TAL Database Application

Zach Dilliha

Western Kentucky University

[**Design**](#_8qz8f8sp3xcr) **3**

[**ER Diagram**](#_43efg2vww6lj) **5**

[**Evaluation**](#_pu67bd68250w) **5**

[**Screen Captures**](#_1kns2m1qdyxo) **8**

# 

# 

# Design

I designed this application as a simple interface for the users to interact with. The index (picture 1) includes the login/signup, a Home button that takes the user back to this screen, and a snippet of text saying “You are logged out!” The signup button takes you to a signup screen (picture 2) where you can create a new account by inputting a username, email, and typing in a password twice for verification. I included an email option because in the future it would be possible to implement a password recovery method which would need to send an email to the user to be able to change/recover their password. The Signup process has a few requirements, such as requiring a valid email, requiring a unique username, and that the two passwords match. If a user is trying to sign up and encounters one of these errors, the kind of error will appear in the url. For example, here is the error when the user inputs a username that is already in use and has their email be mail@mail.com:

localhost/talDatabaseApp/php/signup.php?error=usertaken&mail=mail@mail.com

The url will give the user back the email they used after the “&mail=” token. Similar errors will return values that the user inputted, either the username, email, or both, but never the password for security reasons. Eventually a method of retrieving that information and placing it back inside the input fields could be implemented to further save the user time.

When logging in, the user can use either their username or their email along with their password. When logged in, the user will be taken to the home screen (picture 3) which includes from top to bottom:

* Text saying who is currently logged in
* A Home button that takes the user back to this screen
* A Logout button that takes users back to the index (picture 1)

These elements make up the header, and are included in each subsequent page, other than the index. Unique to the Home page, is the menu of operations the user can perform, and selecting any of the operations takes users to their respective pages.

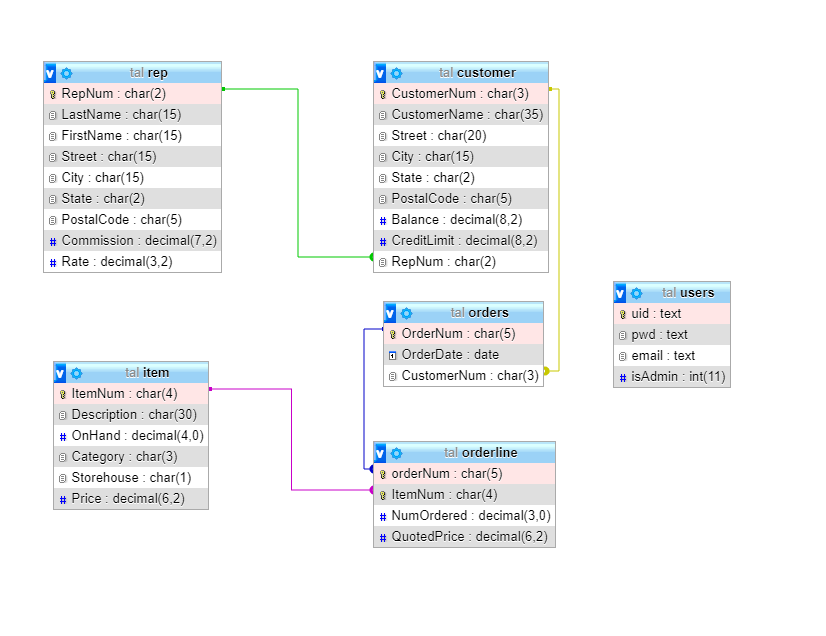
The Add Rep option takes users to a page (picture 4) where they can add a new representative to the database by filling in the appropriate fields. After filling out the fields the user presses the Submit button on the bottom of the form, and then the form will update, displaying text above the form stating “\_\_\_\_ successfully added.” The form remains on the page so the user can add another representative.

The Update option takes users to a page (picture 5) where they can input a customer's name and a new Credit Limit. After filling out these fields, the user presses the Submit button, which will update the page, displaying “\_\_\_\_\_\_’s credit limit successfully updated!” (picture 6). The form remains on the page so the user can reenter a customer name and credit limit.

The Representative Report button takes users to a page (picture 7) that displays the amount of customers they have, as well as the average balance of their customers in the format: “FirstName LastName has # customers, with an Average Balance of $X.”

The Order Report button takes users to a page (picture 8) that has the user input the name of a customer. When the user presses the submit button, The form will update (picture 9) and display the information in the form: “Order Report for: \_\_\_\_\_\_” then “The Total Quoted Price of all orders placed by \_\_\_\_\_ is $X.” The form will remain on the page so the user can reenter a customer name.

