Assignment 5: Medium Fidelity Prototype
Job Search Tracking

William Richard and Guvenc Usanmaz

March 29, 2013

1 Prototype Video

A video of the prototype being used can be found here.
`http://youtu.be/gJs-ZeQcFUA`

2 Features from Previous Assignments

We incorporated much of the design from previous assignments into this prototype. The syntactic flow of the user actions, as well as the general layout of the pages have remained the same from the original specifications and the storyboards. We realized that the ability to delete objects had not been included in the previous specifications, so we decided to add that functionality. We also discussed about an alternative design where system is organized around operations instead of objects. There would be Create, Edit, View and Associate pages instead of objects pages. However, we decided that this design would be more confusing for users and sticked with old design.

3 Differences from design and implementation

There aren't many features that are different from the design, rather there are things that we were unable to implement. These include, but are not limited to

- Editing objects
- Specifying relations between objects
- Full insert functionality for objects. When a user wants to add a new object, the form is already populated with a sample object of the correct type. There is only one object of each type that can be added.
- Deleting any object. The user can delete the sample object they added in a previous step, but they are not able to delete objects that were in the system when they started.

- Contact import.

4 Limitations of Prototype Caused by the Software

We decided to use HTML to prototype our project. With just HTML, without any database or other code running, we were unable to have truely dynamic pages. As mentioned above, we didn't allow the user to add any arbitrary object - when they try to add an object, the form is already filled in. Furthermore, ~~they~~ are only able to add one object of each type - one company, one job, one note and one person.

The main problem to overcome to simulate a dynamic system was storing the system state. Once the user adds a company, they should be able to navigate to any other page, and still have the company they added present when they return to the company list page. We did two things that allowed us to store system state, and easily simulate a dymanic system without having any code running in real time. First, we made a page corresponding to each possible page the user may see - the company page with just the canned data, and the company page with the same data and the single new object added. We made pages like this for every object, as well as an 'add object' page for each object. We then wrote a script that made many copies of these original files, putting them in a directory structure that stores the system state.

The state denoted of if each of the objects had been added or not - whether the company had been added, whether the job had been added and so on. Since there are 4 objects, there are 16 possible states. We represent the state as a string with a letter for each object, if it has been added. For example, if the company and job had been added but the person and note had not, the state would be 'cj'. If the person is then added, the state is changed to 'cjp'. If, finally, the note is added, the state is 'cjnp'. The script made sure that the "Add Object" page linked to the correct new state directory when it copies that page from the original location. Thus, when the user adds the object, they get redirected to a page that looks identical to the original list page, but in a different directory that reflects the new state, and the object added to the list. When they navigate to other pages, without adding or removing objects, they are staying on pages in the same directory, so the state is not lost. Removing the added object was achieved in a similar way.

5 Demo Creation Script Usage

5.1 Dependencies

The demo creation script relies on Python version 2.7. It can be downloaded from here, or using your system's package manager.

`http://www.python.org/download/releases/2.7/`

5.2 Running the script

Execute the script by calling:

```
python2.7 <Path to script location>/create_demo.py
```

This will create a directory called "demo" in the same directory as the "create_demo.py" script. To run the demo, simply go to the demo directory and open the demo/START/start.html in your web browser.