WhiteBoard: Final Report

CS 360-001: December 1st, 2017

Prepared For:

Trevor D. Brown

Prepared by:

Bailey Henson, Maddy Van Hulse, Mary Odenthal, Zachary Stovall, Daniela Zieba

Western Kentucky University

[bailey.henson531@topper.wku.edu](mailto:bailey.henson531@topper.wku.edu)

[madeline.hulse061@topper.wku.edu](mailto:madeline.hulse061@topper.wku.edu)

[mary.odenthal1034@topper.wku.edu](mailto:mary.odenthal1034@topper.wku.edu)

[zachary.stovall466@topper.wku.edu](mailto:zachary.stovall466@topper.wku.edu)

[daniela.zieba727@topper.wku.edu](mailto:daniela.zieba727@topper.wku.edu)

**Table of Contents**

1.) Introduction ...............................................................................................................................1

2.) Strategic Plan .............................................................................................................................1

2.1. Overview

2.2. User Estimations

2.3. Approach

2.4. Milestones

2.5. Schedule

2.6. Benefits

3.) General Requirements…………………………………………………………………..…….7

3.1. Purpose

3.2. Scope

3.3. Product Description

3.4. Analysis Models

3.5. System Hardware Architecture

3.6. Change Management Process

3.7. Definitions, Acronyms and Abbreviations

4.) Specific Requirements……………………………………………………………….……….19

4.1. Purpose.

4.2. External Interface Requirements

4.3. Functional Requirements

4.4. Use Cases and Diagrams

4.5. Inverse  Requirements

4.6. Non-Functional Requirements

4.7. Design Constraints

4.8. Logical Database Requirements

5.) Frontend……………………………………………………………………………………....30

5.1. Bootstrap

5.2. Design

5.3. Font Awesome

6.) Backend……………………………………………………………………………………....32

6.1. Docker

6.2. PHP

6.3. MySQL

7.) Features………………………………………………………………………………………32

7.1. Calendar

7.2. Gradebook

7.3. Messaging System

8.) Visibility .................................................................................................................................33

8.1. Documentation

8.2. Client

8.3. Team

9.) Risk Analysis………………………………………………………………………………...34

9.1. Process

9.2. Technology Risks

9.3. People Risks

9.4. Organizational Risks

9.5. Requirements Risks

9.6. Estimation Risks

9.7. Tools Risks

9.8. Prioritized Risks

10.) Progress………………...……………………………………………………………………38

10.1. Initial Progress

10.2. Second Milestone Progress

10.3. Third Milestone Progress

10.4. Fourth Milestone Progress

11.) Performance……………………………………………………………………………...….50

11.1. Performance Requirements

11.2. Possible Hardware Bottlenecks

11.3. Possible Software Bottlenecks

11.4. Possible Solutions

11.5. Performance Benchmarking

12.) Human Computer Interaction………………………………………………………………..54

12.1. Usability Requirements

12.2. Users and Stakeholders

12.3. Test Cases

13.) Software Testing Strategies…………………………………………………………………56

13.1. Strategic Approach

13.2. Verification and Validation

13.3. Unit Testing

13.4. Integration Testing

13.5. Validation Testing

13.6. System Testing

13.7. Alpha and Beta Testing

13.8. Verification Testing

13.9. Debugging Strategies

14.) Pseudocode…………………………………………………………………………………71

14.1. User Account Creation

14.2 Announcements

14.3. Creating a Course

14.4. Posting Assignments

14.5. Calendar Postings

15.) Reviews……………………………………………………………………………………..72

16.) Acceptance Testing…………………………………………………………………………72

16.1. Functional Correctness and Completeness

16.2. Backup and Recovery

16.3. Competitive Edge & Usability

16.4. Performance

16.5. Documentation

17.)Final Project Adjustments……………………………………………………………………74

17.1. Final Project Overview

17.2. Final Project Scope

17.3. Final Project Requirements

18.) Conclusion ……………………………………………………………………………….75

19.) Future Work………………………………………………………………………………75

I.) Appendix .…………………………………………………………………………………75

A. Raspberry Pi Setup

B. GitHub Repository Setup

**1. Introduction**

WhiteBoard is a project designed to recreate the primary features of Blackboard. This software is commonly used by professors to digitally manage classes with ease. It provides students the ability to have an overview of all their courses. WhiteBoard allows professors and students to be mutually aware and in agreement of happenings in their courses.

WhiteBoard allows teachers to assign homework, post class announcements, and upload or download useful materials. Students also have uploading and downloading capabilities. A messaging system provides a lightweight and seamless outlet for users (including teachers, and students) to be able to communicate with one another. The grading system will be responsive and allow students to see grades soon after the instructor has entered them.

These goals will be accomplished by utilizing several technologies.  Bootstrap and Thymeleaf will be used to assist in the development of the user interface. Spring framework and Docker will support the backend development. This will allow the development team to streamline WhiteBoard’s development process and deliver a complete, polished project under the deadline.

Our client for this project is Trevor Brown and he will be providing us with goals, requirements, and guidance when needed throughout the course of the project. Mr. Brown will also help to make sure we stay on track with our planned milestones.

**2. Strategic Plan**

**2.1 Overview**

WhiteBoard’s main goal is to provide a central hub where students can access all of their class-related information anytime and anywhere with ease. Our project will provide instructor accounts with a management system for each of their specific classes. Institutions will be able to register through WhiteBoard to create instructor and student accounts, as well as create courses.  These levels of account separation will be convenient for the hierarchy of the typical mainstream university. Administration, faculty and students will each have separate functionality designed specifically with each of their needs in mind.

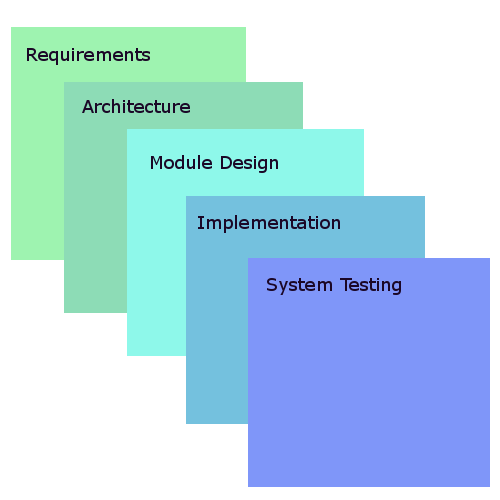
    Additional features of WhiteBoard will include a gradebook interface that instructors will be able to update with grades for all assignments completed by students.  This gradebook interface will be accessible for viewing by student accounts.  A calendar interface which will automatically update as instructors add new course content will be available for viewing by students as well.  Lastly, WhiteBoard will come with its own messaging system, so students can communicate with their instructors and other students.

Displayed in Figure 1 is the proposed design for our primary home page. It Highlights our project’s primary features and the bases for our site wide user interface. The two containers displayed by the user interface will dynamically update with the information selected by the user. For example, what if Figure 1 was a student account. In the left container would be all the announcements and assignments that were posted by instructors. The right container contains all the courses that a particular student is registered for.  Content on Whiteboard is easily erased and replaced with new, more relevant content..  This will be an improvement upon WhiteBoard’s top competitor, Blackboard.  Currently, BlackBoard requires users to navigate through several static web pages in order to completely understand their grades and requirements. We wanted to make sure that would not be an issue with WhiteBoard’s interface.

**2.2. User Estimations**

As Whiteboard’s intended use is for an institution such as a college or university, the user estimation for Whiteboard would be approximately 1,500 - 15,000+ depending upon the size of the institution using the website. The majority of users would be students, the rest comprising a much smaller amount and being staff of the institution. Despite the projects intended use as a tool for universities, the only likely user of our project, and the one we will be taking our input from, is our client. Therefore, the only likely users of our project will be the people in our group, our client, and anyone else asked to review the project for feedback or discovering errors in the intermediate versions of the project. In addition, the final goal of our project is to serve the client and adapt our plans accordingly, not necessarily to make changes in anticipation of an institution or other such possible client.

**2.3. Approach**

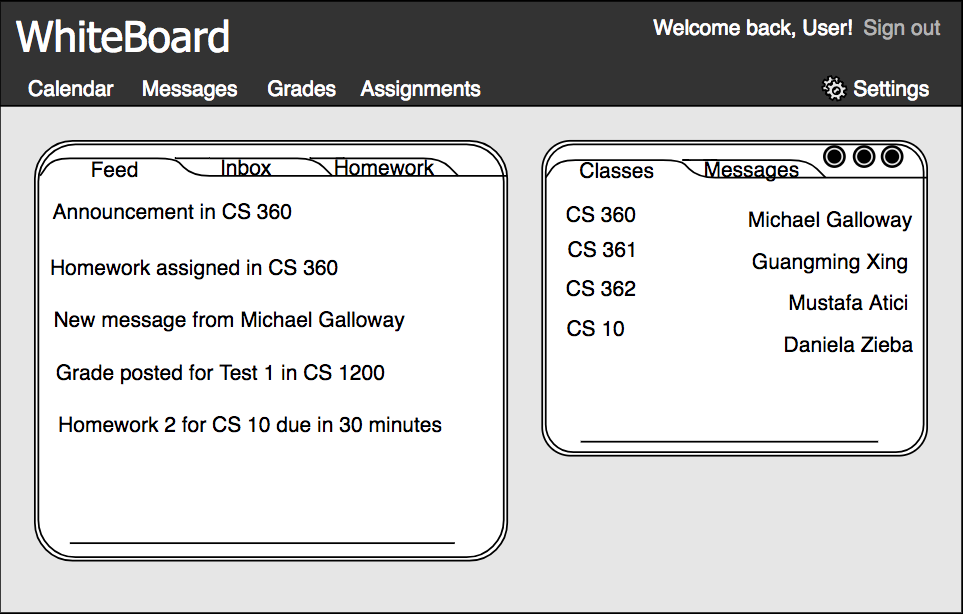


**Figure 1: Sashimi Software Development Model**

**2.3.1 Software Process Model**

We plan to develop our project with an iterative, bottom-up approach, meaning that we will provide various iterations of our project at each milestone and begin development with deployment and configuration of Docker containers.  This is how our development team will outline tasks and achieve increments by specific milestones. This ensures the base features of the project are fully functional so that as more features are added on throughout the course of the project, we have the ability to return to a functioning version should any serious problems arise during the implementation of new features. The development team will also be following the Sashimi software development model as we achieve the increments. This will provide us with direction on what each increment should attempt to achieve.

    The Sashimi model begins with system requirements and moves sequentially through architecture, module design, implementation and system testing. This model is different from the waterfall model because the steps are layered.  System requirements cannot be completed until much of architecture has been developed and at least a little bit of system testing has been explored. This allows more flexibility than the waterfall method, as you can return to the beginning layers and revise as needed while you complete the later layers. This software model is illustrated in Figure 1.

****

**Figure 2. Initial Web Interface Prototype**

**2.3.2 Operational Prototyping**

A technique we are using in development is operational prototyping, where we aim to combine the techniques used in both throwaway and evolutionary prototyping techniques. Throwaway prototypes are often created with the goal of aiming for a better understanding of requirements, and result in more detailed requirements for a project, while evolutionary prototyping has the end goal of becoming a part of the fully-functional system. So, the above progress regarding our initial interface and backend design serves as our base evolutionary prototype. From this initial prototype, we will build a series of throwaway prototypes as well as further evolutionary prototypes to create our various project components and features such as the gradebook, calendar, and messaging; a throwaway prototype of a messaging system could be made using our basic evolutionary foundation, for example, to better define our requirements. Then, an evolutionary prototype of the messaging system could follow, that could meet our newly-defined requirements and successfully integrate as a component to our final, fully-functional system.

    The image above (Figure 2) is an example of our operational prototyping where the development team designed a prototype interface.  This interface has evolved into the current WhiteBoard interface after gaining client approval.

**2.3.3. Quality Control Steps**

Methods carried out in development to ensure software quality were to define and negotiate non-functional software requirements, which are documented in another section. Other than adhering to these non-functional requirements, general steps taken towards software quality involved providing well-documented/commented code, pair programming, and consistent testing (as also documented in a larger testing section of this document).

**2.4. Milestones**

There are three primary milestones which involve the creation of a report. Throughout each milestone meetings, presentations, and reports will be given and conducted to ensure progress in running smoothly and meeting the requirements of the client. In addition to these set milestones, our project involves more specific milestones and deadlines relating to our project features, which are listed as such below:

*September 8 - Feasibility report completed, initial project environment setup*

This is the first milestone for the WhiteBoard project and the focus has been on system requirements and setting up the development environment.  Our goals included the purchase and initialization of a Raspberry Pi, which will be used to house the implementation of the project in an environment that is accessible to all team members. In this environment we will be using Docker containers to compartmentalize and protect our program.

Other areas for progress include team members familiarizing themselves with their planned roles as well as conducting research into what languages or tools they will be using throughout the semester. The team should also begin to access their personal limitations and the potential risks that are involved with proposed ideas.

*October 4 - Report 2, Backend architecture prototype completed*

By the second milestone, the majority of our backend architecture should be completed.  This will allow the development team to lay the foundation for the rest of the project. This will include a database of user accounts and information, content, grades and messages. The front end team will collect information about what needs to be learned in preparation for the next milestone in addition to helping out the back end team when needed. If progress allows, front end work may begin.

*November 3 - Report 3, Frontend interface prototype completed*

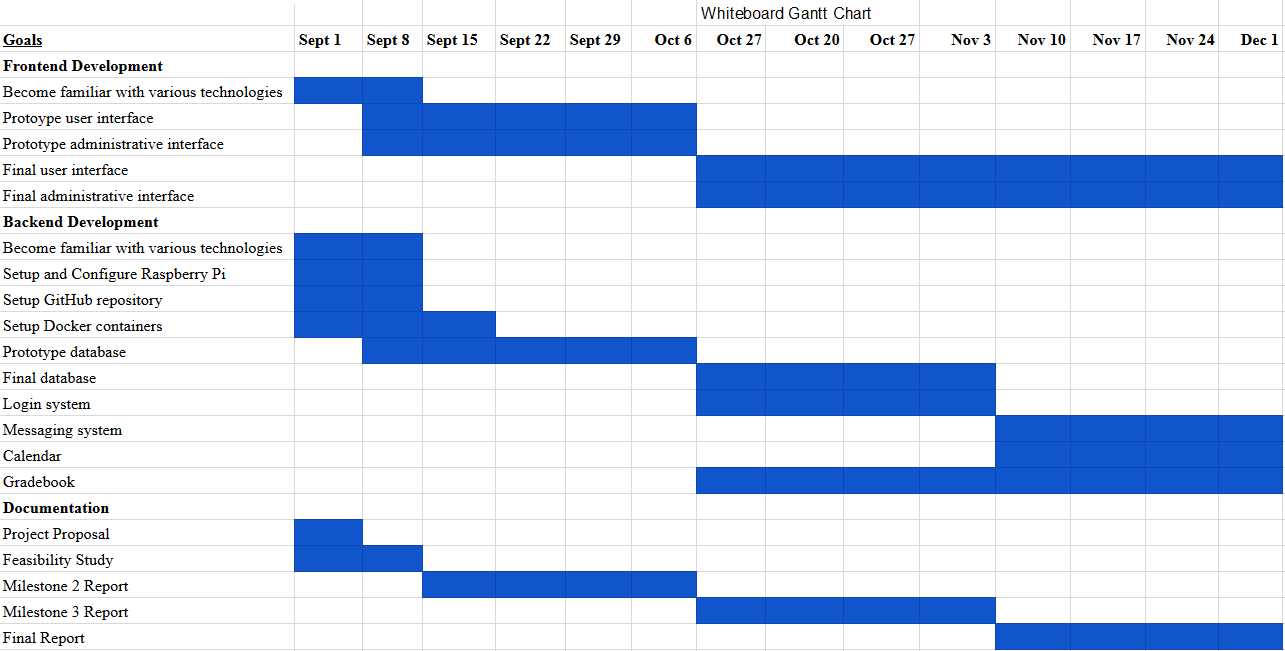
By the third milestone, the WhiteBoard user interface is to be completed.  This section of the program will deliver information from the database built in the previous milestone to the user in a dynamically changing interface. Depending on how far progress has proceeded, further features may be implemented.

*December 1 - Final report, features completed, frontend and backend refinement*

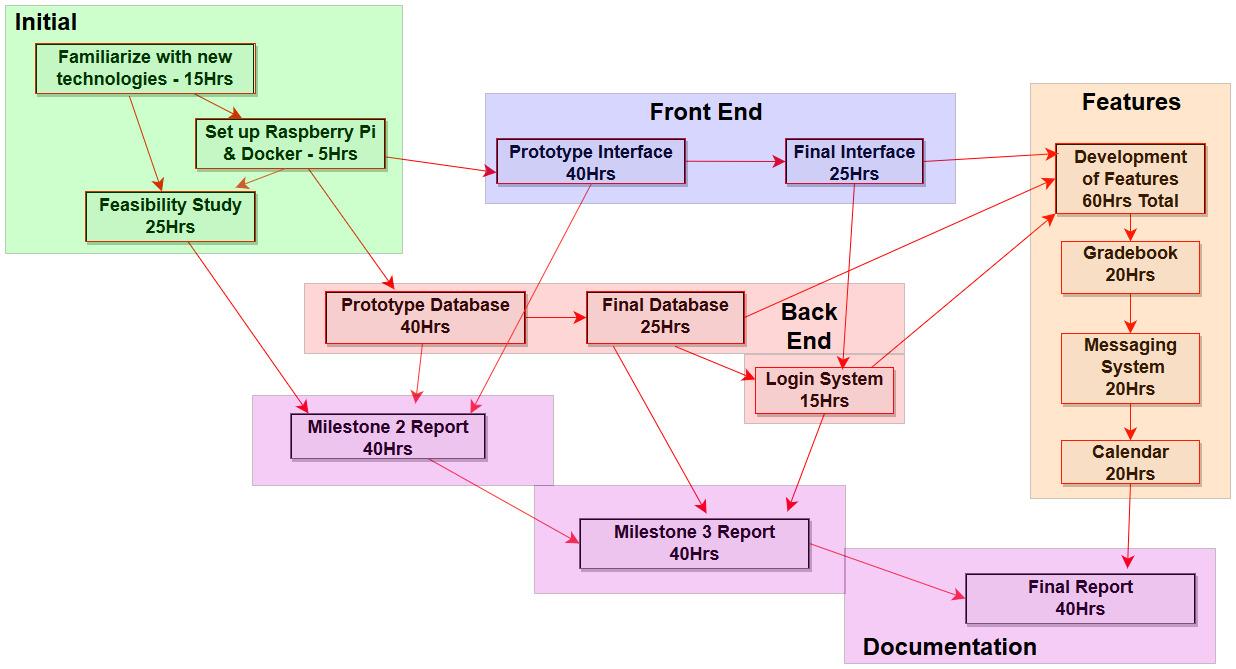
By the fourth and final milestone the development team will complete the additional features.  These features include the messaging system, gradebook, calendar and content posting.  The time before the fourth milestone will also be spent on refining and polishing the frontend interface and backend architecture.

**2.5. Schedule**

Displayed as figure 3 is our Gantt chart, which describes the various tasks necessary for our project and when we aim to develop them. This chart will most likely be subject to change if the needs of our client changes or our development schedule does not progress as planned - some tasks may take longer or shorter than estimated, especially if features must be changed. The roles have been classified as either front end tasks, back end tasks, or documentation-based tasks.



**Figure 3. Gantt Chart**

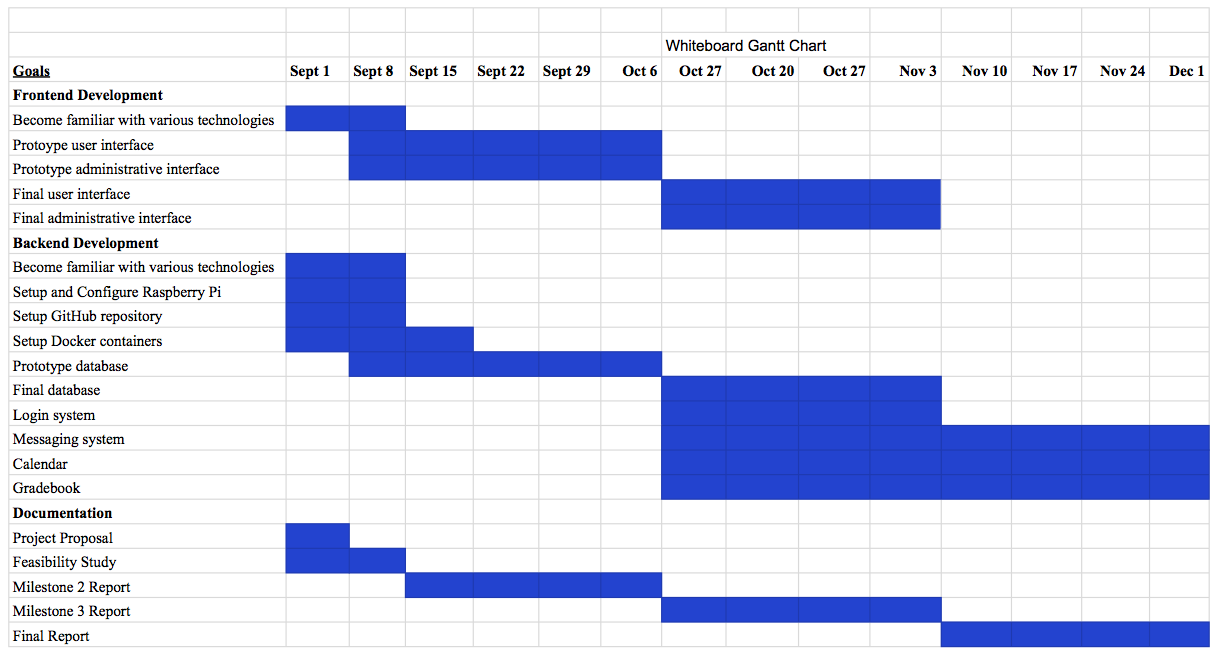


**Figure 4. Activity Chart**

Another aspect of the WhiteBoard development schedule is our activity chart, displayed in figure 4.  This chart will model the relationships of dependencies between certain tasks as the project progresses.  Tasks are categorized as initial, front end, back end, features, and documentation.  It is crucial to the success of the WhiteBoard project to organize tasks in sequential order, so that the time spent on each task is optimized.  This is accomplished through thoughtful planning of tasks so that no time is wasted revising code due to incomplete dependency requirements.

