Conversations.php

- This page deals with two different user viewpoints. The mentor viewpoint and the mentee viewpoint. For both users, their conversations are loaded on page load. From here, conversation information is constructed into a 2d array consisting of the last modified date of the conversation, the conversation id, the other users unique user id, the other users name, and the status of the conversation (whether it is locked or not).
- The conversations are sorted first by Active, Pending, unviewed, Denied – And then they are sorted by most recent within those four groups.
- On page load, if the user has a conversation it will load the messages within that conversation on the right of the screen.
- There are five important fields to consider with this page.
    - The conversation field is where you will see snippets of the last sent message, the mentor name, etc. Additionally each individual conversation is put inside of a button that when clicked loads the conversation associated with that button (this is gotten from the id of the button being set to 'selected_id' + the users id. This field is set to update every 5 seconds for conversation integrity (during the same time the messages area populates with message integrity). This is used so that if a user is on the page, and they get a message, they will be shown the conversation and message within 5 seconds.
    - The head area of the messages screen contains the other users name, possible accept, pending, deny, and end partnership buttons, status messages for the user, and a collapsing class that has a menu item for reporting the user. This area is repopulated when a conversation is clicked or when the send message button is clicked so that it can respond to the users actions.
    - The messages area is re-populating every 5 seconds, same as conversations, to ensure that the user can see new messages within 5 seconds.
    - The typed message area is where a user types and sends their message to another user. This area is re-populated whenever the user sends a message, or when accept, pending, deny, or end mentorship is clicked so as to show the user changes with respect to their actions.
- All buttons on the page are utilizing javascript for their onclick events and have associated request pages that will be gone into further depth down below.
- The typed_message area dynamically grows with the 'grow()' javascript function so that a user can continue typing a message.

Known Bugs

- The collapsible menu for reporting the user is fickle and occasionally gets stuck for a couple seconds.
- No other known bugs.

Future Works

- I would like to add a notification to the user that their message is too long before they send it, since we only allow a message of up to 4000 characters, this way a user won't lose their entire message when they attempt to send a long message.

- I would like to adjust the ‘grow()’ javascript to allow for shrinking when a user deletes more of their message, but currently it does not have the logic for this.

Associated Request Pages;

- endPartnership.php
    - Programmer
        - This is the request page that handles setting the conversation as denied and unpairing the mentor from the mentee and setting the innerHtml of the head area and typed message area to display the proper message for the user.
    - Known Bugs
        - No known bugs.
- mentorSendPopUp.php
    - Programmer
        - This request page takes four $_REQUEST variables; ‘q’, ‘c’, ‘o’, and ‘m’. ‘q’ is the message that is being sent, ‘c’ is the conversation id that the message is being sent to, ‘o’ is the conversation selected option since on this page it could be accept, pending, or deny, and finally ‘m’ is the current number of mentees that the mentor has. On this page it checks if the option is accepting the mentee, if so it checks to ensure the mentor has not reached their maximum number of mentees. After this, the conversation is accepted and the message is sent.
        - For pending, the conversation is set to pending and the message is sent.
        - For deny, the conversation is set to denied and the message is sent.
        - For all messages, they change the innerHtml of the head area, typed message area, and messages area of the Conversations page to allow for a smooth transition and no ‘lag time’ on the users side.
    - Known Bugs
        - No known bugs
- populateConversations.php
    - Programmer
        - This page queries the database to populate new conversations associated with the user. Then it sends back the information as innerHtml for the conversations section of the Conversations page.
        - This request page constructs the conversation button on the left side of the Conversations.php page. It will also check if the user is a mentee, and create a button for them to go to the search page if they do not have any current Mentor conversations.
    - Known Bugs
        - No known bugs
- populateHeadArea.php
    - Programmer
        - This page queries the database concerning the current conversation to adjust the header area to reflect the status of the conversation as described above in the main conversation documentation.

