Designing Bartr

Yan Lin Feng

Database Design

The phpMyAdmin database was designed to be as efficient as possible. Messages, usernames, and offer content were kept to a relative minimum to maximize database storage efficiency and search speed (CHARs are faster than VARCHARs if a certain cap is set on the values). Additionally, they were designed to have identifying IDs which allowed data from one table within the whole database to be used to retrieve other information from other tables. The most important of these IDs: user IDs and offer IDs were present in all tables so that they could be easily retrieved and used for display or further querying.

Modal Dialog

Bartr utilizes modal dialog windows to display information. In this way, users are provided smoother transitions through the functionality of the site. The modal window operation relies on CSS styling to define the properties of the box display. To open the window, an href link was used to submit "#openModal" to the URL to display it. Closing the window utilized the same method. Two modal dialog boxes were created, one larger one to accommodate the bulk of a table displaying messages.

Offer Images

The Bartr posting mechanism is relatively simple: the post form simply submits to a PHP file. The complexity lies mainly in the submission of an image. While the phpMyAdmin database can't store actual image files, it can store the names and extension of files in a directory. Therefore, image uploading was implemented by allowing the photos to be downloaded onto the CS50 IDE and then accessed from there. The script checks to see whether an image was actually submitted. If so, it checks to ensure that it is an image file and that the file size is reasonable. Furthermore, the system accounts for identically titled images, and using a while loop, will check to see if the file already exists and if so, give it an incrementing numerical extension (in the form "_n" until is unique. Images are displayed in a modal window simply by using an <img> tag. The location is determined by They are scaled to size by defining CSS properties for the "img" identifier.

Confirming/Deleting Offers

Confirming and deleting an offer both utilized onclick handlers to send information to JavaScript functions, which would call PHP to work through the confirming/deleting processes. Since there was no need to transmit information to another part of the page, I found this the most efficient way to communicate with the server. Deleting an offer simply required the offer's ID, which would allow it to find the offer's photo (if one existed) and delete it from the directory using the unlink() function, delete the offer from the databse, and delete all messages pertaining to that offer (since once the offer is gone, there is no reason to maintain communication about it).

Confirming offers ultimately led to deleting the offer, but first inserted the transaction information into the database. In this way, users could keep track and remember what transactions they are a part of.

Displaying Conversations

I decided that under the messages tab, it was important to show all messages somehow. However, I felt that displaying all messages at once would be very cluttered and difficult to sort through. Therefore, I felt it would be best to display a list of *conversations* that could be individually clicked to display all messages from said conversation in a modal window. However, I also found it important to display the user's own message in the case that no true "conversation" has begun, but they have simply sent a response to someone else's offer. Users should be able to see which offers they responded to, even if they hadn't yet received a reply. Unfortunately, it seemed that there was no effective way of retrieving just these conversations, so I developed a method. First, the script would select all messages sent to the current user and then fill an array with all of the offer IDs. Next, all messages with the current user as the sender would be retrieved. This latter messages were filtered for those in which there has been no previous conversation about the offer by looping through each item of the messages and comparing their offer IDs to those in the created array. If it matched none of those IDs, it must be a new conversation started by the user and was therefore added to the master messages array.

Searching

The search function was admittedly rather primitive, but understandably so. I found no particularly need for advanced searches. In this scenario, users would mostly be searching for simple items (couch, books, food, etc.) which would all be similarly titled. Therefore, a simple LIKE statement comparing to % . query . % retrieves any offers that contained the queried string.

helpers.php

Many PHP functions were written to get user IDs from usernames or offer IDs from offers (and vice versa). Oftentimes, conversions were needed since PHP would receive from displayed HTML values (which would certainly be usernames rather than numerical IDs) or PHP would need to display a username rather than the ID values it drew from querying the database. Other functions such as getresponders() and getmessages() were incredibly useful in that they allowed views to communicate with the server, collect returned information, and display it. This was particularly useful in modal windows that retrieved GET requests from the URL to get and display data to the user.

Sending Information to Modal Windows

GET requests were used to transmit information from one part of a view to another (ie. The main view to the popping-up modal window). There were difficulties getting JavaScript/getJSON functions to cooperate with the views, so this was an excellent and efficient alternative: links could simultaneously open a modal window (with the addition of "#openModal" to the URL) and transmit information through the URL with a single click. This information in the URL would be accessible to the code within the modal window, which could thus use functions such as getmessages() to display information using foreach loops. While this is admittedly not the most secure method, it was an effective and very structured method of data transmission within a view.