PHP was the easiest language for me to work with. It didn’t fundamentally feel different than something like Java, and it is nice to be able to declare a variable and have it be any type of object that I want. Learning how to use MYSQL and databases was interesting, but after research, reading the book, and having the slides I found it to be simpler to work with than I had anticipated.

When designing the page layout, I used PHP for the header and footer of the page: it covers everything except what’s actually inside the main division of a page. This drastically cut down the amount of lines I had to write for each page.

Layout for the database is generally:

* Users have relevant information stored. Primary Key: userId
* Topics have relevant information stored to be displayed. Has a property super\_Topic, which determines if it the object is a child or not. Only child topics can have threads related to it. Primary Key: topicId
* Threads have information and status information stored, as well as the userId of the person who made it. Thread has topicId stored in the object (threadTopic) Primary Key: threadId
* Posts store relevant information and have threadId stored in object(postThread). Primary Key: postId
* When displaying a thread, display\_Thread.php collects all posts where the postThread is equal to the threadId and displays them in ascending postId order. When displaying a topic, display\_Thread determines if the topic is a parent or not. If so, displays children, and if not, it displays all threads with threadTopic = topicId in ascending orders.
* Search generally finds things where words end or begin with the key phrase, then displays a list as relevant to what I wrote above.
* User.php determines if the user has permissions to alter information. If so, enable buttons for this, but otherwise it only displays the user’s info.
* Some pages use GET to display some info and POST to display other, while other pages relegate the POST to other pages. I should have been more consistent with this. As I practiced I became more comfortable with the former.
  + Example: When making a post or thread, information is sent to post/thread.php. However, searching is all done within search.php

Most php pages were formatted to work like this:

* Does the user have relevant permissions?
* Does the user have relevant information?
* Is the sent information actual information that can be used?
* (Continues to use the information sent then)
* If any of these fail, or there are connection/insert/search errors, outputs errors
* Otherwise, declare success and link to page OR display requested page

The reason I went for this layout is that it makes it clear what the issues are when developing the page, any user accessing the page accidentally can see the errors/mistakes, and there are no issues with redirects then (although I found a fix that addressed issues I had with redirects, so no reason I couldn’t use them now. Just need to use ob\_start())

For a list of implemented features, check out the list of features document.

I managed to get most of the features I wanted. I only really feel like I’m missing the mailbox (Which is functionally a private thread) and freezing a thread so it can be read, but not posted in (which would have been doable in a few hours but it’s exam time currently).

JavaScript is pretty minimal, although I tried to not use intrusive formatting for it. The exclusion of this is the user’s page buttons. Everything else checks on submitting forms, or checks on click for the login button at the top. Pretty straight forward.

---Known Limitations:

* Might have a spot or two where I overcoded checks, so all the php could be cleaned up
* I could have included more Javascript in relevant areas
* I’d like to have had the ability to upload an image to be saved to the forum for topics/threads to have images