Faculty of Engineering
Department of Mechanical and Mechatronics Engineering

The Design Process of UWCoopJobFinder

JSI Telecom
Ottawa, Ontario

Prepared by

WenChao Jiang

ID 20512856

W52Jiang@UWaterloo.ca

2B Mechatronics Engineering

September 21th, 2015

WenChao Jiang
Room 2, Suite 702, 208 Sunview St.
Waterloo, Ontario, Canada
N2L 3E1

September 21th, 2015

Professor David C. Weckman
Associate Chair Undergraduate Studies
Department of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Sir:

This report, "The Design Process of UWCoopJobFinder", prepared for my 3rd Co-op and 2A work term. This report is intended for the WKRPT 200 and it was written during my work term at JSI Telecom. I worked under Ralph Malek, my supervisor, as a software developer. JSI Telecom is a dedicated software company that create software for law enforcement and intelligence agencies. Due to security reason, I cannot fully disclose my department and other sensitive information. It is due to the same security reason and the lack of professional engineer within my department that I choose to write this report on a self-study project, which features the application of the same technology I used during my work term.

JobMine is a web application used by University of Waterloo students to find co-op jobs. The aged web application contains many problems and lacks regular updates, which in term impede students from finding jobs efficiently. This report

documents the creation of an application that is designed to improve the job finding process. The application, UWCoopJobFinder, is specifically designed to enhance only the job search process, one of the longest activity in the job application process. I wrote this report with the intention that some future readers might be programmers that might wish to contribute to this project. I hope this report will give these potential readers a better context of project and insights to how the final design take form.

I would like to thank Ayo (Paul) Soladoye Jnr III, a previous co-worker, for his contribution. Ayo completed a component of the project that is used to retrieve data from RateMyCoopJob.com. I would also like to thank Lixar IT Inc. and JSI Telecom for providing me with the skills to create this application. In addition, I would like to thank numerous students of The University of Waterloo for their feedback on the job search process. I hereby confirm that I have received no further help other than what is mentioned above in writing this report. I also confirm this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

WenChao (Bill) Jiang

ID 20512856

Summary

Although JobMine, University of Waterloo's internal web application, contains a wealth of information about the job postings, however, it lack some basic searching and filtering features. Without these feature, the thousands of job posting that students need to sort through each term consumes a massive amount of time.

The report begins by analyzing the entire job search process and identify areas of improvement. With opinions and feedback from other students, the report deduces that JobMine lacks important features such as keyword search, complex filter, and fast response time. In addition, the report analyzes the job search related tasks that students do outside of JobMine and find potential areas of improvement. In short, the job search related task that people perform outside of JobMine are mostly searching for the employer on Google, the job location on Google Maps, and the ratings on RateMyCoopJob.com. With all these criteria in mind, the report presents and analyze possible solutions.

Although a web application that contain all the missing feature will a great fit, however, a desktop application would be much better. A desktop application would have no cost of upkeep, less work to implement, and better performance. With imminent arrival of WaterlooWorks, a newer version of JobMine, a solution in the form of a desktop application would also retain more of its usefulness as opposite to a web application, which will likely be completely replaced by the new official software.

The application, UWCoopJobFinder, follows an N-tier design with data, business, and User Interface (UI) layers that perform individual duties. This design will separate each layer's responsibility to help produce clean and easy to maintain code. UWCoopJobFinder also features a straightforward user interface with complex filtering and support panel that makes job search faster and easier.

The report encourage students to use this application to save time and effort on job search. After the initial data download, every action and query are performed on the user's local database. This means that not only will students save time job searching, they will also generate less traffic on JobMine. The report also encourage students who knows programming to contribute to this project. Lastly, since it is against university policy to redistribute JobMine data, every user will require to parse JobMine for their own copy of the data. This repeated parsing put strain on the JobMine system and waste user's time. Therefore, it is recommended that Co-operative Education & Career Services officially sponsor this application so that data can be redistributed efficiently. Although this project is currently only for personal user, but the project could be extended to support MacOS and other programming language if wide adoption occurs in the future.

List of Figure

List of Table

Table of Contents

1.0 Introduction and Background

JobMine is University of Waterloo's internal web application that "helps students with the co-op process" [1]. It is an all comprehensive program that enable students to find, apply, and interview for jobs while allowing employers to post jobs, find applicants, and schedule interviews. However, the complex nature along with slow updates of JobMine causes much performance and feature lacking issues.

