eric Entenman for deploying our code into the AG system, together with the staff members for welcoming us as co-workers. We really enjoyed our stay here and we are very grateful to the company IT Department for their willingness to take time allowing us to bring our ideas and contributions, answering our questions, and providing guidance and support.

# Table of Contents

I.	<b>Abstract</b> .....	<b>ii</b>
II.	<b>Acknowledgements</b> .....	<b>iii</b>
III.	<b>Table of Figures</b> .....	<b>vi</b>
IV.	<b>Executive Summary</b> .....	<b>vii</b>
V.	<b>Background</b> .....	<b>9</b>
	1 Workflow.....	9
	2 Web-based Application.....	10
	3 Single-page Web Application .....	13
	4 Model-view-controller (MVC) .....	14
	5 Full-stack Software Development.....	16
VI.	<b>Chapter 1: Introduction</b> .....	<b>20</b>
	1.1 Angelo, Gordon & Co. IT Department .....	20
	1.2 Current Situation .....	20
	1.3 Motivation .....	22
	1.4 Making the process easier .....	22
VII.	<b>Chapter 2: Angelo Gordon Hire Direct (AGHD) – A Workflow Web Application</b> .....	<b>24</b>
	2.1 General Goal Overview .....	24
	2.1.1 Process Overview .....	24
	2.1.2 Process Flowchart .....	25
	2.1.3 Form Examples.....	27
	2.2 User Interface Design.....	28
	2.2.1 Initial Setup.....	29
	2.2.2 Log In .....	33
	2.2.3 Create New Functionality .....	34
	2.2.4 Vendor Selection.....	36
	2.2.5 Administrator Process View .....	37
	2.2.6 Tracker Drop-down Selections and Buttons .....	39
	2.2.7 Quick Emails and Reminders within Process.....	40
	2.2.8 Interview Scheduler.....	41
	2.2.9 File Uploading and Downloading.....	42
	2.2.10 Making Final Decisions .....	44

2.2.11	Submit Files to HR.....	45
2.3	Implementation.....	46
2.3.1	Frontend Development.....	46
2.3.2	Backend Development.....	47
2.3.3	Database.....	47
<b>VIII.</b>	<b>Chapter 3: Technology.....</b>	<b>48</b>
3.1	JavaScript (JS).....	48
3.2	Node.js.....	48
3.3	Angular.js.....	50
3.4	HTML.....	51
3.5	CSS.....	52
3.6	Express.js.....	53
3.7	SQL Server.....	54
3.8	Modal Window.....	55
3.9	RESTful Web Service.....	55
<b>IX.</b>	<b>Chapter 4: Results and Limitations.....</b>	<b>58</b>
4.1	Results.....	58
4.2	User Experience.....	58
4.3	Limitations.....	59
<b>X.</b>	<b>Chapter 5: Future Work and Recommendations.....</b>	<b>64</b>
5.1	Extensions and Recommendations.....	64
5.1.1	More functionalities.....	64
5.1.2	Database View.....	64
5.1.3	Candidate Information Scanning.....	65
5.1.4	Registration System.....	65
5.2	Future Work.....	65
<b>XI.</b>	<b>References.....</b>	<b>66</b>
<b>XII.</b>	<b>Appendix A: Candidate Application Form.....</b>	<b>68</b>
<b>XIII.</b>	<b>Appendix B: User Guide.....</b>	<b>69</b>
<b>XIV.</b>	<b>Appendix C: Code.....</b>	<b>80</b>

# Table of Figures

---

Figure 1 Workflow Example .....	9
Figure 2 Web Application Illustration.....	11
Figure 3 How Applications Work.....	12
<b>Figure 4 Web Application Structure.....</b>	<b>13</b>
Figure 5 Modern Web Application Architecture.....	14
Figure 6 MVC.....	15
Figure 7 The Modern Developer's MVC Pattern.....	16
Figure 8 A Modern Web Development Stack.....	17
Figure 9 Full-stack.....	18
Figure 10 Hiring and On-boarding Process Overview .....	25
Figure 11 Candidate-Searching Process Breakdown.....	26
Figure 12 On-boarding CAO Steps .....	26
Figure 13 On-boarding Final Steps.....	27
Figure 14 Sample On-boarding Form.....	28
Figure 15 Initial Setup 1 - Settings .....	30
Figure 16 Initial Setup 2 – Add vendors.....	30
Figure 17 Vendor MSA.....	31
Figure 18 Initial Setup 2 – Email Settings .....	32
Figure 19 Log-in View.....	33
Figure 20 Banner .....	34
Figure 21 Initial Search Progress Bar.....	35
Figure 22 Create New.....	35
Figure 23 Vendor Selection .....	36
Figure 24 Home Page.....	38
Figure 25 Drop-down Menu.....	39
Figure 26 Vendor Email Template.....	40
Figure 27 Interview Scheduler.....	41
Figure 28 Date Picker.....	42
Figure 29 New Candidate Entry.....	43
Figure 30 Final Decision Page .....	44
Figure 31 Submit Files to HR.....	45
Figure 32 Frontend.....	46
Figure 33 Node.js Server.....	49
Figure 34 AngularJS.....	50
Figure 35 HTML.....	51
Figure 36 HTML Tag.....	52
Figure 37 CSS.....	53
Figure 38 RESTful Web Service.....	56
Figure 39 REST Web Service design structure.....	57
Figure 40 CIO Approval Request View.....	61
Figure 41 Approve Message.....	62
Figure 42 Decline Message .....	62
Figure 43 Application Form.....	68

# Executive Summary

---

Angelo Gordon & Co. is an investment firm that is primarily dedicated to alternative investing. The IT Department of the firm is currently on-boarding a few consultants, and the process of hiring is not ideally optimized in a way as the administrator desired. The administrators are now communicating via emails and other media to perform back and forth information exchanges, which is complicated and time-consuming. Our goal is to create a web-based workflow application to handle all information coming from or to the consultant vendors, IT Departments, and Human Resources, including department administrator, hiring managers, managers, and the Chief Information Officer (CIO).

There are two major parts of the entire selection procedure: the hiring process and the on-boarding process. After identifying a need of a consulting position, the administrator of the department passes all related information including job title, department and job description to the CIO for approval. Once it has been approved, an email is generated by the administrator and sent to selected vendors with all information provided as attachments. If they have recommended candidates for the position, they will fill in the forms with the candidates' files attached. Then it follows the interview selection process. Once a candidate passes the interview and is selected by the department hiring manager, the on-boarding process begins. The administrator then gathers all the files of the candidate such as background checks, work orders, Master Service Agreements (MSA) for their vendors, resumes and cover letters, etc., and then forward them to Human Resources for the next steps.

The web application that we created is called **Angelo Gordon Hire Direct (AGHD)**. Based on the current situation within the AG IT Department, this app was designed to function as a platform for different department administrators to enter information related to the consultants they are planning on hiring, which provides them an easy path of sharing required information across departments. Instead of making phone calls or generating emails back and forth requesting files, or delivering files in person, what an administrator needs to do now is simply sending files required by both of the hiring and on-boarding processes within the system to the vendors or other departments, and after they fill out the fields, the information would be automatically stored into the system for further use.

In addition to information exchanges, AGHD tracks the hiring and on-boarding status for the department. The goal of AGHD is to provide an overview of each of the hiring processes to the managers and hiring managers so that they have a general knowledge of what the current situations are. The current app is designed with single user login system. The department administrator is allowed to access all hiring positions with all editing



permissions, while the CIO performs approval actions toward a certain position. On the contrary, when the CIO decides to reject a request, the administrator is required to stop proceeding with the rejected process, and authorized to reprocess every entry and then re-submit. The system would send the managers and hiring managers notification emails, which allows them to supervise along each process.

Specifically, AGHD authorizes the administrator all access including editing the information stored in the system for both selecting new consultants and on-boarding employed consultants in all aspects, together with providing managers and hiring managers who are involved information to the system.

The application involves functionalities of initiating a process, requesting for approvals, selecting vendors, scheduling for interviews, make final candidate selections, and forward all files to HR. We designed an Initial Setup page that functions as a settings tab for the application. The user may add new vendor, candidate or possible position information, modify all records, and pre-enter all hiring manager names as preferred. After entering all information, the names of each category appear within the drop-down lists as options to prevent complicated text entries, thus reduces error rate.

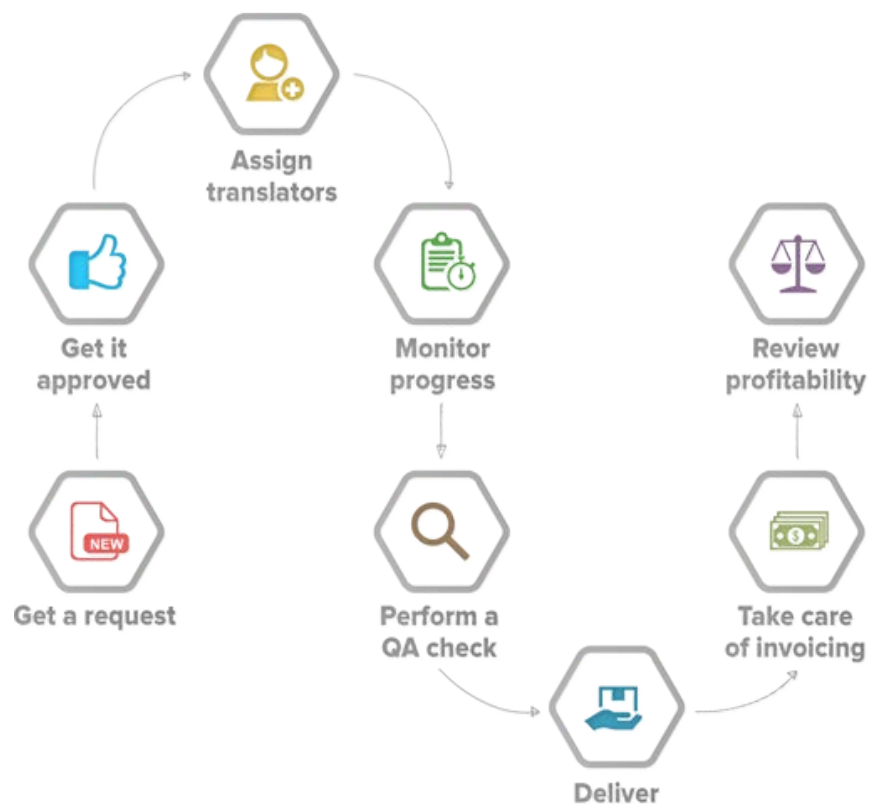
The delivery of this program is to bring the IT Department a tracker system in the business processes that involves information exchange and data storage. This increases the efficiency of the entire consultant hiring and on-boarding processes and reduces redundant communication among departments. It also shows clearly about the current situation such as where the files are, which position is currently been processing by which hiring manager, and what needs to be done by which department. Our final goal is to let the departments with hiring needs widely adopt our system, facilitate the process, and also treat the system as a documentation platform to provide the Legal Department sufficient proofs for each position searching procedure.

# Background

## 1 Workflow

Workflow (Workflow, n.d.) consists of the progression of steps, including tasks, events, interactions, that comprise a work process, create or add value to the organization's activities. Each step is dependent on occurrence of the previous step which involves two or more persons. It may be viewed as one fundamental building block to be combined with other parts of an organization's structure such as information technology, teams, projects, and hierarchies.

Workflow is the definition, execution and automation of business processes where tasks, information or documents are passed from one participant to another for action, according to a set of procedural rules (Workflow Tutorial, n.d.). Workflow trackers and information sharing functions are necessary components for business firms to keep a record of all information cross departments that saves time and reduces work. Organizations use workflows to coordinate tasks between people and synchronize data between systems, with the ultimate goal of improving organizational efficiency, responsiveness and profitability. An example of an order workflow is listed in *Figure 1*.



*Figure 1 Workflow Example*

Workflows automate the flow of employee tasks and activities, reducing the time the process took to complete as well as potential errors caused by human interaction. It makes processes more efficient, compliant, agile, and visible by ensuring that every process step is explicitly defined, monitored over time, and optimized for maximum productivity.

There are three different types of workflows: Sequential Workflow, State Machine Workflow, and Rules-driven Workflow. Sequential Workflow is typically flow chart based, and it progresses from one stage to next and does not step back. State Machine Workflow progresses from state to state, which is more complex and may return to a previous point if required. While Rules-driven Workflow is implemented based on a Sequential Workflow, whose rules dictate the progress of the workflow.

Workflow exists in various industries: Human Resources, Banking, Manufacturing, Customer Service, Defense/Emergency, and Travel. Workflow software provides organizations with the technology functionality, enabling them to innovate and create new working environments that not only define the business process and tasks, but execute and run them through common day-to-day applications like Outlook and Word. It improves productivity which reduces time spent on manual tasks, and it enables organizations to react quickly and smoothly to market changes through process modifications. It provides the feature of executing business functions, creates productivity measurement and improves continual process, thus increases accountability. Additionally, the managers would be able to see what is happening with business critical processes at every point at any moment.

## **2 Web-based Application**

A Web application, also called Web App, (What Is Web Application (Web App), n.d.) (Web Application, n.d.) is an application program that is stored on a remote server and delivered over the Internet through a browser interface. It is an application in which all or some parts of the software are downloaded from the web each time it is run. By having a web application system, the users would be able to use web browsers as clients to update and maintain web applications without distributing or installing software on many computers. It may refer to browser-based apps that run within the user's Web browser. In addition, web apps support cross-platform compatibility. Common web applications include webmail, online sales and auctions, wikis, instant messaging services, and many other functions.

Web application has three types: Browser Based, Client Based, and Mobile Web Apps (Definition of: Web application, n.d.). From a technical point of view, the web is a highly programmable environment that allows mass customization through the immediate deployment of a large and diverse range of applications to global users (Web Applications: What are They? What of Them?, n.d.). Similar to web browsers, software applications that allow users to retrieve data and interact with content located on web pages within a website, web applications are computer programs allowing website visitors to submit and retrieve data to/from a database over the Internet using their preferred web browser. The data is then presented to the user within their browser as information is generated dynamically by the web application through a web server. The documents are generated in a standard format to allow support by all browsers.

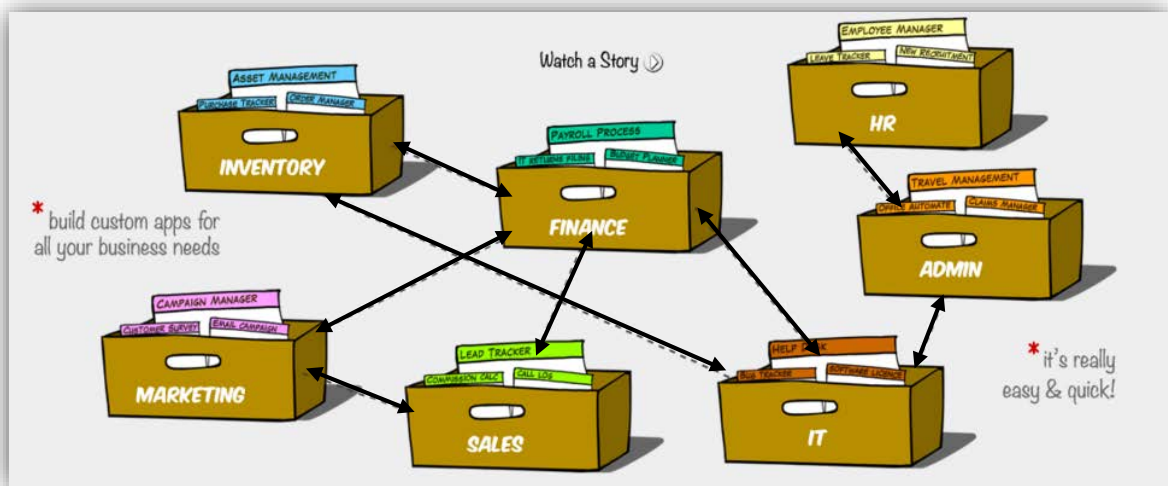
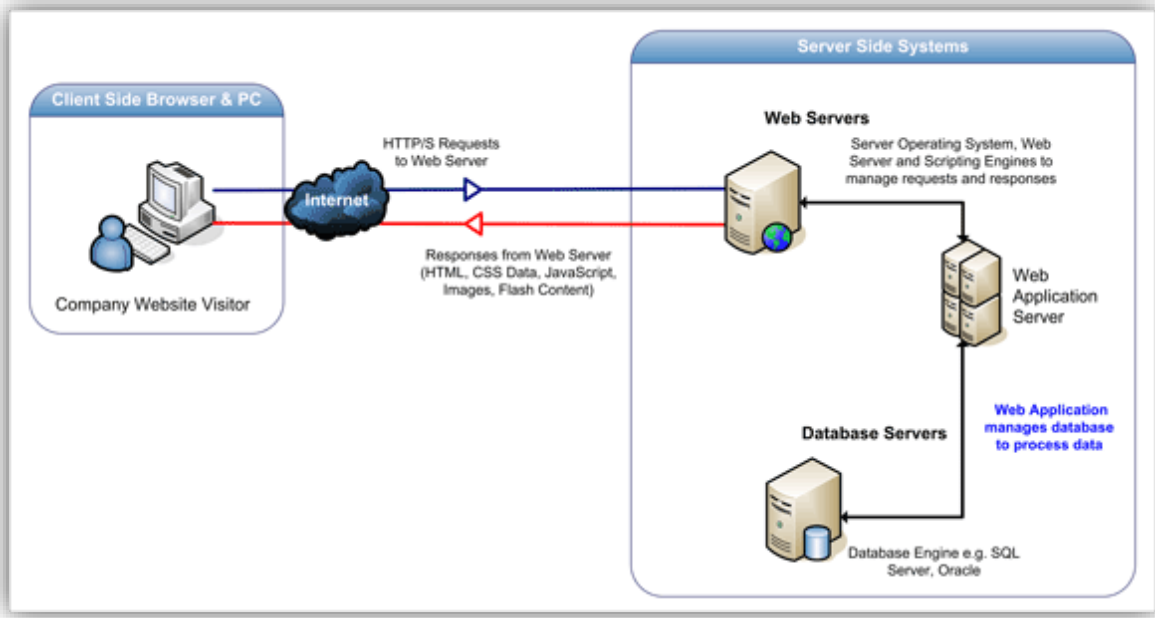


Figure 2 Web Application Illustration

Above is an example from a website called Zoho, which illustrates a complicated relationship among departments. A significant advantage of building such web applications is that they perform their function irrespective of the operating system and browsers running client side. Web apps are quickly deployed anywhere at no cost and without any installation requirements at the user's end.



*Figure 3 How Applications Work*

*Figure 3* details the three-layered web app model: first layer is a web browser or the user interface; second layer is the dynamic content generation technology tool (JSP or ASP); third layer is the database of all data.

