CPSC4360-01, CPSC5360-01 - Software Engineering
Spring Semester, 2011
Project Specifications and Instructions

Submission date: Monday, April 18, Wednesday, April 20, and Friday, April 22, 2011
Submission venue: MA111
Time venue: 10:10-11:00

Project Description ('Matching your resume to the best job'):

This project ('Matching your resume to the best job') requires you to adopt a use-case driven object-oriented approach to develop a system to support the **process of matching your resume to best job in terms of preferences.**

Motivation:

Job hunting is a very delicate issue nowadays. Employers have increasingly relied on digitizing job-seeker resumes, placing those resumes in keyword-searchable databases, and using software to search those databases for specific keywords that relate to job vacancies. Many companies now use these technologies. In addition, many employers search the databases of third-party job-posting and resume-posting boards on the Internet. According to the National Resume Writers' Association, more than 80% of resumes are searched for job-specific keywords [http://www.quintcareers.com/resume_keywords.html].

Difficulties:

The bottom line is that if you apply for a job with a company that searches databases for keywords, and your resume does not have the keywords the company seeks for the person who fills that job, then your resume is unlikely to be considered.

Structure:

This project deals with a job specification published by a company and a resume written by a job hunter. There exist rules to write a job specification, such as most of them contain preferences about the languages, require work authorization or sponsor that, salary range, goals/expectations, recommended and required qualifications, and so on. However, there is no unified description of a standard pattern to write a job specification. Likewise, there is no unified description about how to write a resume. For example, few jobs are published at the address: http://www.structurestudios.com/website/company/jobs.html

The 'Matching your resume to the best job' Computer Support:

Previously, the automated job-seek software engines look only for a limited number of about 100 words from the resume. Recently, professional tools consider using keywords throughout the entire resume. Many resume designers recommend to front-load the resume with keywords to maximize the matching between the job company and the job hunter. Other career experts still advise a bare-bones spewing of keywords labeled "Keyword Summary," a more accepted approach is to sprinkle keywords liberally throughout a section early in the resume labeled "Summary of Qualifications," "Professional Profile," or simply "Profile."

The 'Matching your resume to the best job' Requirements:

1. To develop a system that will hold information about registered job companies searching for employers.

- **1.1.** To display the job offerings in a public website/magazine;
- **1.2.** To be able to match a received resume (and/or cover letter) with the job offer.

1.3. To inform the job hunter about his/her eligibility to be short interviewed.

2. To develop a system that will help the job hunter to find open jobs.

2.1. To use the resume keywords to match with the public website/magazine the job offerings;

2.2. To be able to match a received resume (and/or cover letter) with the job offer based on a thorough search (such as parsing, or re-searching for the second category of keywords).

2.3. To calculate the 'matching function' between the hunter resume and the job offerings. This 'matching function' can be a mapping between two text files and the interval [0, 1], with the meaning that the closer to 1, the higher matching between the two files.

3. The system must be able of future expansion to incorporate information about existing competitive 'matching your resume to the best job' and improve the current system.

Your team will work together to perform Object-Oriented Analysis and design and implementation of the proposed System. If you find that certain important information about the requirements is missing or ambiguous, you may explore on your own and suggest additional requirements or refine given requirements. While doing so, make reasonable assumptions. Your assumptions must not conflict with each other or with the given requirements. The problem and solution space needs to be decomposed among the classes. The system is to be modeled in UML and implemented in Java. It is not however compulsory to program in Java, hence you can actually choose any other object-oriented programming language, such as C++, Python, etc.

Instructions

You are required to use Java (or another object-oriented programming language) to implement classes according to specific software requirements, especially the persistency requirement. You should be able to demonstrate the functionality of the system through a menu-driven interface. It can be a text-based menu, GUI or web-based interface. Type of interface does not have any impact on project evaluation.

You should make all functions very easy to use (user friendly). For example, you should not expect the user to enter all parameters required in one long string. You can use GUI, but you may have to bear the risk of having a system that does not work correctly with the test driver (see following paragraph).

Testing

You are required to perform **black box testing** on the functionality of classes and the interaction between them. To carry this out you may either implement a test driver or use JUnit to execute automated testing. Ideally the menu-driven interface and the test driver can be combined as one program. If you have difficulty in combining them, it is acceptable to separate them into 2 programs.

Data Persistency

You are not allowed to use a database management system (DBMS) to meet the persistency requirement. However, you are allowed to use Java file system to retrieve and populate the data by using the **object serialization** to meet the persistency requirement.

Deliverables:

I. Project Report:

Your report should include the following:

(a) A complete **use-case diagram** of the proposed system illustrating functionality to be developed by your team. A **domain model** is also requested for later refinement into the class diagram.

(b) A **class Diagram** to illustrate design for the proposed system. The diagram should include classes, associations, multiplicities, attributes, and methods. You are required to use UML notation for drawing the Class diagram. Please choose relevant and meaningful names for classes, attributes, associations, and

methods. Level of details for class diagram e.g. how detailed method signature should be, is left to the team members' decision.
(c) Any assumptions, interpretation, and limitations that you have included while developing the models. Please be concise. Write all these as a separate section at the beginning of your report or after the introductory note, if you have any.
(d) An **interaction diagram** of the most complex use case scenario chosen by your team; a **state chart** for any object from your design; and an **activity diagram** for the system you propose. If you find that drawing one activity diagram is difficult or not possible, you are allowed to provide multiple activity diagrams. However, a brief reasoning for the same should be included in the report.
(e) **Test cases** used for automated testing.
Important: The total page limit for report is 15 pages, single spaced letter report (include all diagrams, appendix and references if any). The report will be assessed for abstraction, completeness, clarity, correctness, structure and innovative design /discussion /presentation.

II Project Demonstration

Each team is required to demonstrate:
- functionality of its system through a menu-driven interface;
- testing of its system through an automated test driver.
Each team is to submit its project source code along with report. Put the source code in a CD. The content of the CD should at least include the following:
a) Source code of the system. Put them in a subdirectory called **"source"**.
b) Test driver and test cases that team used for testing your system. Put them in a subdirectory called **"testing"**.
c) A "`Readme.txt`" file containing instructions on how to compile and run your system. Indicate important configuration information, e.g., the class-paths (if any) that need to be set. Any special requirements, e.g., a web server must be installed, or MySQL must be installed etc., should be indicated as well.
Note: As we will be using the source code on your CD for demonstration, please ensure that all the required files in your CD are not corrupted. You will not be allowed to use a fresh copy of your source code during the demonstration. Remember to label your CD with your team members' names (at least).
There is no need to include additional third party software in your CD.
What to bring for the demonstration: Each team is to bring one laptop for doing the demo. We will run your system on this laptop. Ensure any third party software necessary for running your system is installed.

Important points to note:

Each team has about 10-15 minutes to do the demonstration during the week starting April 12, 2010. You will be providing your own test data. Size and variety of data should be adequate to demonstrate the functionalities of your system. It is preferable that your system has a 'bulk loading' function to load all the test data in a single operation (say from text file(s) containing the test data), rather than adding all the test data one-by-one into your system (which can be painfully slow).