Bookkeeping Phase 1 Report

Jennifer Brady

Matthew Dobson

Andrew Eissen

Kevin Ramirez

Christian Rondon

Steven Wu

University of Maryland University College

CMSC 495

Dr. Robinson

November 19, 2018

Contents

Achieved Functionality Milestones

Overview

As of the fifth week of the class, the bookkeeping project team (Team 4) has made significant progress towards the overall completion of the application, with nearly a hundred commits merged to the project GitHub repository and over two thousand lines of code written by both the front-end and back-end teams in their respective languages. Though work remains to be done related to the development of the aesthetic design of the interface and the linking of the back-end and front-end sides of the application to each other, the application framework presently has a clearly defined form and works in a limited testing capacity in the supported browsers.

Beginning with the formation of the group, the front-end and back-end teams have spent the last few weeks in the planning and preparation phases of the development process, laying the groundwork for the implementation of the application functionality in its various forms. This has involved the respective writing of everything from PHP classes defining the layout of the database tables and retrieval of data thereof to the writing of JavaScript functions managing the dynamic generation of HTML content to the writing of CSS style rules handling the interface appearance.

Implemented Back-end Functionality

As far as presently-implemented back-end functionality is concerned, the back-end team members have thus far defined a total of nine (9) PHP, SQL, and JSON files related to the definition of the database design and the handling of the user login and account creation processes. Relatedly, these files also deal with defining the various levels of account access provided to user accounts, with certain types of account

granted special administrative privileges and user rights as needed to manage the database's collection of user data.

Implemented Front-end Functionality

Regarding the presently-implemented front-end functionality, the front-end team's JavaScript engineer has since defined the builder and assembly functions responsible for the generation of the interface HTML and constructed a set of HTML document object model designs (called "scenes") defining where HTML elements are to exist with respect to one another on the page depending on the scene being viewed. Additionally, he has implemented an example REST-compliant `GET` request handler for use with the ongoing development of the PHP API endpoints, a set of dynamic HTML table row addition and removal handlers, a rough set of event handlers related to the document and entry addition actions, and assorted utility functions and enums.

Concurrently, the front-end team's CSS specialists have built off of the rough CSS framework provided by the JavaScript engineer and evolved the design even further, making use of a unified design aesthetic to give the various page scenes a more consistent shared appearance and associated polished design. Additionally, they have begun the development of the device width breakpoint framework used to handle the display of page elements differently depending on the device used to view them.

Remaining Functionality Milestones

Overview

Though significant progress has been made towards the implementation of the required application functionality, there still remain a few critical aspects and components that require development and testing before the program can be

considered complete and ready for deployment. Though these specific aspects discussed below are known to the team, the group is also prepared to handle any unforeseen issues arising during the development of the application. Additionally, in the event that time remains for the addition of optional functionality, the teams are prepared to handle the configuration and addition of new components that may aid the clients in more easily manipulating and interacting with their data.

Remaining Back-end functionality

As far as remaining back-end functionality milestones are concerned, the back-end team is presently occupied with the development of a series of individual PHP class API endpoints that will be used to return JSON data related to the various program functionality components upon the receipt of `GET`, `POST`, `DELETE`, or `PUT` requests from the front-end. This REST-compliant system will allow for the dynamic return and handling of user data in real time as the client interacts with the various event-driven elements present in the interface. Additionally, the back-end team will work to develop the server configuration in the near future, focusing on the development of an HTTPS-powered system for the secure transmission of data. Lastly, the back-end team is finalizing the essential bookkeeping functions.

Remaining Front-end Functionality

On the front-end side of house, the front-end team is presently concerned with the development of a few major elements of note. First, as discussed in the above section, the development of a RESTful system implemented via a series of `XMLHttpRequest`-driven `GET`, `POST`, `DELETE`, or `PUT` request handlers is a primary concern related to the integration of the front-end and back-end. Though an example

`GET` handler has been implemented and successfully used to grab JSON data and display it in a sample ledger HTML table, work continues on the construction of the other handlers in preparation for the planned back-end and front-end integration process. Secondly, the development of a configurable popup modal window is the primary concern of the JavaScript engineer, as this popup will eventually be used to display mini-scenes related to the addition of new ledger entries or new documents on the press of the appropriate interface element option.

As far as aesthetics are concerned, the front-end team's CSS engineers are presently working towards the development of a polished interface design that is fully capable of displaying all interactive elements on all devices. In accordance with the responsive design and content portability paradigms of web development, this will be implemented via the handling of specific device width breakpoints, a system that will allow the user to use the application both on external monitors of large conventional widths and on small mobile devices as needed.