Project by Chris Stiteler
ResumeCMS
for CSCIE10B
Project Write-up

Description:

This program is a GUI based candidate management system for organizing resumes in a neat and organized manner. The user will be able to add resumes manually by inputting candidate data by hand, or they can import a resume, which gets parsed automatically for email, phone, state, and zip code information. A graphical data table represents the entire database of candidates, and the user can double click on each candidate to bring up a window showing the resume for that candidate. The user can also delete candidates from the database.

Instructions:

To add a resume, you can either input candidate information on the candidate panel on the left side of the application, or you can click the "Parse New Candidate" button. The parsing makes a best guess at the candidates phone number, email, zip code, and state. If you want to delete a candidate, select the row represneting that candidate in the table, and then click the delete selected button OR you can selected delete from the edit dropdown menu. To exit the application, you can hit the X in the window, OR you can hit "ALT-F" for the file menu and then "CTRL-X" for the exit shortcut. (Instructor note: On the first start up of the program, if it's not finding the "candidates.db" file I included to seed the database, it may throw a background error to the console, but after that it should save correctly). I have supplied a few sample resumes for you to test the parsing yourself.

Features:

Automatic Resume Parsing
Persistent "Database" that loads your candidate records on start up.
Simple intuitive UI

Expansion:

To expand the project in the short term ("If I had one more day to work on this…"), I would implement search by name and keyword and an easy way to edit a candidate already in the database. More advanced features that I have the opportunity to add in the future are emailing a resume out (as in, to a client, or hiring manager), scoring a resume based on keyword search (would be a fun and engaging algorithm), checking for duplicate candidates on parse in, add more functionality to the candidate table (sorting, right click functionality, etc), and perhaps a method by which you can download the candidate database in .CSV format or other related format for safe storage locally.

Reactions:

My reaction to writing this project was overall very positive. My number one goal for this project was to focus on "Model View Controller" design to both improve the separation of

concerns between each of my classes. I also took it upon myself to learn the package model to help with this pursuit. The most fun I had during the project was implementing the resume parsing for a candidates email, phone number, state, and zip code. This could be expanding to guess a person's name (maybe? it's the first thing on the resume, right?), their GPA, their major, years experience, etc. I think it was a great choice for me since I'm a recruiter by trade during the day, and I always think I can improve upon the system that we use internally. One of the biggest challenges for me in the model view controller architecture was making sure I wasn't allowing the view to touch the data, and I must say, it was very tempting at times to do this and go around the controller. I can say, though, that in the end, when the project became ever more complex, having structured the program in this way was a godsend when it came to debugging the program. At no point was I confused where an error was happnening, if it was a GUI problem, the error was in the view, if it was a problem with the data, the error was likely somewhere in the database or the controller. Thanks for the opportunity to work on this project! (Also, thank you Dimitri for all your hard work and feedback as my TA).