**2.5.1. Final schedule**

Our final schedule did not change much from our original plans, and below is our updated, final Gantt chart:



**Figure 5. Final Gantt Chart**

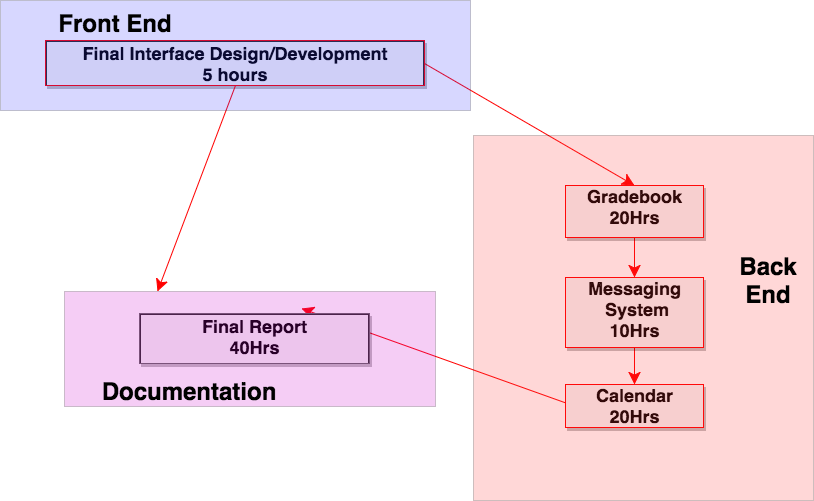
As one can see, our new Gantt chart has hardly changed. The only difference between this and the above chart is starting certain components earlier and finishing other components earlier than expected. The update in our Gantt chart proves our success as software developers in sticking to our time-based budget.

**2.5.2. Final schedule organization**

Our final schedule organization was able to overcome time constraints and maintain weekly or bi-weekly group meetings. Our weekly meetings were always about an hour long on Fridays, while additional meetings switched dates, and either occurred on Mondays after Software Engineering I or Tuesdays. One limitation of this system was that group meetings typically involved around 3 members due to availability. When time was not spent in physical meetings, group members would message each other roughly every other day.

Another issue plaguing the final schedule for WhiteBoard development involved the external factors such as a group member taking a vacation and not spending any time on coursework, as well as frequent work imbalance across the team. Where two members often recorded time of around 8 hours a week (the expected time for this class), other team members would document 1 to 3 hours of work. However, we were able to successfully complete this project, even if some may wonder how more balanced workloads would have affected development.

**2.5.3. Milestone 4 Activity Chart**



**Figure 6.Final Activity Chart**

Shown above is our activity diagram, specifically created for Milestone 4. This diagram only shows minor changes to our Milestone 4 schedule, thanks to our surprisingly accurate predictions on our development cycle. The only major difference between this and our original schedule is the downsizing of the time needed to complete our frontend development.

**2.6. Benefits**

Whiteboard will have several benefits as compared to its top competitor BlackBoard.  The largest advantage of using the WhiteBoard educational system will be its in-house messaging system. This is a feature that BlackBoard does not offer at all.  Blackboard messages are transmitted through the corresponding university e-mail accounts.  This requires students to check two separate interfaces to keep up with their assignments and course content, and also to keep track of their messages. WhiteBoard will solve this issue by providing a all in one package messaging system within its interface. Now it will be easier for students and instructors to use without having to login to a separate interface.

Another issue of BlackBoard that WhiteBoard will improve upon is the amount of navigation required to reach relevant content. When using BlackBoard students and instructors must navigate through several different static web pages before being able to reach course content.  WhiteBoard will address and resolve this issue by providing one static webpage with an interface that dynamically updates with new content. This will cut out the need to constantly reload the same interface across multiple static web pages.

**3. General Requirements**

When taking on a project of this particular magnitude there will be several clear cut requirements.  An account and login system is needed that can differentiate between institution, instructor, and student accounts. The login page should return different visuals depending on what type of user accessed that page. That means that our front-end has to fully functional and work with our backend to retrieve and display data. Organizational accounts will behave differently compared to instructor accounts and student accounts. The backend of our site will be designed so that it can communicate with the database accordingly and provide accurate user related content..

A messaging system will accompany the account system, allowing professors and students to communicate through WhiteBoard. One underused and/or non-existent feature of Blackboard is the ability to directly message a instructor or student outside of the realms of email, which is something we intend to improve upon. The collaboration possibilities are limitless when we give our users the tools they need to work together.

Our software will allow for content posting through file uploads. Our initial goal is to design our site to handle basic text uploads. As we make progress hopefully we will be able to add features like multimedia. Each assignment will be equipped with specific announcements and instructions that will make each assignment easier to understand. As new content is added the calendar features will automatically update to reflect due dates and release dates of new content for all courses.

Finally, there will be a gradebook to keep track of completed assignments and overall grades in each course.  With all of these features, WhiteBoard will enable professors and students to efficiently manage their classes online.  This is especially desirable for students who commute or who work full-time, as online management and completion of assignments allows for more flexibility in scheduling.

The project must be accessible through a mobile-friendly web interface. Nearly all of WhiteBoard’s features will rely heavily on excellent database organization and functionality such as the login system, messaging system, gradebook and calendar. Best practices must be employed relating to aspects of the project such as the database to ensure the convenience and the security for WhiteBoard users.

**3.1. Purpose**

The purpose of the general requirements section is to outline all included functions of WhiteBoard. This is necessary to ensure the development team has correctly understood the expectations of the client.  It is also helpful in keeping the entire development team unified in purpose and scope during the course of the project. It will outline the capabilities and limits of WhiteBoard for any future users. This document is intended for the development team, the client, and any potential users. This section will also discuss the features that are not included within the project. This document is targeted at those without a deep knowledge of computer science concepts.

**3.2. Scope**

There are several requirements set out by the client for this project.  The discussion of scope will be organized by these different requirements.

**3.2.1 Account and Login System**

The account system for WhiteBoard will need to accommodate three separate types of users: institution, instructor and student.  A large aspect of the scope of this project will be to create three alternative layouts of WhiteBoard’s basic interface. Each of these modifications will adjust the functionality based on what type of user is logged in and what types of actions those user can execute.

**3.2.2 Institution Accounts**

Institutions such as a college, university, or even a high school program will need to register and make an account of their own just like any other user. Institutional accounts have unique abilities because they serve as the administrator to the student and teacher accounts.  After signing up, the institution account is responsible for creating all instructor and student accounts that are involved with that institution.  The institution account will also create the courses, and assign the instructor and all students to these courses.  These are the basic functions of the institution account. Alternatively, we as a group could throw together mock up classes students and institutions for the sake of testing. However, to release a product that would not allow institutions the capability to lay out their curriculum and members would be substandard and shortsighted.

A form must be provided for institutions to create new accounts. If possible forms need to have the capability of adding just one student or adding multiple students at the same time.  Any changes to the organization should update content on the site and store the information on the database to reflect the new account creation. After an institution account has been created institution administrators may log in to access the WhiteBoard institution home page.  This home page will present the institution with options to create new instructors, students or courses.  These separate options will be the same basic function to update our stored information. Institutions will also have the capability of uploading files and making announcements that can be seen by both students and teachers. However, each one will update a different element of our stored information based on what the institution is creating: instructor account, student account, or new course.

The another function for institution accounts will be to assign instructors and students to specific courses. This function will update the course information in our database to allow access to courses for those instructor or student accounts. Lastly, institution accounts will be able to post announcements to all students and instructors within that institution.

**3.2.3. Instructor Accounts**

Once instructor accounts have been added to a course and allowed access the account will have permission in our databases to upload course content and also be provided with an interface to submit assignments that are viewable by the students enrolled in their course. Most likely instructors will submit content in some type of form which will allow them to add comments as well as file uploads. Uploaded content could contain materials that are crucial for the assignments completion. This course content will be stored within the course information and displayed on the main page. The simplest form of course content is a reading document uploaded by the instructor. Another type of course content WhiteBoard will allow is an assignment.  An assignment will be an interface made available to students of that course to upload their own documents to complete the assignment.  These uploaded documents will be made available to the instructor and a student identifier will be attached so that the instructor may distinguish who in the course submitted each assignment.

The final type of course content allowed will be announcements posted by instructors.  These announcements will appear on students’ course feeds.  Instructor accounts will also be given access to update and edit the gradebook for each student within one of the instructor’s courses.  This will be made available under the course interface for instructors.  Instructors will be able to creates grades for each assignment and correlate these grades with specific students in the course.  Grade updates and announcements will all be accomplished by creating relationships within our database information so that the correct information reaches the correct students in the correct courses.

**3.2.4. Student Accounts**

Student accounts will be provided with an interface to access all course content uploaded by the instructor.  This will be accomplished by establishing relationships within our database information to deliver all of the course content to the student through the WhiteBoard interface.  Students will also be given access to assignments posted by the course instructors.  These assignments will allow students to upload documents or other files to complete the assignments.  Students will also be able to access a calendar summarizing their courses and a gradebook containing grades of completed assignments as submitted by the course instructor.  Lastly, students will have access to the messaging system to enable communication between instructors and other students.

**3.2.5. Calendar**

The calendar function of WhiteBoard will display the dates and times of announcements, assignment due dates, and reading due dates in a way that is visually easy to understand. Each time an instructor updates course content by adding a new grade or even finishing a grade on a particular assignment the database will show that there was update was made. These changes will immediately be reflected on a student calendar so that those enrolled in the course can be on the same page as their instructors. This will be accomplished by submitted new calendar entries automatically when instructors update new course content. These calendar entries will be associated to the proper courses using relationships within our database information.

**3.2.6. Gradebook**

The gradebook feature will allow instructors to update each assignment completed by a student with the grade achieved on that assignment.  The gradebook interface will be accessible for viewing by students, and will update with the proper information submitted to the database by the instructor.  The gradebook will display individual grades and also an overall grade in the course. This overall grade will be able to have different calculation methods, which will be specified by each instructor. For example, some professors may use an overall point calculations for the overall grade while some may use a weighted calculation so that some assignments, such as tests, are worth more than others.

**3.2.7. Communication**

The communication features of WhiteBoard will enable messages to be sent between instructors and students as well as message sent from students to other students.  This will be managed by dedicating a section of WhiteBoard’s database to message content and establishing relationships between sending and receiving users. This information will be relayed to the messaging interface so that users may view their messages.  This feature will not be enabled for institution accounts.

The design and functionality of the messaging system will be similar to that of Facebook messenger in the sense that students and instructors will be able to send messages individually to other accounts and form groups for discussions. Each user will have a list of contacts whom they will be able to message at any time. Students contact lists will be automatically updated with students in their classes and the professor teaching that class. When the class is over, these contacts will remain in their list unless that are manually removed. Students and teachers will also be allowed to lookup other users at their institution and add them to their contacts list by either name or email address.

The messages sent between users and in groups will be saved indefinitely unless deleted by the user. This will allow users to view past conversations whenever they would like.

**3.2.8.** **Excluded Features**

Features that will be excluded from WhiteBoard include overly ambitious aspects of Blackboard such as video lectures due to the sheer size of saving videos on our Raspberry Pi; because of our hardware constraints as well as our user estimations, the amount of estimated professors and their potential lectures would far surpass the storage capabilities of the Raspberry Pi. Another excluded feature present on Blackboard is that of goals - schools can define the outcomes and goals for various classes, which we believe to be unnecessary in comparison to the core features planned for WhiteBoard. Going along with the goals feature is the feature of goal performance, which is yet again another feature that will be excluded under the basis of unnecessary.

Yet another feature that will be excluded from WhiteBoard is that of being able to create tasks as a student, as it is deemed unnecessary for our primary goals, which are to emphasize communication between students and professors through features such as the ease of electronic homework submission and an electronic gradebook so students can manage their progress in classes.

**3.2.9. Final scope**

The above scope descriptions were drafted up and defined in our initial feasibility research - they have not been subject to much change. Our only major modification to scope involves SMS integration into our messaging system; this was a feature originally planned as a stretch goal, and was ultimately decided against because of the development costs needed and security lowered by integration with third party software. We decided that a better use of our time would be honing and testing our internal WhiteBoard functionality.

**3.3. Product Description**

With the advent of web and the following revolution in web-based technologies, organizations of every kind have found it a necessity to integrate with the web. Educational institutions are no exception, and wide-spread success (and sometimes frustration) can be found in their use of web technologies. Every institution has a website for prospective students, and the majority have digital applications; however, past this, web technologies are used to improve educational opportunities for their own students. Integration of various technologies in a central hub is the standard of today, and the return to a day where gradebooks weren’t available online or teachers weren’t available to talk to by email seems unimaginable to many.

Technologies such as Blackboard, Sapling Learning, WebAssign, and Pearson Revel are used for various purposes. Blackboard-type technologies provide a central hub for students to view grades and upcoming assignments with ease, while the other technologies are specific to one field such as chemistry, math, or computer science. They often integrate with hub technologies to automatically log in students so that they can complete their homework assignment externally and have it automatically graded, much to the ease of instructors.

**3.3.1 Product Perspective**

WhiteBoard is our proposed solution to the demand for aforementioned web-based educational hub technologies, and is aptly named for building on and improving the features of Blackboard. Blackboard is absolutely usable, however WhiteBoard seeks to improve components such as the visual user experience as well as simplify homework assignments and communication between members of a class.

**3.3.2 Product Functions**

General functions Whiteboard will provide for students include the ability to complete homework assignments, access class materials, keep track of due dates, take quizzes and exams, message other students and professors, and view course grades. Instructors will be have access to the same messaging and calendar functions as students, as well as the ability to create assignments and tests, upload class materials, organize class course work, put in grades for student assignments, and customize grade weighting. Institutional administrators will be able to create student and instructor accounts and courses.

**3.3.3. User Characteristics**

As the intended usage of WhiteBoard is in educational institutions, the users of the software will fall into three primary categories: institutional administrators, instructors, and students. So, our design must keep in mind the needs of these intended users. Institutional administrators will need interfacing to easily manage their institution. Especially when many administrators and instructors may not necessarily be technologically experienced, ease of use is key. Students on the whole will most likely be younger and have more experience using web technologies, however some institutions do accept unconventional students.

**3.3.4. General Constraints**

Constraints associated with our project come in a variety of topics, ranging from technological to organizational. One constraint is that of timeframe, where we have a very strict deadline of completion by the end of this academic semester. This means that the features and development of the project must be planned such that we can feasibly deliver the product successfully to our client. Our solution to this is to clearly differentiate between the core features necessary for WhiteBoard and the satisfaction of our client, and then the features which are additional features not necessary but that would overall heighten the experience of using WhiteBoard. There are also constraints in the capabilities of the team; if we were more experienced with the proposed technologies to deploy to develop WhiteBoard, then the project would theoretically run more smoothly. However, issues come with the fact that many team members are inexperienced in web technologies - more time will be spent out of necessity on research of web technologies simply to be able to complete our product. Physical constraints come in the hardware resources available to us, as the Raspberry Pi has limited capabilities in various ways.

**3.3.5. Assumptions and Dependencies**

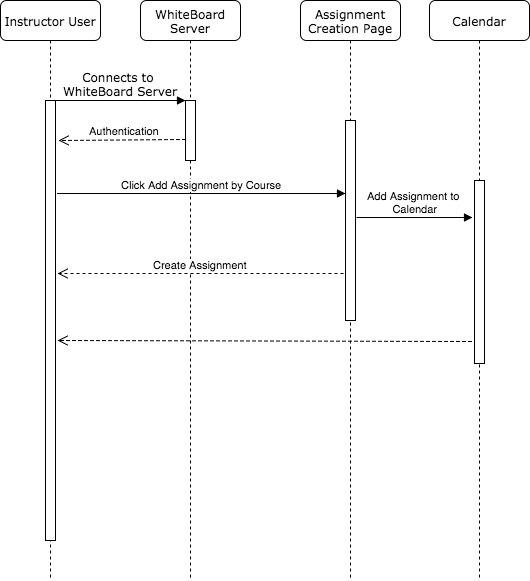
An assumption about our product is that it will be used by educational institutions with the presumed target of universities. Additionally, we must assume that the hardware used by clients will be recent enough for the various technologies that form the components of our website, and that clients are accessing our project through web browsers that are compatible. Of course, our project depends primarily on the hardware capabilities of our server, the Raspberry Pi. Features such as storage and RAM are necessities for the project.

External dependencies for our project relate to the various APIs, frameworks, and technologies planned to accomplish the features of WhiteBoard. For example, using Twilio for SMS notifications would be considered a dependency, as well as the use of the Spring framework for our backend and Thymeleaf for web component templating. More dependencies come in the reliance on a pre-existing and functional e-mailing system associated with a particular educational institution. If features are implemented regarding integration with other educational web tools specific to particular subjects, then our features will rely on the functionality of technologies such as WebAssign.

**3.4. Analysis Models**

**3.4.1. Sequence diagram**

**Instructor Posts Assignment:** the next diagram displays the sequence of actions when an instructor user logs on Whiteboard to create an assignment. The user must first be authenticated as an instructor, then taken to the assignment creation page where the user can enter assignment details. When finished, the assignment will be created and added to the Calendar for all associated users in the specified course.

****

**Figure 7. Assignment Post Sequence Diagram**

**3.4.2. Component diagrams**

This section consists of component diagrams and their associated documentation, which describe the various web interfaces accessed by users for particular components. The diagrams were broken down by user group with the intent of increasing readability.

**3.4.2.1. Student component diagram**

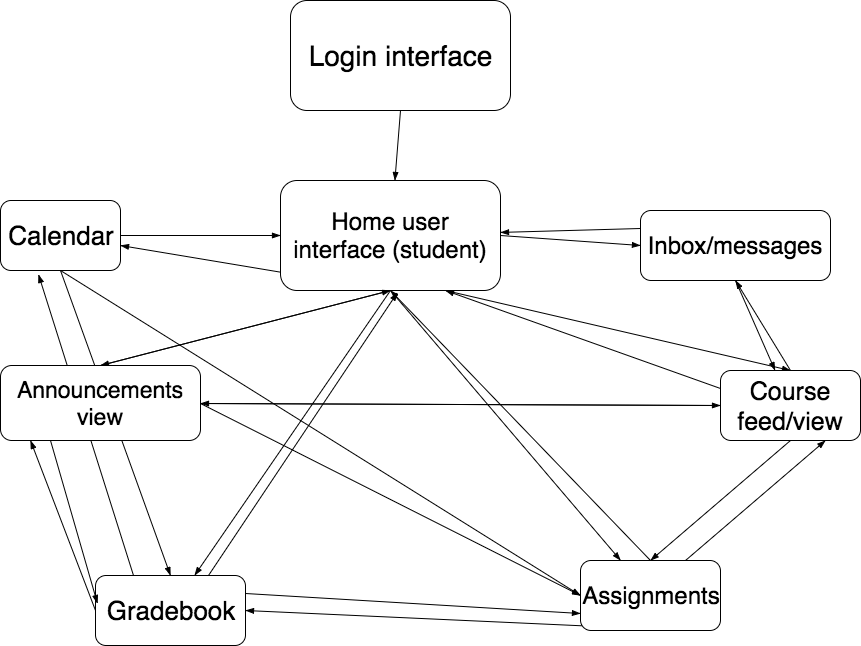
The above diagram details the web interfaces accessed by students, and the pages that then interface with each other. After logging in, students can then view their home user interface, from which a variety of features can be accessed. These features include core project features such as the calendar, messages, and gradebook, as well as views relating to announcements, assignments, and course feeds. All of these interfaces can then interconnect back to one another.

**3.4.2.2. Instructor component diagram**

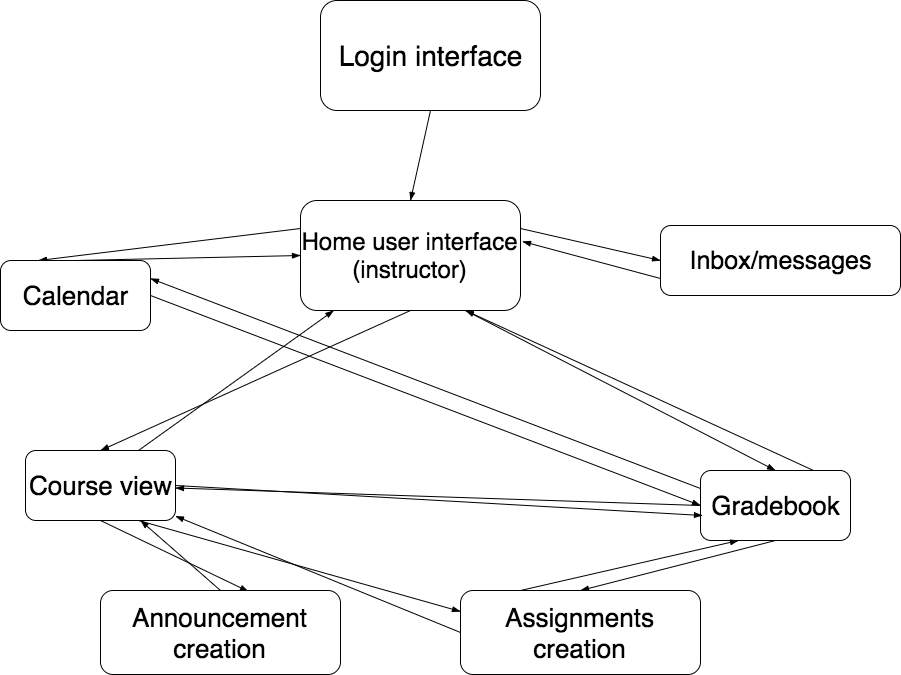
Shown above is the component diagram for instructors. Similarly to students, instructors first access a login page and are authenticated. Next, they have access to a home user interface in which they can view the majority of the interconnected instructor features. For example, the inbox/messages can be accessed, and more importantly, features such as the gradebook. The gradebook then connects to allow instructors to make announcements and assignments, as well as view the students enrolled in their courses and the calendar of assignments.

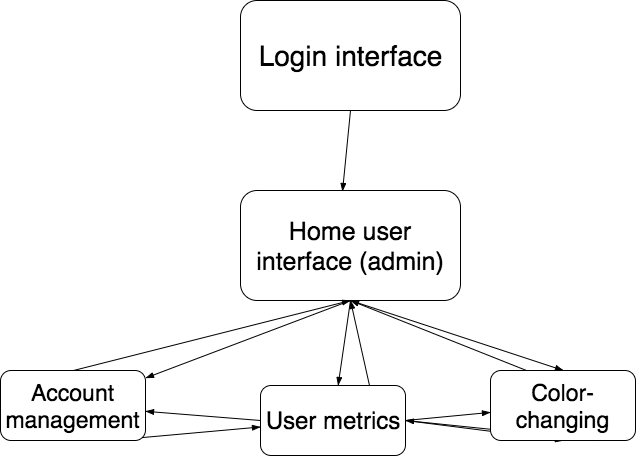
**3.4.2.3. Administrator component diagram**

Displayed on the next page is the component diagram for administrative users. They can manage student and instructor accounts, view user and performance metrics, and color-changing features after successfully logging in through the login interface. These features interconnect with each other to allow for easy interface navigation and enhance the user experience.