        - This page performs the logic necessary to display the buttons depending on the status of the selected conversation.
        - If the status is ‘unreviewed’ or pending mentor approval, then it will populate the Accept/Pending/Deny buttons if the user is a mentor, or it will display a brief message if the user is a mentee informing them of the status of the conversation.
        - The report user dropdown menu is always available for any conversation status.
    - Known Bugs
        - No known bugs
- populateMessages.php
    - Programmer
        - This page queries the database concerning the current conversation for messages and repopulates the innerHtml of the message area to show any new messages associated with the current conversation.
        - This page currently grabs the entire conversation that is currently selected, and recreates every message as a new message. It isn’t the most efficient, and I would prefer to change this request page to just recreate the last 10 or 20 messages that are in the selected conversation, and then do something with javascript to continue to repopulate older messages when the user scrolls up in the chat. However, I wasn’t sure how to do this more efficiently, since if I just repopulated the last 10 messages received or sent then it is possible that it would miss a message since the request page to repopulate messages is only called every five seconds. Despite it being unlikely, it is possible that more than 10 messages may be sent in that time frame, and so it wouldn’t populate all new messages if it was done this way.
    - Known Bugs
        - No known bugs
- populateTypedMessage.php
    - Programmer
        - This page queries the database about the current conversation to get the status of the conversation and reflect that in the innerHtml of the text area in the Conversations page.
        - This page deals with the logic for whether the send button has the javascript function attached to it to send a message, this way if a conversation is not accepted the users are unable to send messages. Additionally, the textarea used for the message is locked so that a user can’t type into it.
    - Known Bugs
        - No known bugs.
    - Future Works
        - I would like to mention that by having disabled textareas, it is still possible for a user to inspect the page, and change the html to make the textarea visible. There is a lot more testing I would want to do to ensure this was not abused.

- popupMaker.php
    - Programmer
        - This page is sent two $_REQUEST variables, ‘q’ and ‘o’. ‘q’ is the text prompt associated with the option that the user selected, and ‘o’ is the option that the user specified (being accept, pending, or deny). Then the PopUp div in the head area is sent the innerHtml for the sendPopUpMessage box that the user will type a message to alter the status of the selected conversation.
    - Known Bugs
        - No known bugs.
- sendMessage.php
    - Programmer
        - This page is sent two $_REQUEST variables, ‘msg’ and ‘id’. ‘msg’ is the message that the user typed, and ‘id’ is the conversation id that the user is currently viewing. After doing an escape on the string for sql injection, the message is then sent to the conversation and appends the message to the current innerHtml of the message area to create a seamless experience for the user.
    - Known Bugs
        - No known bugs
- setVariable.php
    - Programmer
        - This page is used to set the view user id in the session before redirecting to the ViewProfile page. This is called when a mentee clicks on the mentors name in the head area of the Conversations page to go to the selected mentors profile.
    - Known Bugs
        - No known bugs.

ViewProfile.php

- This page checks to make sure that the variable ‘view_user_id’ is set in the session and then proceeds to load the Mentors information from that user id. After using the procedures to pull down all information about the mentor from the database, a check is performed on whether a conversation exists between the mentor and the current user that is viewing their profile. If there is a current conversation, and the conversation status is not denied, a message will be displayed regarding the status of the conversation and will inform the user to go to their conversations page to proceed with communicating with the mentor.
- The three javascript functions on the page are makePopUp(), sendPopUpMessage(), and cancelMessage(). Together, these contribute to creating the start conversation message popup for the user, and the sendPopUpMessage and cancelMessage functions deal with the send/cancel buttons attributed to that popup message area.

Known Bugs

- No known bugs.

Further Works

- I plan on changing the Start Conversation button to become disabled after the mentee sends a message to the viewed mentor. This will not break the page with how it is, but it would be more intuitive for users.

Associated Request Pages;

- popupMaker.php
  - Programmer
    - This request page is given two $_REQUEST variables; ‘q’ and ‘o’. ‘q’ is the placeholder message that is shown to the user attempting to send a message, and ‘o’ is the option that this request page is being used for. For the ViewProfile page, this will echo the inner html for the message area that a user can send a message from.
  - Known Bugs
    - No known bugs.
- sendPopUpMessage.php
  - Programmer
    - This request page is given one $_REQUEST variable; ‘q’ which is the message that the user is attempting to send. This escapes the message string for sql injection, then proceeds to find out if the conversation exists, verify the status, and if the status is denied (3) – it will add the message to the current existing conversation. Otherwise it will create the conversation, add both users to the conversation users list and then create the message in that conversation.
  - Known Bugs
    - No known bugs.