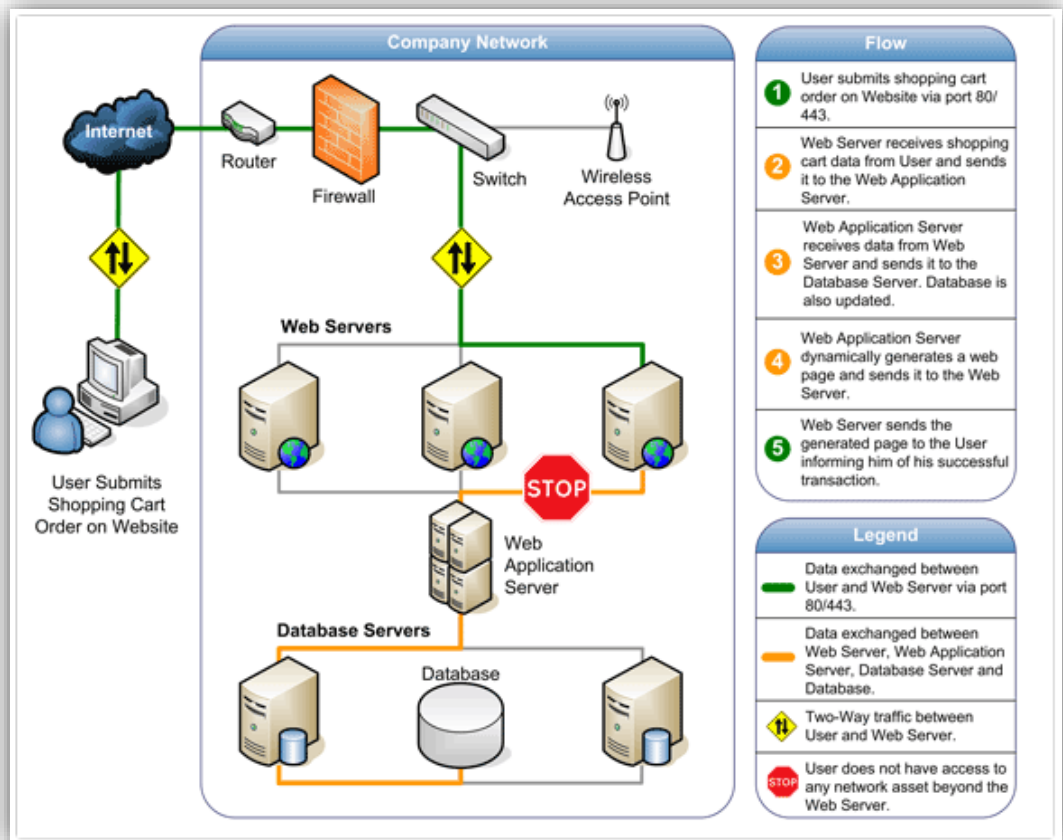


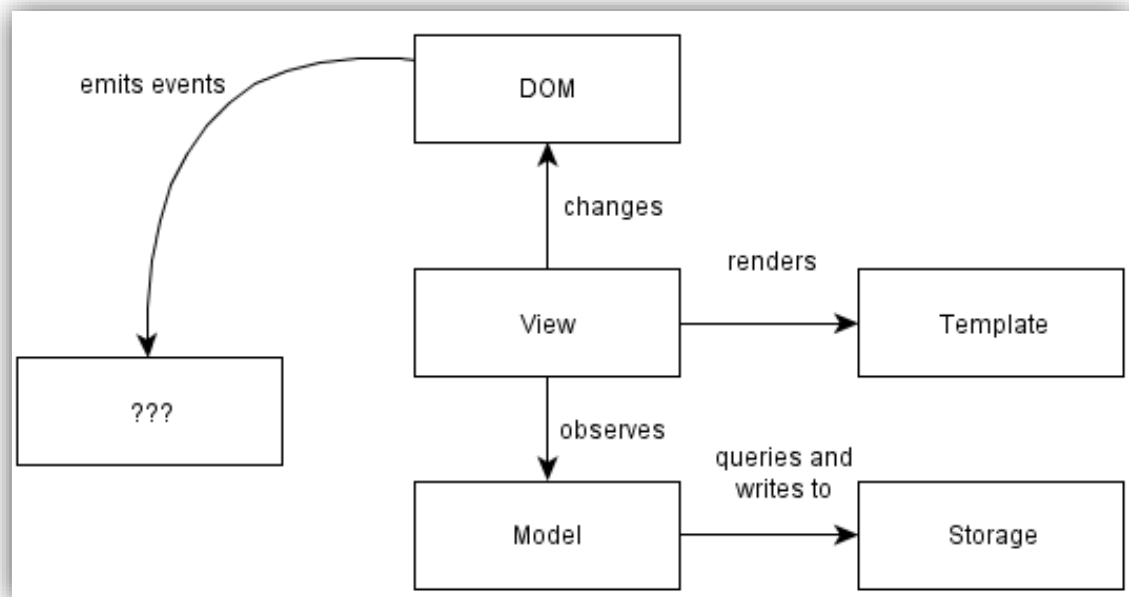
Figure 4 Web Application Structure

Figure 4 shows how the initial request is triggered by the user through the browser over the Internet to the server. Web apps access the database servers to perform the requested tasks, and update and retrieve data within the database. Then they present the information collected to the user through the browsers they use.

### 3 Single-page Web Application

A Single-page Application (SPA) (Single-page application) is a web application that fits on a single web page with all necessary code retrieved with one single page. The page will not reload at any point of the process or control transfer to another page. It allows people to redraw any part of the UI without requiring a server roundtrip to retrieve HTML (Single page apps in depth, n.d.), which can be achieved by separating data by having a model layer that handles data and a view layer that reads from the models. SPA always allows us to offer a more native-app-like experience to the users.

A modern single page app is generally structured like *Figure 5*, which indicates that programmers would be able to write code that takes advantage of standard mechanisms:



*Figure 5 Modern Web Application Architecture*

Supporting rich interactions with multiple components on a page means that those components have many more intermediate states. Server-side rendering is hard to implement since small view states do not map well to URLs. A modern web app is structured from three different perspectives: Architecture, Asset packaging, and Run-time state. Architecture is the conceptual part that considers the different parts of the app, while Asset packaging cares about the structural aspect of the app. Run-time state is about when to load into the browser and what is stored in the memory.

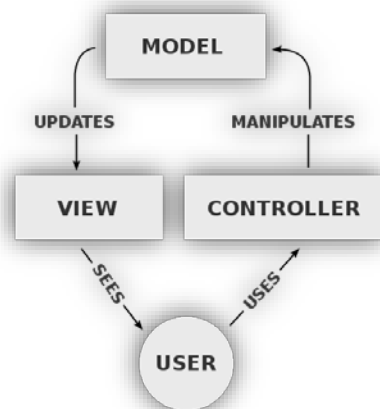
Experienced programmers keep the amount of code at a minimum to achieve lower maintenance costs. Keeping things on one page and reducing complexity has been a goal for most programmers and this single-page web application is so far one of their best options.

#### 4 Model-view-controller (MVC)

Model-view-controller (MVC) is a software architectural pattern for implementing user interfaces (Model-view-controller, n.d.). It divides a software application into three parts

so as to separate internal representations of information. Traditionally, MVC was used for desktop graphical user interfaces and it was one of the first approaches to describe and implement software constructs in terms of their responsibilities. The MVC pattern has subsequently evolved and the use of the MVC pattern in web applications exploded in popularity after Spring Framework. MVC web frameworks now hold large market shares relative to non-MVC web toolkits.

MVC expresses the “core of the solution” to a problem and allows it to be adapted for each system. It is an architectural pattern primarily used in creating Graphic user interfaces (GUI) which is based on modularity to separate three different aspects of the GUI. The three parts are: model (data), view (visual representation), and controller (interface between the view and the model). The idea for keeping these components separate is so that each one is as independent of the others as possible, and changes made to one will not affect changes made to the others, which makes it possible that the GUI can be updated with a new look or visual style without having to change the data model or the controller (Model–view–controller, n.d.). A typical collaboration of the MVC components is attached below (*Figure 6*):



*Figure 6 MVC*

The central component of MVC, the *model*, captures the behavior of the application in terms of its problem domain, independent of the user interfaces. It directly manages data, logic and rules of the application. A model can respond to requests for information, instructions to change the state of its information, and even to notify observers in event-driven systems when information changes. This could be a database, other structures or storage systems. In other words, a model is the data and data-management of the application. A *view* can be any output representation of information such as charts or



diagrams, and multiple views of the same information can be represented by bar charts or tabular views for accountants. It effectively provides the user interface element of the application and it will render data from the model into a form that is suitable for the user interface. *Controller* accepts input and converts it to commands for the model or view to perform appropriate actions. A *view controller* generates an output view and an embedded controller.

Although MVC was originally developed for desktop computing, it has been widely adopted as web applications in major programming languages. It is usually best to allow the model to do all necessary validation of values so that any changes to the allowed values, or changes simply to the validation process, only need to be made in one place. In some cases, this might not be possible. The model could be made to have an additional method that takes text as input, or the controller will do initial parsing of the text to get the numeric value and then pass it on to the model to do further validation. When either the controller or the model determines validation, the controller will need to tell the view that the value is invalid. Sometimes, the view may then issue an error dialog or notification for these cases. The MVC design pattern has a UI that is encapsulated in a View, the Browser Web Page.

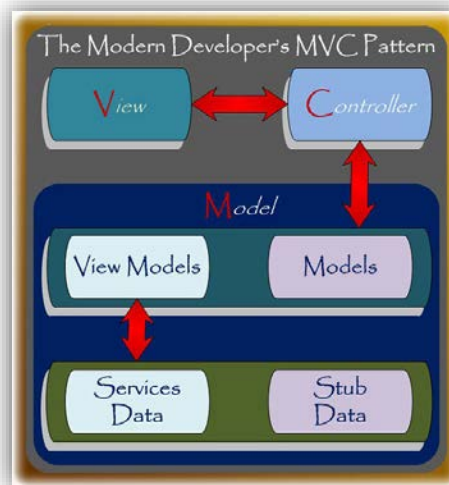


Figure 7 The Modern Developer's MVC Pattern

## 5 Full-stack Software Development

The most important thing for a web developer to learn today is a robust client-side MVC framework. Users will simply not use apps which behave poorly. To perform well and provide a rich experience, the client can no longer be requesting whole pages from

the server and waiting for the response to render them. Thus the latest evolution of web development looks like this (Figure 8):

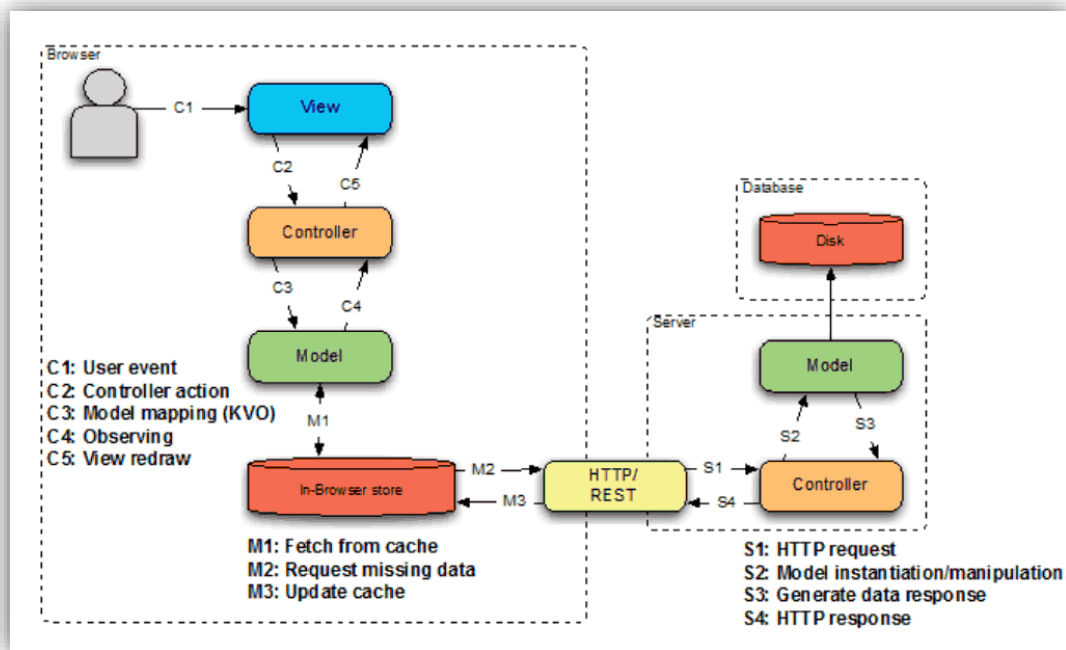


Figure 8 A Modern Web Development Stack

The client and server talk in HTTP via a Representational State Transfer (REST) pattern. Server views are used very little and client MVCs take raw data from the server's controller and render HTML on their own in the browser. This allows a much more dynamic and rich experience for the user and enhances performance due to the user's machine taking over a lot of computing instead of the central server doing all the heavy lifting for everyone. People can use a variety of different tools to put a stack together (Visualizing a modern web development stack with a diagram, n.d.).

Full stack development has many layers (What is a Full Stack Developer?):

- Server, Network, and Hosting Environment:
  - This involves understanding the possibility of breaks, appropriate use of file system, cloud storage, network resources, and data redundancy and availability.
- Data Modeling:
  - Know how to create a reasonably normalized relational model.
  - Complete with foreign keys, indexes, views, lookup tables, etc.

- Business Logic:
  - This is the heart of the value the application provides which requires solid object oriented skills and frameworks.
- API Layer/ Action Layer/ MVC:
  - It relates to how the outside world operates against the business logic and data models with heavy frameworks.
  - Full stack developers need to have the ability to write clear, consistent, and simple to use interfaces.
- User Interface (UI):
  - Full stack developers need to understand how to create a readable layout so that a good visual design can be implemented.
- User Experience:
  - Users basically want things to work which requires a good system that doesn't give its users a hard time getting things done.
  - Useful error messages are essential for the design when things break.
- Understanding what the customer and the business need:
  - With the understanding of what has been going on within the field and when the customers use the software, the developers may have a grasp of what their users really need.



*Figure 9 Full-stack*

(Being a Full Stack Developer)

The term full-stack means developers who are working with both back-end and front-end technologies. More specifically, it means that developers are able to work with databases, PHP, HTML, CSS, JavaScript and everything in between, together with converting Photoshop designs to front-end code. Developers doesn't need to master all of the areas and technologies, but be comfortable working with those technologies. Being a full-stack developer today means that a lot of skills are required: Linux and basic shell scripting, Cloud computing, Background processing, Searching, Caching, and Monitoring. Some web development tools for Version control and Virtualization are necessary as well. Having separated development environments on a project basis is easy for virtualization tools. Back-end technologies include Web Servers, Programming languages such as PHP and Ruby, Databases like MySQL, MongoDB, etc. Front-end technology covers HTML, CSS, and JavaScript, compatibility quirks across browsers, Responsive design, AJAX, JSON, XML, and WebSocket. In the designs part, converting website design into front-end code, UI and UX are all counted as the crucial section within the field. In addition to front-end technologies, full-stack developers also understand what is possible and what not to create with the constraints of HTML/ CSS/ JavaScript, and convert the design accordingly (Being a Full Stack Developer).

With an open mind towards new technologies, developers need to have an understanding of how a web application is done from the concept to design to the finished product, and the ability to communicate intelligently between team members and to be a good asset if the situation needs it.

# Chapter 1: Introduction

---

## **1.1 Angelo, Gordon & Co. IT Department**

Angelo, Gordon & Co. (AG) is a privately-held registered investment advisor which was founded in 1988 and manages approximately \$27 billion. The company has expertise in a broad range of absolute return strategies for both institutional and high net worth investors. The company focuses on fundamental research complemented by strict adherence to the disciplined style – focus on the downside, diversify and use leverage judiciously. The research analysts specialize in a number of different but related investment strategies, with a goal of meeting the common goal of understanding the risks and rewards of each investment opportunity. The investment process involves significant analytical experience in equity and debt markets. The Internal and external resources are used to come up with ideas and analyze investment opportunities that generate attractive risk-adjusted returns. Some strategies include Credit Strategies, Real Estate Strategies, Private Equity Strategies, and Multi-Strategy Strategies (Angelo).

Before starting the project, we did some research about the hierarchy of the IT department, which is the department we were mainly focusing on, and got a brief idea of how things work here at AG. The hierarchy starts with the Chief Information Officer (CIO), then it leads to four Managing Directors. Each of the Managing Directors has an average of ten Hiring Managers (HM) who are full-time employees. Then follows a number of contracted consultants, provided by preferred vendors. The project was designed with a focus on the consultant positions by one of the Managing Directors, who is also our sponsor, Scott Burton. We primarily worked with the Department Coordinator, Cindy Aguilar, to discuss functionalities throughout the project.

## **1.2 Current Situation**

After researching about the current situation within the IT Department, there are mainly two parts of the entire hiring process. The two parts are: the hiring process and the on-boarding process. From identifying a need of a position, the administrator of the department would pass all related information including job title, department names and job descriptions to the CIO for approval. Once it has been approved, an email would be generated and sent to selected vendors by the administrator with all information provided. If they have recommended candidates for the position, they will reply back with the candidates' information attached. Then follows the interview selection process. Once a candidate passes the interview and is hired by the department, the on-boarding process

begins. The administrator would gather all files of the candidate such as work order, MSA for their vendor, resume and cover letter, etc., and then send it to Human Resources.

Both sections of having a newly-hired consultant on-board are not efficient. Since everything was done manually, the administrator gathered required information, printed out files, got the signatures, scanned it, and then processed all of the paperwork. Our team found out that starting from when a hiring manager identified the need for a position, it took a long time to construct emails for vendors to receive the position information and then there was a long process for afterward communication of requesting different files. The vendors then had to send in information for a recommended candidate and the administrator had to process them all before he or she can proceed to the interviewing step.

Between steps, a lot of communication was going on when collecting information, thus when requesting and sending files via emails back and forth, it took longer than it should within the IT Department. Since there was not an existing database to store all information, the IT administrator had been using an Excel spreadsheet to manually track all relevant information, or forward it to different hiring managers requesting for updates. When onboarding roughly about 20 consultants after they were selected for different positions, the administrator required a few business days to manually fill out all information fields again on a different form for each hired consultant and then passed all paperwork to the Human Resources Department. Then the HR Department received the files from the IT Department and continued with the processes. Sometimes when more detailed information was missing, the two departments had to communicate through phone calls or emails for file exchanges. Especially when a consultant was leaving for a position, if trying to pullout previous records for this specific person, the administrator had to dig through all emails and giant piles of paper copies, simply trying to look for historical data, which sometimes was lost and very hard to find as time goes by.

This process could be really inefficient and it may somehow affect a person's onboarding time frame or the IT Department as a whole. The administrator has more than enough to coordinate throughout the day for all aspects and it would be really beneficial for them to minimize their workload if there is a system that merges many of the steps into a single step.

### **1.3 Motivation**

Besides Angelo Gordon's excellence in investment, there was not a focus on either hiring or on-boarding consultants for the IT Department. When it comes to hiring for a selected position and inviting other professionals to on-board, the IT Department does not have an efficient enough way for exchanging required information or even storing them into a database for future purposes. After a hiring manager requested to hire a consultant, he or she will ask the CIO for approval and then pass it along to the department administrator. Currently the administrator is managing the process based on her own preference, which is basically paper-based. No matter what method the administrator chose, whether to print everything out and pass it along to the next department to process, or communicate over emails, or deliver in person, the process doesn't really have a standard format for handling the situation. In regards to making things easier for cross-department data sharing and data storage, our main goal is to create a certain web-based application for the IT Department in order to have a streamlined system or platform to store and request information, send file packages to other departments, and receive email notifications along the hiring processes.

Moreover, it would be necessary for the administrator and hiring managers to retrieve data as quickly as possible, especially when they intend to search for a specific role and see which of the steps the hiring process has reached. The system should provide a general view of all steps when started seeking candidates for a position, together with indicators with each ongoing process. The more clearly the application shows on the main home page once logged in, the better for the users to understand the current situation and respond to it. What the department really needs is an easier way of storing data collected overtime and being able to retrieve requested information as desired. This did not seem to be a focus within the department prior to the hiring season. Now the department administrator is dealing with a huge workload every day and it would be necessary to come up with such a web application to ease her work life. Our sponsor was considering this process optimization with the administrator and put it into effect as a project for our team.

### **1.4 Making the process easier**

Since no system had been previously implemented within the firm, this project was more of an innovative programming implementation in the business filed. Both hiring and on-boarding processes included gathering files such as candidate resumes, cover-letters, manager approvals and job descriptions, sending request emails, and scheduling for interviews, etc. To optimize this process, our idea was to create a single web application containing characteristics of file exchanges which could be done with a single button,

together with a tracking system that showed everything in detail as a progress status indicator, and a database to store all data. Once the administrator logged in to the system, there should have been a status bar that provided detailed information about each position.

