

Richard Loomis  
CS 50 Final Project  
Design  
December 7, 2009

## **LifeLongHealth.org – <http://www.LifeLongHealth.org>**

### **Background**

LifeLongHealth.org was designed as a tool to provide a personalized report of tests, interventions, and vaccinations that are recommended by the United States Preventive Services Task Force (USPSTF).<sup>1</sup> The site was loosely modeled after the USPSTF's Electronic Preventive Services Selector (ePSS) tool.<sup>2</sup> The ePSS tool is intended for clinicians and the results are not in a patient friendly format. The goal of LifeLongHealth.org is to provide a simple, straight forward interface for visitors to obtain a report of recommendations, while also providing links to obtain detailed information for each item. By design, the site is easy to use, with all necessary instructions on each screen, and should not require any prior knowledge.

### **Database**

The first step in this project was to acquire the USPSTF rules from its website. Unfortunately, I was unable to find any publicly available database of these recommendations, so I created one.<sup>3</sup> In designing my database, I needed to determine what fields would be necessary so that I could compare user input with the recommendations and provide customized feedback. I eventually decided on using minimum age, maximum age, sex, sexually active, pregnant, and smoker as the basis for my rules. I also included columns test, question, vaccine, grade, and ask your MD. Test indicates whether the intervention is a test. I was not sure how I would use it, but thought it might be helpful information to have. The question column indicates whether there is a question on the web form about that recommendation. Vaccine is for immunization entries. I did not use grade (the level of evidence supporting the recommendation) and ask your doctor, but might in future implementations, so I included these. After creating the initial database, I decided that it would be helpful to have links to more specific details about each recommendation, so I created a column containing the URL to that page on the USPSTF site.<sup>4</sup>

Initially, I put the rules in an Excel spreadsheet and then converted it to a CSV file. Using a modified version of the import script from pset8, I imported this data into a MySQL database. Further modifications were made through the phpMyAdmin interface, so the CSV file does not match the actual database.

<sup>1</sup> <http://www.ahrq.gov/CLINIC/uspstfix.htm>

<sup>2</sup> <http://epss.ahrq.gov/ePSS/search.jsp>

<sup>3</sup> I was told that the NIH is currently funding a project to do this. Unfortunately a little late for the CS50 FP deadline!

<sup>4</sup> Finding the URL's specific to each recommendation was great fun... but not really.

## **Index.php**

The index page contains a web form requesting user input. The first six entries are required, and the rest are optional, but will remove any recommendations, if it doesn't apply to the user. I used a PHP script to perform validation of user input. I first checked to make sure that all required fields were completed. Only six fields are required, since I did not want to deter users by having to fill in other details like weight or respond to information that they are unsure of. Validating radio button groups turned out to be more complicated than I anticipated and I ended up using for loops to perform this function.<sup>5</sup> Age validation ensure that users are between 18 and 130 years old. There are different USPSTF recommendations for pediatrics and my database is only applicable to adults age 18 and over. A male obviously cannot be pregnant and is alerted if those radios are checked. Error checking of optional fields only occurs if they have been completed. If a height is present, a weight must also be present, since this information is used to calculate BMI and both pieces of data are necessary for this. Height and weight must also be non-negative. The year of last tetanus shot must be between this year and 130 year ago. BP ranges are checked to ensure they fall within a reasonable (generous) range.

Upon submitting the form, the javascript input validation function is called. If an error in input exists, the specific error is printed at the top of the form in red. After correction, and resubmission, the form is again checked with validate. If no errors are present, the form is sent to recommendations.php.

## **Recommendations.php**

Recommendations.php is a php/html that generates a custom report for the user. It first connects the USPSTF rules database. All user input passed from index.php is escaped. The first details weight specific information. BMI is calculated based on height and weight using the BMI formula (assuming a height and weight was given). The BMI is then categorized (underweight, normal weight, overweight, obese). If the patient is overweight, the number of pounds needed to achieve a BMI of less than 25 (normal weight) is calculated and reported. Since BMI categories do not apply to pregnant women (not to mention suggesting that a pregnant woman lose weight could be harmful), no category is assigned. Links are provided to webMD relevant content if the patient is overweight or pregnant.

The next section reports specific tests and interventions that are indicated. All of the rules are contained within a while loop. The while loop iterates through each recommendation in the database, comparing it with each rule to check if it applies to the patient. I broke down the rules based on the fields in my database, handling rules for smokers, sexually active, male/female specific, etc. Since there are so many specific rules and exceptions to rules, it was hard to condense the code. Though it may (likely) have required less code to hard code the logic for each rule, I tried to make the rules as general as possible, so that new rules could potentially be added to the database without having to change much code. An example of a rule that is particularly hard to get at is abdominal aortic aneurysm (AAA) screening, which only applies to male smokers, or past smokers, from age 65-75. I had to add the required field past smoker just for to

<sup>5</sup>It required a lot of code. Is there an easier way to accomplish this?

get at this rule. In order to provide accurate feedback, however, this information is absolutely required. A AAA screen in a past smoker is absolutely necessary.

A particular challenge was determining whether the user answered yes to having a specific test or intervention. The ultimate strategy that I chose was to store the answer to the question, along with the test ID (same ID present in database) in an array. I then stored all of these in an array (2D array). When I needed to determine if the user had answered yes, I iterated through the array containing the tests to check if the patient had already had that specific intervention.

The diabetes screen rule check is the patient's blood pressure and responds appropriately. The recommendation is for all patients with a systolic blood pressure greater than 135 or a diastolic blood pressure greater than 80 be screened for diabetes.

Each time a rule is printed, a counter increases, so that the results can be numbered.

The third section recommends vaccinations and uses a similar method to determine if a user has already had that vaccination. It also checks for sex specific vaccinations. If the user enters the date of last tetanus booster, the code will determine the number of years since that shot. If greater than ten, it prints the recommendation and gives the number of years since the last vaccination. Tetanus boosters are needed every ten years.

The last section checks if the user is pregnant and if so, prints general recommendation for all pregnant women. Since most pregnant women are under the care of an obstetrician and all of these items are standard practice, these recommendations are less relevant.

Since I included the URL of each rule in my database, I was able to make each recommendation a hyperlink that points to the USPSTF link to that specific guideline.

### **Comments**

The name of my project was chosen purely on the basis of domain name availability. After many failed attempts at godaddy.com, I finally stumbled upon this one, which was available and seemed relevant to my project.

In order to provide only the relevant recommendations and make feedback as user specific as possible, a great deal of logic was required to handle all of the specific cases. The USPSTF ePSS tool only has five fields making the resulting report very general.