Although it would be great to improve every aspect of the system, however simply tackling the job search process, one of the most used functionality of JobMine, would go a long way to perfect the process. The purpose of this report is to document the design process of UWCoopJobFinder, an application that enhance the job search process on JobMine. This report will analyze the current job search process and pinpoint the problems that occur with the job search activities. In the following section, solutions to the previously identified problems will be explored. The report will explore the advantages and disadvantages of possible solutions and document the process of implement the chosen solution. Then, the exact implementation of the solution will be explored in-depth. Detail documentation will cover special areas of design that is not intuitive. In short, this report intent to explore the problem with the current system, identify the appropriate solutions, and explain the general software design so the audience will have a better context of the project. By the end of this report, the reader should have a good idea what problems the project try to solve and how it was solved.

2.0 Analysis of Current Process

The current job search process usually involve the students reading job information data on JobMine as well as elsewhere on the web. The analysis will be broken down into two components and analyze actions that will need to be performed inside and outside of JobMine. The analysis will explore the weakness within the process and focus on the objective of improving the process in terms of efficiency and ease of use.

2.1 Analysis of the Job Search Process within JobMine

Overall, the process of job search within JobMine is simple yet lacking in some aspect. To begin the job search, the student will have to first log in and enter the basic information like term, discipline, and levels. Although this is fairly polished, there are still limits of 3 discipline, 1 location, 1 job title, and 1 employer name in the selection. The limits of 3 discipline is especially problematic for first year Mechatronics student, who would try to find work in multiple fields, and Art faculty students, who would search in many related disciplines due to a lack of postings.

After searching for postings, the next step in the process requires the student to look through the table of job postings and pick out the ones that fit the criteria. The job posting table provides a good set of basic information like employer name, job title, and location. However, the table lacks important deciding factors like minimal duration of the coop and important information in the comments. Even with all these information, it is still probably not enough to give students a good idea of what the job is about. A quick way to give students a glance of the job description, the full detail of the job, would be much desired. But sadly the current implementation requires students to open job detail page in a separate web browser window in order to see the job descriptions.

The separate nature of the Job Detail page and job inquire table might be by design. This design choice could be the reason why the job inquire page does not have search functionality within the job description, a feature that would be very useful. Often time, keywords in the job description could give students a good idea of what the job is about as well as letting them filter out the unwanted jobs. If there are two postings with the same title of "Software Developer", the word appearance of "JavaScript" and "HTML" as opposite to "SQL" and "C#" would easily allow students to distinguish the skill sets required. Although JobMine does have search functionality on the job title and employer name, it is often not enough as a search tool to narrow down the selections because those two fields contain too little information.

In the end, one of the bigger downsides of the current JobMine implementation is the speed and response time of the user actions. Many times a job search with thousands of postings would require upwards of minutes for the results to be displayed in the table. This is made worst by the fact that students have to open each of "Job Details" page separately, which dramatically increase the time wasted while waiting for webpages to load.

In conclusion, JobMine could really benefit from the removal of 3 disciplines limit, more detail in job posting table, a quick glance of job description, keyword search within the job description, and faster response time.

2.2 Analysis of the Job Search Process Outside of JobMine

Although students could usually rely on JobMine, but sometimes external resource are needed in the job search process. One of the most expected action is to Google

search the employer in order find out more about the company. In addition, the exact location is very important if the position is near the student's hometown and other special situations. All these information are highly desirable and are a consistently the same type of information.

Another set of important information that the user could obtain is the ratings of the coop job posting. This information is readily available on www.ratemycoopjob.com, a website dedicates to coop job reviews. Although the information is readily available, it further require the student to search.

The three actions, Google search, map search, and review search, are estimated to be the most performed action that is applicable to all postings. But the common problem with the job search process outside of JobMine is that students are required to search for the information, instead of the information presenting themselves to the student. A solution that can change the paradigm of how information is delivered would significant improve the efficiency of job search and reduce time.