By having the main goals in mind, the IT Department really needs to have a file-storing and data-sharing board to provide easy job posting request, and prevent data loss or redundant communication. With fill-in forms built into the application, the administrator or hiring managers may easily generate a job search request and send it to as many vendors as they desired. When requesting data from others, or sending data to other departments or vendors, it is necessary to eliminate some actions when processing.

During email communications, it was complicated to search for historical files since the data has not yet been organized. The more steps it takes, the longer it takes to have a candidate processed. Thus our job was to provide them an easier method to store a candidate's information who had been selected for a consultant position. During the onboarding process, when a specific list of information was required, the task could be easily handled as a one-button action, such as clicking on notify, send, or email buttons.



# Chapter 2: Angelo Gordon Hire Direct (AGHD) – A Workflow Web Application

---

## **2.1 General Goal Overview**

### **2.1.1 Process Overview**

Given instructions by the IT Department, our general goal was structured as shown in *Figure 10*. The Hiring Managers inform the department administrator to generate a job description together with other information such as job title, hiring manager name, and other related fields that were required when searching for candidates. Then the administrator would forward the form to the CIO for higher level approval. Once the hiring request has been approved, an email with a job title and a job description would be sent to the selected vendors. After applications were submitted by vendors that have recommended candidates, the interview process may proceed. Then the hiring managers would schedule interviews and select their desired candidates for the consulting position, and that led us to the on-boarding process.

Once it came to the on-boarding process, the administrator of the hiring department would have filled in the required forms for the position, uploading background checks, resumes, cover letters, vendor information, etc. The vendors were also required to submit the work orders for the position which were generated by themselves or the administrator, and meantime the administrator would have checked if there was an MSA for the vendor on file. When everything was submitted and saved on file, all of the files would have been passed to the Human Resources department for signatures, account setups, and more access granting. In the end, the Helpdesk would have been notified to set up building accesses or RSA tokens if applicable.

Once we finish the candidate selection portion, the system would ask the administrator to gather all files required for each candidate who applied for the position and proceed to the Human Resources department. The system would no longer track the progress in detail once the files get to Human Resources but gives a brief notification to the IT administrator about the current progress each department makes, and provides a final notification email if the entire on-boarding process is finished by all related departments.

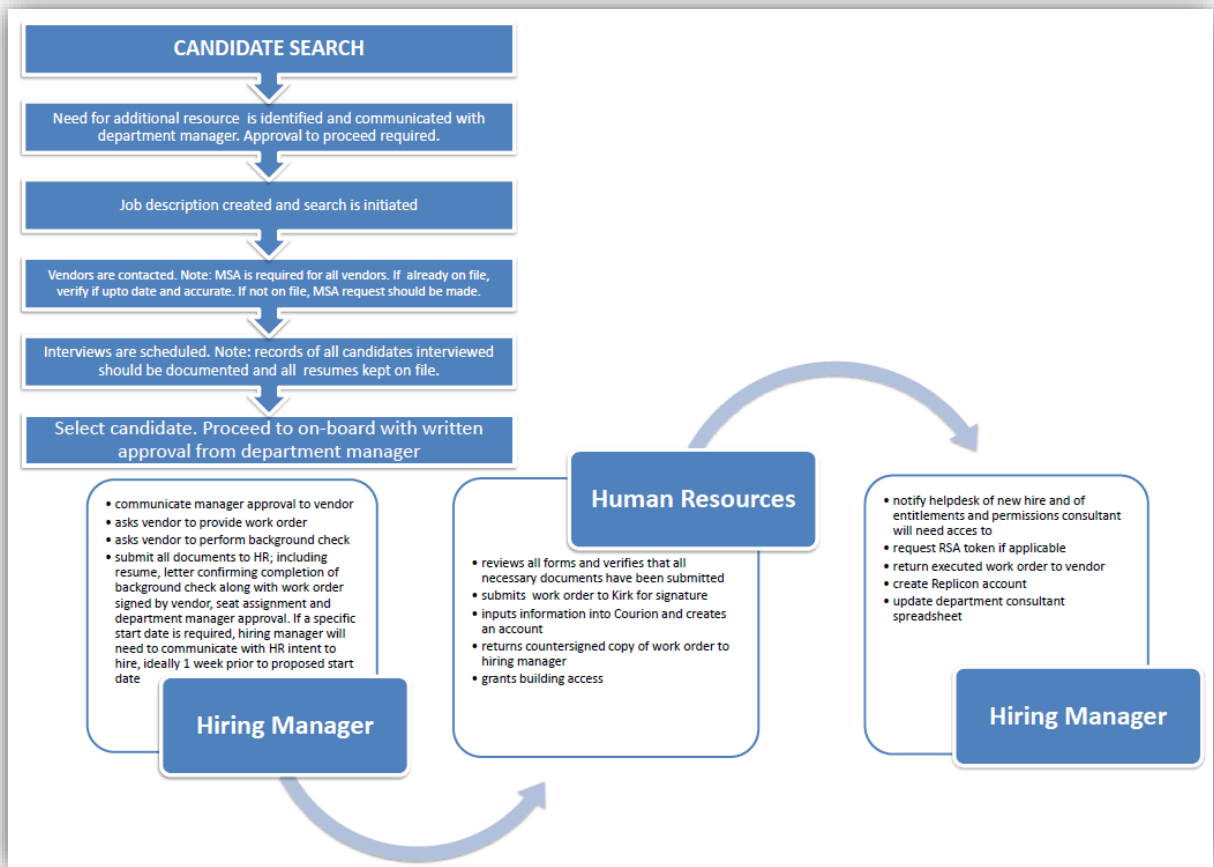


Figure 10 Hiring and On-boarding Process Overview

We decided to follow the basic processes for hiring a consultant, optimize the flow to reach the goal of saving extra time, separate and regroup basic processes to make the workflow more smoothly.

### 2.1.2 Process Flowchart

The entire process involves many departments which may include a large number of employees. Based on our understanding, we first constructed a process flowchart based on the company’s hiring and on-boarding process structure that breaks the process into small steps, and then created user stories and use cases for each step listed. As shown in the figures attached below, the blue boxes indicated what the IT administrator’s tasks are, and the yellow, red or green boxes indicated what CAO, Hiring Managers, and Vendors are required to achieve within the process, respectively. We then proceeded to the design of the user interface with the knowledge of how the process should be handled in detail.

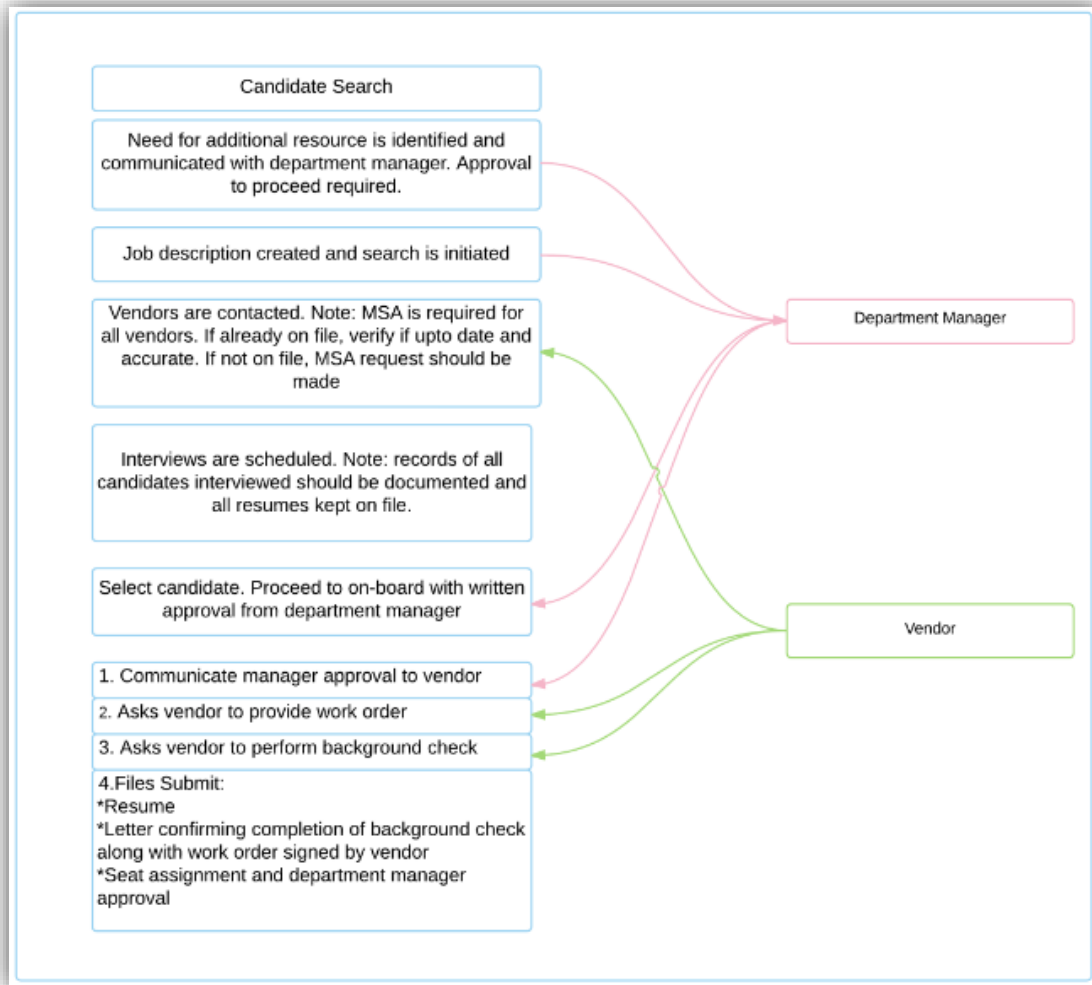


Figure 11 Candidate-Searching Process Breakdown

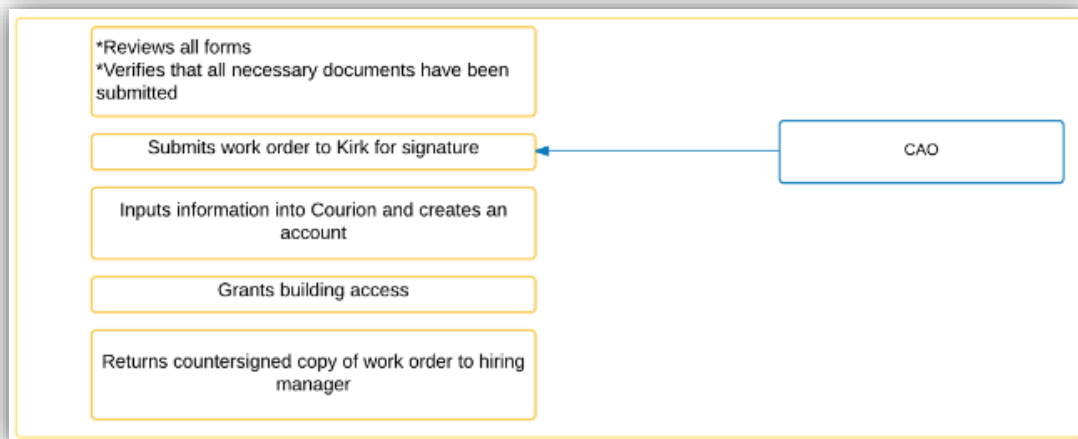


Figure 12 On-boarding CAO Steps

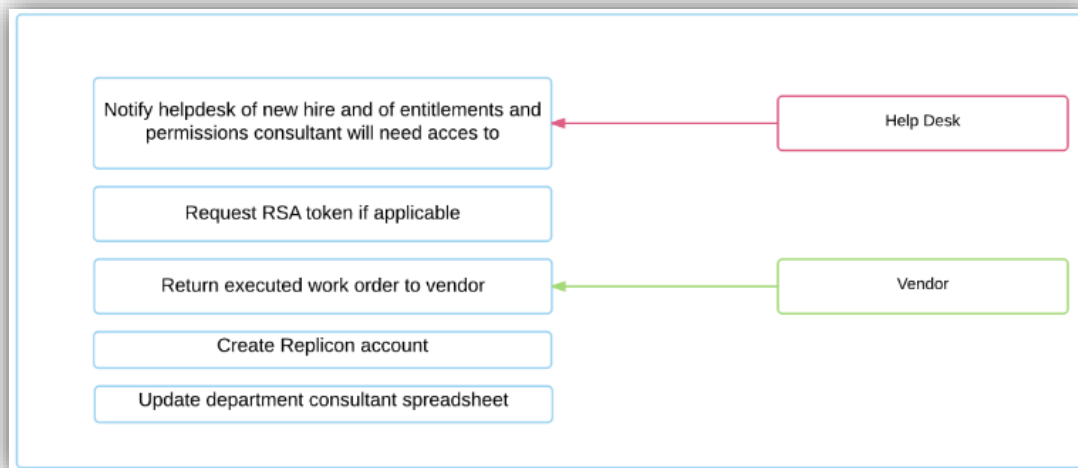


Figure 13 On-boarding Final Steps

### 2.1.3 Form Examples

Given what the IT Department administrator currently had for both hiring and on-boarding forms as references, with the information required listed on the forms (*Figure 14*), we were able to refer back to what is needed specifically. This form appeared in the system as a web page so that it could be forwarded to whomever was selected to view the link. Then the receivers may click on the links, fill in the information required and submit it afterward. Currently these forms were printed versions and they were all stored in different folders by the administrator so that she did not have to go through all emails to look for them. However, the original processing method required a lot of paper usage and storage space. When it reached to a certain amount, the administrator may take a long time organizing all sheets of paper documented previously, and there would never be enough space to save all of the files.

We decided to create a similar application form compared to the original pdf file in our designed application so that the administrators were familiar with what they used to have been doing. We remained the style of the forms to be fill-in forms, attached in the appendix (see *Appendix A*). The form still contains all of the necessary information, with an additional notes section for the vendors to insert comments for recommendation. Fill-in forms are self-explanatory, which requires little memory from users so that they allow users to refer rather than recall. Within this one-page form in the web app, users were provided with guidance in each field for suggested answers in order to minimize possible errors.

**IT CONSULTANT ON-BOARDING FORM**

Today's Date  New Hire Approved by

First Name  Last Name

Vendor  Vendor Contact Name

Contact Phone Number  Contact E-mail

Start date  End Date

Reports to  Project Name  Role

Desk Location

Access Request

Building  Key  RSA Token  Remote Access

Other

*Figure 14 Sample On-boarding Form*

Since the form example covered a lot of information required for different stages, we decided to break them down into different smaller forms for clarity.

## **2.2 User Interface Design**

Before starting to design the user interfaces (UI) of each screen, we created a list of user stories so that we can have a clear structure of what needs to be done by the system. The user stories involve the functionalities of the system we need to design to satisfy users' needs, the relationships of each steps to ensure usability and efficiency, and what the system should authorize the users access based on the requirements for each possible procedure.

We first manually drew out our brief idea of the basic layout, then we decided to use *Moqups*, an online mockup user interface design tool, to create a simple, straight-forward

view to show our project sponsors how we wanted our ideas to be implemented. We created different views including the Log-in Page, CIO Approval/Denial Page, Administrator Process Overview page, Create New Processes/Candidates functions for generating new position searches or candidate creation, and other buttons for sending files or notifications. We also had an Initial Setup Page to allow the users to personalize all possible entries to ease the process of searching. Having a mockup UI designed for this system is easier for new users to have a better understanding of our web app, and it can be used as a reminder of what we actually need for the tool. The actual design of the application varied from the mockup design where more functionalities were added to the application.

Each process can be cancelled at any time within the procedures. When a process is cancelled, it will be shifted to the Cancelled tab with all information saved. The user may also resume any process and keep proceeding without reentering any information.

### **2.2.1 Initial Setup**

When first logging into the web application, the administrator is led to a page to set up preferences and enter related information for future use. This mainly included vendor listing preferences, hiring manager contact information, and a list of possible positions. For both of the general vendors list and the specialty vendors list, the administrator had the ability to put in different vendors names for the categories thus whenever he or she intended to create a job listing, the only thing that needed to be accomplished is to click on the checkbox in front of each category based on the preferences for a particular job search.

After entering a list of hiring managers, all information would be stored in the database for future entries. When creating a new job search, the names of the all hiring managers would show up within a drop-down list in the *Reports To* field as options (*Figure 15*).

Hiring Information Settings:

Project Managers	Email	Edit
Danny Fong	DFong@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Robert Graffeo	RGraffeo@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Eric Entenman	EEntenman@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Masha Leyn	MLeyn@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Scott Burton	SBurton@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Christopher Hansen	CHansen@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Chris Vozzo	CVozzo@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Tom Dargan	TDargan@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Robert Luisi	RLuisi@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Hosain Malick	HMalick@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Meaghan Kelly	MKelly@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>
Sam Velishka	SVelishka@angelogordon.com	<a href="#">edit</a> <a href="#">del</a>

Figure 15 Initial Setup 1 - Settings

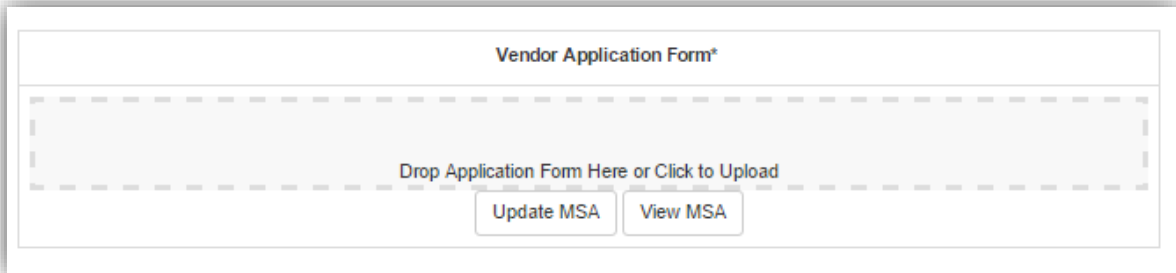
General Vendors*	Email*	Phone #*	Fax	MSA*	Edit
Privosoft	example@example.com	100010	10101	Drop MSA Here or Click to Upload Update MSA View MSA	<a href="#">edit</a> <a href="#">del</a>
JK Partners	example@example.com	100010	10101	Drop MSA Here or Click to Upload Update MSA View MSA	<a href="#">edit</a> <a href="#">del</a>

[Add General Vendor](#)

Special Vendors*	Email*	Phone #*	Fax	MSA*	Edit
Modis	example@example.com	100010	10101	Drop MSA Here or Click to Upload Update MSA View MSA	<a href="#">edit</a> <a href="#">del</a>
Cra Advisors	example@example.com	100010	10101	Drop MSA Here or Click to Upload Update MSA View MSA	<a href="#">edit</a> <a href="#">del</a>
IVP	example@example.com	100010	10101	Drop MSA Here or Click to Upload Update MSA View MSA	<a href="#">edit</a> <a href="#">del</a>

[Add Special Vendor](#)

Figure 16 Initial Setup 2 – Add vendors



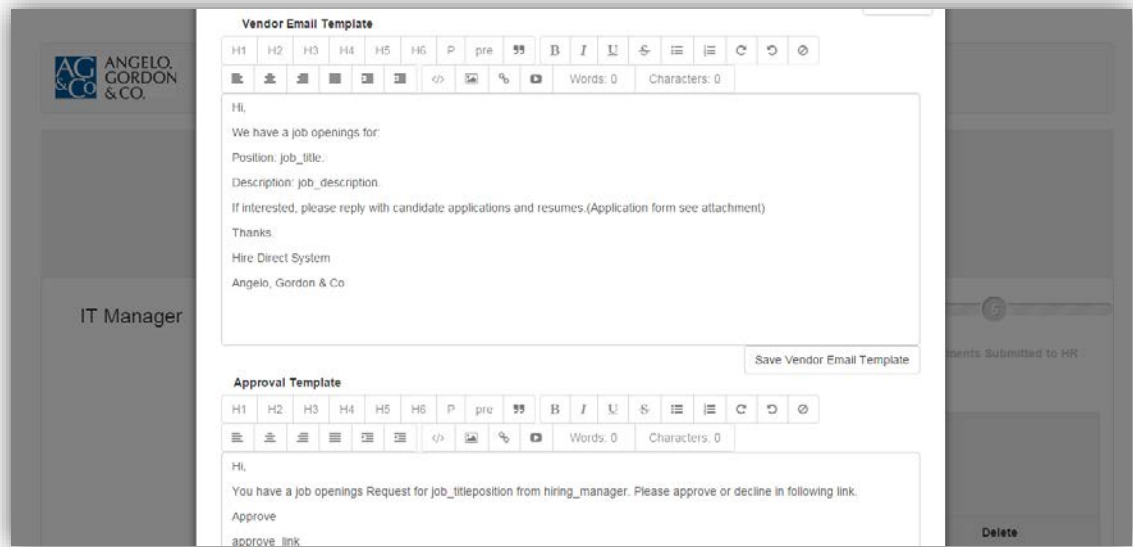
*Figure 17 Vendor MSA*

The edit and delete buttons allowed the administrator to change the existing records of any category, or delete the entire row. The add buttons to the lower right corner of each section authorized the user to add a new record to the system which would appear in the dropdown lists. The user may go back and edit the information in *Settings* at any time of the process and proceed after modifying the record.