AdminConsole.php

- The left side of the screen are the admin commands, Review Pending Mentors, Review Reported Users, Review Banned Users, and Search for Mentor are all collapsible buttons that display the pending mentors, reported users, banned, users, and the search bar respectively. The pending mentors, reported users, and banned users are queried when the page loads and the information regarding each mentor is put into an array that is then used in corresponding innerHtml variables to display each mentor as a button for the admin. Upon clicking the mentor for pending mentors and banned users, the javascript function showSelectedMentorProfile is called to populate the right side of the screen with the mentor profile. This sends information to a request page to get the information for the mentor.
- When a user is selected under the reported users command, the right side of the screen is populated by the request page associated with the showReportedUserCircumstances javascript function. This has two buttons in the top for banning the user or dismissing the report, both of

with have javascript functions associated with them and request pages of banSelectedUser and dismissSelectedUser respectively.

- The search for mentor command populates a search bar that calls the searchMentors function which then populates mentors below the search bar that fit the search constraint.
- Change Automated Email Credentials in the final tool on the page. Clicking it loads the showAdminEmailCredentialnput.php file on the right.

Known Bugs

- The collapsible commands occasionally get stuck for a few seconds.

Future Works

- I would like to add a review all reports command so that the administrator can view all reports that have been submitted to essentially review previously dismissed or banned reports.

Associated Request Pages;

- adminSearchMentors.php
  - Programmer
    - This page is the request page that searches for mentors with attributes that are akin to the search constraint. This page is sent one $_REQUEST variable, 'q' which is the string that the admin entered. This searches the database for any mentors that have a name, expertise, skill, education, email, etc that is the same or similar to the search constraint. After this query, the innerHtml for the searchedMentors area is replaced with the results of the search constraint, each mentor is displayed as a button that then allows the admin to click the mentor to view their profile on the right side of the screen.
  - Known Bugs
    - No known bugs.
- approveMentor.php
  - Programmer
    - This request page is associated with the pendingMentors command and a button is populated on the right side of the screen that then sends this request page one $_REQUEST variable 'id' that is the mentor id that is being approved. This enables the account and sends an email to the user notifying them of the change. Additionally, the mentors name is removed from the pending mentors list and the right side of the screen is reset to a blank page.
  - Known Bugs
    - No known bugs.
- banSelectedUser.php
  - Programmer
    - This request page is associated with the reviewReportedUser command that populates the conversation of the reporter and the reportee as well as a ban button that sends this request page two $_REQUEST variables 'id', and 'eId'.

    - This then sets the users account to a banned status and marks the report as reviewed so that the report doesn't continue to populate the unreviewed reports of the admin. Upon clicking this button, the mentor is removed from the list of reported users, and the right side of the screen is reset to blank.
  - Known Bugs
    - No known bugs.
- denyMentor.php
  - Programmer
    - This request page is associated with the pendingMentors command and a button is populated on the right side of the screen saying deny mentor that then sends this request page one $_REQUEST variable 'id' that is the mentor id that is being denied. Then the account is deleted and the mentor is notified of this change. Additionally, the mentors name is removed from the pending mentors list and the right side of the screen is reset to a blank page.
  - Known Bugs
    - No known bugs.
- dismissSelectedUser.php
  - Programmer
    - This request page is associated with the reviewReportedUser command that populates the conversation of the reporter and the reportee as well as a ban button that sends this request page two $_REQUEST variables 'id', and 'eId'. This then marks the report as reviewed so that the report doesn't continue to populate the unreviewed reports of the admin. Upon clicking this button, the mentor is removed from the list of reported users, and the right side of the screen is reset to blank.
  - Known Bugs
    - No known bugs
- showReportedUserCircumstances.php
  - Programmer
    - This request page is called when a mentor is clicked under the review reported users command, it queries the database for the report information including the reported user, the reporting user, details concerning the report. Then is queries the database for the conversation between the two users. This is then set up as innerHtml for the right side of the admin page so the admin can view the details of the report and the conversation between the two users to verify that the terms and conditions were violated. Additionally, a ban user and dismiss report button are populated in the top right of the screen for the action the admin wishes to take concerning the report. These buttons are associated with banSelectedUser and dismissSelectedUser respectively.
  - Known Bugs
    - Some messages are not displayed in their proper container properly. This is due to me adding the name of each user at the top of each message so that an outside user (being the admin) can follow along with the conversation. It only

seems to occur with one word messages where part of the word will populate outside of the message container.

