**Backend Documentation**

The backend for the website (the work the server performs) is operated on a WAMP Server. Installing and running a WAMP Server is as simple as fetching the files and installing it like any other program. This is the case for desktop PCs but may be different for a cloud-based server option. Also note that this is designed to run on a Windows PC.

WAMP Server Download: [**https://sourceforge.net/projects/wampserver/**](https://sourceforge.net/projects/wampserver/)

Install with all the basic installation options – do not make any changes to anything except for default apps the program links to. These are personal preference and are prompted at the end of installation. Run the WAMP Server by running “Wampserver64” – a shortcut for this program should have appeared on the desktop. An icon in the corner of your taskbar should appear green, shaped as a green “W”. If this is yellow or red, something has gone wrong during startup or installation, and the server will not run.

Graphical user interface

Description automatically generated with low confidence

Unfortunately, the package supplied is not recommended for production use. This is a demo and shows how it might be done. While functional on a local machine, proper security measures have not been taken and it has not been field tested across the internet. Further, it has not been integrated with the frontend work that the rest of the team has provided, as the demo has been bound to my local machine due to the nature of how PHP is done, and how our team was structured.

With that being said, the main components of the demo are the following:

* Database Integration
  + Blog functionality
    - Blog entries can be created and uploaded via webpage.
    - Entries can be viewed in a digestible format on its own page.
    - Structure exists to support editing and toggling visibility of individual posts but is currently not implemented.
    - An example landing page has the most recent post to the blog printed to it.
  + Account functionality
    - Support for user-level and admin-level accounts.
    - Functional log-in and log-out pages.
    - Basic encryption is implemented for password storage but does not use a secure hash.
    - Cookie-based log-in system with encrypted tokens for authentication (insecure hash).
* PHP Dynamic Webpages
  + Content is populated using a PHP backend to deliver requested content in a flexible manner.

**File Structure**

ROOT

index.php

login.php

loginResult.php

logout.php

permissionDenied.php

blog

newEntry.php

upload.php

view.php

css

barebones.css

templates

dbconnect.php

loginHeader.php

tokenValidate.php

index.php

Landing page. Contains a welcome, and the most recent blog post.

login.php

Log-in page. Redirects to loginResult.php after user data is submitted.

loginResult.php

Performs log-in operations. Checks database for existing username and password hash match. If there’s a hit, a token is passed to the client to validate their log-in status, and the same token is stored server-side for validation.

logout.php

Erases the login token both on the client and the server.

permissionDenied.php

If a user-level account tries to access an admin-level page (blog/newEntry.php, for example), they are redirected here and told that they do not have access.

blog/newEntry.php

Admin-level accounts can create and submit new blog entries for upload here.

blog/upload.php

Upload operations are processed here, formatting and sending the data provided to the database. If anything goes wrong, the user is informed that the post was unsuccessful.

blog/view.php

Any account may access this page to view all blog entries, sorted by date posted and displayed 5 items per page. Navigation allows users to request another set of 5 entries.   
Admins have access to “Edit” and “Mark as Invisible” buttons, though these do not currently function.

The CSS and “templates” files are purely for code modularity and are not navigable via the webpages.