Similar to adding hiring manager information, administrator may add vendors' details to the database as well. Possible roles could be accessed in the settings tab, and the application would also memorize each entry and save automatically for future inputs. The color contrast of blue and red allowed the users to differentiate the functionality of the buttons and provide actions accordingly.

Specifically about adding a vendor to the settings, the MSA files are required (*Figure 17*). Unless an MSA file is attached, the user may never add a new row to the vendors list successfully.





*Figure 18 Initial Setup 2 – Email Settings*

The users are allowed to modify the email templates in the Settings menu. Thus each time when initiating a new process, the template would show up on top of the main view as an editable template before actually sending the emails (*Figure 18*).

We were originally planning on entering the information for all categories as built-in information, however, considering all of the possible future changes, we decided to leave this decision to the administrator and Hiring Managers. By the time they actually use this web application, they will have the option to modify all records based on their references. This feature actually saves the administrator time from manually typing in every piece of information each time. When she has multiple applicants at hand, it would be ideal for her to choose from a list of options instead of being left with blanks that requires free texts.

The email templates also prevents multiple text entries. The administrator would send at least one email to each vendor per position, along with several notification emails, and approval emails to different persons. Having an editable template saved in the system would definitely assist the users of the application and increase task efficiency. The unfilled categories will later grab information from the uploaded files and automatically fill in the entries.

### 2.2.2 Log In

The system platform followed the theme color combination of blue, white and gray to ensure consistency. By having a logo and a name of the system on the top left, we provided users a guidance for correct path indications. The texts located on top of the username and password section functioned as a welcome message from the system to remind the users to sign in before proceeding to the next stage. In order to make the users feel comfortable when using this web application, we kept the background of the page gray to keep the consistency of a normal interface nowadays.

The “*Remember Me*” check box provides the users the option of saving their username and password combinations to the local computer browser thus it saved them extra work for typing everything again and again each time. The “*Forgot Password*” option functioned as a link that sent a temporary password to the users’ email address, which allowed users to reset their account password when they were having troubles logging in.

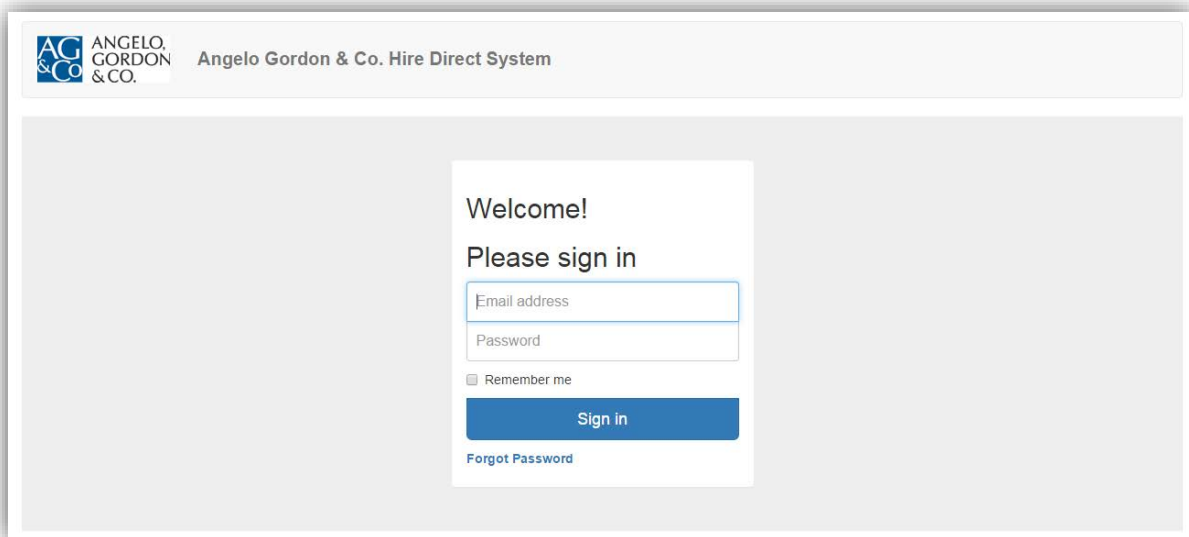


Figure 19 Log-in View

The figure attached (*Figure 19*) above is a view that would show on users’ screen when they opened our web application as guests by using a web browser. The system does not allow any further actions until a user has entered correct username and password and signed in with a recognizable account saved in the system. We created a page with those credentials as the initial view the users would have once they opened our web application. We created accounts for specific users so that the system is secured in a way that only certain users have access to view or edit the files stored in the database. In the case we have

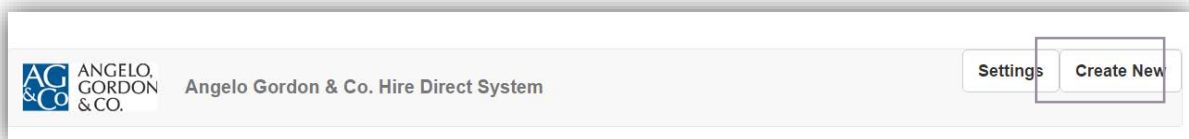
new Hiring Managers or other employees who are recently assigned to supervise the hiring process, the administrator were authorized to generate new access to the person that was involved in one or more hiring procedures and supervise the progress along the way. The account information is stored in the database and can be modified within the *Initial Setup* page.

To keep the design process simple, we decided to set up the log-in functionality as single-user temporarily. We determined the only user to be the administrator for IT Department. The CIO and Hiring Managers were authorized to view each of the hiring process when they have the login credentials, and the CIO will also get all notifications generated by all actions together with all approval requests.

### 2.2.3 Create New Functionality

To start a hiring process, the administrator was required to enter all information for a position into the database, including: *Job Title, Reports To, Department, Skills, and Job Descriptions*. This led the process to the initial stage, which would generate an email to the CIO for further official approval.

This first step was *Initial Search*, which was designed to be automatically highlighted with navy blue after creating a hiring process by the. Once a position searching process has been initiated, the position along with its progress bar will appear as a new row on top of the existed ones.



*Figure 20 Banner*

To achieve the *Initial Search* stage, a user needs to click on the *Create New* button located on the upper right corner of the main page to access the form for the position.

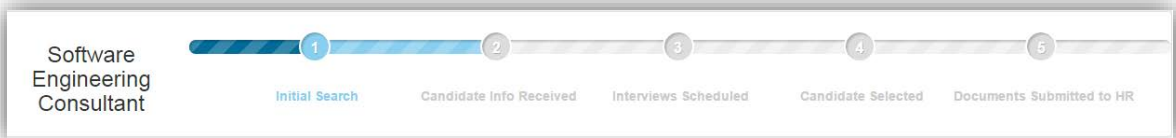


Figure 21 Initial Search Progress Bar

The *Create New* function allowed the administrator to add a new searching process after identifying a need for a specific consultant position. The database saved all previously entered information into the server, so that if there is a future need for the same position, the saved files could be easily pulled out from the database.

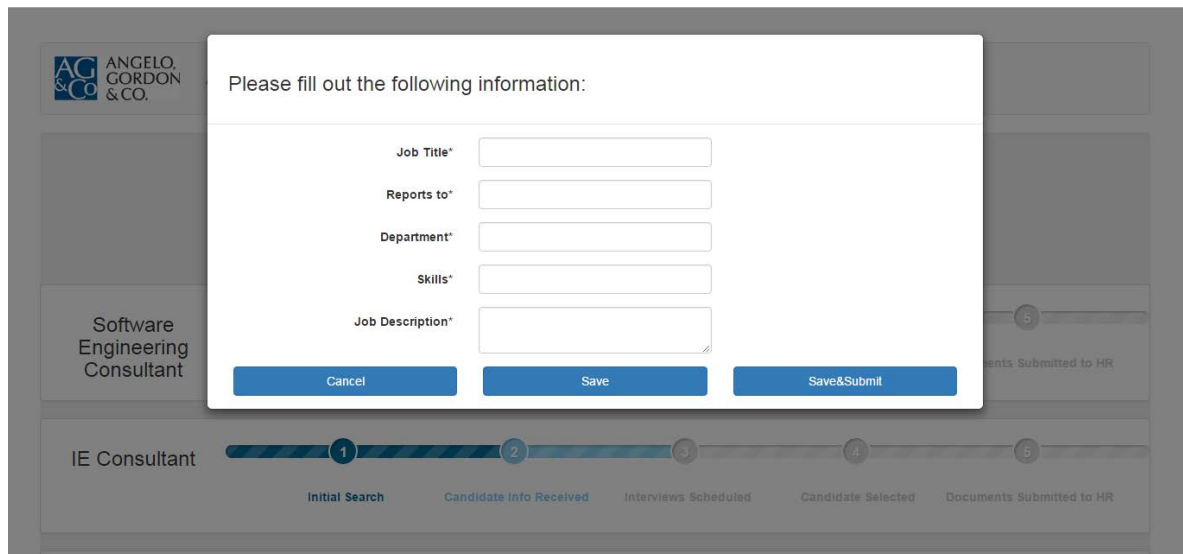


Figure 22 Create New

The stars were created as an indication of required fields. A user may never proceed to save and submit the application until every field has been filled out. We placed three buttons on the bottom of the modal window: *Cancel*, *Save*, and *Save & Submit*. Each button represented different functions: *Cancel* meant withdraw all information entered and go back to the main menu; *Save* was designed to save the current information and proceed later; and *Save & Submit* provided the user the ability to save everything and submit the application. If everything has been done correctly and been approved by the CIO, the users may proceed to the next step – vendor selection process.

The form was designed as a pop out modal window to indicate major priority for the specific window. In the meantime, the background would be greyed out and disabled for the moment until the users hit either one of the buttons for desired actions.

When clicking on the first three categories, *Job Title*, *Reports To*, and *Department*, a drop-down list would appear after each category to provide lists of information for selection. Having drop-down lists lowered the error rate, decreased the required time to type each word, and reduced user memory necessity.

#### 2.2.4 Vendor Selection

As one of the preferred functionality of the department, being able to select a list of vendors to send the emails to with the hiring requests was the best way of forwarding related information to the vendors. From initiating the hiring request, after entering all requested information for the position, the administrator or Hiring Managers may have access to the vendor selection step.

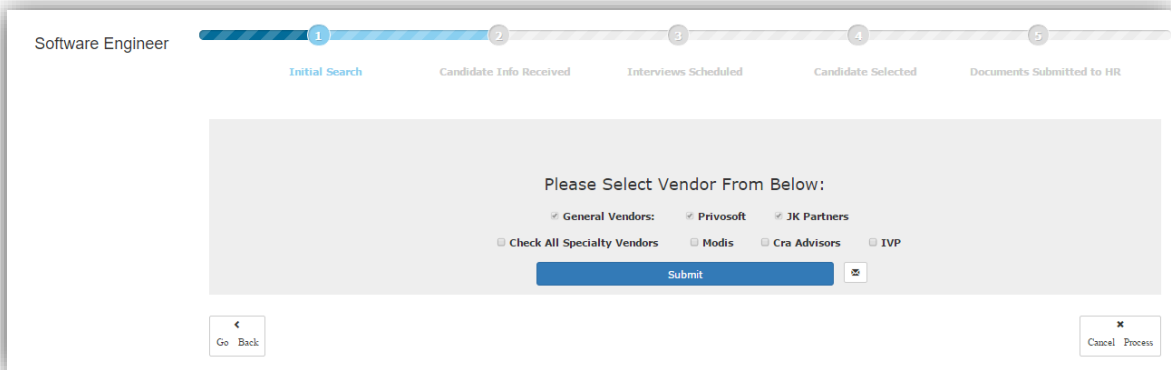


Figure 23 Vendor Selection

We separated different vendors into groups: *General* vendors list, and *Specialty* vendors list. *General* vendors referred to all vendors that the company had previous interaction with, which are selected by default and cannot be deselected. The *Specialty* vendors were a list of vendors that may provide candidates with several specific skill sets, and the user can personalize selections. We moved the *Add New* vendors feature into the *Settings*, which was controlled by the administrator only. An MSA file is required to add a new vendor to the existing list. Any viewers, namely hiring managers and managers, would only view the current existing lists without the ability for modification.

When the vendors receives the emails containing the detailed information about a position, the *Managers* and *Roles* sections are not included for security reasons. The emails are sent on behalf of the AG IT Department thus the content were pre-generated briefly asking for recommended candidates, along with a subject of the hiring request and a signature of Angelo, Gordon & Co. IT Department. All email templates can be modified in the Settings tab. The administrator may select a specific email template base on position preferences.

The email icon placed to the right of the *Submit* button is another way of editing the email templates. It has the same functionality as the email editor in the Settings tab. The users may have access to the email templates on both of the places so that when realizing the email template has not yet been edited in the middle of the process, the user may just click on the email icon to open up the template window.

### **2.2.5 Administrator Process View**

Logged in as an administrator, the user was able to see all in-process positions (*Figure 24*) listed on the main page, together with all detailed information on the progress bar. The progress bar was a step-based indicator that showed each stage of the candidate-searching positions. When multiple candidates are applying for the same position, their detailed information will all appear under the progress bar of that position. The unreached step was indicated by grey circles while the finished ones were navy blue, and the on-going process was filled with a lighter blue so that the progress could be easily understood by the administrator.

# On-boarding Process Status

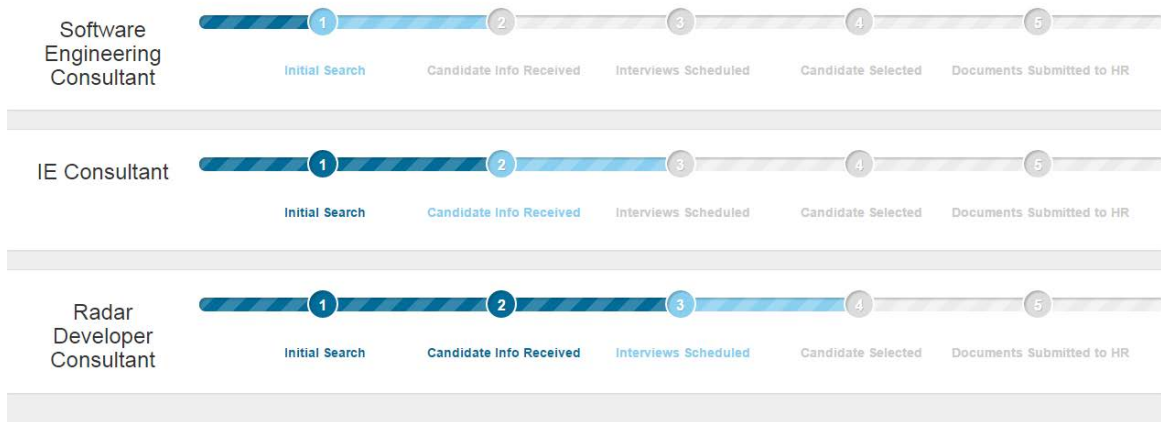


Figure 24 Home Page

As shown in *Figure 24*, the progress bar of a candidate search for a position includes *Initial Search*, *Candidate Info Received*, *Interviews Scheduled*, *Candidate Selected*, and *Documents Submitted to HR*. Each step contains special functions that are designed for different purposes. The color of the bar is designed to change along with the progress of each process. Each process covers all candidates who have applied for the position and with their information listed underneath the bar, the users may have a clear view of the current procedures and make decisions accordingly.

## 2.2.6 Tracker Drop-down Selections and Buttons

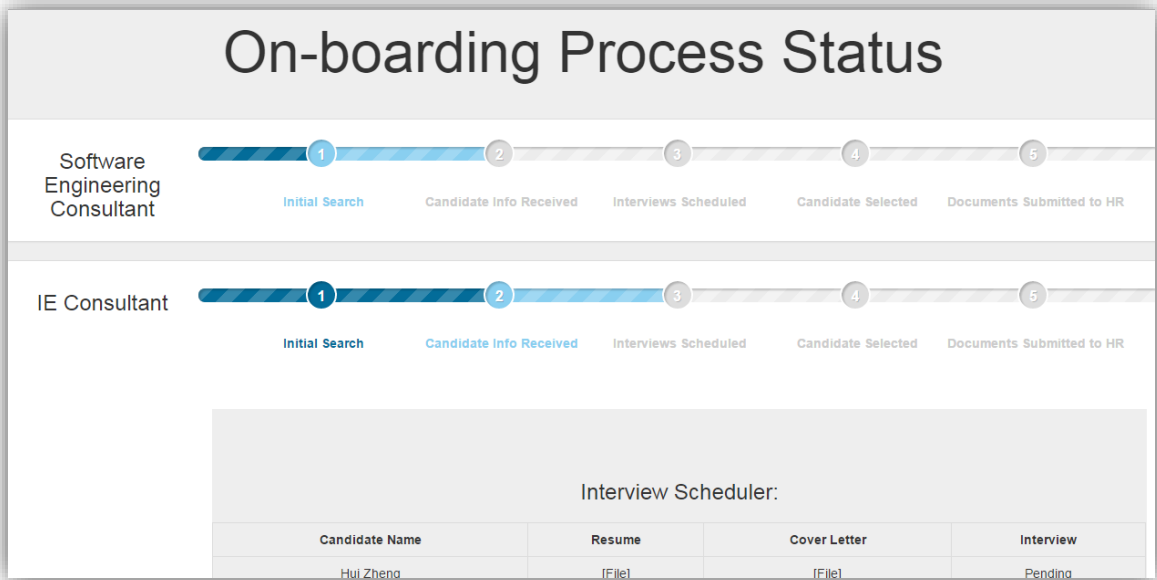


Figure 25 Drop-down Menu

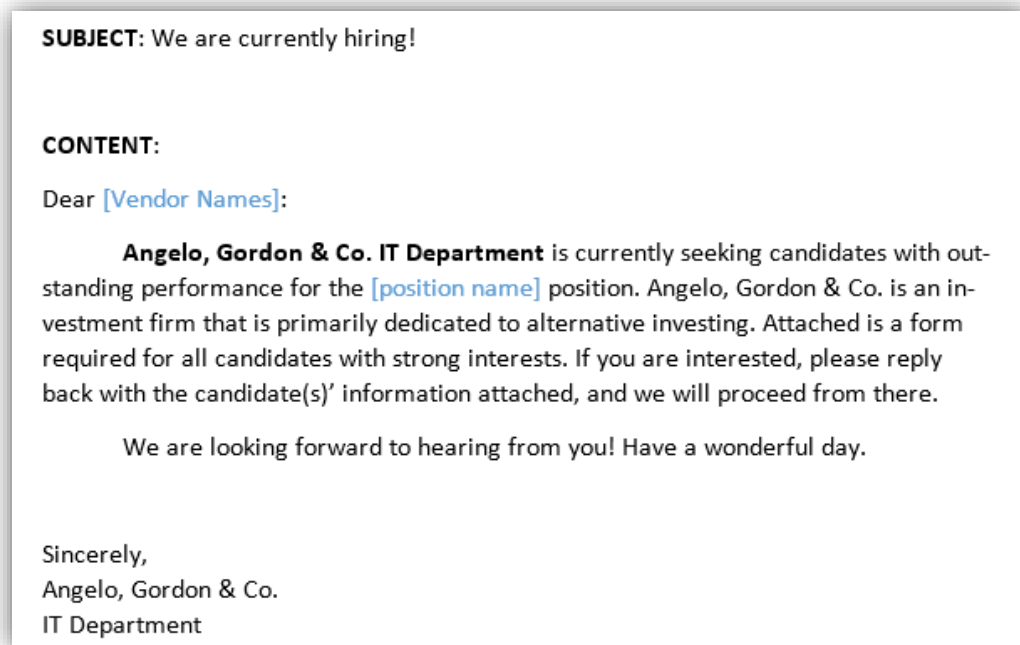
When the progress reaches to a new step, the new drop-down lists are activated by clicking on the ribbon-shaped status bar. When the step is highlighted in grey, the functionality of that step could not be accessed until all previous steps were accomplished. Similarly, once the users hit the buttons for submission, they are not allowed to go back and re-submit the files. The only drop-down menus that the users may access are the ones designed for the current on-going step.

