Total Life Manager Update Document

Sebastian Hardesty, Logan Grisham, Stephen Johnson

**Update Documentation**

The current state of Total Life Manager is only moderately different from the original design. Several of the changes made were only stylistic in nature, such as the color scheme and positioning of elements on the page, which will not be included in this document. This is due to the disjoint nature of these decisions and the success of the project as a whole. The document will instead show the changes to the design and implementation of modules currently working and planned for the system.

While the overall structure and use of TLM has not shifted, there are several aspects that have been adjusted. The first is our toolset, and more specifically the libraries and frameworks we are using. Originally, we had planned to use ReactJS as a framework to help us build our pages and JavaScript more dynamically. As we began though, only one group member has had experience with React and it seemed like it would become a hindrance instead of an asset, so we reverted to basic JavaScript and HTML for the majority of our content. Due to dropping React, which has a few graphing libraries associated with it, we had to determine a new way to graph the data of our users. Initially, we were going to try to use plain HTML and JS canvas, but it seemed too cumbersome, so we instead chose to use ChartJS. None of the team has had experience with ChartJS before, so much of the last few weeks have been spent attempting to learn the ins and outs of ChartJS while simultaneously building a website using it. We also initially planned to have separate graphs, toggled by a dropdown menu. While this menu will be maintained for determining which data set the use will be adding to, all the information will be displayed on a single graph. Chart.JS allows users to toggle which sets of data are showing by clicking on each in the graph’s key, and this will be clearly indicated.

After initially planning the system using the LAMP (Linux, Apache, MySQL, and PHP) stack, we considered using the MERN (Mongo DB, Express, React, and Node) stack, but it seemed like overkill for this project, especially since none of the team was very familiar with these tools. By our first presentation, we had switched back to the LAMP stack. We are still trying to fully figure out PHP, but at this point we have a decent grasp over the LAM part of LAMP, and are making substantial progress on the P.

There were a couple of pages that we felt were erroneous for a project prototype at this level. We have elected to not continue work on the FAQ, Landing, and Meals pages because they are largely just standard HTML pages and do not functionally contribute to our project. If, by the end of the semester, we have some leftover time, they may be added back in, and if this were a commercial product, they likely would be necessary. Similarly, we have decided to not try and implement a slideshow on the login page. We have also chosen not to go forward with the idea of having the user-login be remembered. Dealing with complex cookies safely and effectively seemed outside the scope of our current skill level and, once again, a distraction from the main goals of our system.

Currently, we have completed a moderate amount of components to the system, most of which are behind the scenes. The server went up without a hitch, as that was an area of expertise our group held already and exploited to save time. Shortly following, the server got loaded with our initial github setup and everything worked accordingly. The first big hurdle the team encountered was the login and signup page. We lacked the understanding of PHP needed to set this up, as well as how to properly communicate between the front end and the back end. The initial syntax and language barrier were all that stopped us, and once those were overcome the methodology was simple enough. The system will login current users and spit out an error message for invalid entries. This works on the live server, so that crucial step is out of the way. We now have to implement the actual modules the user’s will see when they use the site. The login and signup forms are separated and do each respective task when submitted through their buttons.

The next hurdle was navigation between menus on the website. Our team initially planned for a bar across the top of the page to link between all the pages, but later decided to change to a menu system. The specific style we adopted was the “Hamburger” menu that displays the information in a vertical menu as opposed to the bar across the top. This works functionally the same way, but streamlines the user experience, while not distracting the user from the on-screen information. We also have much of the layout and visual structure set up on many pages, such as the login and dashboard, as well as the color scheme for the website.

The partially complete sections, which are the sections we are currently focusing on, are the graphs and dashboard screens. The graphs are still under construction due to the learning curve of Chart.JS and its capabilities. The page has backend integration set up but cannot be properly tested until the charts are fully implemented. Similarly, things like users setting personal weight goals are dependent on this page. One of our group member’s main focus is to get this page up and running, and it is vital that the graphs and user information input are in working order so that the other components who use those graphs and that data can further advance.

The next item is the dashboard and all its associated widgets. One of the widgets is the aforementioned graphs. Once the graphs page is working properly, linking this widget and that data will be one of the primary goals. The circular graphs also rely on this data. The calendar and weather widgets have not been started yet, leaving the dashboard only partially complete. Similarly, the calendar page has not been started in earnest. We likely will use a calendar plugin, if we are able, in order to simplify this process. The dashboard is fully connected to the menu and the rest of the pages through the hamburger menu, which is now present on all pages in which a user is logged in. Finishing these widgets and getting them appropriately placed in the dashboard is one of our biggest goals, as the dashboard is the main page the users will see.

The list of what needs to be implemented still is the longest list. Finishing the graphs page and dashboard, and starting on the calendar page are the biggest chunks left to do. The service does not need as much backend work as it needs integration between the layers of the system and increased frontend functionality. The graphs page and dashboard being our current workload means that not much needs to be designed but there is a lot of work needed on functionality. The expected outlook would have the calendar and weather components coming last, as they are the least important to the website's completion of the established requirements. While important to the project, things such as login/password recovery, email verification, and editing account information have fallen to the wayside in favor of increased focus on getting the main parts working.

The unresolved issues in the system largely currently lie with unfamiliar technologies. The chart.js setback, as well as figuring out how to get actual frontend and backend communication, required a huge amount of research that we are still working on implementing. This will clearly be a success or fail component of the system, because it will not actually do anything without pulling the users data and displaying it appropriately. Some previous aspects of the design have been scaled back to better accommodate our time and skill restraints, but the basic functionality of the website is still moving along as intended.

**Updated Design Diagram:**