****

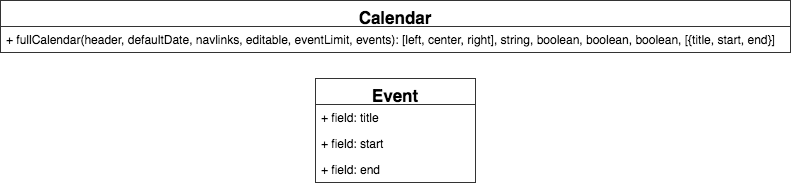
**Figure 8. Student Component Diagram**

  
**Figure 9. Instructor Component Diagram**



**Figure 10. Administrator Component Diagram**

**3.4.3. Class Diagram**

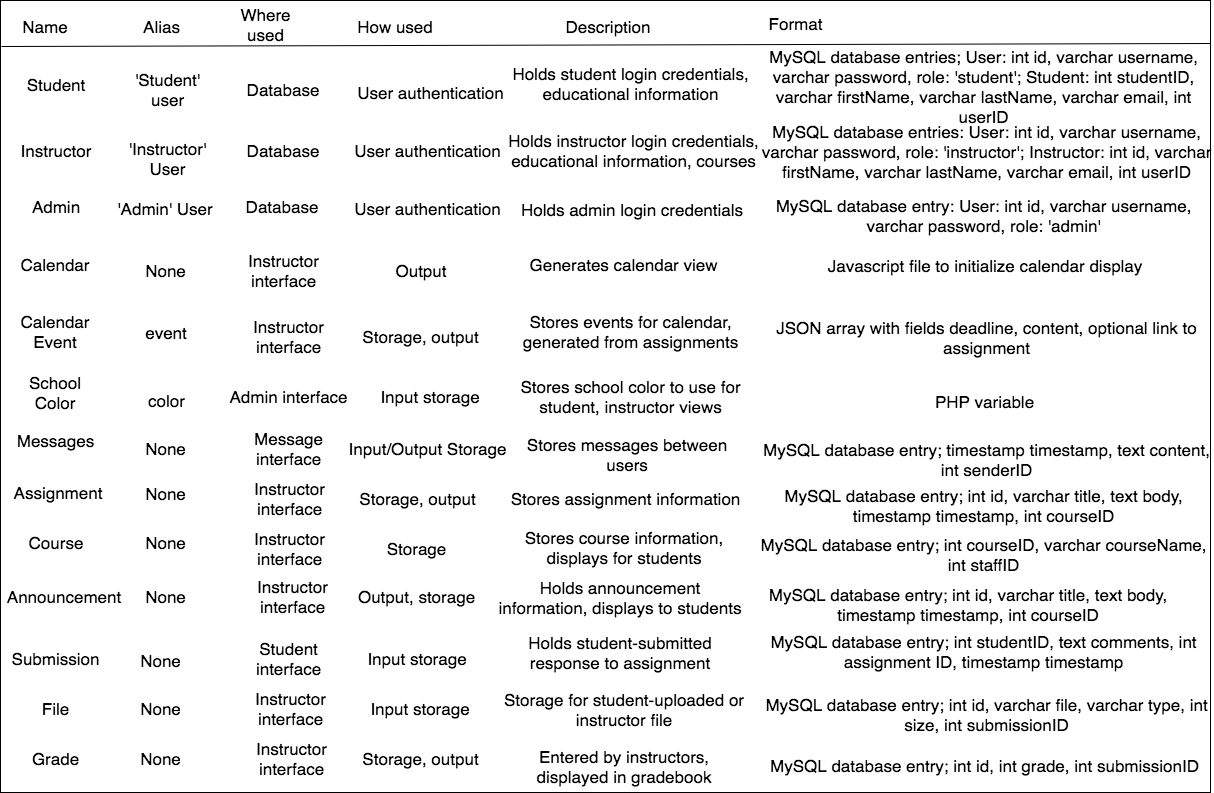


**Figure 11. UML Class Diagram**

Shown above is a class diagram detailing the minimal use of classes in Whiteboard. Classes are only used as a part of the Fullcalendar framework, and otherwise, all of our information is stored in our database entities. The Calendar only has one function, fullCalendar, that initializes it. Some sample fields are shown above, but there is a large amount of fields that can be chosen to initialize the calendar that are both purely visual and logical. The events are also simple, consisting of a title, start, and end date.

**3.4.4. Data dictionary**

Displayed on the next page is the data dictionary for the various types of data needed for our project. As one can see, the majority of our data is stored in our database, and otherwise comes in the form of Javascript or PHP. This data dictionary encompasses entries relevant both across all three user types as well as those needed for the individual user types of student, instructor, and administrator.



**Figure 12. Data Dictionary**

**3.4.5 Deployment Diagram**

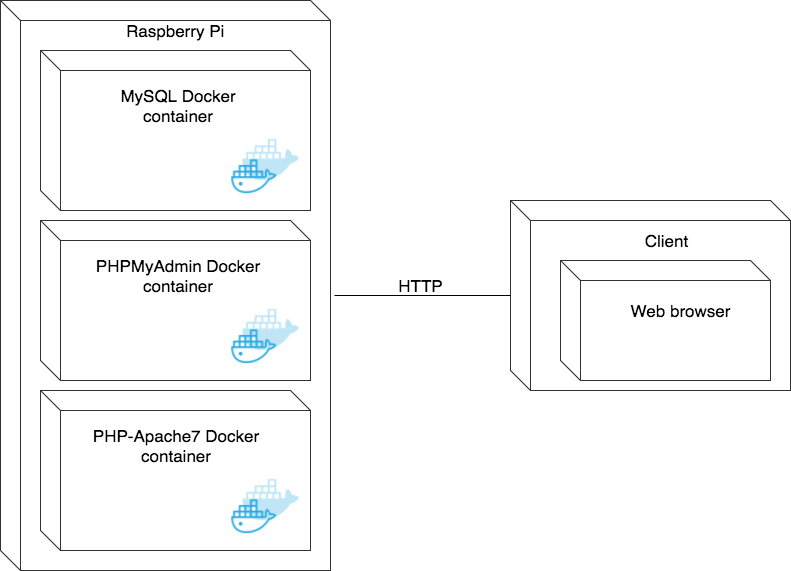
Shown on the next page is our deployment diagram, visualizing deployment of WhiteBoard. As one can see above, the Raspberry Pi serves as our web server on which our three docker containers are held. A client accesses WhiteBoard through their compatible web browser and is connected to our Raspberry Pi through sending an HTTP request.

**3.5. System Hardware Architecture**

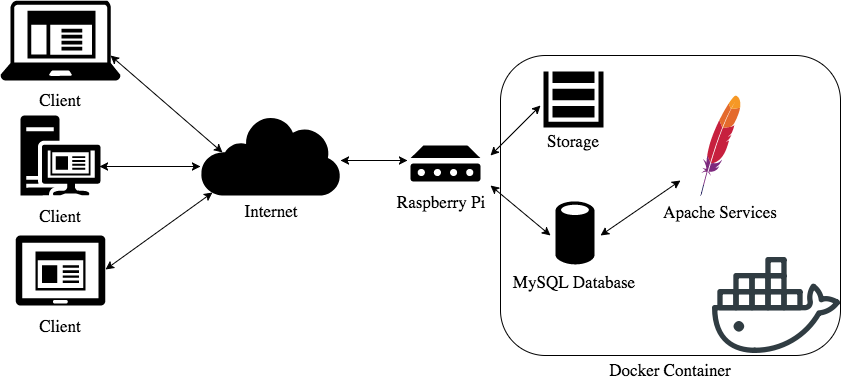
Pictured on the next page is a diagram showing the overall architecture of WhiteBoard: clients access the project through the web, and our server, the Raspberry Pi, handles requests. It connects with our database for user system-associated features and offers storage for homework assignments or files uploaded by professors.

**3.6. Change Management Process**

As development on WhiteBoard progresses, it is inevitable that requirements and plans will change based on the success or failure of planned features and technologies to be implemented. Changes to the requirements of the WhiteBoard project will be discussed within the development team, and between the development team and the client.  When a change is agreed upon by all parties it will be documented and the requirements section updated.  Information on the original



**Figure 13. Deployment Diagram**

****

**Figure 14. System Diagram**

requirements will be documented as well as the change that was made, and the justification for that change.

**3.7. Definitions, Acronyms and Abbreviations**

There are many different technologies used in the development of WhiteBoard.  This section will describe these technologies and their function.

**3.7.1. API**

Application programming interfaces are small programs shared to assist in solving a specific issue.  APIs can be relied upon to process specific tasks so that all functionality does not have to be created from scratch.  Using an API reduces the project timeline to complete certain features of WhiteBoard.

**3.7.2. Bootstrap**

The Bootstrap framework is a program used in the front-end development of WhiteBoard.  It works with the HTML and CSS of the web page to adapt WhiteBoard’s user interface to different browsers that may be used to access WhiteBoard.  It also aides in the creation of the user interface by providing blocks of generic code the development team can use to build the web pages.  These code blocks include both static displayed elements and dynamic functionality to apply to the visual elements.

**3.7.3. Cloud9**

Cloud9 is an online development environment and repository that enables remote collaboration.  It allows for the sharing of a project so that the entire development team may access the same code at any time. Cloud9 also stores revision history so that WhiteBoard can be recovered to an earlier state.  Cloud9’s workspace will serve as the main development environment for the front end of WhiteBoard as it comes with code autocompletion that is compatible with the languages being used by the development team.

**3.7.4. CSS**

Cascading style sheets describe how the elements on a webpage should look.  CSS can include descriptions of size, position, style, and color that many different pages of the website can refer to for information on how to display the content within the page.  Condensing the design information to one source allows the development team to easily change the appearance of multiple pages simultaneously.

**3.7.5. Docker**

Docker is a software container management technology.  Whiteboard will be developed in software containers to ensure maximum compatibility with different runtime environments.  When placing software into these containers all needed source information is stored within the container as well.  This ensure the software will always have the information required to operate even if WhiteBoard is being run on an operating system different than the operating system it was developed on. Containers also increase the modularity of WhiteBoard.  Docker is a framework designed to help manage containers.

**3.7.6. GitHub**

GitHub is an online software repository.  It enables the WhiteBoard development team to save a copy of the project in case of a failure within the Raspberry Pi.

**3.7.7. HTML/XHTML**

HTML stands for HyperText Markup Language and it is a programming language designed to format web pages.  HTML allows you to designate different elements on your web page for different types of formatting.  It works very well with CSS to change or update formatting and design.  XHTML is HTML that agrees with XML syntax.  This is important as WhiteBoard will transfer data by using XML encoding, so all HTML used for the web pages will be XML compatible.

**3.8. Final Product Description**

Our above description of WhiteBoard has not changed since initial feasibility planning, and all system diagrams are accurate to the final product we have developed and delivered to our client. Features such as successful account creation and file uploading designed for our user bases have been successfully implemented, and the constraints of our features are documented in system testing and performance data.

**4. Specific Requirements**

**4.1. Purpose**

The purpose of the specific requirements section is to outline the details of how the WhiteBoard development team will be achieving the general requirements described in the previous section.  This will allow those with a deeper knowledge of computer science concepts to understand how the development team plans to accomplish the completion of the WhiteBoard project.  This section is intended for developers or users with a knowledge of computer science concepts, as well as contributors to future improvements upon the original WhiteBoard project.

**4.2. External Interface Requirements**

**4.2.1. User Interfaces**

**4.2.1.1. Account creation interface requirements**

Institutions will be responsible for account creation.  WhiteBoard will provide an interface for institutions to create their own accounts, as well as an interface to create instructor and student accounts.  Inputs for this requirement will include events triggered by user actions, as well as information regarding account entered into the interface through forms.  WhiteBoard will then process the information provided through forms and create new accounts within the system based on what information was provided.  Outputs from this process will include verification of account creation, reiteration of information provided, as well as any generated identifying information such as an account number.  Possible errors include incomplete or incorrect information entered into the forms.  At this point a message displaying these errors will be displayed and the user will be given another chance to complete the form correctly.

**4.2.1.2. Login interface requirements**

The login page for both instructors and teachers accepts two fields: username and password. This login page then has a login button, will allow access with correct credentials, and has an option to reset passwords using email if forgotten. Inputs are the username for a particular instructor or student and the password associated with the account. The output would then be the verification of the user’s credentials and loading of the dashboard/home page. Potential errors may arise when users either have incorrect credentials i.e. a wrong password or credentials that don’t exist at all in the database; a message stating either of these issues will be returned to the client attempting to login.

**4.2.1.3 Central interface requirements**

The central interface requirements are such that the interface must be able to display a large amount of data and showcase the core features of WhiteBoard. So, these core features are as such: the central interface must provide users with a wide range of options. The user must be able to exit and close their session by logging out of their account as well as be given the capability to adjust settings of the web page. This interface must be able to load data from classes in the form of assignment and announcement descriptions, load messages, and display features such as the gradebook and calendar. The inputs into this interface would be events caused by user actions, which would result in outputs such as the user’s session ending or loading more information about a class. Errors could arise if information is not entered correctly by an instructor and is therefore unable to load properly, so it must be handled by either not showing incorrect information or stating that there is an error and catching it such that the rest of the information pulled is unaffected.

**4.2.3. Software interfaces**

One user group of WhiteBoard comes in educational institution administrators who will access WhiteBoard solely for the purpose of account generation. One of our stretch goals is to maximize ease of use for such administrators by allowing WhiteBoard to interface seamlessly with technologies that possess institutional credentials such as student identifications, instructor identifications, and course lists. Additional interfaces WhiteBoard could potentially communicate with include other subject-specific educational technologies often used so that students can complete their homework and have their grade automatically recorded in a central gradebook; the goal would be for WhiteBoard to link to such technologies, and for such technologies to give information to WhiteBoard for entry into the gradebook.

**4.2.4. Communications interfaces**

There are a variety of external communications interfaces and internal features currently planned for use in WhiteBoard. With regards to internal features such as the WhiteBoard-specific messaging feature, security measures are taken, such as best practices in database setup.

A proposed stretch goal for account information will be WhiteBoard’s capability to integrate with institutional interfaces; Western Kentucky University, for example, uses TopNet. A hypothetical link between TopNet and WhiteBoard would allow for optimal user account creation by Western Kentucky University, if they were a user for WhiteBoard. Instead of some form of manual account generation or even the need to export data from a system such as TopNet, an institution would optimally be able to interface their software with WhiteBoard to automatically gather information on instructors, students, and course numbers to add into WhiteBoard.

Another stretch goal for features would be the integration of WhiteBoard with aforementioned technologies tailored towards assignments of a particular discipline. Successful integration of tools already linked to an institution would make this goal as simple as pulling user credentials from the institutional database to automatically login students and provide them with a link. The only additional feature that would be necessary to integrate with this tool is integration with gradebook technologies such that a student could complete an assignment and have the grade automatically entered into WhiteBoard.

**4.3. Functional Requirements**

**4.3.1. Gradebook requirements**

The gradebook must show the current class grade for students by displaying a list of classes and then the assignments for each of the classes. For instructors, the gradebook must allow grades to be entered or modified. The gradebook must automatically calculate grades based on weights. Inputs for the gradebook are the grades for assignments from instructors and then the students selecting the option to view grades for their classes. Outputs, then, are the grades to be displayed to students.A potential error could come in having incorrect credentials when attempting to view a particular class grade for a certain account; a message is relayed to the client accordingly.

**4.3.2. Calendar requirements**

The calendar for WhiteBoard displays the deadlines for assignments pertaining to the classes of a particular student. Necessary inputs for the calendar differ based on the client accessing the calendar feature - the inputs an instructor would give would be their information regarding a particular assignment, which would come in the form of the title, description, and due date, along with a submission link if applicable. The instructor output would be the displaying of the assignment to students. If a student accesses the calendar, their input would be the course identifications associated with them, and the output of the calendar would then be the assignments associated with the courses a student is enrolled in. The calendar will handle errors such as that of an assignment not having a set due date by omitting the assignment on the calendar.

**4.3.3. Messaging requirements**

Messaging must be available to both student and instructor accounts and allow messaging on a one on one or group basis. Contacts for students should consist of other students in their courses and their instructors for those courses. Contacts for the teacher should include other instructors in their department and students in their classes. Each user should be able to create conversations and add however many users to the conversation they would like from their contacts list. If a user should try to message a user who is not in their contacts list, an error message will be displayed informing them of their mistake. All past messages sent in conversations should be stored for future viewing by the user.

**4.3.4. Credential-based requirements**

To use WhiteBoard for classwork relating to a particular institution, both students and teachers must have the proper login credentials for security purposes and will have accounts created by the institution accordingly. Inputs are the  student and instructor data by the institution administrators and the output is the generation of accounts for the students and instructors to use. The institution must provide WhiteBoard with the correct data necessary to generate student and instructor accounts, so an error message will tell the institution what data is missing and not create accounts if anything is incorrect.

**4.4. Use Cases and Diagrams**

**4.4.1 Instructor Posts Assignment**

**Actor**: CourseInstructor

Flow of events:

**1.** CourseInstructor connects to WhiteBoard server

**2.** WhiteBoard server authenticates CourseInstructor

**3.** CourseInstructor selects the course they are teaching and then selects the add assignment option

**4.** CourseInstructor enters the details of the assignment -  title and description - then specifies how the assignment can be turned in; they are given the options of allowing students to upload to WhiteBoard or requesting a physical in-class submission and can specify the due date

**5.** CourseInstructor has automatically added assignment to calendar for both students and CourseInstructors to see

**6.** CourseInstructor logs out and waits for students to complete assignment

**4.4.2. Student Uploads Assignment**

**Actors:** Student, Instructor

Flow of events:

**1.** Student connects to WhiteBoard server

**2.** WhiteBoard server authenticates Student

**3.** Student selects course from dashboard and can view assignments associated with course

**4.** Student reads assignment title, description, and can view the due date of the assignment, as well as the assignment as posted on the calendar

**5.** Student can attach a file through WhiteBoard with text comments if necessary

**6.** Instructor connects to WhiteBoard server

**7.** WhiteBoard server authenticates Instructor

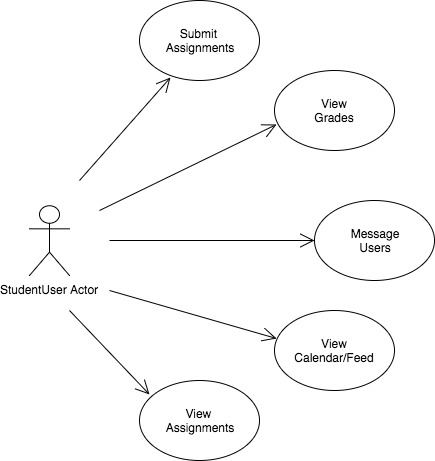
**8.** Instructor is notified of submission of homework and can access assignment for grading

**9.** Instructor assigns grade which is automatically entered into gradebook

**10.** Student views grade on submitted assignment

**4.4.3. Student User Use Case Diagram**

The following diagram details the functions available to a Student User’s account. Each student will be able to complete and submit assignments, view grades from each of their courses and individual assignments, message other students and instructors, view announcements, and view their calendar of activities/assignments.

****

**Figure 15. Student User Use Case Diagram**

**4.4.4. Instructor User Use Case Diagram**

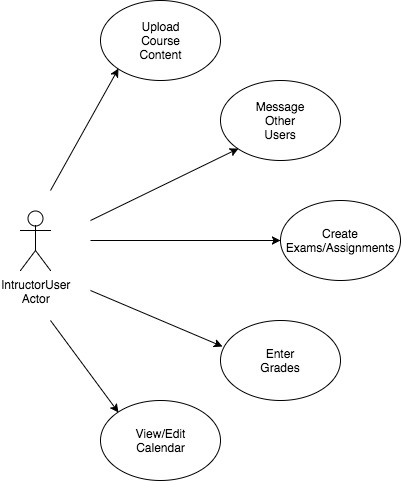
The diagram below displays the functions available to an Instructor User’s account. Each instructor will be able to create and upload assignments, as well as exams and quizzes, manually enter grades for students in each of their courses, message other students and instructors, make course announcements, and view and edit their calendar of activities/assignments.

**4.4.5. Administrator User Use Case Diagram**

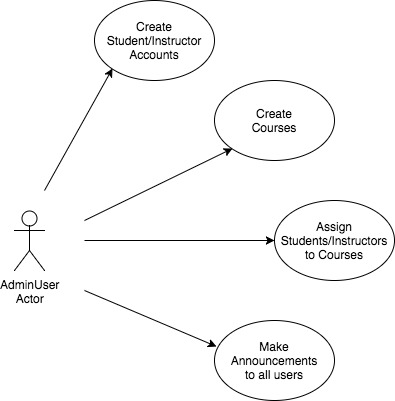
This use case diagram displays the functions available to an Administrator User’s account. Each Administrator will be able to create student and instructor accounts, create courses, assign students and teachers to their respective courses, and make institution-wide announcements to all students and instructors.

**4.4.6. State Diagram**

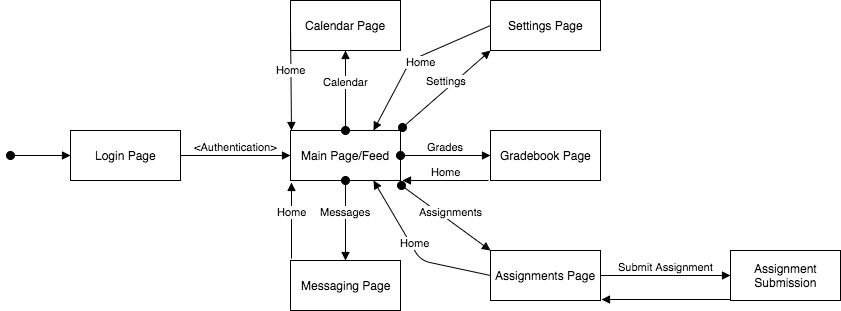
Figure 18 details our state diagram for how Whiteboard should function. Initial access brings up the login page. If the user’s credentials are correct, they will successfully login to access the main page. From there, the user will be able to use a navigation bar at the top of the webpage to access Whiteboard’s various features. Clicking on items in the bar will bring the user to each separate page, where they will be able to utilize the unique features of each page. Clicking home will take the user back to their main page/feed where they started out.



**Figure 16. Instructor User Use Case Diagram**

****

**Figure 17. Administrator User Use Case Diagram**

****

**Figure 18. UML State Diagram**

**4.5. Inverse Requirements**

Any classroom requires rules and limitations for education to be effective. Inverse requirements describe functionalities that are intentionally excluded from WhiteBoard to enhance the educational environment.

**4.5.1. Assignment and Submission Sharing**

WhiteBoard will not allow students to share assignments or submissions with each other.  The transfer of assignments will be strictly limited to two relationships.  The first relationship is instructors may share assignments to students within their course.  The second is that students may submit completed assignments back to their instructor.  This will help to discourage academic dishonesty within students as it will limit their ability to share submissions with each other.