3.0 Improvement Solutions and Technology Selection

Over the past section, the various problems of the Job search process in the existing JobMine system have been identified. In this section, the report will identify how potential solutions could solve the problems. After some initial brainstorming, three potential solutions are found: Cloud hosted web application, independent desktop application, and native mobile applications. While a native mobile solution was also considered due to the growing numbers of smartphone and tablet, the solution was ultimately rejected. The first problem with a mobile solution is the lack of view space with smartphones. On the other hand, tablets are still not as prevalent as computers. In order to reach a very large majority, native IOS, and Android application will need to be written, which adds to the amount of work. Since this self-study project will need to be completed part-time by only one person, therefore the amount of work in order to implement the solution would be an important consideration.

In order to create a web application that can reach higher efficiency, the application must not rely on the existing JobMine infrastructure, and should work separately to ensure the maximum throughput. Since the JobMine job search usage is constantly fluctuating, peaking out at first round job search then decrease throughout the term, the use of a cloud solution could properly handle the change in usage. The cloud application could be configured to auto scale to meet peak demands while also able to save at low usage periods. A separate system from JobMine would also mean highly available access to the job search system, which will alleviate the effect of JobMine's downtime after midnight. If more work were put into to format the website correctly, the web page could be displayed nicely in mobile web browser. However, as mentioned before, mobile support has lower business value and creates

more work. At a high level, a web application sounds very capable and all comprehensive, but there is one major disadvantage: Cost. The amount of funds required to pay for the hosting might be very significant, especially when there are many users. Although an advertising and donation model could be set up to procure funds to keep the website running, these sources of income could be unpredictable and might not be able to sustain a website.

The second solution is to create a desktop application that will be run from the student's computer. The application will scrape JobMine or download existing data packages. The major upside for this solution is the lack of cost. Since each student is running the program on their own computer, there is no cost to host anything. The application is also completely scalable since each computer will run their own instance of the application. A native desktop application will also enable the program to do more instead of being confined to the web browser. In addition, a desktop application will access data from the computer's hard disk, which means it will be faster from the lack of internet delays and will also support offline viewing. Since "90 percent [2]" of personal computers are running Windows, publish only a Windows versions will still reach enough users. Unlike a website, which request more complex web configurations and design to handle many user access, a desktop application will only have one user, which reduces the complexity. Not only a desktop application is less work than a web application, the need for only one OS version will further reduce the effort. However, there are downsides. A desktop application will require an initial installation along with occasional data updates to re-scrape JobMine. In addition to the longer initial setup, the application won't be accessible from computers without permission to install the applications, such as school computers. Not only that the application won't be able to access everywhere, it also will not have future mobile support, which might be desired at some point.

From a high level, each solution has its own strength and weaknesses. Further analysis on how each solution can improve the problems identified inside and outside of JobMine will be explored in the following sections. Since the native mobile solution does not fully satisfy the basic criteria, the latter section will only focus on the web and desktop solutions.

3.1 Improvement for Job Search Process within JobMine

One of the primary problems with JobMine is the separate nature of the Job posting table and the job details page. The separation prevented users from accessing detailed information, such as comments and job descriptions, directly from the table. In order solve this problem, it is necessary to link the table and the actual job details. Once the job posting table has access to these detail information, then it will be able to perform keyword searches and present more information. This backend design of the data structure is universal and can be easily implemented in the web and desktop application.

From a user point of view, In order for a student to access more detail from the job posting table, a quick glance of job detail page can be implemented in the form of tooltips for both solutions. The tooltip would be the perfect way to give students a really good idea of the job while taking up no space in the UWCoopJobFinder's job posting table. As for the problem of lack of information in the existing JobMine posting table, the web and desktop application could be built to simply extend the job posting table. More columns could be added to show students the job comment as well as other information that is currently not presented. The solution could also implement complex filtering in order for students to systematic narrow down the job postings.

Although the web and desktop solution can behave similarly in solving most of the problem with JobMine, but it is ultimately performance that differentiate the two. A web application will always be slightly slower due to the latency and speed limit. On the other hand, modern computers are very capable at handle simple operations regard few thousand records. Although the web application could be optimized to perform similarly as the desktop application, however, the amount of effort required to analyze and performance test a more complex web application will be significant.

3.2 Improvement for Job Search Process Outside of JobMine

The set of activities usually performed outside of JobMine is to simply obtain more information. The web application could either implement direct links to the external resource or somehow embed information within detail panels. The direct links are likely to be a set of hyperlinks that open new web browser tabs of other website, which would be a generic and effect way to give students more information. On the other hand, it is also possible to simply embed another website as part of a slide out panel. These panels would not take up too much screen real-estate and only pop up when the user selects them.