- showSelectedMentorProfile.php
    - Programmer
        - This request page is sent two $_REQUEST variables, 'id' and 'o' where 'id' is the mentor id and 'o' is the option being sent. This page is used for both viewing searched mentor profiles, pending mentor profiles, and banned mentor profiles, thus the option is 0, 1, and 2 respectively to display either nothing, accept/deny for pending mentors, or unban for banned mentor profiles. On this page the users profile information is queried and then sent to the innerHtml for the right side of the screen to display the profile to the admin.
    - Known Bugs
        - No known bugs.
- unbanUser.php
    - Programmer
        - This request page is sent one $_REQUEST variable of 'id' where that is the banned users id, and on this page the account is set back to enabled and the user is then able to log in again. Upon completion, this user is removed from the list of banned users and the right side of the admin console is reset to blank.
    - Known Bugs
        - No known bugs.
- showAdminEmailCredentialInput.php
    - Programmer
        - This request page is called from the Change Automated Email Credentials tool. It loads 2 forms, the top most one containing 3 text entries for entering an Email address, Password, and confirm password. The form requires the passwords match on submission, checked at the top of the page. If the passwords match, it will call the database procedure SetAdminEmailCredentials with the new email and password. The second form has a simple button that sends a test email from the includes/emailFunctions.php file both from and to the admin email credentials set above. This is to test if the credentials have a good connection since if the credentials are bad, no automated emails will get sent out. This tool was required in this form since regular email reset links would not work if the credentials associated to the old admin email expired.
    - Known Bugs
        - No known bugs. Unfortunately stores password in clear text on database. Storing hashes was not viable due to requiring password to send emails and encryption would do little good on the database if it were to be compromised anyways. Credentials and methods that call credentials should all be on server side and un-browsable directories for security.

Search.php

- This page checks at the top whether the user is a mentee, and if they aren't redirects them to the landing page. This page then loads the first twenty mentors as a default list that is displayed on page load and when no search results are found for the given search. Each mentor has their information populated into a button, similar to the conversations page that displays brief information about the mentor including their name. The onclick event sends the user to the ViewProfile page then displays the full information about the user and allows for them to start a conversation with the selected mentor.
- This page has two javascript functions associated with it; showSuggestion(str) and searchMentors(str). Each one has their own request page.
- The showSuggestion is an onkeyup event so that it will show suggestions to the user in real time.

Known Bugs

- No known bugs

Login.php

- This page is the main mentor login page. It also has the most robust checks to see what type of account is logged in, what they have access to, and their enabled status, so many pages redirect to login.php where sessions are not set correctly to have them reapplied and checked centrally.
- Reset password link calls the resetPassword.php page. This page asks for the email address they wish to reset a password for. It checks to make sure that email is a valid mentor email, if it is, the email will be sent a reset link that is coded for a GET request storing the email address and a unique hash generated by hashing a random double between 1 and 1337. This does not need to be a secure hash, it is just used for a unique identifier so that only the person with access to the email can reset the password. It will call the database functions to create a password reset validation entry with the new hash and values.
- Clicking the password reset link takes you to the resetPasswordInput.php page which is a form with 2 password entries. It verifies the passwords match, then hashes and stores the passwords before informing the user they can now log in. It also has error messages for in the link is expired or has been used before or if it has an invalid hash. (Note: This works for the admin account credentials as well, but the admin automated email account is stored separate from the login of the admin accounts)

Known Bugs

- Not a bug, but an idea. The unique hash generated for any of the resets (email rest, password reset) is not very random and collisions are relatively likely. I figured this was not a security focused trait as it was more of a convenience to verify the account you wanted to reset that couldn't be accidently forged or with mild effort. All the reset functions were made very quickly and probably need an overhaul, the password reset one is the most stable and checked, but still rough.