**4.5.2. Available Messaging Contacts**

WhiteBoard will only allow students with at least one course in common to message each other through the messaging system.  Excluding this functionality will discourage the distraction of excess communication with students not in the same class, and instead maintain the focus of communication on related coursework.  This will help to ensure the academic integrity of the web based education environment.

**4.5.3. Unverified User Registration**

WhiteBoard will not allow unverified students or professor to register as part of an existing institution account.  It would cause distraction and clutter within the system for students and professors who do not belong to an institution to register for a user account.  This issue has been blocked by placing the responsibility of user account creation solely with the verified institution accounts.  Once an institution account has been verified and created that account is responsible for creating user accounts for the instructors and students who belong to that institution and then delivering the login credentials to the correct instructors and students.

**4.6. Non-Functional Requirements**

**4.6.1. Performance**

Performance of WhiteBoard must allow students and instructors to load WhiteBoard pages in a reasonable amount of time dependent upon connection speed. If this is not possible WhiteBoard will not be a viable medium for students and professors to complete their academic work within the typical time constraints of University semesters. Additionally, performance must include the ability of WhiteBoard to complete requests without failure.  Failure to process announcements from instructor could directly contribute to students not completing their assignments on time. WhiteBoard must also be able to handle the load of users present within the system.

**4.6.2. Reliability**

The WhiteBoard system must be reliable for students and instructors to access so that both can meet particular course-associated deadlines. If WhiteBoard has questionable reliability with regards to system aspects such as successful file uploading and general accessibility, then this could inconvenience professors given late submissions to assignments as well as students who may lose points for a reason that is the fault of the system.

**4.6.2.1. Practices for increasing reliability**

There are some core best practices that the WhiteBoard development team has tried to follow which fall into eight categories listed below:

1. Limit the visibility of information in a program - this means that various components of our project should not share any data needed past that which is necessary for implementation.
2. Check all inputs for validity - where users can input information of various kinds, we must test to ensure validity. Issues are often caused when programs react adversely to unexpected inputs.
3. Provide a handler for all exceptions - exceptions can be classified as errors or unexpected events (including power or hardware failure), and handling of such exceptions must be accounted for in our software.
4. Minimize the use of error-prone constructs - this means that human error should attempt to be as small as possible by making sure that programmers are on task and do not lose track of system functionality and relationships.
5. Provide restart capabilities - lengthy interactions between users and systems may mean that work can be left unfinished, and functionality should be provided so that users may exit the system and resume unfinished work.
6. Check array bounds - even if we are not using technologies and languages where memory locations can be specified outside of the location of an array, one best practice to mitigate error is to make sure that all calls to locations are within the boundaries of arrays.
7. Include timeouts when calling external components - if certain components of software take a particularly long amount of time to load, then a best practice is for them to timeout and end an otherwise never-ending denial of service.
8. Name all constants that represent real-world values - various variables used in the implementation of WhiteBoard must have names that reflect real-world values and that can be understood by multiple members of the development team.

**4.6.2.2. Operational profile**

Before reliability goals can be defined, the operational profile for our system must be developed. This section will detail the characteristics and usage of different system components, defining the priorities of different components according to their probability of use. Reliability tests will be therefore determined based on the operational profile developed to describe WhiteBoard. Using Western Kentucky University as an example combined with common sense, it is clear to anybody by taking a gander at our three user categories that there is one that is inherently much larger than the other two: students. Administrators and instructors have many functions of WhiteBoard that can be utilized, however in size are significantly smaller than the number of students using WhiteBoard - this is simply due to student to faculty ratios. After that, WhiteBoard would most definitely see more usage out of instructors than it would administrators, as instructors will use the system to manage classes by performing tasks such as uploading assignments, grading assignments, and entering course grades for students to see. However, as a result, instructors may spend more net time using WhiteBoard than each individual student would. So, instructor-related reliability features of WhiteBoard must account for the extensive periods of time instructors may find themselves using WhiteBoard, whereas student-related reliability must instead be defined based on WhiteBoard’s capabilities of serving as many students as they can at one time.

**4.6.2.3. Reliability goals and metrics**

Some classic metrics of system reliability come in the following forms: mean time between failures, mean time to failure, availability, and mean time to repair. So, one of our reliability goals will be drawing conclusive data on these metrics.

    Implementation-based methods that will be used to best work towards reliability goals are in a variety of topics more specific than the above eight methods of ensuring best reliability practices. User inputs will be validated by establishing and checking input that meets standards of input size and representation. Some common programming errors our developers attempt to avoid during development include unconditional branch statements, floating-point number-related difficulties, pointer-based errors, memory leaks, parallelisms, recursive memory overflows, interruptions, aliasing, unbounded arrays and overflows, and default input processing, which is something more detailed in security requirements.

**4.6.3. Availability**

WhiteBoard must be available to students of all backgrounds; some students may have access to laptops and personal computers while others may not. Availability of technology to access WhiteBoard with is an assumption, as educational institutions of every kind today are typically supplied with desktop computers out of necessity. So, metrics of availability will be measured in the ability of WhiteBoard to be available on a wide range of browsers if possible, as many institution-provided computers may be outdated.

**4.6.4. Security**

The WhiteBoard database holds potentially confidential information in the form of grades as well as personal information of both students and professors. A leak of data could leave individuals vulnerable, but also allow students to gain access to certain files or features that they are usually restricted from. So, best practices are followed in database design and file storage with proper design and encryption practices.

One of the most common types of security attacks is a buffer overflow attack. This type of attack happens when a program overwrites the memory adjacent to a buffer that is not supposed to be modified. Languages like C/C++ and Assembly code are often vulnerable to such attacks because they allow direct access to the memory. We don’t plan on using any of those so the threat of this attack happening are already pretty low for us. As a natural precaution we plan on implementing the max length function on our html form. While it would not protect us fully from an attack, this tactic would slow down a malicious hacker from put in a ridiculously amount code in our form.

Many systems are targeted by SQL injection attacks. Attackers pose their SQL queries as trusted commands in order to access or destroy data.  This attack is common and could cause a lot of damage to our system, fortunately it is also very easy to fix. Instead of retaining user input and putting it directly into a SQL statement the input is validated by a prepared statement with parameterized queries. Our security architecture is layered. After analyzing some of the common vulnerables that our site could be victim to, we both validated and trimmed input.

**4.6.4.1. Security Requirements Table**

|  |  |
| --- | --- |
| Identification requirements | The system should identify users as students, instructors, or institutions before allowing interaction with the Whiteboard system. |
| Authentication Requirements | Users will be identified through login information stored in the database. Users will have both a username and password they will be identified by. |
| Integrity Protection | Password protection and encryption will be used to protect users information from possible system attackers. The software must be protected from tampering, corruption, deletions, and insertions. |
| Confidentiality Requirements | Procedures will be established to ensure user information will be kept confidential and that only designated users will have access to the system. |

**4.6.4.2 Security Threat Model**

Our security model is built around our WhiteBoard Site. Students, Instructors, and Admins or Institutions are external entities that can interact with our systems entry and exit points on our site. The whiteboard site is ran on our raspberry pi and Authorized users can send requests to the site. The site pulls data from one the data store housing all our html and javascript files. Since our website adapts based  on users however some data still needs to be pulled from our database on the docker container. Unless a user is authenticated whether through a login or a form they will not be able to access data associated with the pi/docker boundary.

**4.6.4.3. Security design and implementation**

Documented in this section are the specific security measures taken in WhiteBoard’s development. More considerations were taken and researched than implemented due to the cost of security and prioritization of other system requirements.

**4.6.4.3.1. User Authentication and Authorization**

As the security threat model describes, unauthorized users are a security issue. WhiteBoard mitigates this issue by requiring user authentication accomplished by login system functionality. Only users with valid usernames and passwords are able to login and view confidential information such as student grades or university identifications. Every user group requires an account - administrators, instructors, and students all must login before viewing any WhiteBoard-stored educational information. Administrators create the student and instructor accounts, so their methods of keeping login information secure is an external concern dependant entirely on the administrators using our system.

**4.6.4.3.2. Database design and security**

One specific measure taken to mitigate damage in the case of a database breach is encryption of user data. Our passwords, for example, are hashed before entry into the database, so a database breach would mean passwords would remain confidential and secured.

**4.6.4.3.3. Neutralization of inputs**

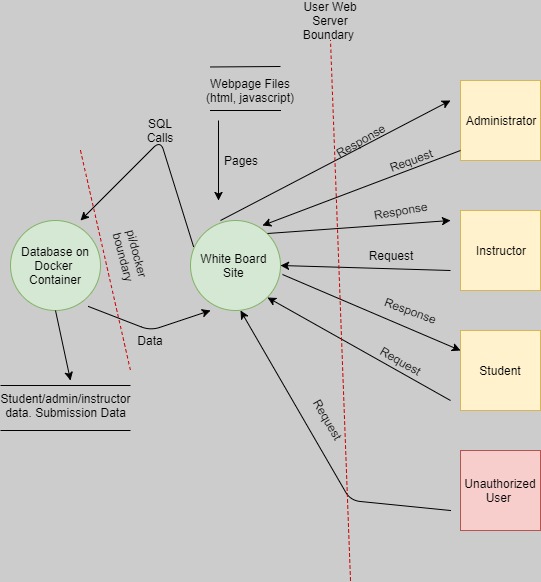
All user inputs are neutralized and checked to ensure that any input does not pull information from our database or is otherwise a command that could be catastrophic to the system. User inputs are possible through essentially every interface of WhiteBoard, and spans from fields such as usernames and passwords for account creation and modification to messages sent between users to the creation of events to post on a calendar to gradebook entries.

**4.6.4.3.4. Barriers**

User authentication and authorization measures are described above, and another feature that relates is the setup of barriers across the system barring unauthorized users from entry. Without successful authentication, users will not be able to simply access pages by having access to a link, and will require login. Barriers are set across all pages with core WhiteBoard features, which includes the three home pages for administrators, students, and teachers, from which all other features are accessed.

**4.6.5. Usability**

Whiteboard should be easy to use for students, instructors, and administrators. Users should be able to use all system functions with little to no training. Everything should be organized in such a way that the user understands how to effectively use each function of Whiteboard with minimal user errors or confusion. The average number of errors made by users should not exceed one per hour. We believe that this error rate can be sustained through the use of easy to read clearly defined navigation bars, content boxes, and labels. One of the principles that Whiteboard is built is that of usability. We want our users to be able to access content with ease.



**Figure 19. Security Threat Model**

**4.6.6. Maintainability**

WhiteBoard must be easily maintainable when it comes to the end of a semester, for example, where students must become unable to contact other students from a class they are no longer taking and institutional administrators and instructors generally don’t need to face any unreasonable amount of time updating classes for a certain semester for students. This will be accomplished by interfacing with pre-existing educational institution databases where applicable, and if such databases are nonexistent, then WhiteBoard will do its best to optimally rotate courses, instructors, and students. As well as maintainability in that regard, WhiteBoard must also be easily maintainable from a more technical standpoint. Web technology standards and browsers may change, and WhiteBoard must be able to change accordingly. The product must be maintainable in yet another regard when it comes to the possibility of adding more functionality to tailor to the changing needs of educational institutions.

**4.6.7. Portability**

Similarly to availability, Whiteboard should be able to be accessed by any user that wishes to on any device. It should be easy to set up by administrators/institutions and function properly on all available web-browsers such as Google Chrome, Safari, Firefox, etc. and different operating systems such as iOS, Windows, and Unix.

**4.7. Design Constraints**

**4.7.1. Physical boundaries**

The Raspberry Pi 3 is the main physical piece of technology that we will be using for our project. We designated it to be our web server for the Whiteboard site. As regards to the raspberry pi’s specifications, our documentation will go into more detail later. We are using the model B version with HDMI out, RCA video out, 2 USB ports, an SD card slot, a headphone jack, and an Ethernet port. Our Raspberry Pi 3 is a Quad-Core 1.2 GHz CPU with 1GB of RAM. This means that we should have all the firepower we need for our project. However, we are somewhat limited in running high power applications on our Raspberry Pi. I probably would not be wise to try and stream loads of video or play flash games on our device. Even some of the programs that we are capable of running on the pi might not run as fast as we are used to working with on our personal computers. From a more practical standpoint the raspberry pi is not an indestructible device so it we must take into account the physical care that goes into making sure the system doesn’t break from a drop, a careless gesture or a bump. We want to try to make sure that we don’t spill any type of liquids on our pi as well.

**4.7.2. Logical boundaries**

Logical boundaries of the system are topics that may restrain us such as the limitations on the system design itself as well as the technologies used to implement the system and what is capable using them. A lot potential boundaries with our logic have to do with our code. For example we are using XHTML on the front end of our site. It is a family member of HMTL and created in a way that fits with the stricter standards of XML. XHTML doesn’t allow us to code as sloppily as HTML does so we have to stay more within a structure to implement the things that we want. We are using several different languages so we want to make sure that everything is talking to each other effectively. We will use thymeleaf and several POST variables to make sure that all the information sent the backend of our website if relevant and correct.

**4.7.3. Social boundaries**

Anytime there is a project that involves serving multiple users with the potential that those users will us interact with one another there must be social boundaries involved. For example, how will we limit who a student or an organization can talk to or not talk to? Ideally, when we deploy our messaging system, we only want to our site to be used for academic purposes. Boundaries will need to be implemented so that our hypothetical students and teachers don’t turn our web application into Facebook Messenger.

Another social boundary of our project is in the form of the technological capability of our users. Certain users may not necessarily employ best practices in technology, for example with regards to selection of a secure password to accompany their password. Additionally, some professors may object to general use of the online system, as there are many who only have vague understanding of today’s technologies such as e-mails and smart devices.

**4.7.4. Economic boundaries**

Obviouslythere will be some economic boundaries associated with Whiteboard. While some of us in the group have jobs, none of the group members have the financial resources to pour a lot of money into this project. Other than the raspberry pi our team has not made much monetary effort in relation to our project. To maximize speed, minimize storage issues, and deploy a sustainable and scaleable site we simply would have to spend more money. Whiteboard is a class assignment and there will be no revenue generated by its use. We have decided to limit ourselves in setting functional and/or non-functional requirements that will require us to put money into our Whiteboard site.

**4.7.5. Political boundaries**

In some projects developers have to be conscious of how their actions in the design or deployment of a project will affect the influence of a particular organization or group. There is little political gain or loss involved in deploying the Whiteboard site. There are also no requirements in this regard in the eyes of our client. However, we would want to avoid any content on our site that would give off the impression that we are a politically non-neutral party.

**4.8. Logical Database Requirements**

Pictured on the next page is a diagram of our current database schemata, which must accomplish various system requirements. The diagram shows how the backend may deal with certain requests and needs of students and professors and how requirements such as grades are dealt with inside the system. Staff and students are managed in the system by use of their student and staff IDs that will be associated with certain institutions as well as unique database-specific IDs to avoid duplicate entries. Additionally, so that they can be notified about any potential system issues such as maintenance or student questions, staff enter their e-mail and phone number. Students also have a login system and enter their email for account safety as well as communication with professors who may contact them about course assignments, announcements, or grades.

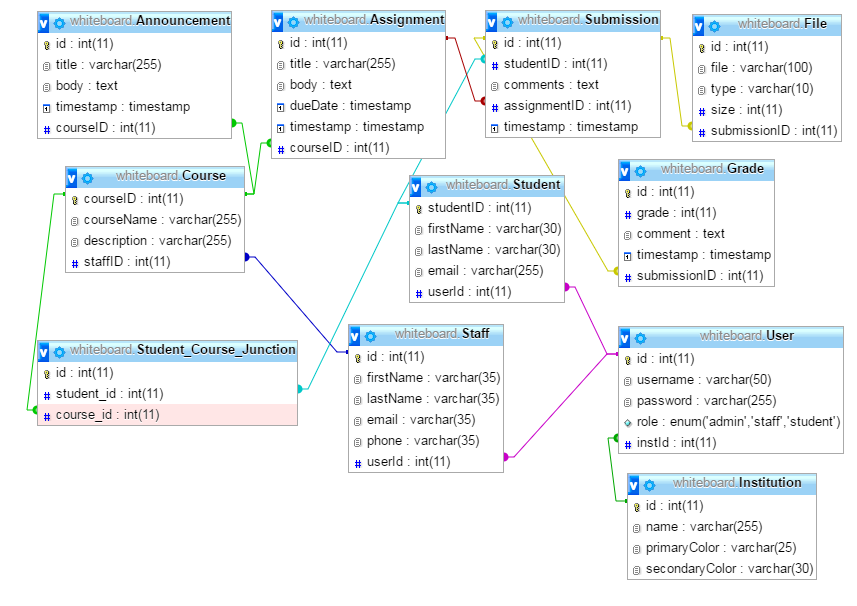
    As well as the student and staff login systems, the database also manages information about courses, grades, assignments, and announcements. Courses are stored with a unique ID, the staff ID of the course instructor, and the student IDs of the members of the course. Data defining the course specifically are the lists of announcements and assignments. Announcements and assignments have similar properties, and assignments tie to specific grades/submissions that come together in the Gradebook feature to automatically calculate total course grades for students.

**5. Frontend**

Much of this project's success depends on a reliable and easy to use frontend. WhiteBoard will use the Bootstrap framework that allows the development team to make a professional looking page without requiring extensive experience with front-end design or utilizing languages such as HTML5 and CSS.

**5.1. Bootstrap**

The frontend framework that will help us design WhiteBoard makes web development more accessible to those lacking in experience. The templates and shortcuts will make WhiteBoard look visually appealing without requiring the development team to invest large amounts of time in



**Figure 20. Database Dependency Diagram**

complex CSS style sheets.  This will allow for more time focusing on improving features of Whiteboard rather than struggling with user interface design and functionality.

Another positive of using Bootstrap is that it allows developers to deploy adaptive web design. This means that no matter what device a user is accessing WhiteBoard from, the Bootstrap code will make adjustments so content is still displayed in a way that is easy to read, comprehend, and utilize. This component is important for Whiteboard because users will be to be able to access it on a variety of devices: phones, tablets, desktops or other mobile devices.

**5.2. Design**

One aspect of WhiteBoard involves the design of the user interface with the goals of being visually pleasing regardless of device used. Some graphics are rendered using SVGs, or scalable vector graphics, created using Inkscape. SVGs are XML-based and can be easily manipulated as a result. The SVGs created specifically for WhiteBoard are sometimes animated to achieve effects such as the planned writing on the landing page for WhiteBoard.

**5.3. Font Awesome**

One technology to improve the user experience is the incorporation of icons created by Font Awesome that are free to use. A large amount of icons are available for free, and will be used on various website components to classify notifications such as announcements and differentiate them from other notifications that could come from WhiteBoard components such as the gradebook. Using Font Awesome will overall ease development while increasing the quality of our user experience.

**6. Backend**

To enable success on the frontend, a reliable backend is required. Our goal is to create an application that can execute using the Raspberry Pi 3 as our web server, which can then communicate information in the database to the browser, giving the end user a seamless experience. As mentioned previously, the backend developers will attempt a bottom-up approach to the WhiteBoard project with the technologies at our disposal - ultimately to help save us from possible future frustrations in the design process.The technologies we plan to use are:

**6.1. Docker**

The development team anticipates the use of Docker as a solution to the problem of collaborating simultaneously on code with other group members. Lightweight containers can be deployed that each run the processes in isolation. The hardships of handling Virtual Machines (such as starting up and maintaining) are removed. These containers only bundle the bare minimum that is needed to run the software, such as the libraries and settings, instead of the full operating system. As a result, a single web server can run multiple of these containers that guarantee the software runs the same in any instance.  Docker will save time in collaboration, and this will allow more of our limited time to be dedicated to polishing and fine-tuning WhiteBoard.

**6.2. PHP**

As a replacement to our initial plans involving a Java backend built using Spring and Thymeleaf, PHP was selected as a solution to our backend-related needs. One reason for this was that our development team is experienced in PHP; the time that would be needed for developers to learn about the Spring framework would cause too much delay in development under our tight time constraint of project completion by the end of this semester.

**6.3. MySQL**

To handle necessary features such as our login system, a database is essential. The majority of the team have at least some experience using MySQL, an open-source relational database management system. Most frameworks also include plugins and features that are compatible with this database tool. Several of our group members are familiar with querying databases.

**7. Features**

**7.1. Calendar**

The calendar necessary for displaying dates and times of assignments for classes is being developing using pre-existing open source calendar code created using Bootstrap for design. Developing a calendar from scratch would require a significant amount of time dedicated both to style and implementation, and this time will instead be spent ensuring that all of WhiteBoard’s features will be functional and reliable for our client.

    Functionality of the calendar comes in the capability to display events not only by month but also by day or week. The original calendar design consists of white and purple with a background picture; as WhiteBoard is planned to be customizable in color by the institution using it, this will be custom, as well as the background picture, if it remains. The original calendar functions by allowing users to add events by clicking on dates, but this feature will be removed; instead, the assignments will be automatically displayed when instructors post the homework or tests that are otherwise visible on WhiteBoard through the class and general feeds.

**7.2. Gradebook**

The gradebook system ties into the database structures, particularly those of the Courses, Grades, and Submissions. Instructors will be able to enter information at the beginning of courses describing how final course grades are calculated to maximize transparency between students and instructors about current grades. Both students and instructors will be able to view the grade for every assignment as well as the final grade of the student.

**7.3. Messaging System**

The messaging system will allow student and teacher user accounts to send messages to each other. This feature will not be available to institution/administrator accounts. The messaging system will be modeled slightly after Facebook’s messaging system Facebook Messenger. It will allow student-student, student-teacher, and teacher-teacher messaging as well as the ability to create chat groups consisting of multiple accounts.

Each user will have a contacts list detailing each account the user has the ability to message. Contacts lists for students will initially consist of the professors who are teaching the classes they are enrolled in and the students with whom they share classes. Teachers will have contact lists with all other teacher users and student users available, however the primary accounts showing will be teachers and students in their classes.

**8. Visibility**

**8.1. Documentation**

Multiple forms of documentation will accompany this project. This feasibility report will gradually grow in size and scope, allowing both the development team and our client to observe growing progress on the project. As the project develops this report will evolve from a feasibility report into a report on the architecture and implementation of WhiteBoard.

This report will discuss the design of the database that powers WhiteBoard, as well as the ways in which the user interface interacts with this database to deliver information through a dynamically changing web page.  It will also discuss the implementation techniques and assisting technologies used to achieve this architecture.

Drafts and snippets of the project report will be deployed during longer intervals between milestones throughout the semester. Smaller, weekly progress updates will be submitted to the client for review at every client meeting. At each milestone we will report on what has occurred over the course of the period in the form of a Milestone Report, which will be turned in to our client for review.