Although similar solution could be implemented in the desktop application, but such solution could take full advantage of the freedom of the desktop. The external resources could be implemented in panels that are in a separate window from the main application. These separate panels could allow the user to freely customize and also take advantages of the multiple screens if applicable.

3.3 Considerations for WaterlooWorks

While JobMine is little outdated, University of Waterloo has invested heavily on the new "WaterlooWorks" that will replace JobMine in 2017. WaterlooWorks is expected to be similar to JobMine but with newer and faster system. Many of the currently problems, such as filtering, keyword search, Google map integration, and speed, are expected to be fixed in the new WaterlooWorks system. Although the solution could be changed to work with the new data structure of WaterlooWorks, but all the new improvements in WaterlooWorks might make any of the proposed solution obsolete. The upside is that UWCoopJobfinder will be open source to students who can keep on improving at a faster cycle, which might make UWCoopJobFinder useful in the case when WaterlooWorks deteriorate over time.

Even with this in mind, building a complete web application might just end up being wasted work, since WaterlooWorks might render it obsolete. The only upside is that the web application might be more specialized, which might make it slight better than WaterlooWorks, and due to the potential mobile support, something that WaterlooWorks currently does not have.

On the other hand, a desktop application could still be useful since it will have offline browsing and zero internet related delays, which make every action faster. A desktop application will also not be restricted to a web browser and utilize multiple windows and multiple screens. Since all the resources are stored locally in the desktop application, it will be even easier for the student to improve or make a plugin for the application. With all this in mind, the desktop application could be a desktop version of WaterlooWorks or more of a companion to WaterlooWorks.

3.4 Final Comparisons and Decision

While from a high level, all the possible solution solutions have its advantages and disadvantages, a more detail decision matrix will systematically compare all the solutions.

Table 1. Decision matrix that compare web and desktop solution

		Web Solution		**Desktop Application**	
Criteria	**Weight**	Score	Weighted Score	Score	Weighted Score
User Accessibility	10%	100	10	80	8
Performance	10%	90	9	100	10
Work Required	20%	60	12	80	16
Cost	20%	50	10	100	20
Improvement on job search activities within JobMine	10%	90	9	100	10
Improvement on job search activities outside of JobMine	10%	90	9	100	10
Usefulness After WaterlooWorks Release	20%	20	4	80	20
Total	100%		63		90

Each of the important criteria in the decision matrix, Figure 1, is given a weight and each of the solutions are given a predicted score base on the discussion early. In general, since the desktop application will only have a Windows version and will

also require installation, it scored lower than the web solution in terms of accessibility. On the other hand, the desktop application scores better than the web solution in terms of performance due the lack of internet related delays. Since the web solution will require a lot of configuration and setup of remote resources, it is expected to require a lot more work. Unlikely the desktop solution where one user access one set of computer resources, the web solution will require many user access the same website, which will add to the complexity and therefore effort required. Furthermore, the web solution will require an initial fund to keep the website running until money from other sources start to come in. It is also hard to judge whether an advertising or donation revenue model could even sustain the website. In short, there will mostly be more cost or work in order to keep a website running. The desktop application also scored slightly higher in terms of the improvement of the job search activities due to the desktop nature of the application. A desktop application will have no internet related performance penalty as well as multiple screen support. The last determining factor is the usefulness after WaterlooWorks release. The desktop solution scored much higher than the web application since a desktop application be used in conjunction to the new system. In the end, the lack of cost, lower amount of work, better implementable features, and resilience to WaterlooWorks release makes the desktop application the best solution.

4.0 Application Design

In order to build the best desktop application for Windows, Microsoft's .NET technology is the clear winner. Although C++ and Win32 technologies provide the highest levels of performance, but the better support for GUI technologies and ease of use simply put .NET ahead. [3] The application will also feature the new Windows Presentation Foundation (WPF) technology for UI display since it has newer and better feature that allow more extensibility than the older Windows Forms technology. [3] C# is also chosen to be the language of choice because it is one of the best-supported language behind all of Microsoft's technology and it is also the language that the creator are most familiar with.