For each step of the entire process, the menus were designed according to the function of the specific step. The buttons placed underneath each form showed different functions based on the current process of hiring a consultant. The process bar would be marked as complete if all steps were accomplished and all files have been submitted to Human Resources. We designed the drop-down menu function to keep everything within the single-page web application so that the users did not have to switch back and forth to view other information while trying to proceed with the current process. Keeping things in one screen was one of our main goals. Thus when viewing and referring to relevant fields, the users simply need to scroll up and down to achieve this goal.



### 2.2.7 Quick Emails and Reminders within Process

Along with each step of the hiring process, automated emails would be generated by the system to notify the user who has initiated the process. For the CIO position, he would be notified of all on-going actions, including file submissions, requests, and step completions, along with the current situations. Similarly, the administrator would receive similar emails of all changes, plus the ability to modify the results of the steps. The email subjects included notifications for position-hiring process details, candidate rejections, interview scheduling, and final selections.



*Figure 26 Vendor Email Template*

As shown in the example above, vendors would receive emails relating to the positions with attachments of the Application Form for the candidates. This email together with all of the other templates could be modified and user specified. Only one vendor name would appear in the *Vendor Names* category within a single email and the labels are case sensitive.

## 2.2.8 Interview Scheduler

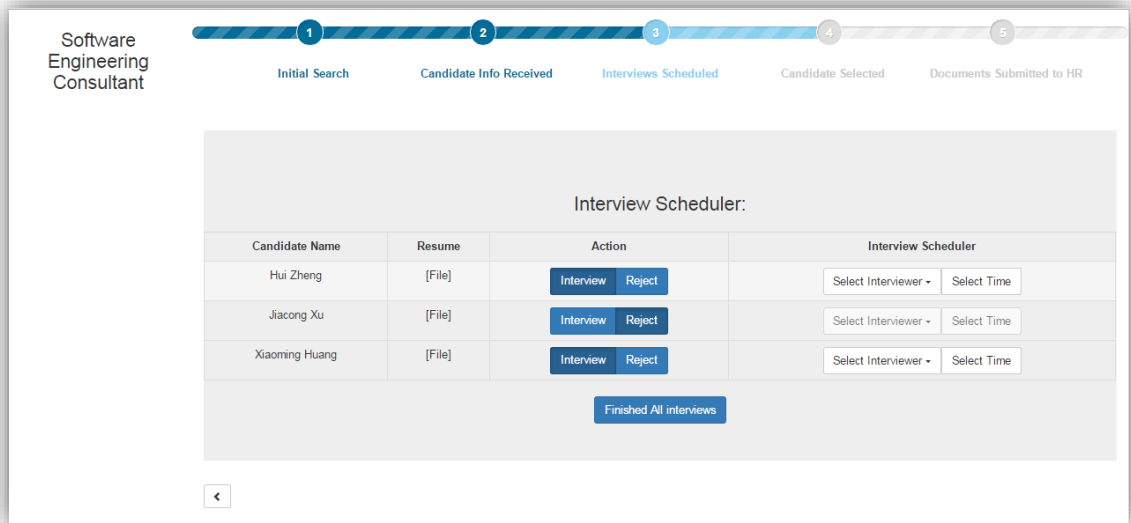


Figure 27 Interview Scheduler

After receiving candidate information from the vendors, the administrator would be able to type in all required fields for each candidate. Under the *Interview Scheduled* step, *Candidate Name*, *Resume* and *Interview* were listed as a drop-down form. Once the information for a candidate was entered into the system, the administrator could either keep adding more candidates to the position, or stop receiving applications and move forward with the process. The status of the interview process would remain pending until the *Finished All Interviews* button was hit.

Once the administrator decided to stop receiving more applications and move forward with the interview process, he or she has to manually click on the *Finished All Interviews* button located at the bottom and proceed. When the administrator or Hiring Managers decided to start interviewing the current applicants and pause from receiving later applications, they were allowed to have an option of stopping the current candidate-seeking process. It was designed as an on-button action for the purpose to ease the users' experiences. This action would lead to a change in the Interview section: instead of a pending status, we inserted a new button after each candidate's name. The button could be toggled on and off, which allowed the administrator to either reject a candidate, or select for an interview process and pick an appropriate interviewer and time slot.

If a candidate was accepted for an interview, the system would provide the administrator a model window to select the interview date and time, and a drop-down list

of possible interviewer's names and proceed after scheduling with the applicants about their availabilities.

Please fill out the following Interview information:

Select Interviewer\*

Select Date\*

December 2015						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
49	29	30	01	02	03	04
50	06	07	08	09	10	11
51	13	14	15	16	17	18
52	20	21	22	23	24	25
53	27	28	29	30	31	01
1	03	04	05	06	07	08

Select Time\*

:

Interview length\*  mins

Figure 28 Date Picker

After the administrator decided to stop the current candidate seeking process, in the *Action* column, a date picker (Figure 28) would appear to the right of a candidate name. When clicking on the box, a calendar should appear right below for choosing interview dates. The two tiny boxes were designed for time selections, and the time was always represented by the twelve hour system.

Each candidate name remained clickable as well as the resume column. Each item led the users to a detailed list of all relevant information about the item. Candidate Name provided administrator all information toward the applicant provided by the vendors. Users may view each resume online, or download as future reference.

### 2.2.9 File Uploading and Downloading

For inserting additional candidate information, we generated a new modal window similar to what we designed for the new position function. The background greyed out on-

purpose when this window appears in front of the main background, which was considered to prevent users from accessing the main page before required fields were filled out. Fields including Candidate Name, Email, Cell, Vendor, Notes, Resume and Original Application are listed. These fields were designed as free text fields which allowed the administrator to enter words as they preferred. The asterisks next to each field label indicated required word entering, so that those fields could not be left blank in order to proceed.

We designed the *Resume* and *Application* fields to allow drag and drop functionality thus when the administrator wanted to upload a file from one device, the only action required was to drag the file from the folder into the dashed box and release the pointer, or simply click on the area within the dashed box and select a desired file.

Please fill out the following Candidate information:

Candidate Name\*

Email\*

Cell Number\*

Vendor\*

Note

Resume

Application

Drop Resume here or click to upload

Drop Application here or click to upload

Cancel Save

Figure 29 New Candidate Entry

After filling all required fields, the administrator may either click on the *Save* button to proceed, or *Cancel* to discard all entries and restart later on. A new row in the candidate information drop-down menu would appear, which was generated based on the information received.

Add and drop functionality saved the administrator time spent browsing, which was useful and efficient, especially since the user did not have to open another window, or move windows around. Instead, users only needed to switch to the source window, drag over target window and drop over the window. It was common across operating systems

environment. When dealing with file based operations, file uploading function isolated files and provided more freedom to the users while providing visibility.

### 2.2.10 Making Final Decisions

When making final decisions, all of the candidate interviews are required to be completed in the previous section. After the the *Finished All Interviews* button was hit, within the *Action* category, two buttons will appear: *Select* or *Reject*. These two buttons can be toggled on and off indicating different results. Select means the candidate is selected for a position and on the contrary, the reject button is indicating not being selected.

Since the administrator is allowed to modify previous steps, we decided not to send automated email messages to the vendors when a candidate was not selected for a certain position. This allows the administrator go back and reprocess the step if a mistake occurs or the hiring managers have second thoughts.

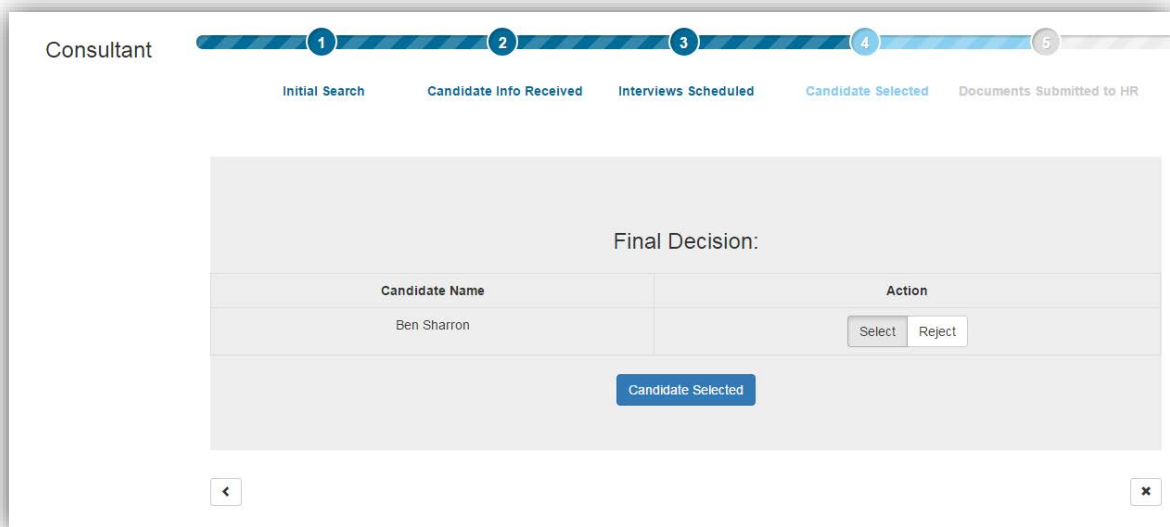


Figure 30 Final Decision Page

Thus if there is a final decision, the administrator may simply press the *Candidate Selected* button to proceed to the onboarding step. This decision was made by the Hiring Managers or whoever conducted the interviews. The Candidate Select button would generate a page containing all information for a candidate. When all required information was filled by the administrator, the candidate is ready to onboard.

### 2.2.11 Submit Files to HR

The files required to onboard a candidate include: *Name of the candidate, Resume and Applications, Background Check, and Work Order*. The administrator should also make sure that a vendor has an up-to-date MSA in order to proceed. These files are all required to onboard a consultant thus without filling all fields, the administrator cannot press submit and send all files.

There is also a field under the HR category to provide the administrator an ability to type in the email address of the HR administrator. The system would then zip up all related files for a candidate and then send it to the typed in email address to proceed with the onboard process.

Name*	Resume*	Application*	Background Check*	Work Order*	HR Email*	Action
John Doe	<a href="#">View Resume</a>	<a href="#">View Application</a>	Drop Background Check Here or Click to Upload	Drop Work Order Here or Click to Upload	<input type="text"/>	<a href="#">Submit</a>

[Finish Process](#)

[Go Back](#) [Cancel Process](#)

Figure 31 Submit Files to HR

To upload files of any kind, the administrator should follow the same steps introduced in the resume and application section by either drag and drop, or click and release. The files are available to be viewed once they are uploaded successfully.

The candidate is ready to onboard when the *Finish Process* button was hit and all files together with the information of the candidate will be sent to the HR Department for further onboarding procedures at the same time. The HR Email field allows the user to enter certain email address(es) to target specific HR administrator.

## 2.3 Implementation

### 2.3.1 Frontend Development

We used Model-View-Controller as an architecture for Web applications. Widely adopted by different languages, several web application frameworks have been created to enforce this pattern. MVC architecture can be used in both the client side and server side. Model stores data for the controller to accomplish logical layer data analytic processes. The View generates different screens to show changes of the Model. A Controller handles all information and presents data within an output view.

A model represents the underlying, logical structure of data in a software application. In our web application, it stores all objects including processes and candidates, and all data structured arrays, which represent a group of objects. We managed the arrays as single objects thus when controller intends to make any possible changes globally, it can be achieved by modifying the object at a single place. The change would be synchronized globally thus it would be processed immediately.

The figure below illustrates this structure:

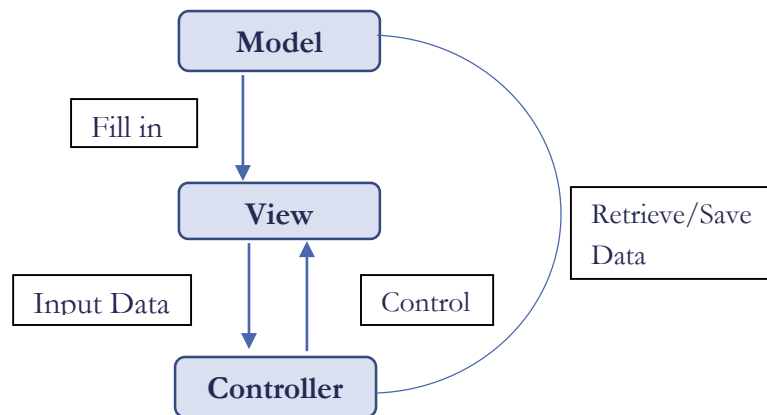


Figure 32 Frontend

### **2.3.2 Backend Development**

The client sends either hyperlink requests or form input to the controller and then receives an updated web page from the generated view. The model exists entirely on the server. We used Node.js for designing the system environment, Express.js for RESTful Web Service architecture, and Mongoose as a MongoDB modeling tool. Node.js allows a fast, minimalist web framework. Many connections can be handled concurrently. Both frontend and backend was written in JavaScript.

Express.js provided an asynchronous event driven framework; it is designed to build a scalable network application. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It provides a thin layer of fundamental web application features, without obscuring Node.js features. Express.js can be operated within different platforms and perform cross-platform tasks.

Both frameworks can be used within JavaScript, which allow web application designers to use the same programming languages, thus reduced unnecessary interaction. JavaScript is a flexible and lightweight programming language and it was beneficial for us to have a grasp of this programming language for designing our application.

### **2.3.3 Database**

We are using MongoDB for the database. We connected database and the server by using Mongoose, which provides a straight-forward, schema-based solution to model the application data. It includes built-in type casting, validation, query building, and business logic hooks. It allows us to save objects to the database and retrieve information as objects directly, and to store the file system. MongoDB also works fine with Express.js, assisted by Mongoose to serialize and de-serialize data within the data-storing processes.



## Chapter 3: Technology

---

In order to implement our thoughts into the web application design, we decided to use a solution named “**MEAN**”. It is a free and open-source JavaScript software stack for building dynamic websites and web applications. With a combination of MongoDB, Express.js, Angular.js, and Node.js, **M** stands for MongoDB which is NOSQL database, **E** is Express.js that is mainly utilized for the server side backend framework design. While **A** is known as Angular.js and **N** is Node.js. AngularJS is commonly used for the frontend MVC framework design and Node.js is used for backend environment.

### **3.1 JavaScript (JS)**

JavaScript is the most popular programming language in the world (JavaScript Introduction, n.d.). JavaScript, alongside HTML, and CSS, is one of the three essential technologies of World Wide Web content production. It is a high-level, dynamic, untyped, and interpreted programming language (JavaScript, n.d.). The majority of websites employ it and it is supported by all modern web browsers without plug-ins. Having the syntax derived from C, JavaScript is unrelated to Java despite naming and some library similarities. It is used in environments that are not web-based, such as PDF and desktop widgets. JavaScript Google V8 Engine is newer and faster than the rest so that it is popular for server-side web applications. On the client side, JavaScript has been recently implemented to perform just-in-time compilation. Utilized in game development, the creation of desktop and mobile applications, JS is also used in server-side network programming with runtime environments such as Node.js.

### **3.2 Node.js**

Node.js is an open-source, cross-platform runtime environment for developing server-side web applications (Node.js). It is written in JavaScript and can be run on OS X, Microsoft Windows, Linux, IBM System z, IBM i, IBM AIX, NonStop, and FreeBSD. Its work is hosted and supported by the Node.js Foundation, a collaborative project at the Linux Foundation.

Node.js allows the creation of web servers and networking tools using JavaScript and a collection of modules that handle various core functionality, such as file system I/O, networking, binary data, cryptography functions, and data streams. It is primarily used to build network programs such as web servers, making it similar to PHP and Python. The

biggest difference is that Node.js is a non-blocking language where commands execute in parallel and use callbacks to signal completion (The Use Cases of Node.js).

Single page, real-time applications require a quick, responsive server that can handle a flood of requests, thus Node.js is perfect for these requests so that people have found it a good fit. Node.js can pipe requests to each other, or stream data directly to its destination without caching, temporary data. It just streams from one place to another, which is often used for file uploads and simple proxies since both use cases are throughputs to an endpoint.

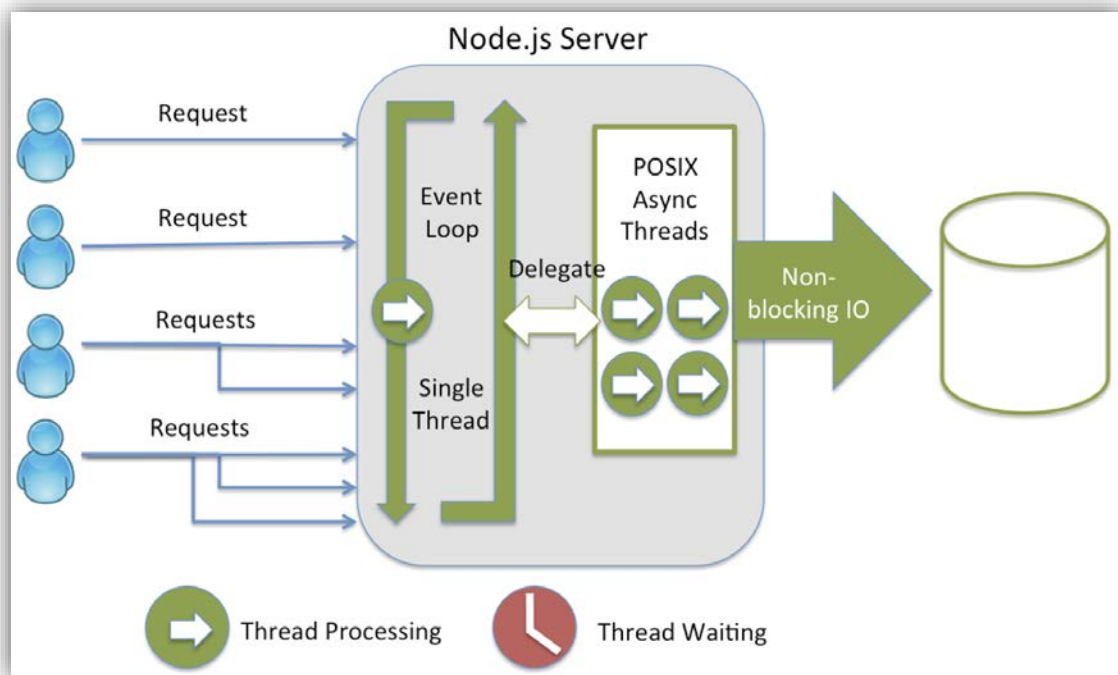


Figure 33 Node.js Server

Node.js is excellent for creating scripts and command line tools because it is just a JavaScript interpreter with a few built in utilities for things like file system management and network communications at its core. We can write a JavaScript script and have Node.js run it for you.