Another important aspect is user documentation.  Information on how to best utilize the features of WhiteBoard will be developed for both students and professors. There will be internal documentation designed to be used by the development team.  This will include meeting minutes summarizing the discussions of each group meeting, and a gantt chart to keep the team on track with development goals and progress.

**8.2. Client**

Progress will be reported as needed to the client through documentation delivered by e-mail and through a weekly in-person meeting with the team in order to ensure that the project development meets the client’s requirements. Additionally, live demonstrations and presentations will be given throughout the development of WhiteBoard along with the previously mentioned Milestone Reports. Daniela Zieba, the project manager, will act as our primary client contact.

**8.3. Team**

Communication among the team is done through Facebook Messenger. Documents and presentations that are related to our project are shared and stored via Google Drive. Code is backed up on GitHub, and the entire team meets at least once a week in person. Meetings are currently planned to take place after class on Monday, Wednesday, and Friday. This schedule provides us all with some flexibility, anyone not present at the meeting will be able to catch up on details through minutes taken throughout the meeting. Additionally, team members working on similar parts of the project may meet outside of designated times to spend time collaborating. If members are for some reason unable to meet in person, Skype or some other form of video/audio calling may be utilized for collaboration and communication purposes. No matter the personal schedule, each of us has dedicated several hours each week that we plan to use on the project.

At meetings, each person will report the amount of progress they have made since the last meeting, including any problems, successes, concerns, etc. that they have. Progress that has been made will them be compared to the Gantt chart, shown at the end of this report, to insure the team is on schedule. If we are discovered to be behind schedule, a discussion will take place as to how we will go about catching up so that the project is successfully completed on time. We also have developed a sheet to log each of our minutes both to show to our client and for the sake of peer evaluations.

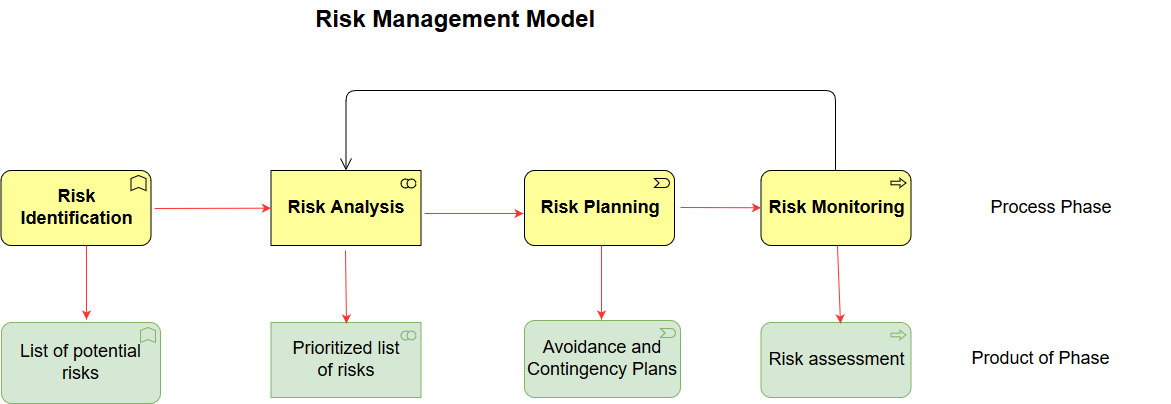
To aid in ease of collaborative web development amongst team members especially with regards to remote development, Cloud9 has been employed and team members have set up accounts. Essentially, Cloud9 functions as Google Drive for web development, with a variety of features ranging to integration with Virtual Machines and GitHub to allowing for setup of common web framework such as Django and Wordpress. We believe that employment of Cloud9 in tandem with GitHub will best allow our team to communicate and develop seamlessly to best produce the software product for our client.

**8.3.1. Team Meetings**

The development team met in-person at least twice a week directly after our class meetings.  These meetings were used to check progress, and delegate individual responsibilities.  Average meeting times lasted between twenty and forty minutes.  Meetings were all held on campus, but the locations varied between Snell Hall, College High Hall and Thompson Central.  There were two occasions where the group met virtually over Skype when not all members were able to be on campus to physically meet.

**9. Risk Management**

**9.1. Process**



**Figure 21. Risk Management Diagram**

The process for risk management the WhiteBoard team will follow includes the steps of risk identification, risk analysis, risk planning and risk monitoring.  Each step in the model will have a specific result produced before moving on to the next step which makes this a sequential model.  However, the risk monitoring step requires the team to reevaluate developing risks as the project evolves. Risk monitoring will divert the development team back to the analysis stage as new issues arise if the need to change risk priority occurs.  This also gives our risk management process a cyclical element, as we will constantly be working through the process as the project progresses.  This section contains our risk management process model in figure 8, as well as descriptions of each phase in the process.

**9.1.1. Risk Identification**

In this phase of the risk management process the development team will create a list of possible risks associated with the project. The WhiteBoard development team has worked together to identify risks which may affect the project development in six different categories.

**1. Technology:** involve the hardware and software used during the course of development.

**2. People**: regarding the members of the development team and their responsibilities within the project.

**3. Organizational**: associated with the project administration who is directing the development team.

**4. Requirements**: have to do with establishing and achieving client expectations.

**5. Estimation**: involve the accuracy of time allowances for increment completion.

**6.** **Tools**: may arise with the use of many frameworks and support softwares such as Spring and Thymeleaf.

The development team believes this to be a comprehensive list of possible risks associated with the project.  Establishing these categories enhances the risk identification phase, and helps to ensure that no possible risks are overlooked by the development team.

**9.1.2. Risk Analysis**

The WhiteBoard development team will undertake risk analysis by considering two main factors: probability of occurrence, and impact of effect.  Probability of occurrence will examine how likely the event at risk will transpire.  Impact of effect concerns the possible loss caused if the event at risk occurs. These two factors will enable the WhiteBoard team to create a prioritized list of risks at this stage.  This will allow the WhiteBoard team to focus on risks that are most relevant at the current point in the development process. Scrutinizing these two factors will also help the WhiteBoard team develop strategies to handle the identified risks in the next stage of our risk management process.

**9.1.3. Risk Planning**

The WhiteBoard development team will develop strategies to handle identified risks should they occur in this stage.  This will be accomplished through research of the risks identified and discussion of possible solutions within the group.  The team will develop solutions based on three levels: avoidance strategies, minimization strategies, or contingency plans.  The best case scenario will be to have strategies to completely avoid the risks, but this is difficult to achieve.  The next level involves minimizing the damage caused by a risk event if it is unavoidable.  The final level involves contingency planning for risks that are unavoidable and not able to be minimized.  The development team will work together to develop these strategies.

**9.1.4 Risk Monitoring**

The development team will continuously be monitoring known risks.  Updates to risks will be evaluated each week as a group effort.  If the risk probability of occurrence or impact of effect changes the priority then the change will be reflected in our prioritized list of risks created in the risk analysis stage.  Risk monitoring closes the loop in our risk management process, as this step will repeatedly send the development team back to the risk analysis stage as risks change over the course of the project.

**9.2. Technology Risks**

One technology risk that presents itself is the failure of the Raspberry Pi in which we store project information. The probability of a hardware failure is minimized by our Raspberry Pi being attached to a reliable power supply and internet connection, so our development team ranks this as a low significance risk.  However, progress could be easily lost in the case of a failure with the Raspberry Pi, and the effects would be serious. To prevent large setbacks the project will be backed up on our GitHub repository at frequent intervals.

**9.3. People Risks**

People-associated risks that could prove to be an issue to our project are such that the possibility of team members dropping the class or otherwise rendered unable to work on the project through other means like sickness of of extenuating circumstances. These risks are low as the course is past the add- or drop- deadline on Western Kentucky University’s calendar for this academic semester, however they must still be kept in mind. Our remedy to any potential issue with this is in our role assignments, where two team members have been designated to focus on frontend development and two team members for the backend; if one of the two drops, our team member focusing on documentation of our software can fill in one of the positions.  If the team member focusing on documentation drops this position will easily be covered by the other four team members. Another risk is that certain team members may not put in as much time into the project as others. However, we’ve remedied this by having open communication with one another as well as honest peer evaluations so that each team member can see ways that they can grow in their contribution to the project. On the other hand some students who work on the project for the same amount of time each week find it time where their schedules overlap so that they can actually work together and collaborate.

For each of our group members software engineering is a learning experience not one person or even a couple individuals knows all that is needed to know to complete the project. So another huge risk to the project's completion and success is knowledge. Members may lack the knowledge needed to complete some complex tasks or solve out of the box problems. While each of us does go to class each week, it will take each of us doing personal enrichment and research on our own to solve this particular problem.

**9.4. Organizational Risks**

Organizational risks that could arise are unique in this project.  Since the WhiteBoard development team is not a professional development team we do not have a project manager working above the development team.  However, we do have a professor and university administrators.  An organizational risk would be if an unforeseen circumstance caused the professor or administrators to change the due dates for a report or milestone. Changing deadlines is a common risk in the corporate environment.  However, academic environments are more stable and the development team believes the probability of this risk is insignificant. However, progress on reports and presentations will be started as soon as possible to avoid being caught by any abrupt schedule changes.

**9.5. Requirements Risks**

Our requirements risk is such that the group could hypothetically develop the project in a converging manner from the requirements established by our client. This is a low-probability risk due to the constant communication we will maintain with our client through e-mails and physical meetings.  Also, our comprehensive feasibility report should clearly state the development team’s intentions for the project.  At this time, the client should be able to distinguish any discrepancies between the development team’s expectations and the expectations of the client.

**9.6. Estimation Risks**

There is no leniency in our time budget, as the project must be completed by the end of the academic semester at Western Kentucky University - there cannot be any extensions of deadlines. So, one risk lies in the potential incompletion of our project. The best way we can hope to negate this risk is to accurately estimate the amount of time needed to develop our project and then adjust our schedules accordingly.  Another precaution the development team is taking to ensure the successful completion of the project within the time limit is the abandonment of any unnecessary features that may take too much time to complete We must remember that Whiteboard will not be as scalable as Blackboard so to replicate everyone of Blackboard’s features and improve upon them may be a daunting task to complete.  Instead of shoving a large quantity of low quality features into one application. our team will focus on ensuring the highest quality and function in the features we will complete.

**9.7. Tools Risks**

The WhiteBoard development team is using several frameworks and support software to complete this project.  This opens up the possibility for several risks concerning tools.

**9.7.1. Failure to Learn Tools**

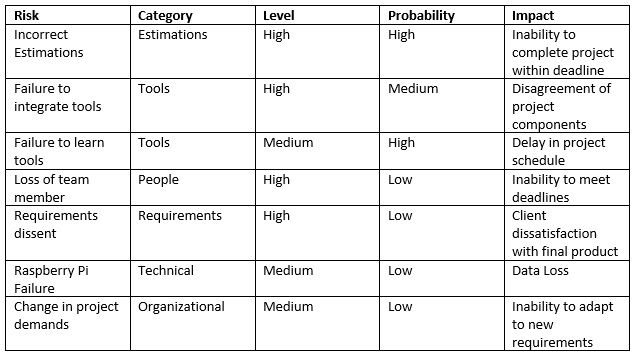
The risk that members of the development team will encounter issues when learning to use these tools is of moderate probability considering the inexperience of the development team. For most of the team the concept of the raspberry pi is completely new, some of the team members have also never used some of the repositories online like GitHub or had a real time coding solution like cloud9. Each week we may use some of these technologies and unfortunately taking the time to learn the tools does not have a direct impact on the speed of the project initially. Even if all team members do manage to learn how to use each of the tools that we’ve discussed on our own and researched together, the time that could potentially be wasted could prove to be a blow to our progress. The impact of this risk would be to the development team’s schedule for completion of the project.  The deadline for project completion is already very short, so delays due to a misunderstanding of unfamiliar development tools poses a serious level of impact.

**9.7.2. Failure to Integrate Tools**

Another possible risk involving the use of development tools is failing to integrate the different frameworks and support softwares together correctly. The development team is currently using Spring, Maven, Thymeleaf, and Bootstrap to assist in project development.  The risk of not integrating these technologies together successfully is of high probability considering the inexperience of the development team.  The effects of such an event could include the frontend interfaces being unable to communicate with the backend databases.  This level of impact is definitely catastrophic, as the success of the project depends upon successful communication between the frontend and backend.

**9.8. Prioritized Risks**

In any development project it is necessary to assess and monitor risks throughout the course of the project.  An excellent way to do this is to create a prioritized list of risks and update this list as risks develop and change over time.  The WhiteBoard development team chose to create a table to represent the prioritized list of risks associated with the development project to provide an easy to read summary of these risks.  The risks within the table have been prioritized based on two factors: probability and impact.  These factors determine how detrimental a risk could be to the project.  Probability is how likely the risk will occur, and impact is how much damage an event would cause should it occur.  Prioritizing risks based on these factors allows the development team to accurately monitor the most dangerous risks.  As these factors are susceptible to change the development team will continually update this table as the project progresses.



**Figure 22: Prioritized Risk Table**

**10. Progress**

As the WhiteBoard project continues all progress will be described in this section.  Progress will be tracked for the duration of the project and documented in every report.

**10.1. Initial**

The WhiteBoard development team completed research in several different areas.  This research focused on the different frameworks that are being incorporated into the WhiteBoard project.  The frameworks being used in the project include Spring and Bootstrap.  The team also investigated Thymeleaf, a java template engine that will assist in development of the user interfaces. Approximately fifteen hours was devoted to the research of new technologies that would expedite the development process.

Another initial activity was the setup of the development environment.  This consisted of acquiring our Raspberry Pi and installing the operating system onto it.  After that was completed the Raspberry Pi was connected to the internet, and can be accessed remotely by the development team. Setting up the Raspberry Pi included installing the Docker container management system. This will enable WhiteBoard to be modularized into independent containers for better efficiency.

The final initial activity completed by the development team was the feasibility study.  This study included a broad outline of the requirements of the project, proposed ideas for achievement of these requirements, and a basic risk analysis. The purpose of this study was to ensure agreement between the development team’s plan and the client’s needs.

As will be described in the following subsections, once the team planned as much as is possible given our knowledge base of various technologies and the project’s requirements, development began on various components of the project. This method of planning entirely before moving onto implementation falls into our Sashimi method of software development which we are following.

**10.1.1 Back End**

After setup of the Raspberry Pi, more extensive backend development could ensue. This went into a variety of directions, including the setup of Docker containers for use in deploying WhiteBoard. Specifically, containers have now been installed and configured for MySQL and PhpMyAdmin. Official images for the two technologies are unavailable, so ports of the images compatible with the ARM (Raspberry Pi) architecture were used. The command used for the MySQL container is:

docker run --name mysql -d -p 3306:3306 -v /home/pi/.local/share/mysql:/var/lib/mysql -e MYSQL\_ROOT\_PASSWORD=WhiteBoard tobi312/rpi-mysql

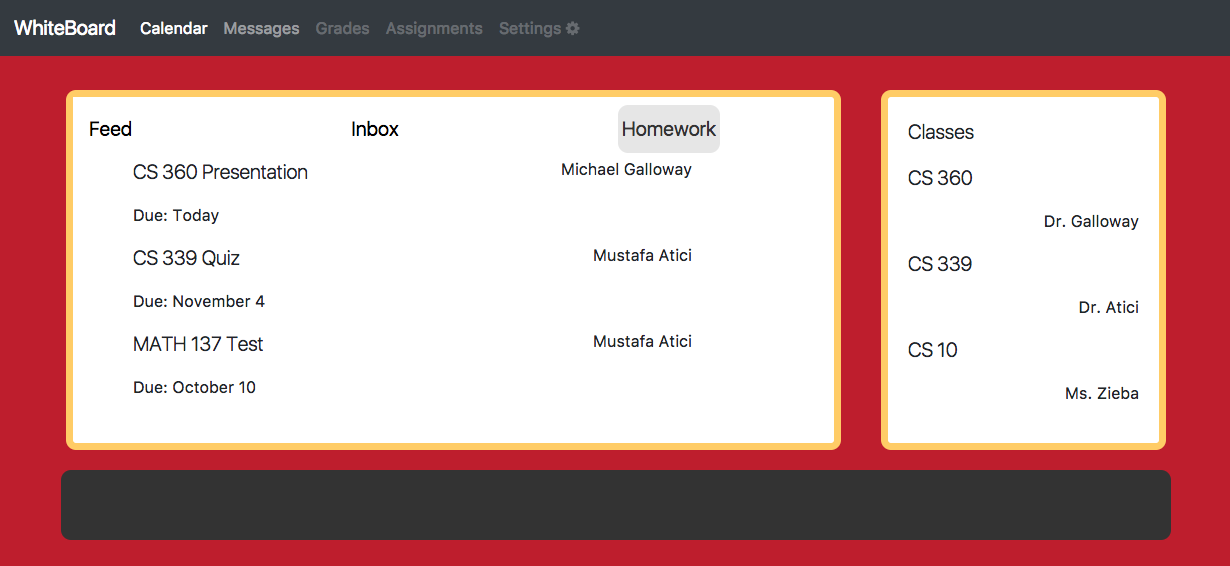
Next, the PhpMyAdmin docker container setup:

docker run --name phpmyadmin -d --link mysql:db -p 8080:80 ebspace/armhf-phpmyadmin

Code has been uploaded and shared through both GitHub and Cloud9 throughout the duration of the project. In addition to the interface described below, an initial and functional administrative interface has undergone construction. The administrative interface features are such that administrators can manage student and instructor accounts with the ability to create, delete, or update accounts. As part of this, following development came in the form of a functional staff creation page. This staff creation page fulfills the requirement established by our client of account creation, as the page pushes information to our database. As part of this progress, our team discovered that we wanted to revise our database, and that the studentLogin and staffLogin tables would no longer be necessary. For ramped up security in our database, hashing of passwords will now occur. This means that, instead of storing plaintext passwords in our database, we only store the hashes of passwords to minimize danger in the event of a database breach.

**10.1.2. Front End**

Frontend development so far has come in the form of the creation of the home web interface, which includes functionality tying to the inbox, announcements, homework, and course loading features. However, the next step that will need to be taken is tying together the database to allow for login features and therefore genuine functionality of various WhiteBoard components. The web interface thus far has been accomplished using Bootstrap components, primarily in the form of the grid layout to create visually-pleasing web pages that are responsive on a variety of platforms. Aside from Bootstrap components, completely original HTML and CSS has been created to accomplish our vision of whiteboards displaying various features. The home interface contains a navbar that will be functional once links to other pages are created. Aforementioned functionality with the inbox, announcements, and homework has been accomplished primarily by usage of the <iframe> HTML tag, as shown here:

****

**Figure 23. Initial Mockup of home web interface**

<iframe src="announcements.html" seamless='seamless'></iframe>

This tag is particularly important in solving our initial goals of displaying all features by loading in web pages that could be toggled through in the home page. The seamless attribute is set to avoid what would otherwise be a default border around the iframe content.

So, initial mockups have been created of messages, announcements, and homework which can be accessed by toggling another navbar built into the larger of the two “whiteboards” that make up the home interface. Design differences between our initial mockup and actual interface vary extremely little - the actual interface looks more visually pleasing thanks to the use of Bootstrap as well as the use of Western Kentucky University website colors. Of course, planned functionality will be such that the website colors will be displayed a custom value as requested by institutional administrators, but for now the page will remain static red, white, and black colors. Another aspect of the home web interface that has been created has been the successful incorporation of Font Awesome icons. Setup of Font Awesome icons is simple; installation is a simple download, and inserting the icon stylesheet is as simple as including this line:

<link rel="stylesheet" href="font-awesome-4.7.0/css/font-awesome.min.css">

As expected, the icons make the user experience much more visually pleasing, and did not take excessive amount of time. Current uses of Font Awesome icons are, as planned, used to differentiate between different types of notifications, and are also used to show whether planned messages are opened or unopened.

Following the above progress, the prototype website has been improved upon, and is by no means a throwaway prototype; instead, this initial interface will serve as an example of operational prototyping, where new features are built upon a foundation that will remain. The next phase in frontend development was in the form of fleshing out initial ideas as well as throwing around new implementation thoughts as well as new features. The goals were to provide the backend development with some sort of aesthetically-pleasing interface to tie progress with for the goal of demonstrations as well as general direction with the project.

**10.1.3. Features**

As part of the setup and research involving the Spring framework, some specific features have begun to be mapped to controllers as well as various models. A pre-existing calendar has undergone initial modifications and will be implemented within the above web interface as well as mapped to the project database before the next milestone. The calendar code was initially created using Bootstrap, and most design components remain unchanged. Although no specific features are complete, general exploration and initial development of features has ensued.

**10.2.** **Second Milestone Progress**

The progress in this section details the prototyping and development progress since our second milestone deadline, in which we were able to present a basic prototype although unnecessary due to our Sashimi method of software development. For the third milestone, however, our goal has to be to create a functional prototype to demonstrate to our client, and the following sections lay out the plans and steps we took to develop a prototype showing basic functionality on all fronts.

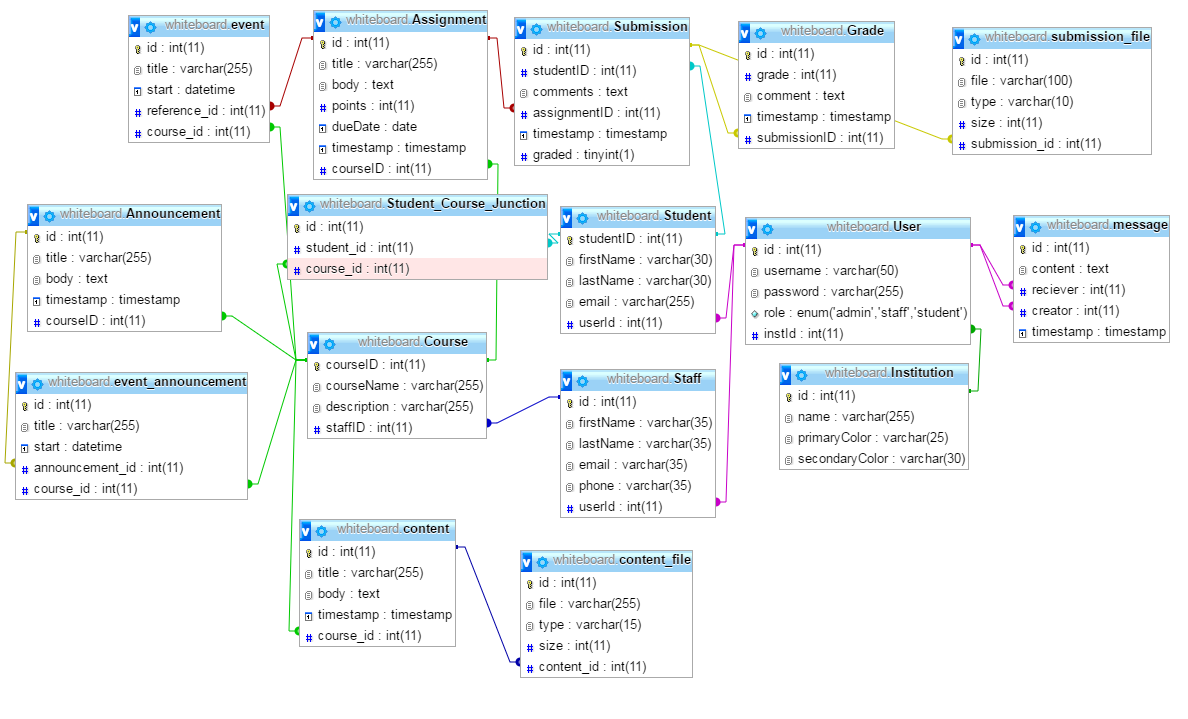
**10.2.1. Development Plan Changes**

Our original plan was to develop WhiteBoard using the Spring framework and Thymeleaf, which would involve using Java. However, we have made the decision as a group to switch to PHP due to time and development constraints. There many more resources available as regard to documentation in more commonly used frameworks like PHP. Team members developing backend components have experience in both Java and PHP, but the Java experience learned from the Western Kentucky University curriculum does not cover its use as it is needed to use the Spring framework. Therefore, all development on our project has been using PHP for the sake of producing a better product as a direct result of being able to spend more time developing and less time learning about implementing the Spring framework and accompanying Thymeleaf templating. However, our plans to use web technologies such as HTML remain unchanged.