4.1 Overall Architecture

The overall solution, like most applications, will feature an N-tier design pattern shown in Figure 1. The application will be divided into data access layer, business layer and user interface layer along with projects that define entity models and contain standalone utilities. The data access layer will be responsible for parsing JobMine and returning information for the program to be stored locally. The data layer will also be responsible for accessing other web data like RateMyCoopJob.com and communicate with the local database. On the other hand,

the business layer will execute any algorithm required and link the data with the UI layer, which will be responsible for user input and output.

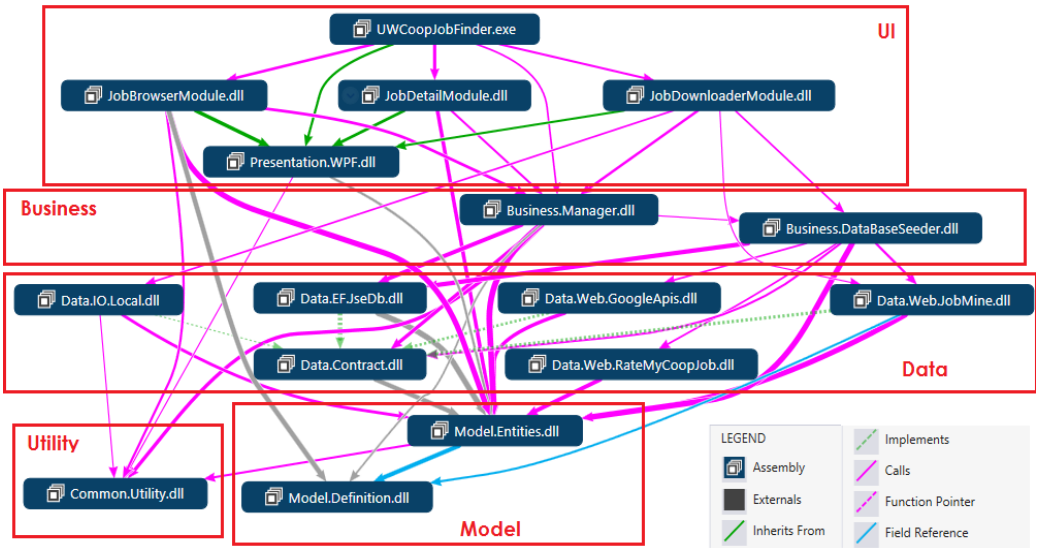

Figure 1. High-level diagram that identify all components and projects

4.2 Data Layer Design

Data access is one of the primary components of this entire application. Without the job posting data, there will be nothing to show users. The very first requirement of this data layer is to be able to scrape JobMine for all the Job posting data. These data can then be stored in memory so that the application can quickly access it. Lastly, other types of data like job rating and location data will also need to be imported. These parts of the layer will be discussed in more detail.

4.2.1 JobMine Data Parsing and Import

Although JobMine does not have an API that allows direct access to all the posting data, it is possible to request the entire web page and parse out the important

information by searching for specific Hyper Text Markup Language (HTML) tags. The JobMine data component essentially emulate a web browser by send requests and retrieve the entire webpage. In order get all the job posting data, the JobMine data component uses .NET WebClient class. The WebClient is modified so that it can store cookies for authentication. The WebClient will follow the standard procedure of getting job posting data. It will first send a Hypertext Transfer Protocol (HTTP) POST request containing credentials to log in. It will then go to the job inquiry page and search for all jobs. The amount of data on the job posting table is an incomplete representation of all the information of that job posting and will be called JobOverView. The WebClient will parse the job posting data page by page then complete all the information by getting the job detail page.

Although there is all this operation happening under the cover, the JobMine data component will still be treat as a normal repository with its own APIs. It is these APIs that the business logic calls in order to populate the database. The initial seeding will only need to perform once, this way all these exhaustive operation will not drain too much JobMine resources. However, regular updates will be required to update information that will change constantly, like the number of applicants for a given job posting.

4.2.2 Other Web Data Access Design

Since Google search and Google map provides a straightforward way to search given a Uniform Resource Locator (URL), they will be embedded as a web browser in the application. Every time the user select a new job, the embedded web browser will navigate to the new URL. There are many upsides in this design. First, it relatively a lot less work to use Google's web page and display. The user can also access another website from the Google search result, which mean more freedom.

The UI of Google and Google Maps is also familiar to users, which mean they will know how to use it without explanation.

