**Group 1 – Jeff Veit, Steven Meier, Brian Schultz, Ximan Liu**

**CSC 436**

**10/25/2020**

**Milestone 3**

**Changes:**

|  |  |  |
| --- | --- | --- |
| **Change** | **Old Solution** | **New Solution** |
| Login Page | Had three potential login pages that would then route to the application. | Now just having login functionality directly on the website. When we incorporate the data source into the project a solution will be created where personal information pages show default or no information until logged into an account. |
| Header formatting | Nav bar HTML written out on every reachable page (excluding the login pages). | Now using the Header component to manage all our navbar/logo logic and formatting to reach out every page. Team members can build pages without hard-coded operation. |
| Footer formatting | Footer information written out on every reachable page (excluding the login pages). | Now using the Footer component to manage all our footer logic and formatting. |
| Index.html | Essentially the home page of the web app. After passing through a login screen you reach this page where you can then use the navbar to link to different pages | Sets header information for browser tab display. Routes directly to app-root (app.component.html). |

|  |  |  |
| --- | --- | --- |
| App routing | Was managed through the copied and pasted header navbar on every reachable page. | Gives us the ability to represent any component we wish to create or allow navigation to. The header component creates nav links that are then tied to the routes defined in the application routing. Represented in the app component section as <router-outlet>. |
| App component | Did not exist. | Created App component to manage the app-root that our index.html displays. Formats the skeleton of the site, putting the header above the router-outlet functionality, followed by the footer formatting. |
| Main component | Was managed through main.html. | Now is its own module with TypeScript, CSS, and HTML file. Still displays the same information as the old main page, but now, with Type-Script, has the ability to integrate forms for data submission once we integrate with our data/API service. |

|  |  |  |
| --- | --- | --- |
| Library component | Was managed through personalLibrary.html. | Created its own module with TypeScript, CSS, and HTML file. Still displays the same information as the old library page but with TypeScript will now be able to pull personal account information and use forms to create new library entries. |
| Goals component | Was managed through goals.html. | Now is its own module with TypeScript, CSS, and HTML file. Still displays the same information as the old goals page but with typescript will now be able to pull goals for your account and use forms to create new account goals. |
| Home component | Was managed through index.html before. | Created its own module completely separated from the index.html file. With the typescript this page will be responsible for submitting login information to the service. As well as displays information bout the creators of the application. |
| Account component | Was managed through the account.html page before. | Displays the account information page just as before. Not much will change with this component, but typescript will be used to display stored account information. |

|  |  |  |
| --- | --- | --- |
| Angular Integration | None before. | By creating his as an angular framework project, we can now use commands like “ng serve” to boot up our websites locally. At the moment we import our dependencies via scripts on app.component.html but we could also use angular’s packaging to import them instead. Using angular also gives us a better folder structures so we now have more separation between functional parts and application assets. Solved some hard-coded problems for creators. |