Our initial plan for the database is found in our appendix, but once development and implementation using PHP began, the above database is what we found best suited our interests and needs with regards to backend development. A few specific differences between this diagram and our initial plan start with the change to the login system. Our previous diagram had separate login tables between students and instructors, but now we have discovered that both the student and instructor-specific login tables are unnecessary, as login information can adequately be stored using the User, Staff, and Student tables. Regarding the User table, it contrasts our initial database design where staff, students, and administrators had entirely separate tables describing their login credentials. So, Users are now assigned one of these three roles depending on their needs from the use of our system.

**10.2.2. Development Role Changes**

Although the number of people in our team remains unchanged, the roles initially planned have changed. Where two team members were going to develop the frontend and two were going to develop the backend while the final team member focused on documentation, development has been going in the form of one primary frontend developer and one primary backend developer, while the rest of the team has worked on writing extensive, high-quality documentation. This change has been more effective, as the frontend and backend developer can focus on using their previous experience to best create the product. Otherwise, as per the wisdom one may receive from reading The Mythical Man-Month, unnecessary time would be spent by developers teaching the other developers with no experience either on backend or frontend.

****

**Figure 23. Logical Database Schema, as implemented in our system**

**10.3 Third Milestone Progress**

**10.3.1. Raspberry Pi Location**

One change we have made that affects our development capabilities is a change in physical location of our server, the Raspberry Pi. One of the issues with our use of the Raspberry Pi as connected to the Western Kentucky University network intended for use by students was the inaccessibility of the Raspberry Pi from locations not a part of campus; this was a major issue in development because of the nature of progress often made from team members’ apartments unable to connect to the WKU network. So, the physical Raspberry Pi has been moved to one of the aforementioned team member’s apartments, and will remain there for the sake of our development capabilities.

**10.3.2. Frontend Design and Interface**

Our design cycle for the frontend interfaces are as such: we create non-functional mockups of components, which are then approved by the backend team, who can then see the requirements defined and connect backend functionality with the frontend components created. So, following the initial mockups, the next prototype that was developed was a mockup of our gradebook feature. Through developing our mockup of a gradebook, we soon realized features we would have previously not considered. For example, the feature of calculating what students need to score on upcoming assignments to raise their grade, if it is not where they want it to be. A tool used by the frontend design team to most effectively create interfaces is wireframe.cc, a bare-bones mockup tool that allows users to draw boxes and use text. This was used over draw.io for the sake of simplicity.

**10.3.3. Logo design**

Another aspect of design that is undergoing is development of website components as SVGs, and the first step in this has been initial planning of our logo. The logo is a minimalistic depiction of a whiteboard with a sinusoidal function scribbled on it, and represents a whiteboard and how it is functional in an educational setting. The creation of our logo as an SVG allows us to easily modify and animate the logo and extend the user interface experience as a result. We decided that a memorable logo would be the way for us to maintain the WhiteBoard brand even when school-specific colors are used.

**10.3.4. Frontend Administrative Interface**

One feature that could also be considered a part of the administrative features is undergoing development from the frontend interface development team, and involves the capability of administrators to set the school colors for all of their students and instructors to see. As one may have noticed for the above evolutionary prototype foundation in the form of our initial web interface, the colors were Western Kentucky University colors, but institutions of all kinds will prefer to use their school colors to maintain web design and branding standards. Therefore, allowing administrative staff to pick the colors they want to use as primary and secondary colors that spread across the various aspects of the web interfaces. This feature will be tied to the administrative interfaces developed by the backend development team once it is finalized; the feature is currently only stored across one webpage, so for the feature to truly work as initially intended, the initial prototype code must be refactored and linked to the database.

Shown in figure 24 is the initial prototype developed as a part of the administrative interface made by the front end development team. Our website maintains a relatively uniform look across the views rendered for our three user groups, but the final administrative interface will most definitely undergo change to scrap features the administrators do not use, such as the calendar and grade features. The next step in development that could tie front end development to backend was to create pages for administrative login and account management.

**10.3.5 Login pages**

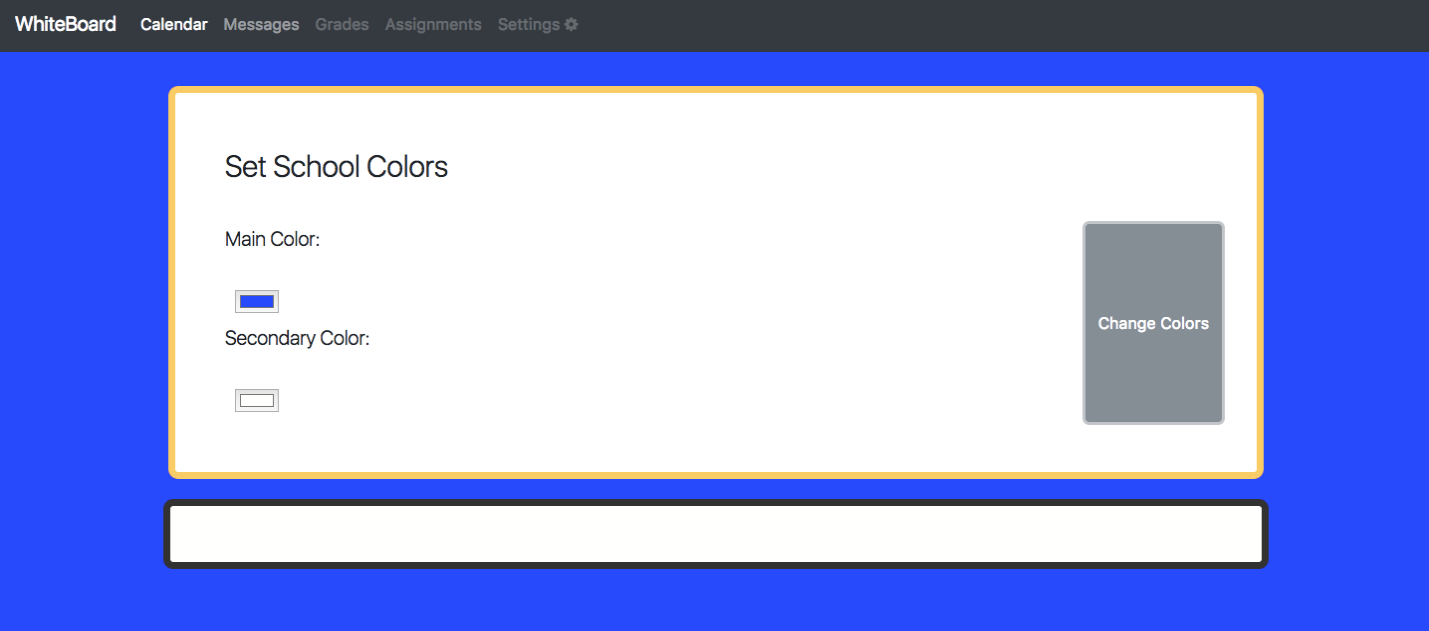
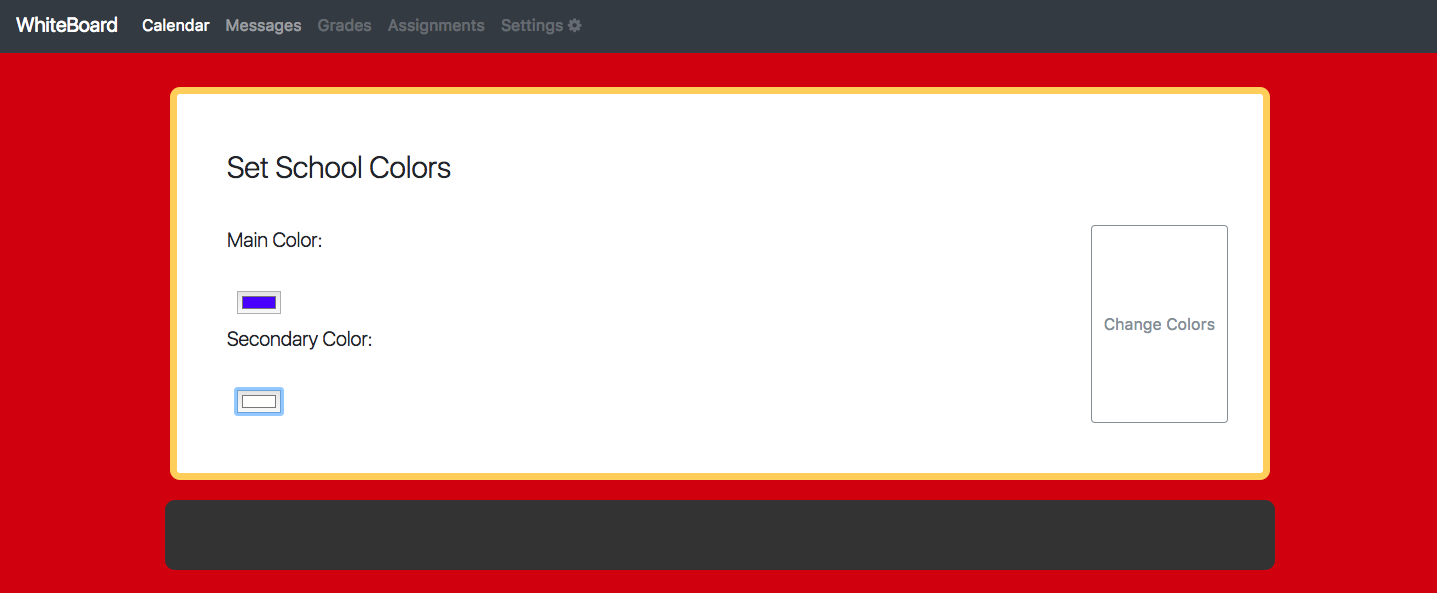
An obvious necessity for login functionality, the frontend team developed a mockup for login pages to be tied to backend development features. Some may ask why development is fully divided when the backend team could develop an interface and vice versa, but we are of the belief that this modularization of our team will result in the most productivity. So, the frontend team developed mockups of login pages with this goal in mind: be simple, and aesthetically pleasing. Our resolution to this was to simply overlay a Bootstrap login form on top of a high-resolution stock image for both student and administrative login.

**10.3.6. Landing page**

One interface that is not a requirement but that we believed would be a pleasant page for potential users is a landing page describing the functionality of WhiteBoard. This could, in theory, be especially valued if we decide to host our project elsewhere post-completion, and until then cleanly explain and visualize our features and ideas to our client.

**10.3.7. Addition of Javascript**

To extend the previous functionality earlier established for the second milestone, the next phase of fleshing out features. Even before connection to the backend development progress, the front-end development team progress came in the step of making web pages functional using Javascript to modify various HTML DOM elements. Whereas the previous initial interface was functional by using the <iframe> tag to generate mockups, the next phase in that development was converting this to Javascript to allow for it to be more compatible with the final system. Additionally, the implementation of Javascript was necessary to solve problems such as allowing administrators to change the colors of their customized WhiteBoard interface and make it feel more tied to their educational institutions.



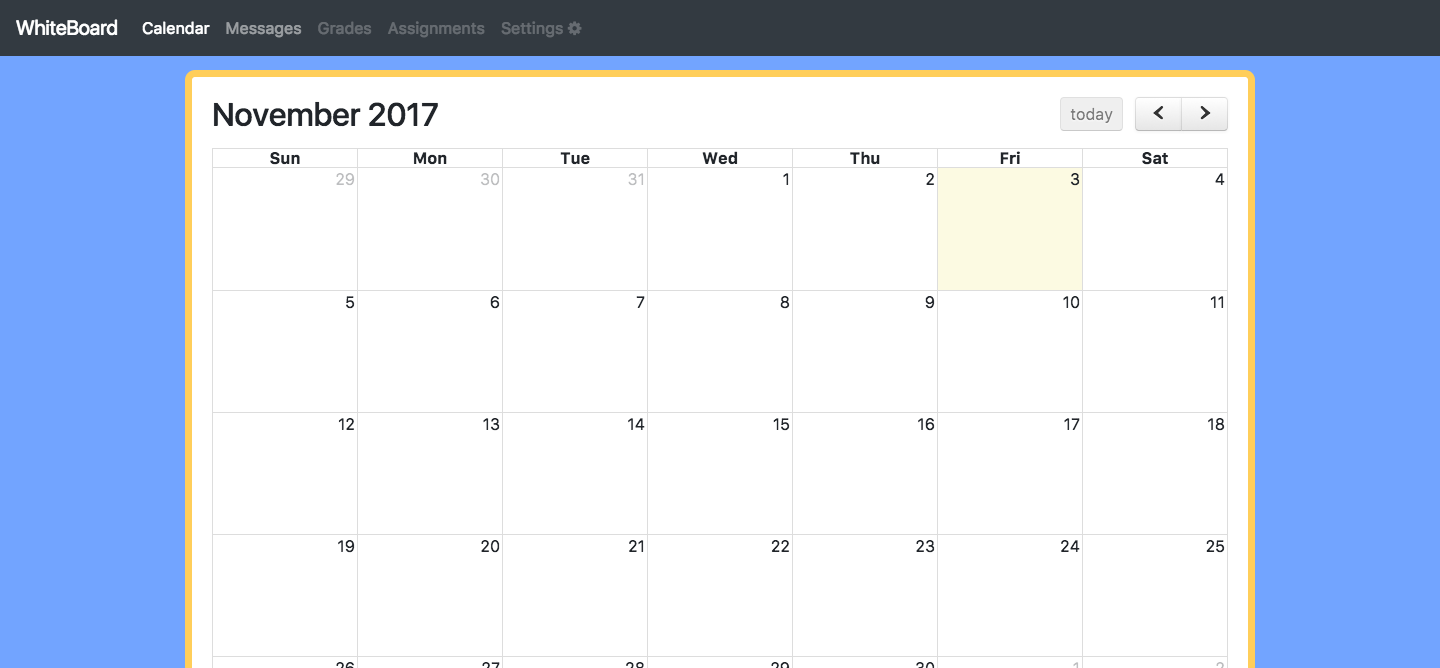
**Figure 24. Interfaces showing color changing feature**

**10.3.8. Calendar**

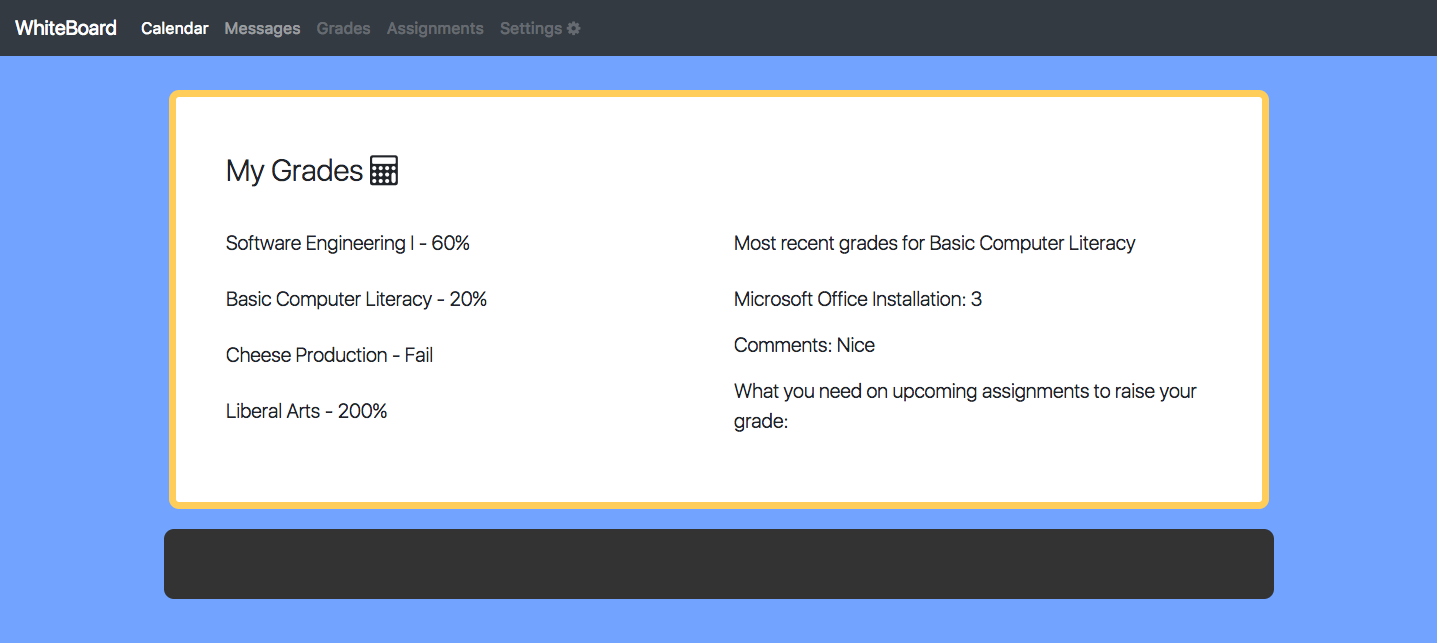
Calendar functionality has been successfully created by the front end development team, such that a calendar with no event functionality successfully displays, allowing users to toggle between months and see what the current day is. This was accomplished using the framework available through http://fullcalendar.io, and further development will involve tying it to the page that has also been created for the staff interface. This calendar will be used for both student and administrative views.

**10.3.9. Gradebook**

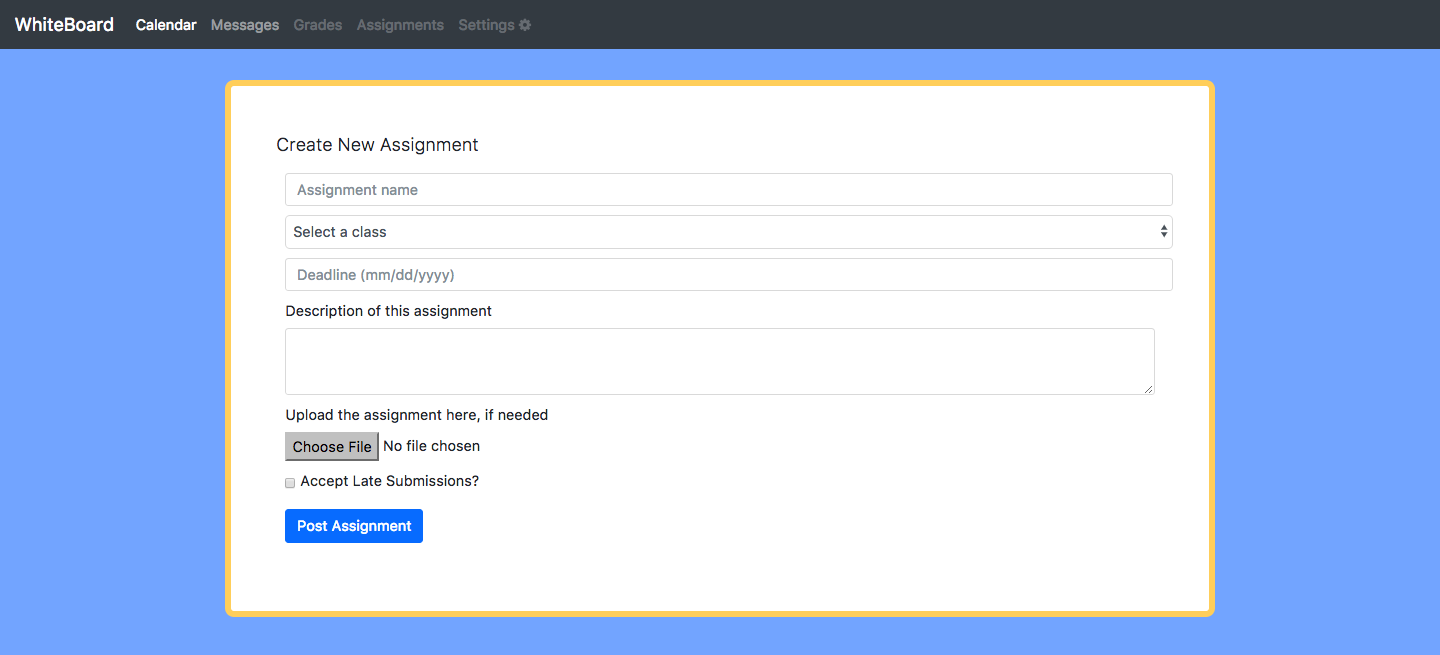
The gradebook shown above has been developed, and is a prototype for the final version. Unlike the previous <iframe> generated mockup, this gradebook uses Javascript, making it much easier to add PHP functionality to.



**Figure 25. Calendar Interface**



**Figure 26. Gradebook Interface**



**Figure 27. Staff Interface**

**10.3.10. Staff Interfaces**

While the staff also have a home page, it is very similar to the student interface previously created. This interface, however, is not, and is where staff can create a new assignment. As one can see, the fields in an assignment allow for a name, the class the assignment is due for, the deadline, description, uploaded/associated file if necessary, and whether or not late submissions of the assignment are allowed. This is currently nonfunctional, but will be tied to the calendar that’s also above by the backend development team.

**10.3.11. Backend Development**

As the backend development and frontend development teams work on their system components in parallel, both the backend and frontend development must be documented pre-combination. While the frontend development team created mockups and used Javascript to add functionality for certain features, the backend development team worked on a lower level of the system, configuring components such as docker containers and database-related features.

**10.3.12. Docker Containers**

The previous docker containers created on our Raspberry Pi were consolidated into a docker-compose, consisting of three containers. The three containers are a php-apache7 container, mySQL container, and phpMyAdmin container.

