An Ideal Headmaster Interface

Joseph Crawley

November 26, 2012

Abstract

As a website, Headmaster, a student database with information associated to each student, is very sleek and easy to use in its current form. However, a command line prompt which takes in simple English commands would be an ideal interface because it is easy to understand and users can access information quickly.

Contents

1 Explanation of Design

When Headmaster is opened and after the user logs in, the only thing to appear on the page is text that would say "What would you like to see?" The user would then enter a command in plain English of what information they want to access or what changes they want to make. The interface would then remove the original text, keep the search bar at the top of the page, and display the information the user requested. If the user asked to change information about a student, a confirmation message would appear. For example, if a user tried to change Matt Damon's GPA from a 3.0 to a 3.5, the program would alert a confirmation screen saying "Are you sure you want to change Matt Damon's GPA from a 3.0 to a 3.5?" When the user confirms the change, they are shown Matt Damon's student profile with the changes made. The following is a list of some of theacceptable commands that the interface would accept to view the GPA of a student named Matt Damon:

- "Show me Matt Damon's GPA"
- "I want to see Matt Damon's GPA"
- "Matt Damon GPA"
- "matt damon gpa"
- "MaTt dAMON gPa"
- "What is the GPA of MAtt Damon?"

2 Usage Scenarios

In this section I will demonstrate how the interface deals with finding basic information and how it deals with errors.

2.1 Adding to the Database

If the user wants to add a new student to the database, the user would type in, "Add a new student." The database would then bring the user to the create a student page where the user would fill in the required fields to complete the student profile. However, the user could type in information about the student while writing the command. For example, the user could type in, "create a student named Britney Spears with a major in Recording Arts" or "create student Britney Spears major Recording Arts" and create a new student in the database with the name and major already filled in with what the user entered. A similar process would occur for creating new grants, awards, and other things that Headmaster contains.

2.2 Search

Suppose the user wants to find a particular student. The user could type in "Search" followed by the name of the student he/she was looking for. Once the user submits the command, Headmaster returns an ordered list of students that match the search input. Next to each student is some identifiable information including major, year, and birthdate. The user can then type in similar commands they would in the home screen except the numbers in the ordered list can be used to refer to a specific student on the screen. For example, if Harry Longbottum is student number 2 on the list of returned results, the user can input, "show me student 2 food allergies", and Harry Longbottum's food information would appear. If the user asked for a student that shared a name with one or more students, the same list page of results would appear. Headmaster would then accept a number input by he user of what student the he/she is trying to access.

If a user tries to search for a student that does not exist, the program will show return an ordered list of students with similar names or attributes inputted by the user. For example, if the user searches for a student named Shaquille O'Neal not in the database, on the ordered list that appears, the user Jermaine O'Neal may be shown by Headmaster. Also, the user will have an option on the bottom of the page to create a new student named Shaquille O'Neal.

3 Rationale

I think this design is the most ideal for Headmaster because it is very easy to use and is efficient. It is simple because the user does not have to look for information. Rather, the user can input what he/she is looking for and have it returned quickly. To the user, the mental model of the interface would be like talking to a secretary. The user asks the virtually secretary to ask or change certain information, then the secretary completes the requested task. I think this mental model is more effective than traditional layouts and interfaces where users go through layers to access desired information. With a command line interface, users can be taken to requested information quickly without going through layers of input.

3.1 Strong and Weak Metrics

I think the strongest metric of my interface is efficiency. It is much quicker on a usability level to be able to specify the exact information and grab that information instantly rather than having a user go through menus to slowly narrow down the choices of information. Also, satisfaction and learnability would be strong metrics for my program. I think users would be happy with the interface because of how quickly they can access information. Also, typing in search terms and complete sentences is easy to pick up on because most users are comfortable speaking and typing sentences on a computer.

Although I feel my interface would do well in all the usability metrics, I feel that it would score not as high in the errors category. When a user has the ability to request anything, the user may request things that are not in Headmaster. While traditional interfaces give users finite choices, the commands the user types in on the search bar are almost infinite. Since the user can type in anything, users may not always receive the best feedback with their errors.

4 Conclusion

A command line interface can be intimidating at first, but can give the user a sense of comfortability. While typing in options rather than clicking them is unconventional, it is more efficient and more pleasing to a user.