# ER Diagram



# 

# Evaluation

When beginning this project, I researched multiple different methods to build the application, and also encountered many obstacles in the process. Methods I attempted included building a graphical app in Java, making my own website and hosting the database on it, utilizing an online database application builder, and finally writing my own web application with php and HTML. I ended up going with my own web application, and this required me to learn php and a little bit of html, so I spent some time reading and watching basic php tutorials and completing tutorial applications. When I felt that I had enough experience to begin, I started on the TAL application. Developing the application was relatively easy after doing smaller tutorial web apps, and I was able to use techniques from those tutorials when developing the TAL application. I started by making the login system for the app, which required 4 different php files and had me creating forms and buttons in HTML that the user used to input information. This was difficult because I had to coordinate these files and also make it work with the forms I needed. I eventually got it to work after much trial and error. This process also led me to creating “dbh.inc.php” which is short for Database Helper in the ‘includes’ folder. This file is used to connect to the database, and is included in any files that need to be able to access the database.

Next was the header file, which I initially wanted to have the name of the app, a home button, and a login/logout button depending on whether or not the user was logged in. As the project progressed I was exploring the uses of sessions and session variables, and I think of them as essentially global variables in a language like C++, where they work throughout a file or set of files, as long as those files have code that allows them to access them. Sessions turned out to be easier than global variables in C++ because I only need to call one function, session\_start(), in order to access any variables I initialized somewhere else. Using session variables, I created conditions within my header that displayed different information based on whether the user was logged in or not. If a user was logged in, the session variable $\_SESSION[‘userUid’] would be initialized to the username of the user. Wherever the header is included it will check if a user is logged in, and if they are, it will display: “Signed in as $\_SESSION[‘userUid’]” to give a visual indication that someone was logged in, as well as showing the Logout out button and having the Home button send people to the main home page, home.php. Conversely, if nobody is logged in, the header will not show any special text, there will be a Login button, and the Home button will take users back to the login/signup screen, index.php, and displays the Login form and the Signup button. The Header file also starts the HTML Document, so that I do not need to reinitialize HTML when using other files, as they are essentially continuing the document. There is a Footer file, but since I didn't need a fancy footer, all that it does is end the HTML document started in the header, and that is its use for every page it is included in.

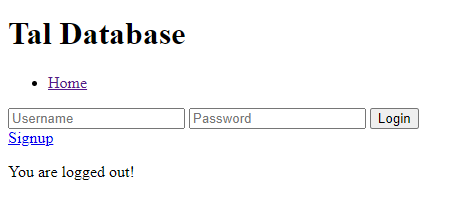
The Home page is simple, as it only needs to display the operations that the user can perform. It includes the Header and Footer to display the Home button, Logout button, and text stating who is logged in. This will be the same for every file that includes the header and footer. Past the Header, it has a list of all the operations a user can do: Add a Representative, Update a customer's credit limit, View the Representative Report, and View a customer's Order Report. These are links that take the user to each operations respective page. This page was easy to make, as it is just a list of the operations with links to their pages, so there was little code needed. The only php I used was to test whether or not someone was logged in. If someone is logged in, they can view the Home page and interact with it, however, if there is not someone logged in, going to home.php redirects the user back to index.php, disallowing anyone without proper credentials the ability to get to this page. This snippet of code was useful, so I formatted it to fit onto all of the operation files, so there is no way a non-authorized user could interact with the various forms and reports.

Next were the individual files for each operation. I worked on these in sequence, and when I would get one to work, I would use and replicate elements from one when creating the next, which saved me time and made the code simpler for me to understand. By doing this, I had built up my own sort of database from which to draw code from, because I had various “modules” of code that I knew worked and could do various things, so I would implement them in certain combinations to fit the needs of the file. The easiest for me to complete was the Representative report file, as it required no user input and just had to display text. To contrast, the hardest was the Add a Representative file, because it was the one I started with, and required the most HTML and php out of all of them. Upon completion it helped me to complete the others, as they could all be completed with slight variations to the php and HTML.