**10.3.13. Administrative Interface Development**

The administrative interface is a necessity for our project to satisfy the needs of one of our three user groups, which is educational institution administrative staff. So, the interface serves their needs of being able to generate and modify student and instructor accounts. Administrative features became as such: administrators could create student accounts, as well as modify or delete them. Additionally, the administrative interface was then given the feature of searching for student accounts to make modification or deletion an easier job.

The next step in our interface development was to complete the administrative interface features from a backend standpoint. So, administrators were able to successfully complete all core tasks requested by our client - using our interface, an administrator can now create, update, and delete student, instructor, and administrative accounts, and from the backend standpoint, the administrative interfacing is complete. The next step in true completion would be tying the functionality to the interface the frontend team made.

So, this next step was taken, and the functionality was united between the two development teams. Features of the administrative interface as described above were also added onto such that the administrative interface could display metrics of users.

**10.3.14. Login Page**

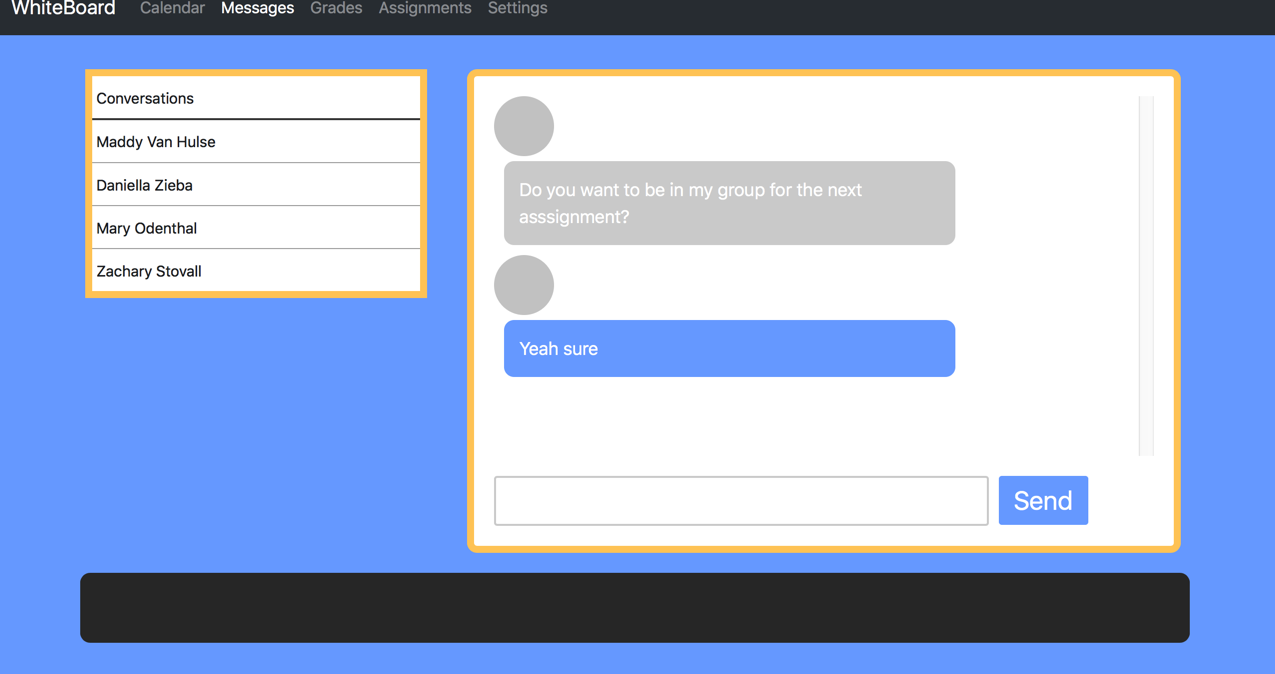
Next, development began for general login features. As one could imagine and assume, a necessity for login features is development of a login page. Our implementation for the login page functionality uses sessions to store variables.

**10.3.15. PHP functionality**

Another vector of backend development following initial development of the nonfunctional student interface was the addition of PHP functionality. Development began with testing the successful display of the student announcement feed using queries for information from our database, and then followed into display of both the announcement and homework feeds successfully.

**10.3.16. Messaging System**

The current mock up for the messaging system’s interface is as follows:



**Figure 28. Messaging Interface**

During this milestone we have created an interface for the messaging system using HTML and CSS. The current interface has two separate main components. One displays a table of the the users list of contacts and the other displays the current conversation they are viewing. Throughout the next milestone we will be adding functionality to this code and tying it to the database, as well as adding Javascript functionality. Once this is completed, the user should be able to click on different contacts in their list and pull up conversations with that user. One consideration that was taken in development with the messaging system explored the possibility of interfacing with the Twilio API for SMS messaging functionality, however this was scrapped in favor of an internal messaging system we concluded would better enhance the user experience and increase security that could be weakened by the use of an external framework.

**10.4** **Milestone 4 Progress**

This section documents the progress made following the third milestone in preparation for the final delivery of our product to the client. So, this work primarily builds off of the previous project made by tying together project components and adding functionality to various web interfaces.

**10.4.1. Frontend development**

Frontend development involved cleaning up various interfaces as well as aiding in the roles that were developed more extensively by the backend development team during this milestone.

**10.4.2. Backend development**

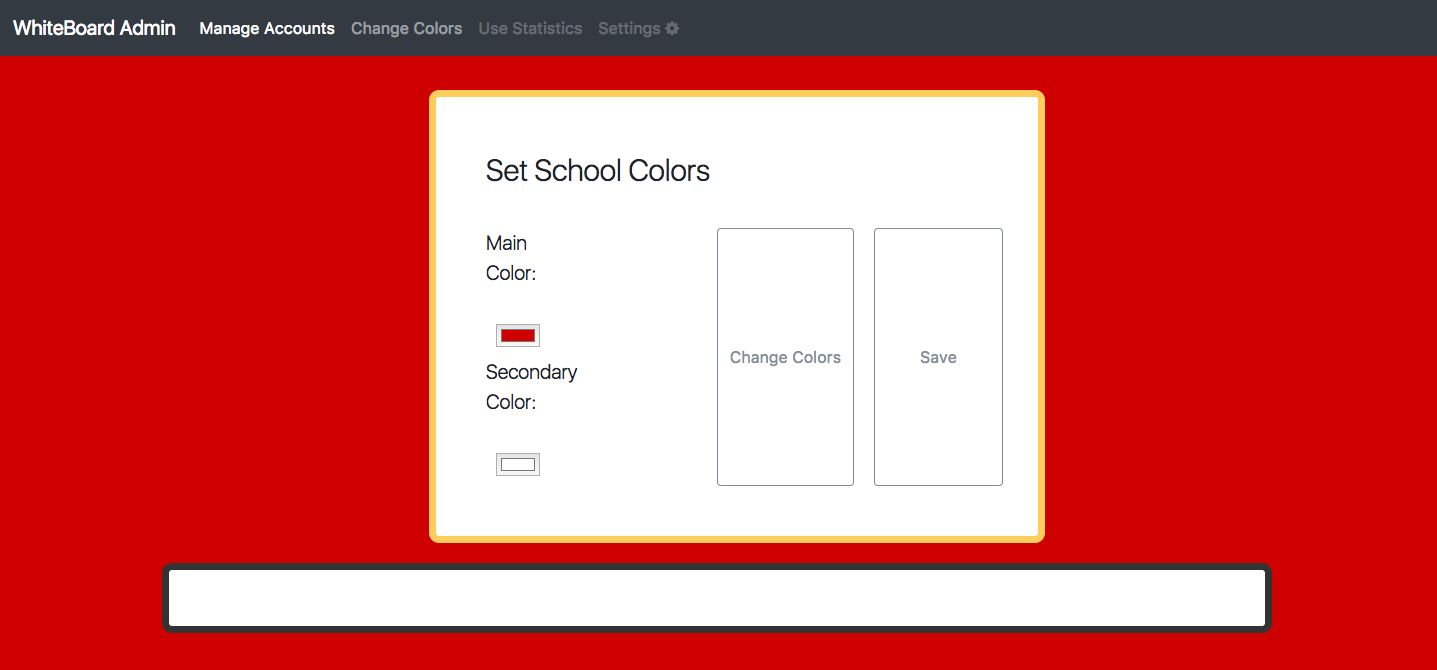
Backend development for this milestone continued initially with development and added functionality to the mockup interfaces completed by the front end development team. The student interface functionality was previously added, but next came progress on other interfaces as described in the sections below.

**10.4.2.2. Staff interface functionality**

The non-functional mockup of the staff interface was developed by the front end development team in the previous milestone. So, a goal for the backend development team for this milestone was to add functionality to the interface, connecting it successfully to components such as the database and login system. So, the feed and assignment tabs became dynamic; the inbox-related components had to wait for the completion of the messaging system, however. So, other progress relating to adding staff interface functionality moved to allowing staff to successfully create and post assignments for their particular classes to view. The previously-created assignment page became functional through the use of AJAX requests.

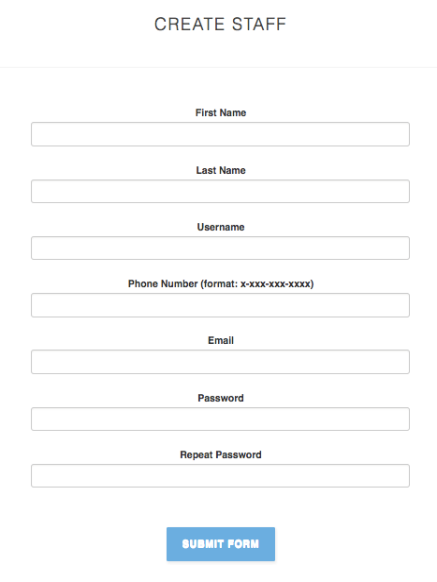
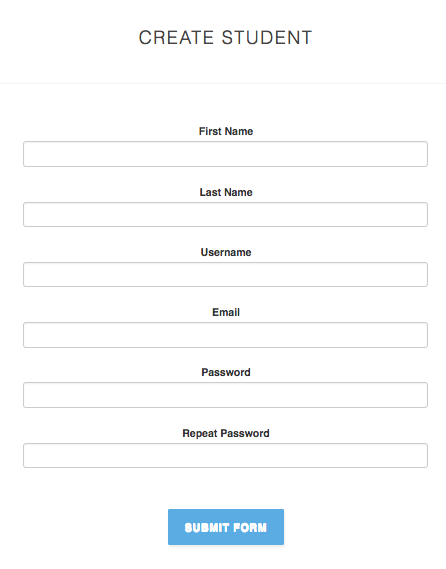
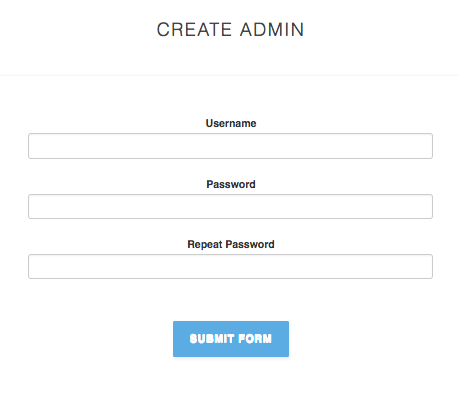
**10.4.3. Finalization of the web administrative interface**

The web-based administrative interface was developed primarily during the previous milestone, but became final during this milestone. Shown below are some screenshots highlighting the final functionality of the interface.



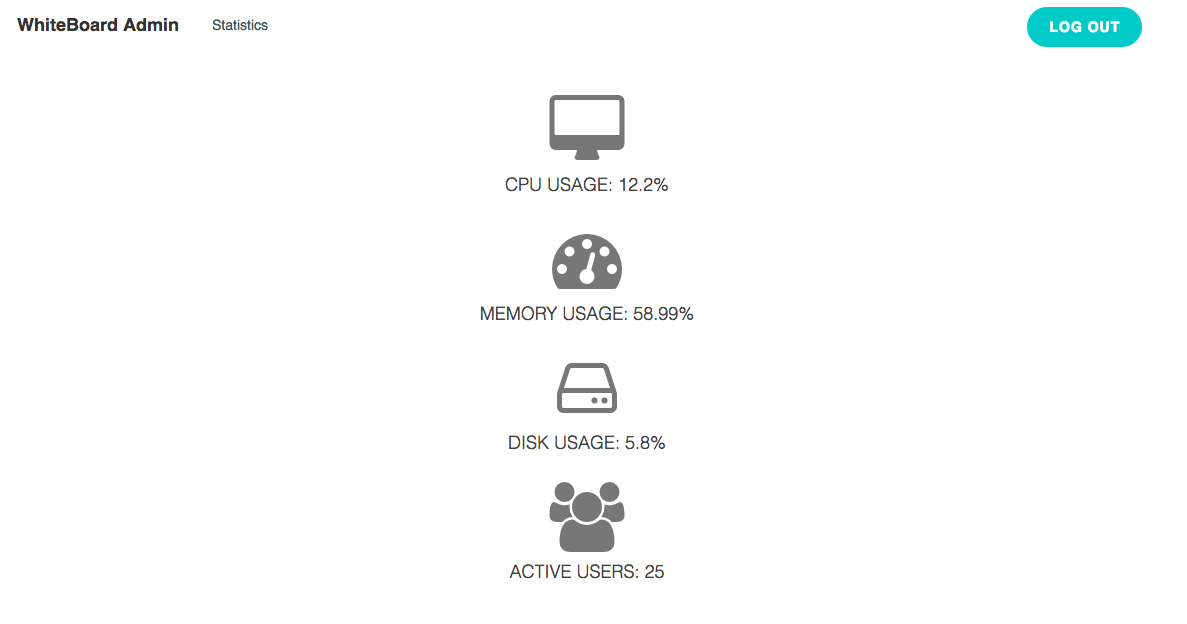
**Figure 29.Administrative Interface**

This interface is the final color-changing interface, allowing administrators to set the colors for all students and instructors for their institution to use.



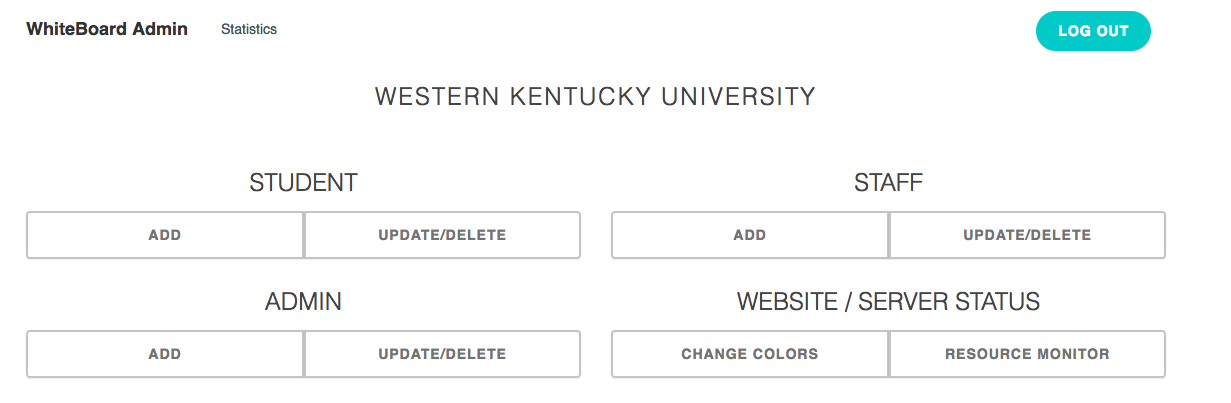
**Figure 30. Account Creation Interface**

The three interfaces above show the fields needed for creation of various accounts. Fields such as passwords are hidden and e-mails are validated.



**Figure 31. Resource Monitor**

Above is the interface showing the system resources used. In our version of WhiteBoard, this of course refers to the resources of the Raspberry Pi. In the future, the portability of WhiteBoard would ideally allow for this to accurately show resources of a server or network of servers with more power.



**Figure 32. Administrator Landing Page**

The above interface is the first interface administrators see when they login, and provides links to every administrative feature and tool. Student, staff, and admin accounts can be created, updated, or deleted, resources can be monitors, and colors can be changed. Overall, we believe that the minimalistic interface is a good design decision for making navigability as simple as possible and keeping the interface neutral institution-wise.

**11. Performance**

Performance engineering is essential for ensuring high system performance on the final product, efficient use of system resources, avoidance of product failure, and reduced maintenance costs. Our web interface should be able to process user requests in near real time so that users are not constantly waiting after each interaction with Whiteboard.

**11.1. Performance Requirements**

Our performance requirements as stated by our client are currently as such: our project must be scalable, defined by its ability to be ported to more powerful machines than our current server to successfully accommodate for a larger number of users. Further in development, we may either be given more requirements for our project to fulfill, and otherwise will define our own performance requirements for our project.

**11.1.1 Performance Engineering Objectives**

1. Eliminate possible system re-work due to performance issues.
2. Provide efficient use of system resources
3. Increase server capability
4. Identify possible/future bottlenecks
5. Eliminate system failure due to performance failure
6. Avoid additional hardware acquirement costs

**11.2 Possible Hardware Bottlenecks**

The CPU may present a bottleneck if it becomes overloaded, possibly from running at a high capacity for an extended period of time or from a long queue of processing activities. This may present an issue with our Raspberry Pi, since it doesn’t have a strong enough CPU to handle large numbers of users attempting multiple server requests at once.

A memory bottleneck could possible occur due to the lack of RAM available on the Raspberry Pi, which only has 1 Gigabyte. This would cause the CPU usage to lower and the device will experience a slow down.

Network issues may occur if the two devices communicating, the Raspberry Pi and the user’s device, lack the processing power or bandwidth necessary to complete tasks in an efficient manner. This can cause the server to be overloaded.

**11.3 Possible Software Bottlenecks**

Software language/frameworks could present possible bottlenecks. The functionality available varies and is limited by our use of css, html, and php. Any frameworks employed will also impose possible restrictions. Some programming languages provide different functionality and restrictions, so choosing the languages to be used is an important decision that could later cause major issues in the development of the project if the wrong one is chosen, because changing languages takes a lot of time and reworking of the system.

**11.4 Possible Solutions**

To fix performance issues with the CPU, it may be prudent to increase CPU power, add more RAM, or improve code efficiency. However, as we would like to avoid additional purchases we will likely attend to the last option.  The memory issue can really only be fixed by replacing the RAM with a higher capacity or faster one. Possible solutions for poor network performance include upgrading existing or adding new servers and upgrading network hardware. Network issues could be resolved by upgrading hardware components or paying for better connection. To fix issues with software bottlenecks in relation to chosen languages and frameworks, a proactive approach is best for solving any problems by ensuring issues do not occur from the beginning. By clearly defining and assessing the needs of the final product, the development team can choose the proper languages and frameworks they will need for an efficient product.

**11.5. Performance Benchmarking**

Although we can speculate about possible issues or concerns of performance, the only way for us to properly identify bottlenecks is in performance benchmarking of various subsystems. Detailed below will be our proposed process, followed by results and interpretation of our data.

**11.5.1. Benchmarking process**

There are various methods in which we can benchmark performance of various subsystems. The options we weighed are as such: we can measure performance through the terminal and bash commands or create timer functions of our own using the web browser. We opted to use both of these methods - the terminal is better suited towards testing backend-related aspects to WhiteBoard, while we can use the web browser to test specific functionality such as the .

**11.5.2. Sysbench**

Sysbench is our primary method of testing aspects of WhiteBoard such as the CPU capabilities of our Raspberry Pi, and is a multi-threaded, cross-platform tool that we will install on the Raspberry Pi and Docker containers. Installation is simple, and only requires this command:

sudo apt-get install sysbench

**11.5.3. CPU**

Our method of testing the CPU is by using sysbench, testing the multithreading capabilities of the CPU. The command used is:

sysbench -num-threads=X -test=cpu -cpu-max-prime=15000 run

The X represents the thread count and should be replaced with the values {1, 2, 4, 8, 16} and graphed. The results of this graph are what will determine the capabilities of our CPU. The graph of the test results shows the change in averages of the total time taken to complete the test (in milliseconds) for each number of thread counts tested.  The graph shows that with thread counts 1 and 2 the time is relatively high compared to the total time from tests 4 through 32, at which point the graph stabilizes around 80 milliseconds.  This is due to the current system operating a quad-core CPU.  The time is high due to only one or two cores being effectively used, and then once all four cores are active the productivity stays consistent.