### 3.3 Angular.js

Angular.js, commonly referred to as Angular or AngularJS, is an open-source web application framework mainly maintained by Google and by a community of individual developers and corporations in developing single-page applications (AngularJS). It is designed to simplify both the development and the testing of such applications by providing a framework for client-side MVC (Model View Controller) architecture, along with rich Internet applications components.

Angular.js library first reads the HTML page and interprets those attributes as directives to bind input or output parts of the page to model that is represented by standard JavaScript variables. The values can be manually set within the code or retrieved from static or dynamic JSON resources. It is the frontend part of the MEAN stack as previously described, which exists in the JS library.

Angular.js is built to create user interfaces and connect software components. The framework adapts and extends traditional HTML to present dynamic content through two-way data-binding that allows automatic synchronization of models and views. It implements the MVC pattern to separate presentation, data, and logic components and brings traditionally server-side services to client-side web applications, which reduces burden on the servers.

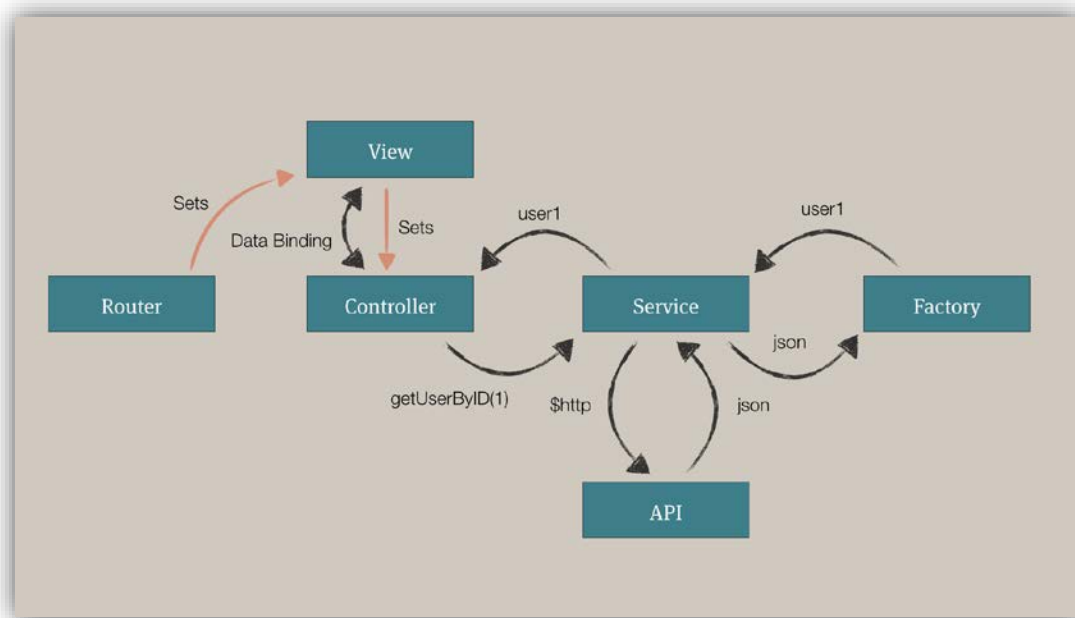


Figure 34 AngularJS

### 3.4 HTML

HTML (HyperText Markup Language), is the standard markup language used to create web pages (HTML). HyperText refers to the hyperlinks that an HTML page may contain. Markup Language refers to the way tags are used to define the page layout and elements within the page (Christensson, 2015). Along with CSS, and JavaScript, HTML is used by most websites to create visually engaging webpages, user interfaces for web applications and mobile applications. Web browsers can read HTML files and render them into visible or audible web pages. HTML allows images and objects to be embedded and can be used to create interactive forms. It creates structured documents by having structural semantics for text such as headings, paragraphs, lists, links, quotes and other items.



*Figure 35 HTML*

Properly formatted HTML pages should include `<html>`, `<head>`, and `<body>` tags. The page title, metadata, and links to reference files are placed between the `<head>` tags, and the actual contents of the page go between the `<body>` tags. Modern HTML relies on cascading style sheets or JavaScript to format nearly all the elements within a page. Even dynamic pages using server-side scripting languages like PHP or ASP must be formatted using HTML. Therefore, scripting languages often generate the HTML that is sent to your web browser (Christensson, 2015).

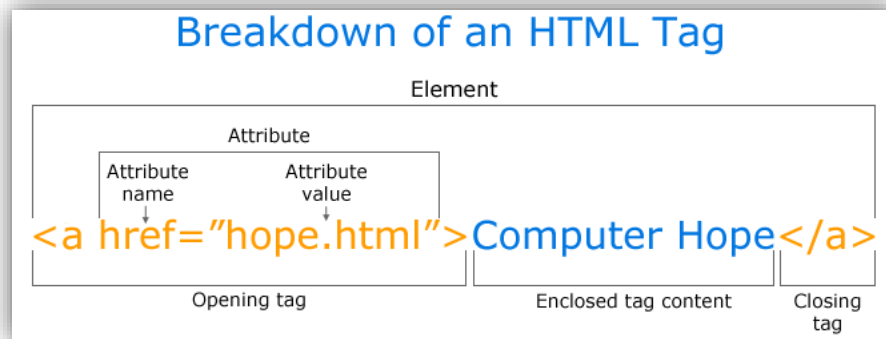


Figure 36 HTML Tag

Almost all HTML tags (*Figure 35*) have an opening tag that contains the name with any attributes and a close tag that contains a forward slash and the name of the tag that is being closed. Each tag is contained within less than and greater than angle brackets and everything between the opening and closing tag is displayed or affected by the tag (HTML). HTML 5 is the update made to HTML from HTML4, which uses the same basic rules but adds some new tags and attributes which allow for better semantics and for dynamic elements that are activated using JavaScript. New elements added include section and forms, together with removing rarely used styling elements.

### 3.5 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. CSS is designed primarily to enable the separation of document content from document presentation, including aspects such as the layouts, colors, and fonts. The separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content. It also presents the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice and on Braille-based, tactile devices. Moreover, it is also used to display the web page differently depending on the screen size or device viewed by users. Readers can specify a different style sheet such as a CSS file stored on their own computer, to override the one the author has specified. If the author or the reader did not link the document to a style sheet, the default style of the browser will be applied.

Another advantage of CSS is that users can apply changes to the graphic design of many documents quickly and easily by editing a few lines in one file rather than modifying every line.

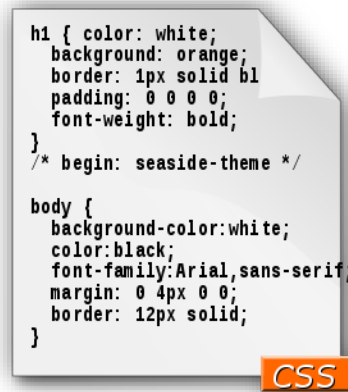


Figure 37 CSS

CSS is a simple mechanism for adding style to Web documents and it tells a browser how to present a document. There are various ways of linking, but the simplest method for starting is to use HTML's STYLE element (CSS Quick Tutorial). A style sheet may be imported with CSS's @import statement, which must occur at the start of the style sheet. Any rules specified in the style sheet itself override conflicting rules in the imported style sheets, and the import order is important in determining how they cascade.

Using an external CSS may help avoid duplication, make maintenance easier, and allow users to make a site-wide change in one place (Mills, 2015). The browser converts the markup language and the CSS into the Document Object Model (DOM), which represents the document in the computers' memory. It combines the document's content with its style. Then the browser displays the contents of the DOM, which has a tree-like structure. Each element, attribute and run of text in the markup language becomes a node in the tree structure. A markup language uses elements to define the document's structure, which can be a container and include other elements between its start tag and end tag.

### 3.6 Express.js

Express.js is a Node.js web application server framework, which is designed for building single-page, multi-page, and hybrid web applications (Express.js). It is relatively

minimal with many features available as plugins. It is also the backend part of the MEAN stack, as well as the SEAN stack that we used for this project.

As a fast, minimalist web framework for Node.js, it provides a robust set of features for web and mobile applications with flexibility. It also provides a thin layer of fundamental web application features without obscuring Node features.

### **3.7 SQL Server**

Microsoft SQL Server is a relational database management system that is a software product with the primary function of storing and retrieving data as requested (Microsoft SQL Server). It may run either on the same computer or another computer across a network.

Data storage is a database with tables. SQL Server supports different data types, including primary types such as Integer, Float, Decimal, Char, Varchar, Text, etc. It makes server statistics available as virtual tables and views, together with other objects such as views, indexes, constraints, and a transaction log. Storage space allocated to a database is divided into sequentially numbered pages. Page type defines the data contained in the page while page is a basic unit of I/O operation. For physical storage of a table, its rows are divided into series of partitions whose size is user defined. Rows in each partition are stored in either B-tree or heap structure. With a B-tree providing the index values, the rows are stored in order within the tables for faster retrieval.

SQL Server allows multiple clients to use the same database. To ensure data integrity, SQL Server provides two modes of concurrency control when multiple clients update the same data or clients attempt to read data that is in progress. The two modes are: pessimistic and optimistic concurrency. It controls concurrent access by using locks when pessimistic concurrency control is being used. Locks can be shared or exclusive. The former allows multiple users to read while the latter grants the users exclusive access to the data.

For retrieving data purposes, SQL Server database needs queries. SQL is a Structured Query Language, which is a special purpose programming language that is designed for managing data held in a Relational Database Management Systems (RDBMS), or for stream processing in a Relational Data Stream Management Systems (RDSMS). The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

### 3.8 Modal Window

In user interface design, a modal window is a graphical control element subordinate to an application's main window which creates a mode where the main window cannot be used (Modal window). It is a child window that requires users to interact with it prior to the parent application, so that the workflow on the application main window can be prevented.

Modal windows draw attention to vital pieces of information and block the application flow until information required to continue is entered, such as usernames and passwords in a login process. They also collect application configuration options in a centralized dialog, thus typically access to the application is disabled while the edits are being made. Modal windows are common in GUI toolkits for guiding user workflow. They provide a way to deliver contextual information, notifications and other actions relevant to the current screen (Making Modal Windows Better For Everyone). Also they quickly shift visual focus from one part of a website or application to another area of content.

### 3.9 RESTful Web Service

Representational State Transfer (REST) is the software architectural style of the World Wide Web (Representational state transfer). REST gives a coordinated set of constraints to the design of components in a distributed hypermedia system that can lead to a high-performing and more maintainable architecture. Systems which conform to the constraints of REST can be called RESTful. Typically, but not always, they communicate over Hypertext Transfer Protocol (HTTP) with the same HTTP verbs, which web browsers use to retrieve web pages and to send data to remote servers. REST interfaces with external systems using resources identified by Uniform Resource Identifier (URI), which can be operated upon using standard verbs.

The uniform interface constraint is fundamental to the design of any REST service. The uniform interface simplifies and decouples the architecture, which enables each part to evolve independently. The four constraints for this uniform interface are: Identification of resources, Manipulation of resources through these representations, Self-descriptive messages, and Hypermedia as the engine of application state (HATEOAS).

Web service APIs that adhere to the REST architectural constraints are called RESTful APIS. HTTP-based RESTful APIs are defined with these aspects:

- Base URI, such as `http://example.com/resources/` ;



- An Internet media type for the data. This is often JSON but can be any other valid type, for example, XML, Atom, microformats, images, etc.);
- Standard HTTP methods;
- Hypertext links to reference state;
- Hypertext links to reference-related resources.

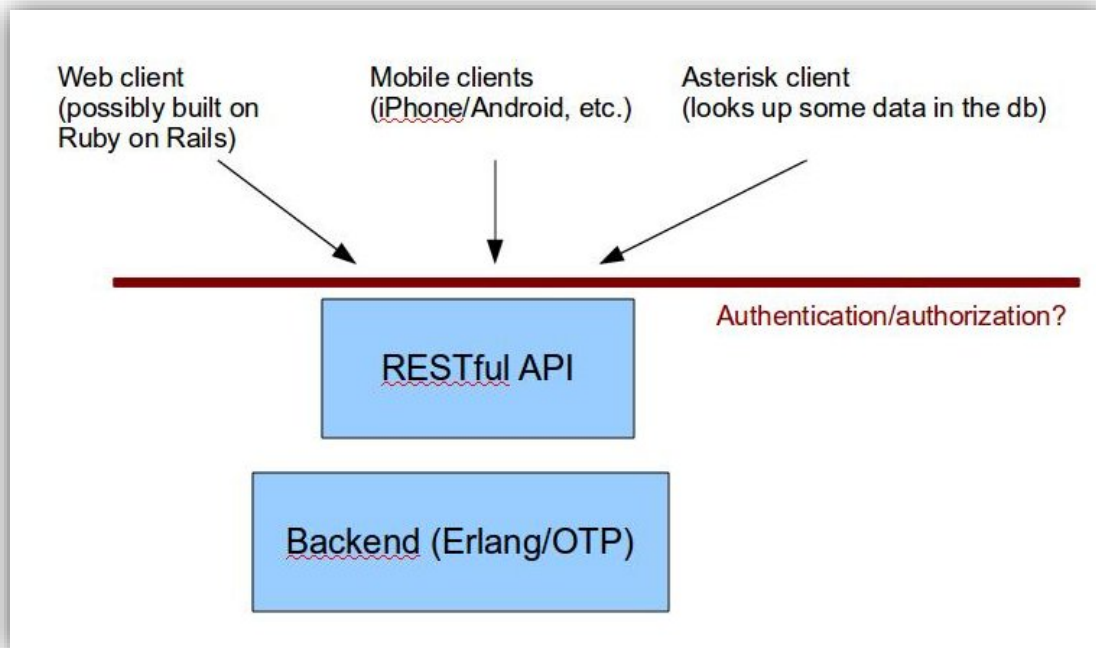
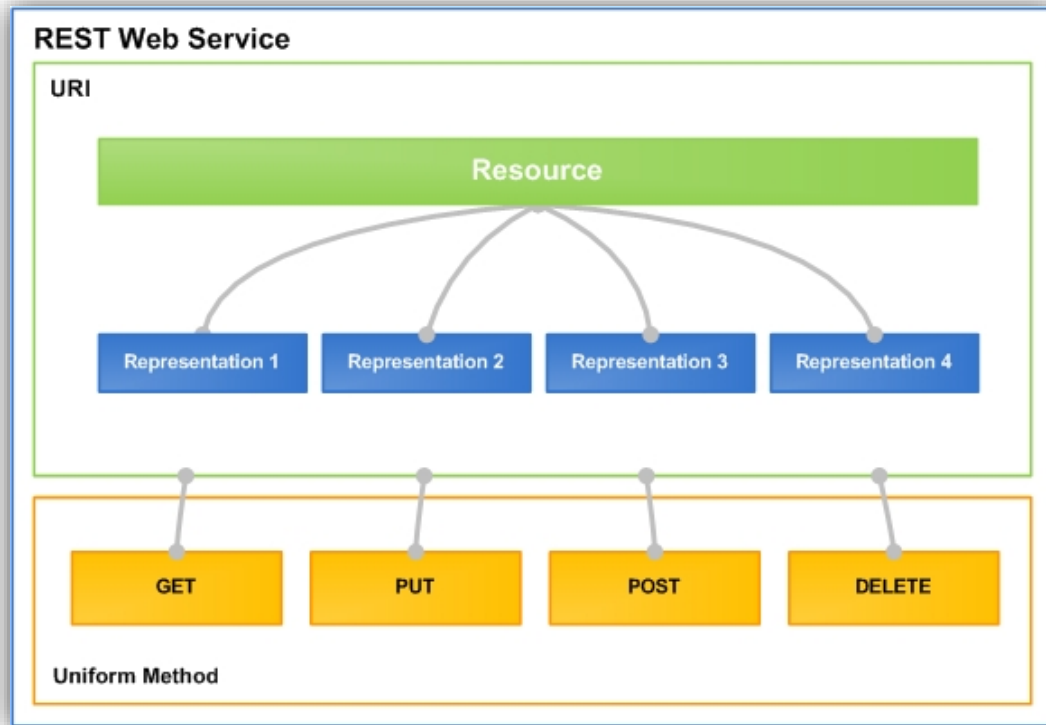


Figure 38 RESTful Web Service

REST defines a set of architectural principles by which you can design Web services that focus on a system’s resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages (RESTful Web services: The basics). If measured by the number of Web services that use it, REST has emerged in the last few years alone as a predominant Web service design model. A complete, independent request doesn’t require the server, while processing the request, to retrieve any kind of REST Web service application, context or state. A REST Web service application or client includes within the HTTP headers and body of a request all of the parameters, context, and data needed by the server-side component to generate a response. Statelessness in this sense improves Web service performance and simplifies the design and implementation of server-side components because the absence of state on the server removes the need to synchronize session data with an external application.



*Figure 39 REST Web Service design structure*

*Figure 38* demonstrates the design principles and components that comprise a REST web service. Wink reflects these design principles in the implementation of web services (Introduction to Apache Wink).

## Chapter 4: Results and Limitations

---

### 4.1 Results

The outcome of implementing this web application within the IT Department of Angelo, Gordon & Co. was to provide the entire department a standardized hiring process for all consulting positions. From an intention of storing data and documents into a uniform tool for future references and reuse, Angelo Gordon Hire Direct allowed its users to perform data exchanges between departments and Hiring Managers. Instead of communicating and sharing all information via emails, the web application stored all things as the minute they were entered into the system, thus prevented data loss.

AGHD is a standardized application we designed for the convenience of the department to easily retrieve data from previous entries and saved time for all users. Optimizing the workflow process dramatically reduced storage space for placing massive paper files. Having several discussions within our team and with sponsors, we reached a new level of increasing both efficiency and usability in real life. When searching for a desired candidate for a position, the administrator logged into the web application, inserted all information required for hiring a consultant, the process would automatically start. Most functions were click-based to reduce user interaction thus prevented future mistakes. Edit and Redo buttons authorized the user abilities to always go back and re-process or cancel all entries.

Features such as modal windows, drop-down menus or lists, check lists, date and time pickers, automatic email reminders, and approval functionalities were designed as a way to present the users a smart application to reduce workload. We performed analysis on usability maximization and modified our design along the way. All entries could be retrieved at any time from the database. The user could log in to the system and download or upload files at any part of the hiring process. This web application also functioned as a renewable and adaptable process tracker of all possible pieces of information.

### 4.2 User Experience

We set up some usability tests to improve user experiences. After making some major developments on the web application, features of the app included:

*a. Reduced reinvented patterns.*

We tried to keep consistency with the company website, both in themes and colors, icons and font styles. It would reduce the need for users to learn a new

pattern each time. Together with a company logo located on each screen, users may feel a sense of belonging whenever using our application.

*b. Grouped related elements.*

We grouped items of the same category together to ensure pattern consistency. Features like log in, settings, view all processes belong together, thus were kept closely together. This design was less confusing and more intuitive to users.

*c. Intuitive with target users.*

When looking at our interface for the first time, users may get a brief idea of how everything works based on the labels of each button. We named each button intuitively so that users could simply tell the function of it by reading the text. Since our web app was mainly designed for the IT Department, we kept most entries based on their preferences.

*d. Clear path.*

Users were clearly indicated where they are when using the application. The ongoing processes and buttons being pressed were all highlighted. There were color changes when hovering a mouse over a button or field. Our application would display data while showing transition to avoid surprises.

*e. Enough testing.*

We performed several trial runs for our app and passed it along to our sponsor for more testing. All data were successfully stored into the database and the interface were performing well. We did a lot of modification through communicating with the target users to ensure the best quality of our design and improve efficiency and usability.

