ITMD 523

Advanced Topics in Data Management

**Career Services**

Team Members

Siddharth Singh – [ssingh108@hawk.iit.edu](mailto:ssingh108@hawk.iit.edu) (A20401776)

Prateek Deshmukh - [pdeshmukh@hawk.iit.edu](mailto:pdeshmukh@hawk.iit.edu) ()

Vighnesh Sawant - [vsawant@hawk.iit.edu](mailto:vsawant@hawk.iit.edu)

**Introduction**

* The project is aimed at developing an offline application for career services of Illinois Institute of technology.
* This system can be used as an application for the career services of the university to manage the student info with regards to co-op, internships and full-time jobs.
* With the help of student login, they will be able to upload the info and can generate cv automatically.
* Employers can also log into the system and can search candidate profiles based on their request.
* It is centralized repository of student’s profile.
* Also, students will able to view employers list along with positions they are offering.
* The hybrid recommender systems exploit the jobs and user profile to generate personalized recommendation of jobs for candidates.
* All student data will be warehoused according to year and department.
* Companies will disclose positions and job availability to career fair services.
* Eligible student can apply for the job based on the criteria provided by the company.
* Student can give pre- aptitude test before the interview through electronic exam in the software.
* When a student logs into the account, he/she can see their status.

**Modules**

The project contains following modules:

* **Administrator module**: Add, view and mail to functionality for a faculty.
* **Career Services module**: Registration of Companies, Arrange Job Fair for eligible students, can send mails to eligible candidates, displaying current Jobs offers.
* **Student module**: View, update, add photo to a profile, Generate and view CV.
* **Faculty Module**: View the profile, Student list, can view Job Fairs and can see the eligible candidates list
* **Admission Module**: It consists of registration form containing student information filled by Student itself.
* **Contact Us Module**

**Program Interface**

This section explains how the user communicates with the program. It must be stated how the user will start running the program, i.e. how the programming environment will be set and activated, what is the command to be entered to execute the program, and what are the parameters (if any) that must accompany the execution command. It must also be stated how the user terminates the program. This section is just for activating and deactivating the programming environment; the details of the communication steps will be explained in the following sections.

* Open any web browser and enter localhost to view the home page of the offline career fair website.

**Program Execution**

This section explains in detail the execution of the program from an end-user’s point of view. The descriptions of the algorithms, data structures, implementation, etc. should not be included. This section covers what types of inputs will be supplied to the program by the user, what are the outputs of the program, what is the functionality of the program, the descriptions of menu options and different parts of the program, etc. This section can be thought as the core of the “user manual” of the program. It is good practice to include screen outputs and some figures (e.g. expressing the structure and parts of the system). Do not give all the explanations in a single section; instead, divide this section into subsections for ease of reading.

**Input and Output**

The format of the input and output will be explained in this section. If the program also gets some input from input files and produces some output on output files, then the exact format of the files must also be given. It is advisable to include some example inputs and outputs, and their explanations.

**Program Structure**

This is the “technical manual” of the program. An overall structure of the program should be given. Then, each part of the program (subprograms, modules, classes, etc.) should be explained in sufficient detail such that the reader can understand the statements, algorithmic logic, etc. The data structures (data types, variables, domains, etc.) should be explained. If desired, the explanation of the data structures can be covered in a separate section (this may be better if the entire program makes use of similar data structures) and this section is reserved for detailing the program code. Do not give all the explanations in a single section; instead, divide this section into subsections for ease of reading.

**Examples**

It will facilitate understanding the program and the document if some examples regarding the execution are included. For instance, a particular input can be given and the operations and the output of the program under this input may be specified. The examples should be chosen in a manner such that different behaviours of the program may be observed under different inputs.

**Improvements and Extensions**

In this section, the parts of the program that need improvement and some possible future extensions are discussed. The weak and strong points of the program can be identified. Any shortcomings or defects of the program can be stated. Also, the deviations from the plan at the beginning of the project can be indicated, i.e. the features, user options, data structures, etc. that were initially planned to be included, but could not be done so for some reasons.

**Difficulties Encountered**

It is quite valuable for an educational project to identify clearly the difficulties encountered within the project period. The difficulties due to the programming language (maybe it is a new language for you), due to the nature of the problem, etc. should be stated. Also, the effects of these difficulties (forcing to use different ways, abandoning some planned features of the program, etc.) should be explained.

**Conclusion**

This section is for the conclusions about the project.

**References**

The references referred to within the document, if any, should be listed in this section in an appropriate format. It is good practice to include some references that show the materials from which the ideas, algorithmic concepts, etc. made used in the project have emerged. This implies that the project is research-oriented.

- Appendices

Program source code must be included as an appendix. Also, if necessary, some information can be given inside appendices. For instance, an appendix is a suitable place for the execution details for some complex sample executions.