Associated Request Pages

- searchSuggestions.php
    - Programmer

        - This request page is sent one $_REQUEST variable ‘q’ which is the string that is in the search box currently. This page looks at the suggestion list that is stored in the session array and verifies whether the string matches or is a substring in that list.
        - Then the page echos the suggestions into a suggestions div tag on the search page to show the user suggestions for their search. This suggestion list is limited to ten suggestions.
    - Known Bugs
        - No known bugs.
- searchMentors.php
    - Programmer
        - This request page takes one $_REQUEST variable ‘q’ which is the string that the user is using as a search constraint. The page then queries the database for any mentors that have an expertise, skill, education, name, etc that is associated with the search constraint and populates arrays with the mentor information. InnerHtml is then constructed in the request page and then replaced the mentor list that is below the search bar with the new found mentors that match the search constraint. Should no results be found, the default mentor list will be displayed instead.
    - Known Bugs
        - No known bugs.
    - Future Works
        - I would like to change this so there is a message to the user notifying them that no search results were found.

AccountCreationSecondEdition.php / ProfileSecondEdition.php

- This is my version of the Profile page that was created by my teammate. I did not connect this page with any of the current pages for the project, but at the end of the project I was attempting to add html injection and sql injection protection to the AccountCreation.php page and the Profile.php page, however it came down to how the page was originally constructed. This was largely due to putting user input into value fields within html elements.
- EX; <textarea value=”<? php echo $expertise ?>”></textarea>
- This made it impossible as far as I could tell to do html injection protection since when I attempted to replace all quotation marks with &quot and all apostrophes with &apos, the value field would display them as direct text and not convert them back to a quotation mark or apostrophe. However, if I left them as a quotation mark or apostrophe, it would end the value assignment operator and thus destroy the view ability of the page. This can be seen if you use this string for input in Account Creation: !@#$%^&*()_+-={}[]\|;:’”,>?.< or any string with ‘ or “ for that matter. But that was my test string. When I tried to move the user input inside of the textareas so that it would be displayed as apostrophes and quotations, the page broke due to different sections of javascript attempting to get the values of parts on the page.

- Hence I started from scratch to make a more readable version of the AccountCreation.php page and the Profile.php page. Both AccountCreationSecondEdition.php and ProfileSecondEdition.php function identically to AccountCreation.php and Profile.php respectively, but protect against html injection and sql injection.
- These pages in reality can just be used for reference, or used instead of the previous pages.

Future Works;

- I would like to transfer this new look for the ProfileSecondEdition page over to the ViewProfile.php page, and the AdminConsole.php page where it uses previous code to display user information.

Known Bugs;

- No known bugs.

Styles

- This is the section regarding which css files are used for which pages.
- adminStyle.css
    - This css page is used for styling the AdminConsole.php page. It is similar to the conversations page in the fact that it is segmented the same, with Admin commands on the left and the display area on the right depending on user input.
    - The .admin_vew_command table styling and .admin_view_command td styling are a part of a previous form of the admin page, and are unused currently.
- login.css
    - This page has the styling necessary for the login container that is used by mentors and the buttons on the login container.
- messages.css
    - This page has styling for messages and conversations. There are sections in here that deal with a search bar in the conversations area, but that was later scrapped due to complexity of trying to search through messages. However the styling is still available for a later date.
    - This has the classes used for displaying received vs sent messages, and the time stamps below each message, etc.
- pageOverlay.css
    - This is the base white overlay page to allow for a consistent container size for all of our pages to use.
- PopUps.css
    - This contains styling for the buttons and textareas of popup messages (such as the one used on the ViewProfile.php page for mentees attempting to start a conversation with a mentor, as well as the popups created for the Accept/Pending/Deny/Report User functionality on the Conversations.php page.
- profileStyle.css

    - This has styling for the header and other portions of the AccountCreation.php page, Profile.php page, and ViewProfile.php page
- search.css
    - This has styling for the search bar in the top of the Search.php page, and the results section down below.
- style.css
    - This is the base style.css that was given to us along with the header/site-header/footer files to allow for the format Eastern Washington University uses on their websites. We did have to adjust the header/site-header/footer files to reflect our page, especially with the possibility of mentor/mentee viewstates.