On the other hand, RateMyCoopJob.com does not provide a way to search base on URL. All employers are stored within the main table which is then filtered by the UI when user input information. The downside to this is that it is impossible to follow the same pattern and simply navigate to a new webpage to get the desired information. However, the upside is that since all the employer information is in the same table, it is easy to scrape the data. Once the employer data has been received, the same pattern as the JobMine parsing process can be followed to get the rest of the job rating information. Another WebClient entity will use the employer Ids it retrieved from the main page table to get to the employer rating page. After scraping the data from the employer rating page, WebClient class will then use Job Ids from that employer rating page to get to the job rating page, which is then parsed as well.

4.3 UI Design

The user interface is one of the most important aspects of this application because it will define much of the user experience. The UI design will follow a very established MVVM (Model-View-ViewModel) design pattern for Windows Presentation Foundation (WPF). The MVVM will create “a strong separation of display from application state, logic, and behaviors”, which will in term “create clean and maintainable code” [3].

The user interface will feature most of the basic controls that user already know, such as buttons and tables. In order to browse job postings, the application will first need to download the job posting data from JobMine. This entire download operation is completely separate from job browsing and will, therefore, be separated

into its own complete view. On the other hand, the job browsing view will have a main table displaying posting data as well as side helper panels that take care of filtering, job detail, Google, Maps, and rating operations. Although many of these control and designs pattern are intuitive to an experienced desktop C# programmer, there are some design patterns that are more notably different and will be covered in following sub-sections.

4.3.1 Filter Centric Design

One of the most needed features of JobMine is complex filters. The lack of filtering result in a cluttered job posting table and reduce the student's efficiency in find the right jobs. Job posting table is currently designed so that each column of the job posting table is independently filterable, which correspond to the employer and job title search bar. Those two search bar are each column's independent filters that will remove any posting that does not contain the keyword that is being searched. Although this type of filtering is simple, it does have significant limitations: there only can be a maximum of one filter for each column of the table. In addition, the current JobMine implementation lack the important search/filtering on the job description.

In order for UWCoopJobFinder to allow users to build and manage many complex filters, it follows a filter-centric design. In this new design, filters are treated as a first class citizen and as an individual entity instead of an addition to the posting table columns. The filters will be shown in a separate table by itself with a check box to indicate if the filter is active. In addition, filters can be named, stored, modified, and disabled. All of this ensure that users have the maximum control of the filters system. Within filters, different categories of the filter are defined. Users can set the filters to search for keywords, define unlimited disciplines, and so on.

Although all these emphasis on filtering might seem excessive, a look at job search use case will demonstrator the usefulness. Bob is a hypothetical first-year mechatronics student who is looking for coop. He knows he does not want work in quality assurance related jobs. He would like to work in mechanical related jobs since he has Computer Aided Design (CAD) experiences and also a programming job that utilizes C++, a programming language he learned in high school. In the new UWCoopJobFinder, he would simply create an anti-filter, a filter that filter out a job if the filter criteria are meet, which search job title for words like "quality" and "assurance". This filter will remove any quality assurance job. Then he could create a separate filter that search for the words such as "CAD" or "C++" in the job description. This second filter will only display jobs that are likely to be CAD related mechanical jobs or programming jobs.

If Bob changed his mind at any time, he could uncheck the quality assurance anti-filter and allow him to see testing jobs. When user starting to build multiple filters, this quick control will allow them to easily see result from multiple filter combinations. With the addition of the score filters, a filter that raises the score of a job when the filter criteria are meet and does not remove job posting from the table when it is not meet, users are likely to create many filters that tailor to their specific skill sets, location, or other needs. The multiple filter capability will allow the users to easily manage filters and separate each filter's responsibility in a transparent manner as seen in figure 2. There is no equivalent solution in the current JobMine system that can replicate the same functionality of the two filters that was mentioned previously. The filter system will allow users quickly narrow down choice instead of wasting time looking through all the job posting.