### **4.3 Limitations**

There are limitations about this web application. This app was primarily restricted by its limited users. Due to security reasons, the content of the app were not available until a user was logged in. Limited by the seven-week term system, we only had enough time to work on most of the functionalities required by the department. Since there were different updates about the design during each meeting we held with our sponsors, the login feature was pushed back on our priority list. Ideally, there should also be a substitute account for the person who was planning on taking over the administrator's tasks and logging in when she is away.

Throughout the entire project, our deliverable was a web application with standardized functionalities. Basic functions successfully covered all required features; however, considering a bigger picture of this application, future unforeseen problems may occur which requires more add-ons to it.

Specifically, we originally were going to create a page that leads the CIO to perform approve/decline actions. However, we created the page layout and did not have enough time to insert the functionality of generating temporary links. Our original thought was to design a button to send email messages containing those links, and the administrator would receive email notifications after an action is made.

*a) Approval Request Page*

When the administrator submits a hiring request for a position to the CIO for an official approval, the CIO would view the following page when the request is sent successfully. A list of information will be forwarded to the CIO together with the request: *Job Title* the Hiring Manager was seeking, *Skills* for the position, *Department* that requires hiring a consultant, *Budget*, *Reports To*, and *Job Description* for the position. These information lists are generated by the system after the administrator entered the information for these fields. The system provides a view to the CIO to show a general overview of what position is currently being requested by which department and provide decisions accordingly, namely, to approve or decline.

The page is made by the system after the information is pulled out from the database. It can be accessed through a corresponding link by the system. The person who started the candidate-searching process would be the one sending the link to the CIO for approval via emails. The links were temporarily made by the server for security purposes which would be invalid after a decision was made.

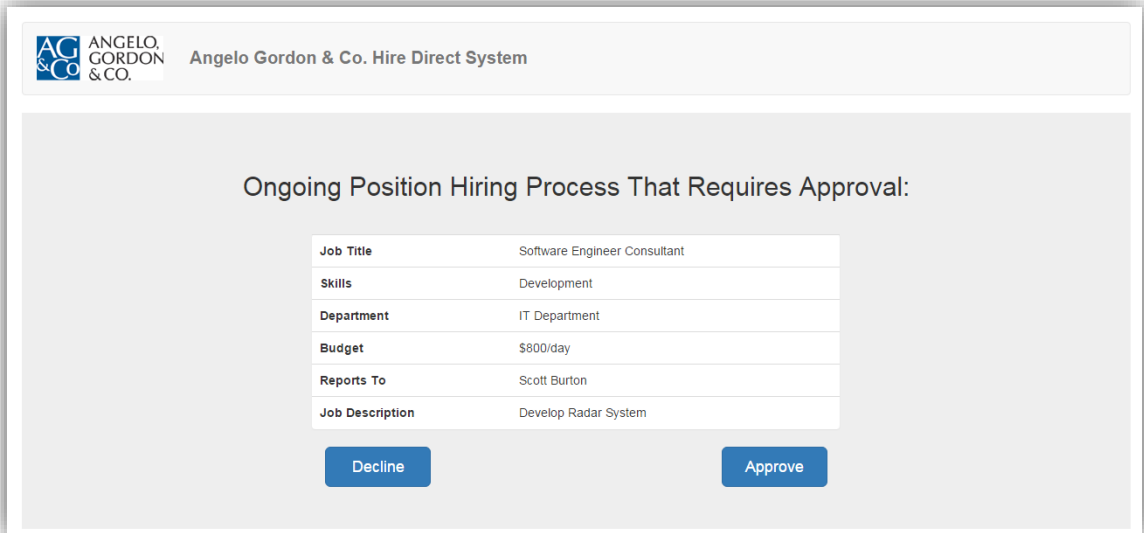


Figure 40 CIO Approval Request View

*b) Approval Message*

When the CIO decides to provide approvals to proceed with the current hiring process, an approval message would appear on a new window to remind the CIO that he has approved a position and it would be saved for later. The *OK* button located on the lower right corner of the message box was designed to close the current page and take the users back to the main page.

An email would be generated to the user who started the requests indicating that the CIO has approved it and he or she may continue on processing the hiring steps. The email is automatically generated with the click on the *OK* button. It also allows further modification if the sender wants to add additional comments or notes to it.

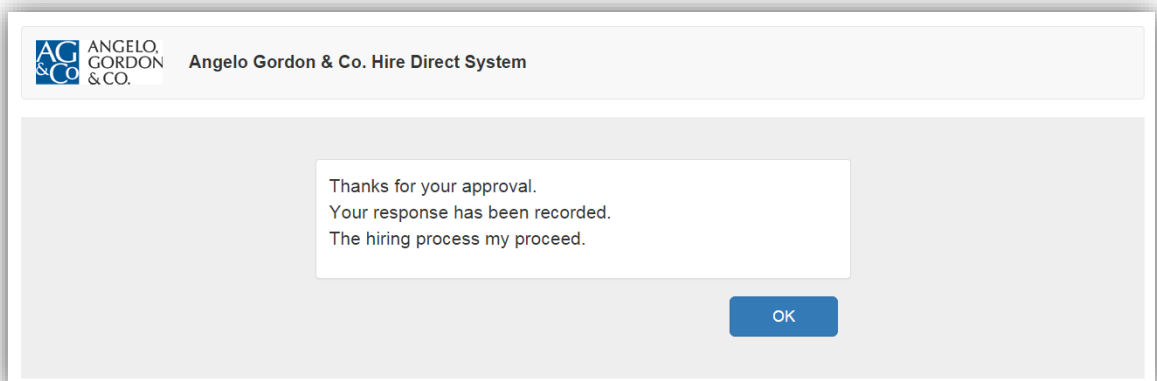


Figure 41 Approve Message

c) Decline Message

If the CIO decided to reject a position seeking request (Figure 39), he simply needed to click on the *Decline* button on the *Approval Request View* and a new window with a paragraph indicating there is a need of modification, would show on the screen, and functioned as a reminder for declining a request from the Hiring Managers.

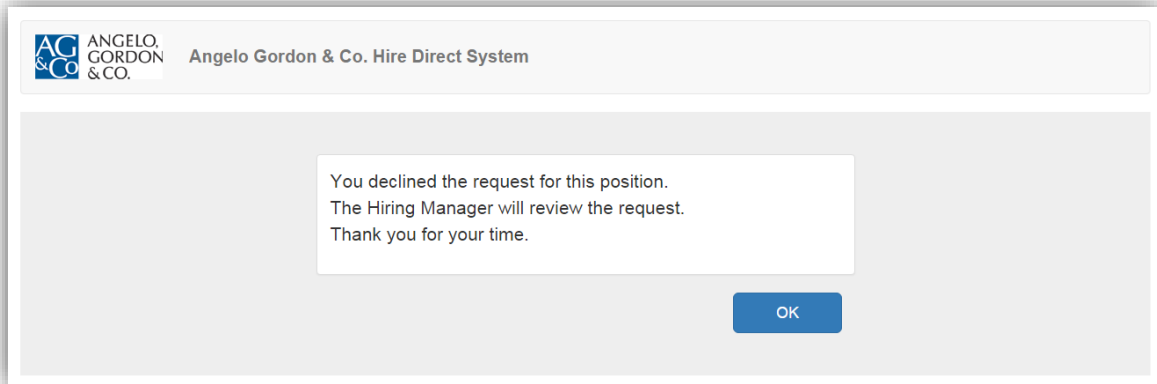


Figure 42 Decline Message

Along with the Ok button, the Hiring Managers would also receive an email about the request for hiring being rejected. Different from the emails for approval, this rejection email also contained a category of the reasons for rejection. This may include small errors, missing information, wrong numbers, and all types of reasons that may occur.

Moreover, the system does not provide the user a search button to look for records based on categories. The user may only view processed, processing, and canceled listings; however, we should insert a function to allow the users to search by candidate name, job titles, or dates, etc. Searching is a standard functionality of a mobile or a web application system. With the ability to search for a desired category, the users will save time going through all possible records trying to retrieve one piece of information. We suggest the IT Helpdesk deploy this feature when trying to maintain this web application for better performance.

Personally, finding a way to cooperate and collaborate with all of our knowledge and skill sets into one project is not always an easy thing to do. Not having enough experiences in web application development, we were learning the skills while trying to apply them at

the same time. It limited our performance level and shortened our available time frame. Learning new things in a real life organization is always an excellent opportunity and we really enjoyed it. Restricted by available resources, we were not able to reach out to other experienced employees all the time yet trying to figure everything out by ourselves. However, we accomplished almost all desired goals and functionalities as expected, thus provided the department an easy and human friendly tool to track all workflow.



# Chapter 5: Future Work and Recommendations

---

## **5.1 Extensions and Recommendations**

### **5.1.1 More functionalities**

So far, our system follows a single-user log in system. If provided with a longer period of time, we would like to setup multiple users to log into the system with different views. We recommend to add a multiuser login system within AGHD. The administrator is authorized with all editing functionalities, and the hiring managers should have the ability to login and view the process along the way to supervise each procedure. Each of the hiring managers should only be able to see the searching processes they initiated, without the ability to get notifications for what other hiring managers are seeking. The administrators will be allowed to authorize other employees the ability to login and view the processes based on their roles.

Moreover, there should be a substitute account that shares the same functionality as the department administrator. Since the administrator may not be available to access the system at all times, a substitute account would be necessary for the department to perform actions as administrators. Each of the users should be able to have individual manageable accounts. The users may log in at any time with the ability to send in-site emails to other users, get notifications by the notification center, and also have the chat functionality to have conversations with the other users. These functions might be developed and added to the system if provided with a longer period of time.

We also have an idea of adding an automatic reminder with Microsoft Outlook so that when the users want to schedule for interviews, the interview date and time should be automatically added to the calendar together with the notification center. For the users' convenience, this also may tell the user when the availability is for the interview to prevent time contradictions. To insert this feature, we need to link the Outlook account to our application to allow accesses to all information stored within the account. It will have the permission to read all data and perform actions. We need to design an Outlook API to grab the information and send it to the server then retrieve data across the two applications.

### **5.1.2 Database View**

Considering the display of the application, we put all processes on the main page. The users had to scroll down and see all of the related candidate information. If we insert a list of tabs on the left side or a ribbon on top of the page, the users may be allowed to view the results based on categories such as candidate, hiring managers, processes, and interviews,

etc. Separating the information based on categories may save the users some time for searching, however, since we did not implement this change into the system, we are not certain this would actually benefit the users.

### **5.1.3 Candidate Information Scanning**

As we moved along the designing process, we discovered that if our system has a function of scanning candidate application forms and auto fill in all fields for the user, it would save more time and be more efficient. Scanning automatically whenever a file is submitted to the system like what most of the current websites are using is a faster way of inserting data into a system. We would like to test on implementing this feature sometime in the future.

### **5.1.4 Registration System**

The ideal situation is to allow the vendors to register to the website, and then submit applications or recommended candidates. We would like to have further updates to create a user registration feature in the future. Vendors can edit all requested field and personalize all data entries, thus prevented possible mistakes when passing information between vendors and the company. Also this registration feature may save the department administrator time and reduce all possible workload.

## **5.2 Future Work**

For the purpose of maintaining the system, if we want to add new possible functionalities to the application, the IT helpdesk needs to find the original code and make changes accordingly. The database is required to be maintained by the third party applications thus it provides a more direct view of the potential changes. If modifying object structure within the backend, the developer is required to restart the server to synchronize both the backend server and the database to prevent the situation of not saving the new data from happening.

If the server shuts down, we may not continue saving any more data to the system. Once we restarted the server, we need to make sure the previous shutdown did not influence any piece of data saved to the database. All data should be rechecked to make sure there is not any unusual data piece stored within the system.

# References

---

- Angelo, G. &. (n.d.). *ANGELO, GORDON*. Retrieved October 29, 2015, from <https://www.angelogordon.com/default.aspx>
- AngularJS*. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/AngularJS>
- Being a Full Stack Developer*. (n.d.). Retrieved from Sitepoint: <http://www.sitepoint.com/full-stack-developer/>
- Brown, D. C. (2015, October 08). *Human Computer Interaction*. Retrieved from Worcester Polytechnic Institute Computer Science CS3041: <http://web.cs.wpi.edu/~dcb/courses/CS3041/>
- Christensson, P. (2015, May 23). *HTML Definition*. Retrieved from TechTerms: <http://techterms.com/definition/html>
- CSS Quick Tutorial*. (n.d.). Retrieved from Web Design Group: <http://www.htmlhelp.com/reference/css/quick-tutorial.html>
- Definition of: Web application*. (n.d.). Retrieved from Encyclopedia: <http://www.pcmag.com/encyclopedia/term/54272/web-application>
- Express.js*. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Express.js>
- HTML*. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/HTML>
- HTML*. (n.d.). Retrieved from Computer Hope: <http://www.computerhope.com/jargon/h/html.htm>
- JavaScript*. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/JavaScript>
- JavaScript Introduction*. (n.d.). Retrieved from w3schools.com: [http://www.w3schools.com/js/js\\_intro.asp](http://www.w3schools.com/js/js_intro.asp)
- Making Modal Windows Better For Everyone*. (n.d.). Retrieved from Smashing: <http://www.smashingmagazine.com/2014/09/making-modal-windows-better-for-everyone/>
- Microsoft SQL Server*. (n.d.). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Microsoft\\_SQL\\_Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server)
- Mills, C. D. (2015, July 17). *Why use CSS*. Retrieved from MDN: [https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting\\_Started/Why\\_use\\_CSS](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_Started/Why_use_CSS)
- Modal window*. (n.d.). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Modal\\_window](https://en.wikipedia.org/wiki/Modal_window)
- Model-view-controller*. (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

*Model–view–controller.* (n.d.). Retrieved from WikiBooks:

[https://en.wikibooks.org/wiki/Computer\\_Science\\_Design\\_Patterns/Model%E2%80%93view%E2%80%93controller](https://en.wikibooks.org/wiki/Computer_Science_Design_Patterns/Model%E2%80%93view%E2%80%93controller)

*Node.js.* (n.d.). Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Node.js>

*Single page apps in depth.* (n.d.). Retrieved from Modern web applications: an overview:

<http://singlepageappbook.com/goal.html>

*Single-page application.* (n.d.). Retrieved from Wikipedia: [https://en.wikipedia.org/wiki/Single-page\\_application](https://en.wikipedia.org/wiki/Single-page_application)

*The Use Cases of Node.js.* (n.d.). Retrieved from Modulus: <http://blog.modulus.io/nodejs-use-cases>

*Visualizing a modern web development stack with a diagram.* (n.d.). Retrieved from { 100PercentJS }:

<http://www.100percentjs.com/visualizing-modern-web-development-stack/>

*Web Application.* (n.d.). Retrieved October 29, 2015, from Wikipedia: [https://en.wikipedia.org/wiki/Web\\_application](https://en.wikipedia.org/wiki/Web_application)

*Web Applications: What are They? What of Them?* (n.d.). Retrieved from Acunetix:

<http://www.acunetix.com/websitesecurity/web-applications/>

*What is a Full Stack Developer?* (n.d.). Retrieved from Laurence Gellert's Blog:

<http://www.laurencegellert.com/2012/08/what-is-a-full-stack-developer/>

*What Is Web Application (Web App).* (n.d.). Retrieved October 29, 2015, from SearchSoftwareQuality:

<http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>

*What Is Web Application (Web App).* (n.d.). Retrieved October 29, 2015, from SearchSoftwareQuality:

<http://searchsoftwarequality.techtarget.com/definition/Web-application-Web-app>

*Workflow.* (n.d.). Retrieved October 29, 2015, from Wikipedia: <https://en.wikipedia.org/wiki/Workflow>

*Workflow Tutorial.* (n.d.). Retrieved from PNMSOFT: <http://www.pnmsoft.com/resources/bpm-tutorial/workflow-tutorial/>

# Appendix A: Candidate Application Form

**Angelo, Gordon & Co.**  
 **ANGELO,  
GORDON  
& CO.**

**APPLICATION FORM**

Position information	
Position Title	
Department	

Vendor Information	
Vendor Company Name*	
Vendor Contact Name*	
Contact Phone*	
Contact Email*	

Candidate Information			
Last Name*		First Name*	
Phone*		Email*	
Address*		State*	Select a State <input type="button" value="v"/>
City*		Zip Code*	
Resume*	[Please attach with email.]	Cover Letter	[Please attach with email.]
Today's Date*			

Skills

Additional comments

Figure 43 Application Form

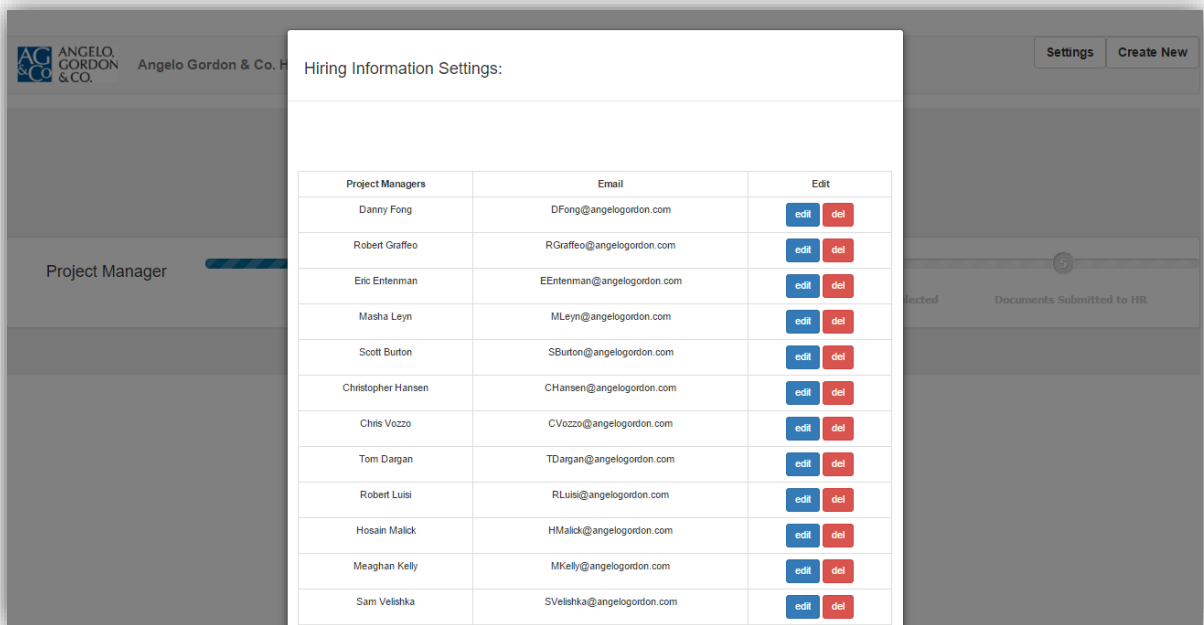
# Appendix B: User Guide

## Step 1: Initial Setup

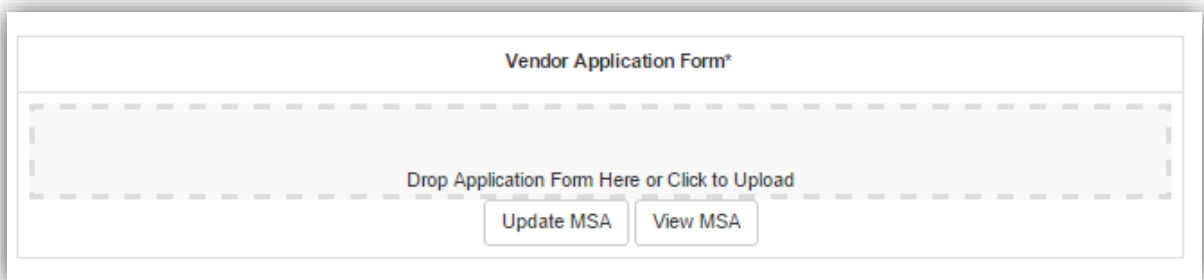
Please log into the system to view all processes.

Click on the Settings tab on the upper right corner of the home page, the Hiring Information Settings modal window will pop up on top of the main page. The detailed information of project managers, vendors, potential position titles, and email templates are all listed within the same page.