LandingPage.php

Programmer

- Div class that is a carousel with three inner classes to hold images, transition effect is achieved through the use of a bootstrap transition.
- Below the carousel are user stories simply placed in paragraphs and styled.

Known Bugs

- Format text better perhaps use a div class

FAQ.php

Programmer

- Header with underlying paragraphs and text

Known Bugs

- Format text better perhaps use a div class

TermsAndConditions.php

Programmer

- Header with underlying paragraphs and text

Known Bugs/improvements

- Format text better perhaps use a div class

<u>Profile.php</u>

Programmer

- Top of page checking SESSION variables and redirecting to applicable pages (AccountDisabled.php, Login.php) for disabled/under review accounts.
- Top also sets some variables based on SESSION useful for the page later such as user_id which is used to identify which user is logged in.
- Top down the next lines of code are based on POST array, this is where the editing of the profiles happens.
    - Editing for profile items is based on the relationship between hidden variables that are stored at page creation and mirror the values placed in user view. These values are posted to the POST array when page is submitted. This is to check if the entry posted is new and hence old needs to be deleted and new added.
- After the adding and deletion code, there are calls to the database to help later with construction of the datafields ie. how many of each there are.
- Next are essentially three javascript functions
    - Function one is for when user clicks edit button, this functions changes the populated fields, not the hidden ones, to not disabled so the user can enter information and click delete and add buttons as well for multirow data entries.
    - Function two is for when user clicks submit changes button, this simply submits the form and inherent data.
    - Function three is actually 4 functions that are for deleting a row from a multi row entry. This is based on a passed in row that will then be emptied of html code but not deleted. Code is repopulated if user decides to add a row again.
- The main form is next in html code mixed with php. Hidden fields are populated for data storage used up above when posted. Then fields user can influence and see are then populated from database.
- The last portion is four javascript functions, one for each row type of multiline entry. These functions bind to their prospective row type, when clicked they edit the hidden field of lastDeleted corresponding to their row type. This allows the user to delete a row from the user view but it is not gone and is overwritten when another is added.

Known Bugs/improvements

- maximumNumMentees needs to add control for less than to 2, do not add then
- currently half multirow lines are working correctly but expertise and skills has a small bug where it will not add a row if you change it in the middle rather than deleting and adding a new row which works.
- Email verification is not ready so editing email is not implemented yet. Email must be verified to work first then can be edited.
- Refactors
    - deleteRow functions could be changed to one and have arbitrary passed in values.
    - Delete and add code using post array could be refactored into one function with two branching paths based on the amount of passed in parameters ie. two types of rows one with two values and the other with three.

<u>AccountCreation.php</u>

Programmer

- Top of page checks if you are logged in, if so redirect to profile.
- Next page checks data using POST array to set variables in database for user and create the account.
- Password hashing is used and the only thing stored on database and passed through is the hash.
- Data entries cannot be empty strings or will ping to create again
- Email is hashed as well and verified before creation
- SESSION variables set next for ease of login.
- Middle of page is html for creating the form entries to be passed to POST array
- Bottom of page is 8 script functions bound to delete and add buttons for the four row types rows (two each)

Known Bugs/improvements

- No known current bugs
- Refactors
    - When checking POST values for account creation some values are checked against hard coded values which could be edited to be changeable by admin. Example maxNumMentees is set to no more than 2000, could be value set by admin.

<u>Login.php</u>

Programmer

- Page checks globals at load to see if logged in, and if disabled etc. page takes you to the appropriate page based on those or stays on page till user enters login info
- Next area down from checks is where password and other info is checked
- Simple data entry for gathering password and username, also button for creation as well

Known bugs/improvements

- No known bugs at this time.

<u>ViewProfile.php</u>

Programmer

- On page load SESSION is checked to make sure it is a mentee searching.
- Then another id is checked to make sure a mentor profile id is set
- If all checks are passed mentor information is pulled from database
- Another check is made to make sure mentor is available due to conditions such as already have a conversation with that mentor etc.
- Next are three functions the first makes a popup the second sends it and the third deletes it

- Lastly data is populated and multi row entries are accessed from the database and populated at the same time

Known Bugs/improvements

- No known bugs at present