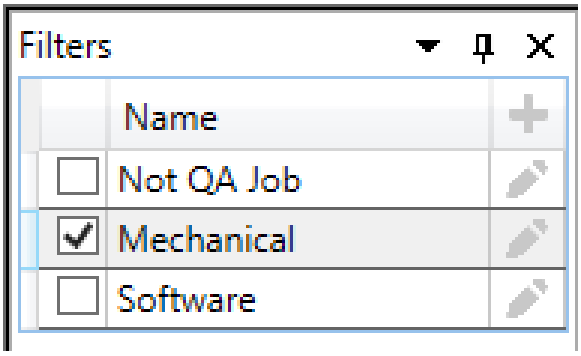

Figure 2. Filter Table

4.3.2 Support Window Design

One of the biggest features is the set of support windows that existing in UWCoopJobFinder. It is these support windows that is used to display job description (Job Detail Panel), Google search of the employer (Google Search Panel), Google Map search of the employer location (Map Search Panel), and job reviews data from RateMyCoopJob.com (Job Review Panel).

All the support panels follow the same structure and implement a base class call "JobDetailViewModelBase". The base class is setup to update when the user selects a new job posting from the table. In order to create a new support panel, the creator of the new panel will need to simply implement the base class and override the "NotifyPropertyChanged" method in order update information for the user to see. This simple structure makes it easy for anyone to make a new support panel that could potentially do all type of things. For example, a new panel could be created to parse the job requirement information. The panel could use some logic to find all the job requirements, such as knowledge of C++ or CAD, and present it to the user for easy viewing.

5.0 Conclusions

This report documented the design process of UWCoopJobFinder, an application that enhances the job search process. The documentation cover every aspect of the design such as identify existing problems, chose appropriate solution, and detail application design. This report provides the reader with a better context of the problems UWCoopJobFinder attempts to solve and how these problems were solved.

The current job search process using JobMine was analysis within this report. The result was that JobMine lacks useful features, keyword searches, complex filtering, and fast response. Actions that student perform outside of JobMine, such as Google search of employer, Map search of location, and review search of the job rating, were also found through this report.

Furthermore, many different types of solution were explored. Mainly, a solution in the form of a web application was discussed in depth along with solution in the form of a desktop application. The advantage and disadvantages of both solutions were compared systematically using a decision matrix. In the end, the lack of cost, lower amount of work, better implementable features, and resilience to WaterlooWorks release makes the desktop application the best solution.

Last by not least, the application design and technology selection was discussed to help the reader understand how the solution was implemented. The application follows a standard N-tier design with a data layer, which parse websites and manager local SQL database, business layer, which execute logic, UI layer, which control user input and output. In the sub-section, the specific of how web pages were parsed are discussed in detail. Most of the parsing operation follows what a user would normally do. In addition, the application contains some unique design patterns, such as filter-centric and support panel design, were also discussed in

greater detail to help readers to understand and potentially allow them to contribute to the project.

6.0 Recommendations

A lot of work has been put into creating UWCoopJobFinder so that students can save time and effort while find coop jobs. It should go without saying that this application should be advertised, distributed and used by as many students as possible. It would be necessary to convince the Co-operative Education & Career Services (CECS) the usefulness of a desktop application like UWCoopJobFinder. If the application is accepted and officially sponsored by CECS, then it would receive wider adoption.

At the time of this report, all instances of the application will need to individually parse all the JobMine information. It would be a good idea to establish a process where all information on JobMine will be parsed once, and the information will be redistributed so that JobMine will not be put under any strain. This is currently not done due to the security complexity behind re-distribution and lack of permission from CECS. Again, a formal sponsoring of this program would make the application faster and reduce JobMine's traffic.

If widespread adoptions occur, it would be worthwhile to further refactor the existing code base so that it allow students of all programming backgrounds to contribute to this project. Everyone will think of new functionalities, which could then be added piece by piece. By making interfaces for popular programming languages like C++, Python, and Java, the growth of this application could be dramatically accelerated. Wider recognition might also prompt the production of a MacOS, or Linux version of the same application so that an even wider audiences can be reached.

References

[1] "About JobMine," JobMine, 2012. [Online]. Available at: https://uwaterloo.ca/jobmine/about-jobmine. [Accessed: 2015].

[2] K. Mackie, "Microsoft Admits Windows Use at 14 Percent," Redmondmag.com, Jul-2014. [Online]. Available at: https://redmondmag.com/articles/2014/07/14/windows-use-at-14-percent.aspx. [Accessed: 2015].

[3] "Choose Your Technology," Windows Dev Center. [Online]. Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/dn614993(v=vs.85).aspx?f=255&mspperror=-2147217396.