The blue Edit button is for editing information for each row. The red Del button helps delete the unwanted entries. Save the entries by clicking on the Save button.



Add an application form by clicking on the dashed boxes to select a file, and then save it to the system.




To add a new vendor of either category, click on the add button on the lower right corner to proceed. Each field is set as a required field. The user will not be able to save the information until everything is filled out.

General Vendors*	Email*	Phone #*	Fax	MSA*	Edit
Privosoft	example@example.com	100010	10101	<div style="border: 1px dashed gray; padding: 5px;">                     Drop MSA Here or Click to Upload  <input type="button" value="Update MSA"/> <input type="button" value="View MSA"/> </div>	<input type="button" value="edit"/> <input type="button" value="del"/>
JK Partners	example@example.com	100010	10101	<div style="border: 1px dashed gray; padding: 5px;">                     Drop MSA Here or Click to Upload  <input type="button" value="Update MSA"/> <input type="button" value="View MSA"/> </div>	<input type="button" value="edit"/> <input type="button" value="del"/>

Special Vendors*	Email*	Phone #*	Fax	MSA*	Edit
Modis	example@example.com	100010	10101	<div style="border: 1px dashed gray; padding: 5px;">                     Drop MSA Here or Click to Upload  <input type="button" value="Update MSA"/> <input type="button" value="View MSA"/> </div>	<input type="button" value="edit"/> <input type="button" value="del"/>
Cra Advisors	example@example.com	100010	10101	<div style="border: 1px dashed gray; padding: 5px;">                     Drop MSA Here or Click to Upload  <input type="button" value="Update MSA"/> <input type="button" value="View MSA"/> </div>	<input type="button" value="edit"/> <input type="button" value="del"/>
IVP	example@example.com	100010	10101	<div style="border: 1px dashed gray; padding: 5px;">                     Drop MSA Here or Click to Upload  <input type="button" value="Update MSA"/> <input type="button" value="View MSA"/> </div>	<input type="button" value="edit"/> <input type="button" value="del"/>

Edit the email details at the template section if any modification is needed. This is a global setting of the emails thus when modifying the content here indicates all future emails would follow the template inserted here.



ANGELO GORDON & CO.

Angelo Gordon & Co. H

Project Manager

Project Manager

Settings Create New

Add Role

**Vendor Email Template**

H1 H2 H3 H4 H5 H6 P pre
B
I
U
↺ ↻
☰ ☲ ☱
C ↺

Words: 0
Characters: 0

Hi,

We have a job openings for:

Position: `job_title`.

Description: `job_description`.

If interested, please reply with candidate applications and resumes (Application form see attachment)

Thanks.

Hire Direct System

Angelo, Gordon & Co

Save Vendor Email Template

**Approval Template**

H1 H2 H3 H4 H5 H6 P pre
B
I
U
↺ ↻
☰ ☲ ☱
C ↺

Words: 0
Characters: 0

Hi,

You have a job openings Request for `job_title`position from `hiring_manager`. Please approve or decline in following link.

Approve

`approve_link`

Approve

`decline_link`

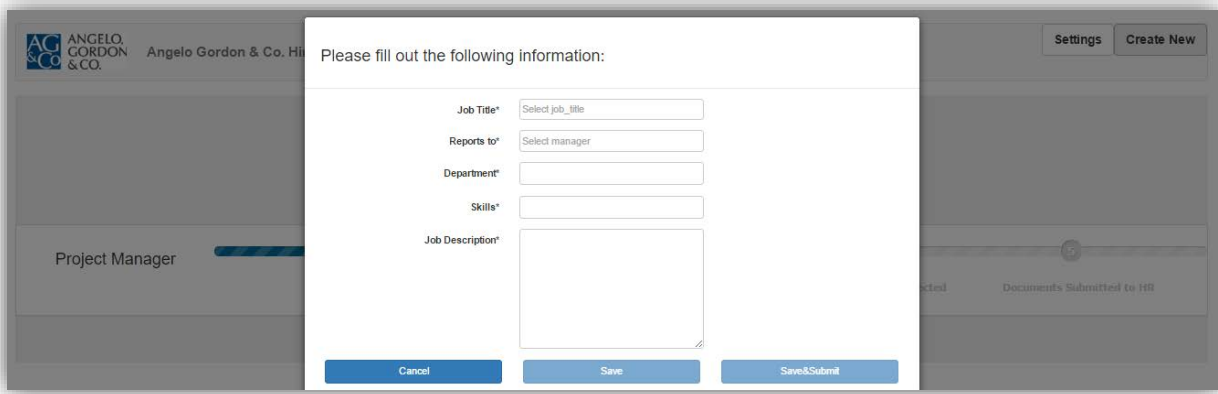
Thanks

Angelo & Gordon Co



## Step2: Create New Process

Click on the Create New button to add a new searching process. A modal window with a blank form looks as follows:

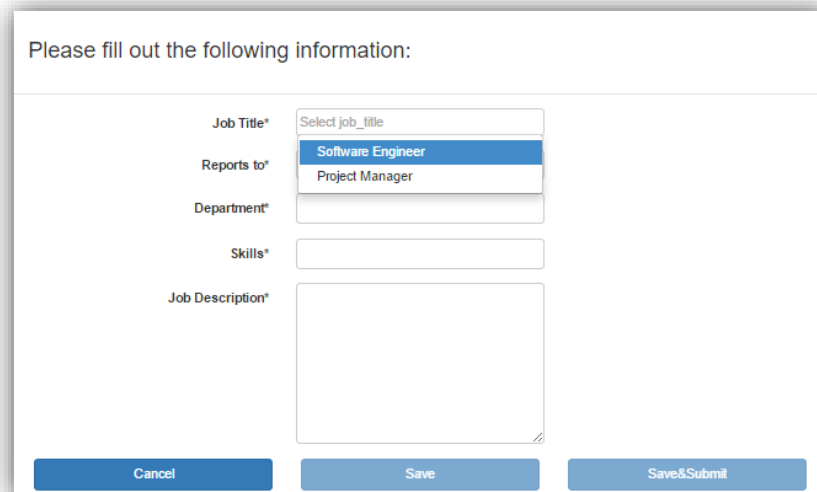


The screenshot shows a modal window titled "Please fill out the following information:" overlaid on a web application interface. The background interface includes the logo for "ANGELO, GORDON & CO." and a "Project Manager" role. The modal form contains the following fields:

- Job Title\*: A dropdown menu with the placeholder text "Select job\_title".
- Reports to\*: A dropdown menu with the placeholder text "Select manager".
- Department\*: A text input field.
- Skills\*: A text input field.
- Job Description\*: A large text area for entering the job description.

At the bottom of the modal, there are three buttons: "Cancel", "Save", and "Save&Submit".

Choose from the drop-down lists of the first two categories. This can be modified in the Settings page. All of the employee detailed information is built into the system thus the user may simply choose from a list of Hiring Managers.



This screenshot shows the same modal form as above, but with the "Reports to\*" dropdown menu open. The menu displays two options: "Software Engineer" (highlighted in blue) and "Project Manager". The other fields remain empty.

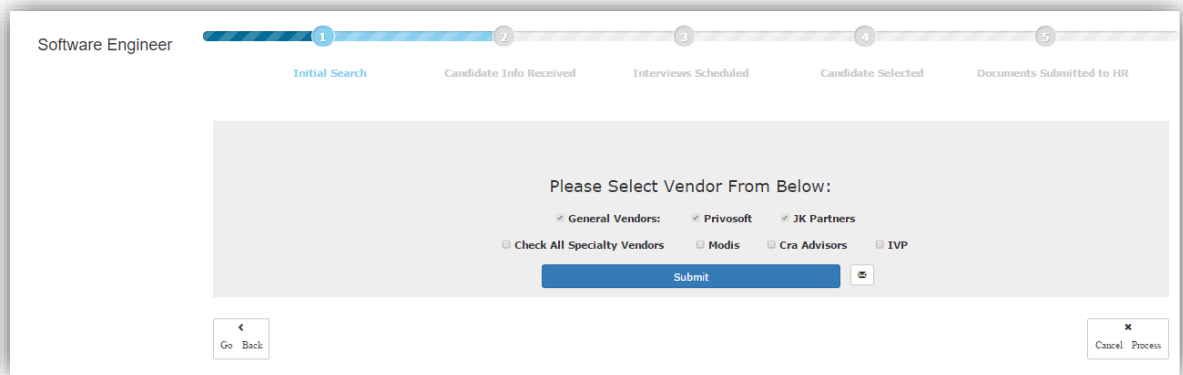
The example shows the result of adding a new process to the system: to create a process called Software Engineer, Reports to Scott Burton, IT department, with a skill of Programming. The new process looks as follows:

Please fill out the following information:

Job Title*	<input type="text" value="Software Engineer"/>
Reports to*	<input type="text" value="Scott Burton"/>
Department*	<input type="text" value="IT"/>
Skills*	<input type="text" value="Programming"/>
Job Description*	<input type="text" value="To program."/>

Click on Save & Submit to initiate the process. It will automatically jump to the vendor selection step. The user may select desired vendors to proceed.

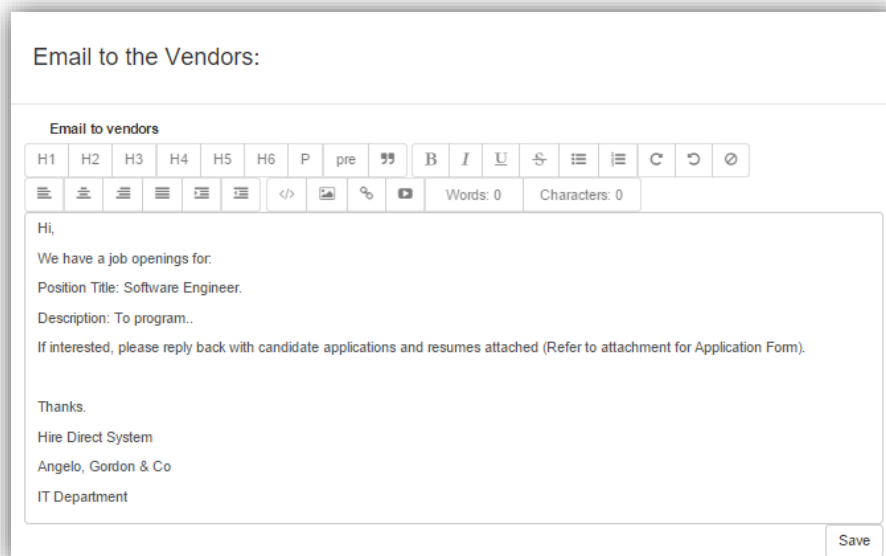
### Step3: Vendor Selection



The General Vendors category is selected by default and cannot be deselected. The user may choose as many Specialty Vendors as he/she wants. All vendor listings can be modified in the Settings Page.

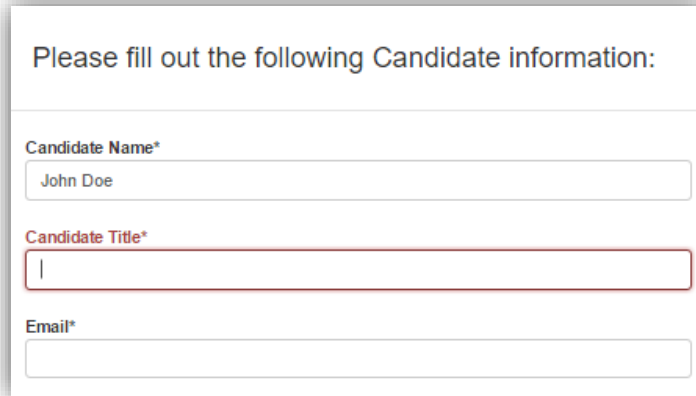
Click on the email icon to edit the content of the emails sent to vendors. Then click on the Save button to save and continue. The submit button would generate the edited email to the selected vendors.

You may go back to the last step by clicking on the Go Back button in the lower left corner, and cancel this entire process by pressing the Cancel Process button in the lower right corner. These actions function at any time of the entire process. If a process is cancelled by the user, it would show up under the Cancelled tab. All information will be stored and the cancelled processes can be re-initiated at any time.



#### Step4: Create New Candidate

Click on Create New to add a new candidate. We add a candidate named John Doe as an example, titled as Consultant. To add a resume and application form, click on the dashed boxes or drag the file and release. An unfilled required field will be marked red and the system would stop the user from proceeding.

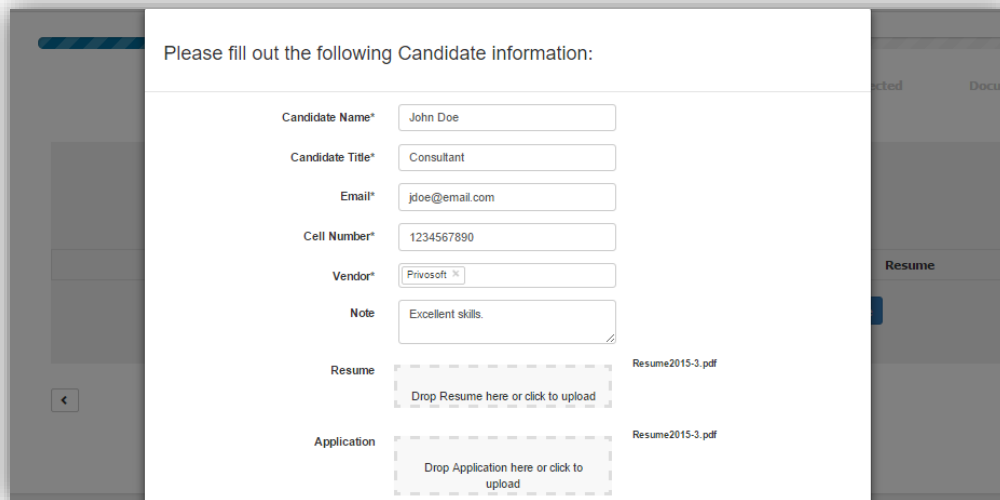


Please fill out the following Candidate information:

Candidate Name\*  
John Doe

Candidate Title\*  
|

Email\*



Please fill out the following Candidate information:

Candidate Name\* John Doe

Candidate Title\* Consultant

Email\* jdoe@email.com

Cell Number\* 1234567890

Vendor\* Privosoft X

Note Excellent skills.

Resume Resume2015-3.pdf  
Drop Resume here or click to upload

Application Resume2015-3.pdf  
Drop Application here or click to upload

After saving the information, the candidate will show up in the table. The uploaded information can be viewed by the View buttons. The file will show up on a separate browser tab.

## Step5: Schedule for Interviews

Candidate Information Received List:

Candidate Name	Candidate_title	Resume	Delete
John Doe	Consultant	<a href="#">View Resume</a>	<a href="#">Del</a>

[Add Candidate](#) [Confirm & Continue](#)

[Go Back](#) [Cancel Process](#)

Click on the Add button located within the Add Interview Category to schedule for an interview.

Interview Scheduler:

Candidate Name	Action	Interview Scheduler	Add Interview
John Doe	<a href="#">Interview</a> <a href="#">Reject</a>		<a href="#">Add</a>

[Finished All interviews](#)

[Go Back](#) [Cancel Process](#)

To create a new interview schedule, select the interviewer from the drop down list, and then select an appropriate date and time for interviews. Then click on Save.

Please fill out the following Interview information:

Select Interviewer\*

Select Date\*

December 2015

Sun	Mon	Tue	Wed	Thu	Fri	Sat
49	29	30	01	02	03	04
50	06	07	08	09	10	11
51	13	14	15	16	18	19
52	20	21	22	23	24	25
53	27	28	29	30	31	01
1	03	04	05	06	07	08

Select Time\*  :

Interview length\*  mins

[Cancel](#) [Save](#)

In the Interview Scheduler section, we can see the scheduled date and time. The user can use the red dashed button to remove the interview time, and schedule as many interviews by as many interviewers as he/she may. The same interviewer can appear more than one time for the same candidate.

To reject a candidate, toggle the Interview/Reject button off to the Reject side. Click on Finished All Interviews to proceed.

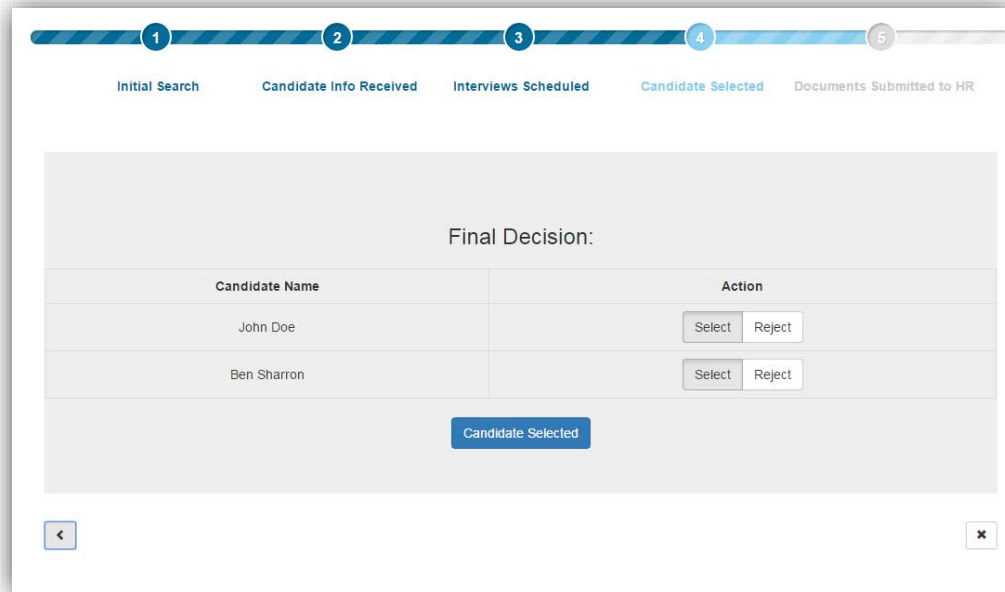
The screenshot displays the 'Interview Scheduler' interface. It features a table with the following structure:

Candidate Name	Action	Interview Scheduler	Add Interview
John Doe	<input type="button" value="Interview"/> <input type="button" value="Reject"/>	Scott Burton 17/12/2015 01:45 <input type="button" value="X"/>	<input type="button" value="Add"/>

Below the table, there is a blue button labeled 'Finished All interviews'. At the bottom left, there is a button with a left arrow and the text 'Go Back'. At the bottom right, there is a button with an asterisk and the text 'Cancel Process'.

### Step6: Final Selection

Click on Finished All Interviews button to stop interviewing all candidates and move on to the next step. The interview scheduler section may disappear from the current page. Select or reject John Doe for final decisions. Here let's accept John Doe as a desired candidate for this position.



Press Candidate Selected to proceed from the current step. This step only provides the user buttons to make final decisions. Then it comes to the onboarding processes.

### Step7: Send Files to HR

In the table below, click on View Resume to view the uploaded resume as pdf on a separate tab, and View Application would bring you to the application form on a separate tab.

Name*	Resume*	Application*	Background Check*	Work Order*	HR Email*	Action
John Doe	<a href="#">View Resume</a>	<a href="#">View Application</a>	Drop Background Check Here or Click to Upload	Drop Work Order Here or Click to Upload	<input type="text"/>	<a href="#">Submit</a>

[Finish Process](#)

[Go Back](#) [Cancel Process](#)

To upload a Background Check or Work Order, click on the dashed box to upload from personal devices, or drag the existed files into the dashed boxes and release. After the files have been uploaded to the database, two view buttons would appear to authorize the user to double check the files and proceed.

A text box with HR Email label is placed before the final action button. It allows the user to add as many targeted email addresses to send all files to as he/she wants.

Press Submit button to end the candidate seeking process and send all files to the HR Department and proceed.



## Appendix C: Code

---

Please refer to the following link for code resources:

<https://github.com/Tommzy/HireDirect>