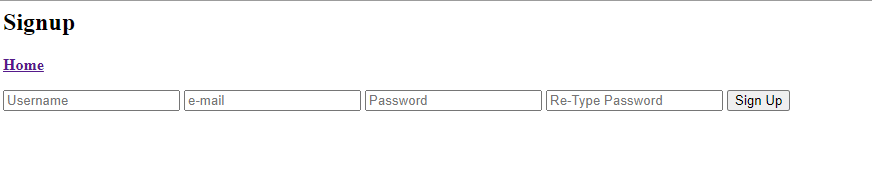
One thing I was not able to complete was a installation file that would be able to install the TAL database for you, so there would be little room for user error, and it was near completion, but I was not satisfied, and therefore its files are in the DEFUNCT folder, to be viewed, but not used, and potentially finished later. The other files will work, as long as the user is able to import the TAL database themselves

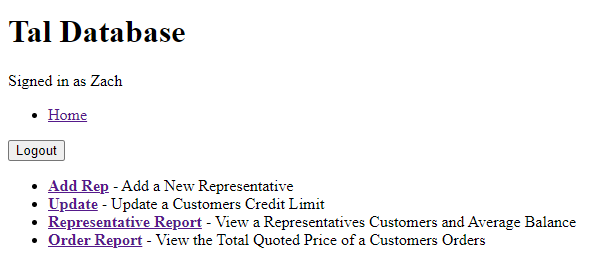
Overall, I feel that my implementation of this database application was done well, I was able to make it work with a sort of rushed knowledge of php and HTML, and I did the best I could with what I knew and could learn. The only things that I would change would be to add style and design to the pages to make it look more appealing and not a mostly blank page. To do this, I would have to learn CSS and, while I could do that, I did not have the time to do it for this assignment.

# Screen Captures

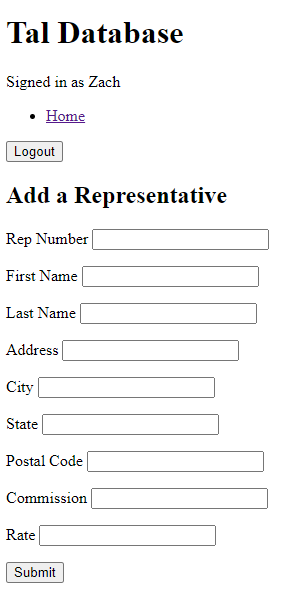


1

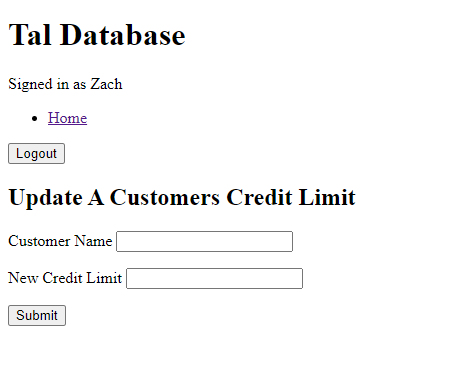
2

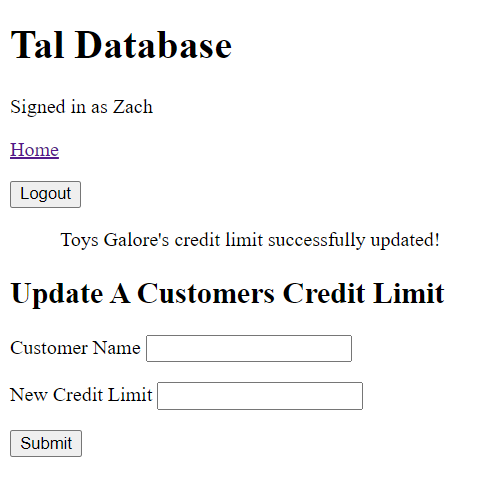
3

4

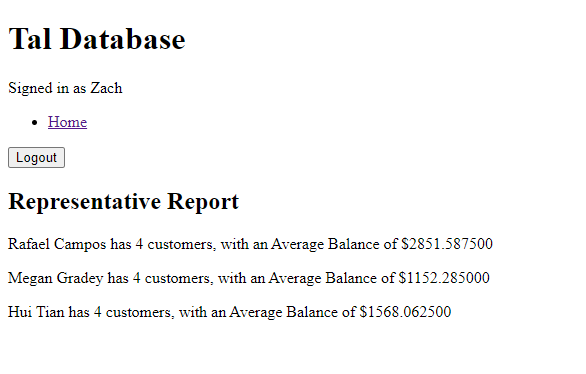


5

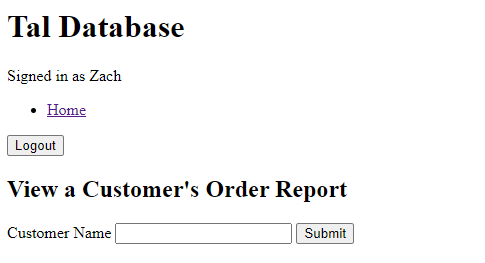


6

7



8



9