**11.5.4. File I/O**

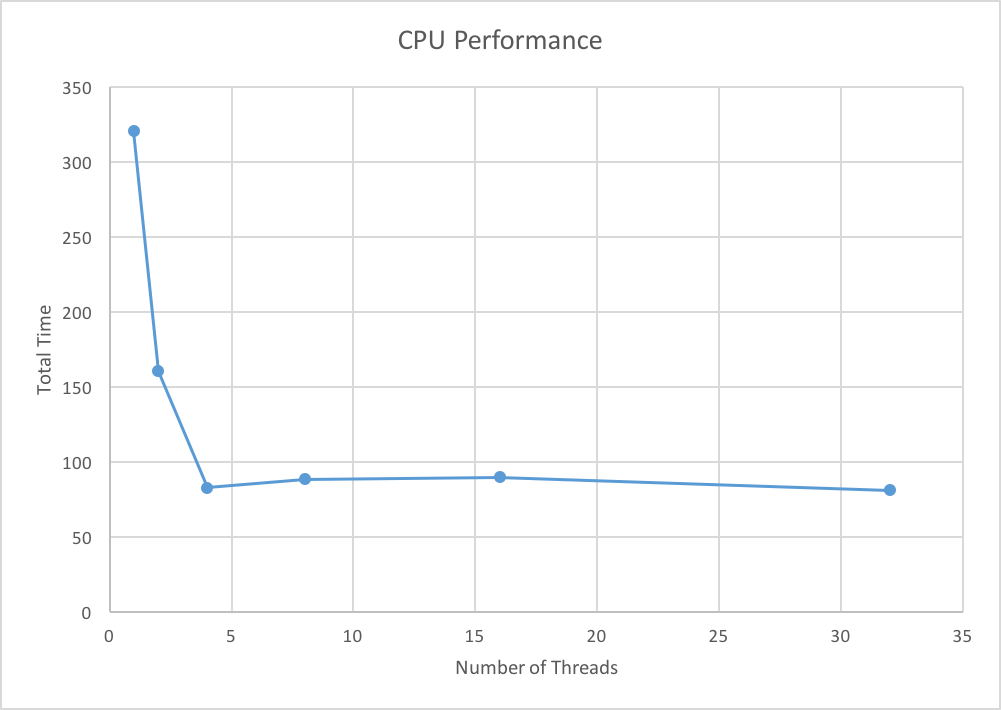
Our method of testing the File I/O is by using sysbench, testing the ability of the system to write and read basic information. The commands used are:

§sysbench --test=fileio --file-total-size=X prepare

§sysbench --test=fileio --file-total-size=X --file-test-mode=rndrw --init-rng=on --max-time=300 --max-requests=0 run

§sysbench --test=fileio --file-total-size

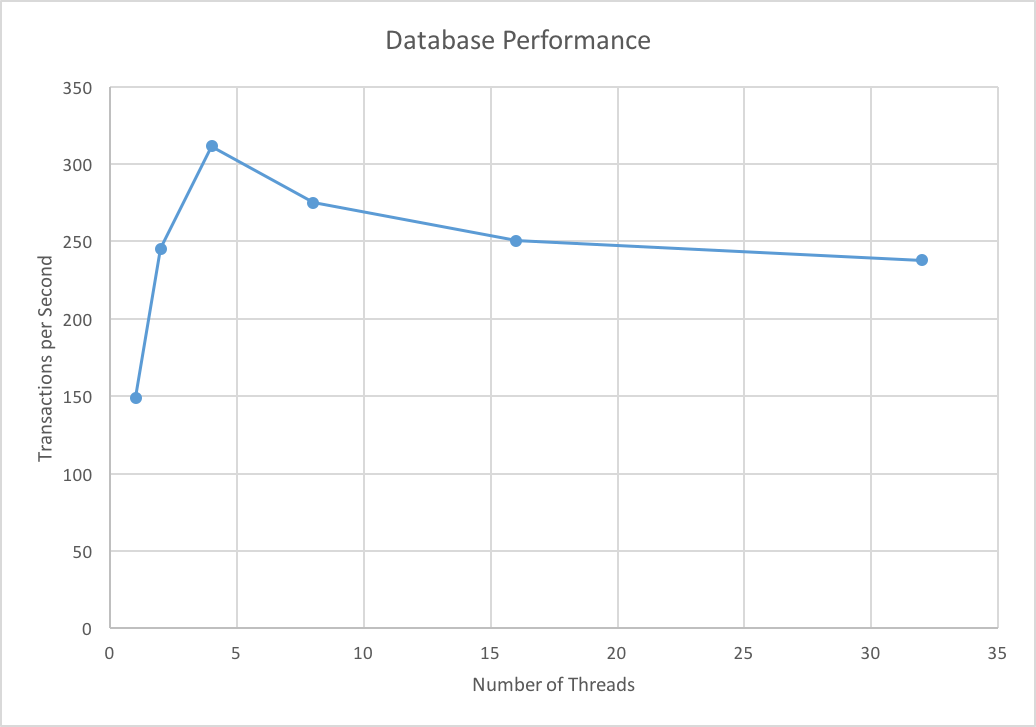
The X represents the file size and should be replaced with the values {128MB, 256MB, 512MB, 1GB, 2GB, 4GB, 8GB} and graphed. The results of this graph are what will determine the capabilities of the system to write and read data.  The graph displays the test results in megabits per second for each file size tested.  As we can see, peak performance was achieved with smaller files and as the size increased, the I/O performance became slower and leveled out.



**Figure 33. CPU Performance Graph**

****

**Figure 34. File I/O Performance Graph**

****

**Figure 35. Database Performance Graph**

**11.5.5. Database Performance**

Testing of the database was completed with the use of sysbench inside the docker container containing the mySQL container. Commands used were:

sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test --mysql-user=root --mysql-password=yourrootsqlpassword prepare

sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test --mysql-user=root --mysql-password=yourrootsqlpassword --max-time=60 --oltp-read-only=on --max-requests=0 --num-threads=X run

sysbench --test=oltp --mysql-db=test --mysql-user=root --mysql-password=yourrootsqlpassword cleanup

The X represents the number of threads used and should be replaced with the values {1, 2, 4, 8, 16, 32} and graphed. The results of this graph are what will determine the databases performance metrics.  The graph displays the test results in transactions per second for the number of threads tested.  Peak performance was achieved with a low-mid range number of threads - in this case four.

**11.6. System and Application Workloads**

This section documents system and application workloads relating to WhiteBoard, aiming to identify workloads and associated data as well as providing estimations by percent of application workload mixes based on our other performance-related tests.

**11.6.1. System Workload Mix**

The overall system workload mix is something viewable through the administrative component interface, interestingly - the capabilities of our web server, the Raspberry Pi, can be viewed at any point. Currently, these numbers are as such: CPU usage is 12.2%, memory usage is 58.9%, and disk usage is 5.8%.

**11.6.1. Application Component Workload Mix**

The application component workload mixes differ based on the component, and this section details the estimations we make regarding workload mixes and based on the performance benchmarking data gathered above.

**11.6.1.1. Administrative Component**

The administrative interface component has a very minimalistic frontend design, and as a result, this can be assumed to be low-cost with regards to the application workload mix. The most demanding aspects of the administrative interface relate to features themselves, and our estimations are as such: 10% of the workload can be attributed to the frontend interface, 20% can be attributed to the color-changing feature, 30% can be attributed to the user account modification, and the remaining 40% can be attributed to gathering system metrics.

**11.6.1.2. Student Component**

The student interface’s design is slightly more demanding than the administrative component as a result of the implementation of various web technologies documented in this report; as the student interface has access to the calendar, messaging, and gradebook/course related features, the workload of the student component can be broken down as such: 20% frontend interface, 30% messaging, 20% calendar, 20% gradebook, and 10% course information. Course information only requires one query from our database, and is therefore a less demanding aspect of our student interface.

**11.6.1.3. Instructor Component**

The instructor interface is very similar to that of the student, however is more demanding in actual functionality due to cases where students can merely view the information that instructors can modify. As a result, our workload estimations are as such: 10% frontend interface technologies, 20% course information, 20% messaging, 30% calendar, and 20% gradebook. We believe that the calendar is the most demanding workload-wise due to the implementation of the Fullcalendar framework to create this component.

**12. Human-Computer Interaction**

Developing user interfaces following the paradigms of HCI is a valuable component necessary to the success of the frontend development team. This is especially important to adhere to, as the later user interface changes are made, the more costly they are to development. Detailed below is the HCI-based research and planning related to our user interfaces, which involves analysis of the needs of our user base and careful mapping of features.

**12.1. Usability Requirements**

Although the ultimate testing of our user interfaces is reliant on user feedback through test cases we generate, we can first hypothesize on what would best serve our users. From there, we will then generate test cases, gather data, and refine our user interfaces. Methods we can use to form our aforementioned hypothesis include common sense as well as theory involving psychological analysis, human cognitive processing analysis, and a study into our user demographics and their tendencies and preferences in user interfaces.

**12.1.1. Satisfaction**

Our first requirement of usability is satisfaction, which can be quantifiably measured as effective functionality and qualitatively as a pleasant user experience.

**12.1.2. Robustness**

The robustness of our system is quantified as stability and low probability of user interface-related errors. Errors could be traced back to problems such as particular technologies and their incapability to function on certain hardware or software. For example, certain fonts used aren’t downloaded on certain operating systems, and formatting can be ruined. Which, as one can imagine, is very detrimental to the user experience. Or, certain HTML components could be too modern for many, older versions of browsers used on older computers. This is an issue that can only be resolved by testing on a diverse but realistic set of computers that may be used to access WhiteBoard by hypothetical students.

**12.1.3. Efficiency**

User interfaces must be efficient with resources both in the form of system resources and user-based resources such as time. System resources that could easily be taxed include RAM and CPU, and can easily be tied to user-based resources. For example, an extremely high-resolution image may look fantastic in development, but become a massive hassle in deployment where an excessive amount of memory is taken up on the limited resources of our Pi and images take significant time to load for our users, thereby hampering the pleasantry of their user experience.

**12.1.4. Learnability**

Learnability of WhiteBoard can be quantified as having a low learning curve for use of our interfaces. This is important for guaranteeing the success of first-year students whose grades in particular classes rely on their ability to learn how to use various web-based technologies, whether they are in the form of software such as WhiteBoard or are instead in the form of software more directly relating to homework assignments such as WebAssign.

**12.1.5. Memorability**

The user interface, in addition to having a low learning curve, must be memorable such that users will be able to retain their knowledge regarding user interfaces even with infrequent usage of our system. This could be important as per the user base defined below because students may use WhiteBoard for certain classes, but then in theory encounter a semester solely consisting of instructors who choose not to use a digital educational system to manage their classes.

**12.2. Users/stakeholders**

Users of WhiteBoard in theory are students of all disciplines, administrators, and instructors. While students of all disciplines and backgrounds are much more inclined towards technological literacy than more elderly administrators and instructors, an example such as Western Kentucky University proves that there can be a large amount of unconventional students typically defined as undergraduates older than twenty-five.

    The above description of our users is in theory, but in reality, the users testing our system are going to be well-versed in technology. Our client is a computer science undergraduate student, while our instructor is a computer science professor, and both are very technologically capable of navigating digital systems. So, an assumption we can make is that our user base will be used to navigation, and will not need as much of a tutorial as less experienced users may need. Even if the above theoretical user base of our system comes into play, our user interface should still be navigable, as everybody currently at Western Kentucky University is required to use Blackboard. Blackboard is an arguably more complicated user interface, and as any student or instructor can find, has many unnecessary features that are not used by anyone.

**12.3. Test Cases**

Our test cases can be generated to adhere to the user interface requirements documented in section 12.2. above. Although we can quantitatively conclude metrics such as providing users with all base functionality planned for WhiteBoard, our problem then becomes how we can establish experiments and conclude the effectiveness of our ideas.

One quantifiable metric of success, and a means of quantifying qualitative data, could be the measuring of WhiteBoard’s user experience against Blackboard, the current software required of all Western Kentucky University students to use. Feedback can be given to us in survey form, as well as the time needed for first-time users to navigate their ways to components of Blackboard compared to the time needed to navigate to the same components in WhiteBoard. Comparable components include aspects WhiteBoard and Blackboard share, such as the gradebook and calendar for students and instructors. Administrative interface success may be more difficult for us to conclude due to the lack of our access to such an interface in Blackboard, but we can compare to existing administrative interfaces for other popular applications for which administrators may need the same requirements and measurements of various users and usages as our project does.

**13. Software Testing Strategies**

This section of the report documents analysis and testing relating to the success of our final product. It includes specific techniques used in development and our reasoning and associated methodologies for such techniques.

**13.1. Strategic Approach**

To successfully test our software, our strategic approach is as such: we must test to ensure that functionality meets all of the requirements we previously designed to cater to the needs of our client. Next, we must ensure that WhiteBoard responds correctly to all types of input, and can perform in a reasonable amount of time. Another requirement is that it can be installed or replicated in its intended environment, and that WhiteBoard must accomplish desired results as stated by the client. Our methodology of determining tests began with the conducting of formal, technical reviews of our product. Generally, best practices of testing then dictated that our testing begin at the component level, followed by integration testing. Software testing will only end when our project is required to be delivered to our client at the end of this semester.

**13.2. Verification and Validation**

Two things we will need to do to ensure WhiteBoard’s success as a deliverable to the client are software verification and validation. Verification entails ensuring algorithms work as intended, so our methods of testing must involve generating various test cases. For example, verification throughout development involved testing input fields such as e-mails needed for our login system to ensure that only the correct input would be able to register within the system. Validation, meanwhile, must make sure that the client’s requirements are met. For us, this entailed ensuring the capabilities of our gradebook, account, messaging, and calendar systems, all core requirements for our project.

**13.3. Unit Testing**

Unit testing focused on testing the functionality of various methods in our code, and listed below are the results of method testing in various system components broken down into as much detail as is possible and/or reasonable.

**13.3.1. Component Testing**

**13.3.1.1. Messaging Component Testing**

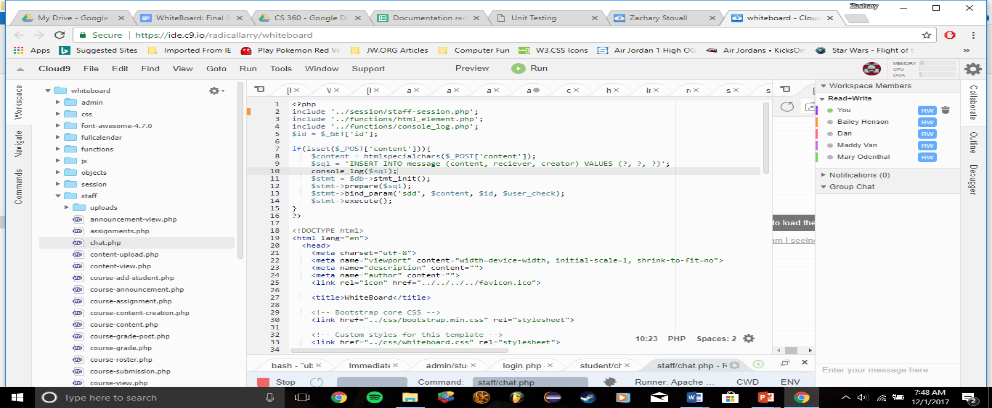
The tests executed for our messaging component will not change based on the type of user (student vs. instructor), and are therefore all documented in this section. Our unit test taught us a lot about the strengths of our messaging system. For example when our messenger inbox gets a message that is extremely long instead of crashing our code it just prints as much as space will allow. Also our messenger prevents users from submitting black messages, requiring them to fill out the submit field before trying to send to another user. We were able to communicate seamlessly between two student accounts. As mentioned the functionality of the messaging system is the same between students and instructors so we did not have test both components.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID # | Test Case | Input Value | Expected output | Success |

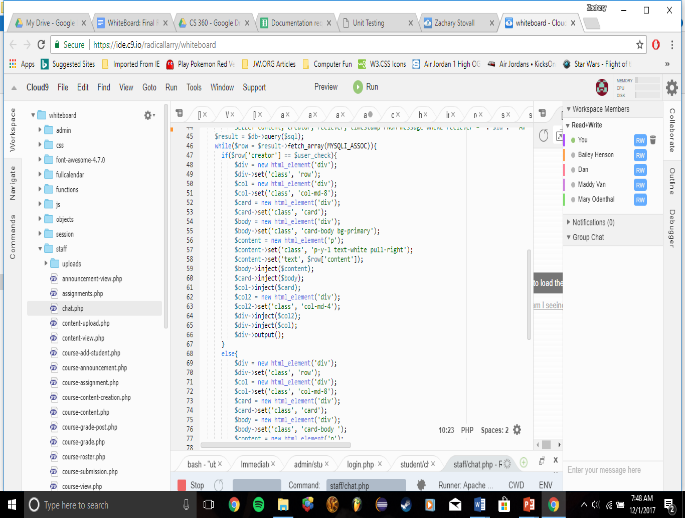
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 111 | Messenger Function Unit Implementation Test  -          Basic Message Sent | Receiver: Bailey Henson  Sender: Zachary  Message:  “Testing 1,2,3,” | BaileyScreen- “Testing 1,2,3” (Receiving)  Zachary Screen-  “Testing 1,2,3,”  (Sending) | Yes  BaileyScreen- “Testing 1,2,3” (Receiving)  Zachary Screen-  “Testing 1,2,3,”  (Sending) |
| 101 | Messenger Function Unit Test Implementation  -          Try to overflow the Buffer | Receiver: Bailey Henson  Sender: Zachary Stovall  Message: (Held down the “j” key for 3 min straight. | Expected some error message of some kind. | Successfully printed some of the “j”s and cut some of them off. |
| 110 | Messenger  Function Unit Test Implementation | Receiver: Bailey Henson  Sender: Zachary Stovall  Message: (Blank Message) | Please fill out this field | Yes  Please fill out this field |

**13.3.1.1.1. Messaging Component Flow Graph**

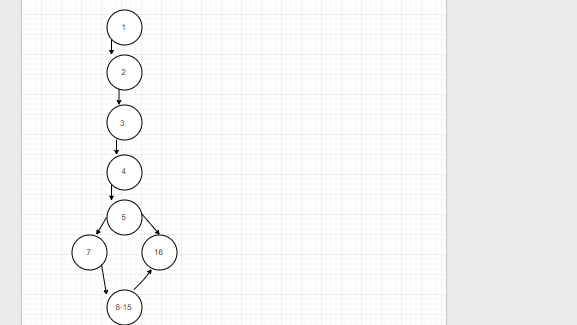
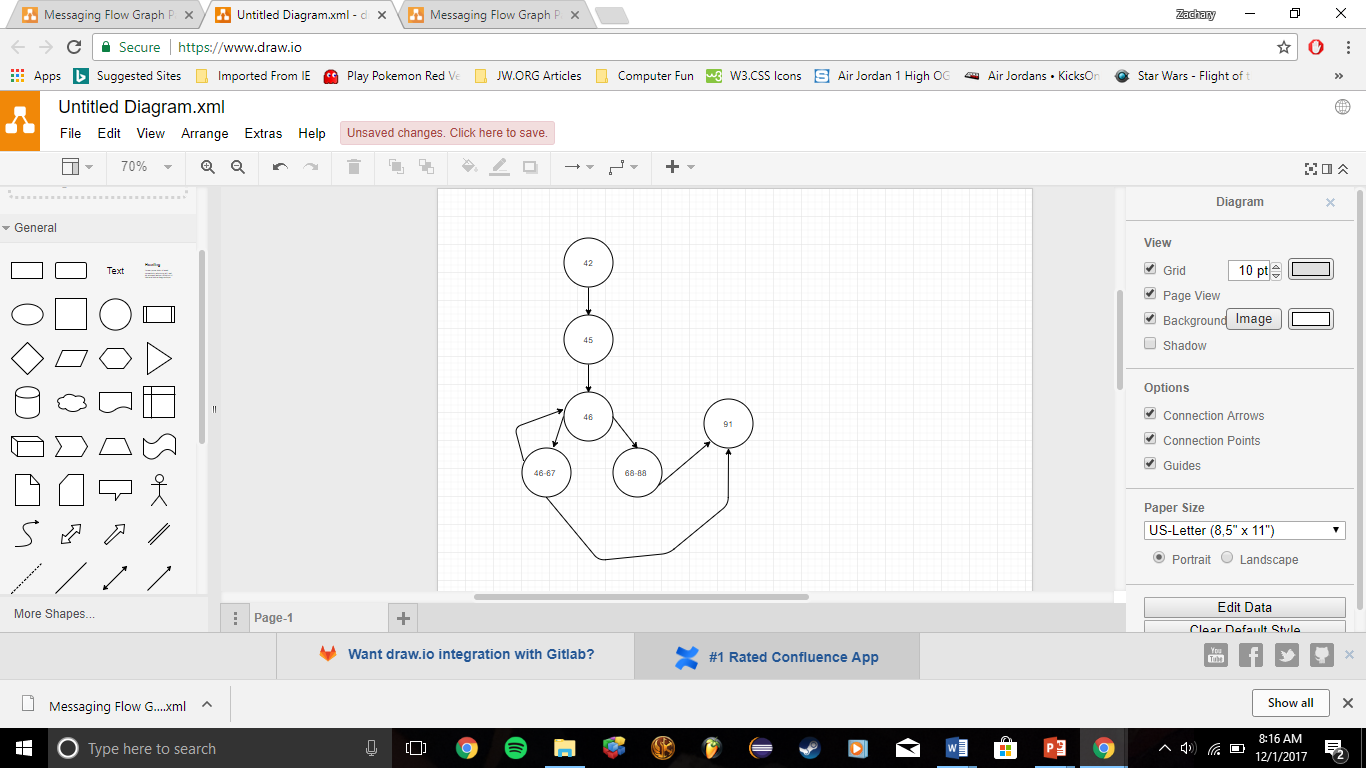
Our  messanging system is very detailed and has several components so we tried our best to summarize the two most crucial pieces of our php code in the following flow charts. We’ve also posted their respective code samples directly below.



**Figure 36. Messaging Source Code**



**Figure 37. Messaging Source Code**

****

**Figure 38. Message Flow Graph**

**13.3.1.2. Calendar Component Testing**

The calendar component test cases, unlike the messaging component test cases, must differentiate between student and instructor views. This is because of our core design and implementation of calendar functionality where students can view assignments but should not be able to modify them, while instructors are able to post and modify assignments they have created.

What we learned from our Calender unit tests caused us to slightly modify our code.  The biggest problem we found is that a instructor did not have to fill in all the fields in order to successfully submit an assignment. In fact not only did our calendar form allow submission without the title field it allowed submission it allowed submission without any of the fields filled out. We immediately fixed this bug once we found it to save memory and provide more seamless functionality.

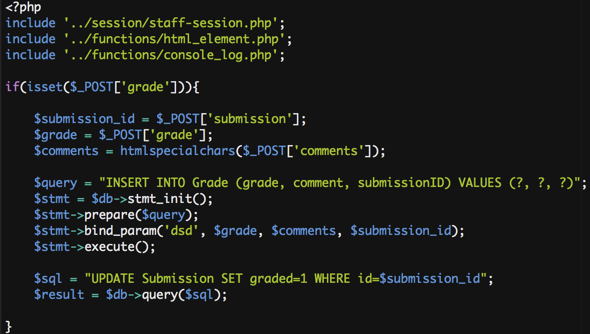
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID # | Test Case | Input Value | Expected output | Success |
| 000 | Full Calendar Implementation Unit Test  -          Basic Successful Post. | Announcement Title: “Discussion Board Week 1”  Class: “VAC 101”  Announcement Description: “What you to write about the newest vacuum on the market”  Make an event: Y  MM/DD/YYYY: 12/1/2017 | Server Output: Announcement Successfully Posted!  Home Page:  Assignment Event Displayed on Home Page. | Yes |
| 001 | Full Calendar Implementation Unit Test  -          Missing Field (Title) | Announcement Title: “           ”  Class: “VAC 101”  Announcement Description: “Assignment”  Make an event: Y  MM/DD/YYYY: 12/1/2017 | Server Output: “Missing Field”  Home Page:  No Display for Assignment, program should catch. | No  Server Output”  Announcement  Successfully Posted!  Home Page: |
| 011 | Full Calendar Unit Implementation Test  -          Missing Field (All) | Announcement Title: “           ”  Class: “     ”  Announcement Description: “    ”  Make an event: Y  MM/DD/YYYY: “     ” | Server Output: “Missing Field or Please Fill in a field” | No  Server Output”  Announcement  Successfully Posted! |

**13.3.1.3. Gradebook Component Testing**

Our gradebook component has functionality for both students and instructors, and each gradebook view must be tested accordingly. Students can view grades, while instructors can enter and modify grades. Documented are separate flow graphs for the different views due to their differing functionality.

**13.3.1.3.1. Gradebook Flow Graph Instructor View**

The picture below is a screenshot of the grade submission code for the instructor. Below the screenshot is a flowchart graph representing this code.

****

**Figure 39. Gradebook Source Code**

**13.3.1.3.2. Gradebook Cyclomatic Complexity/Test Cases**

Test cases for the instructor grading an assignment are featured in the table below.

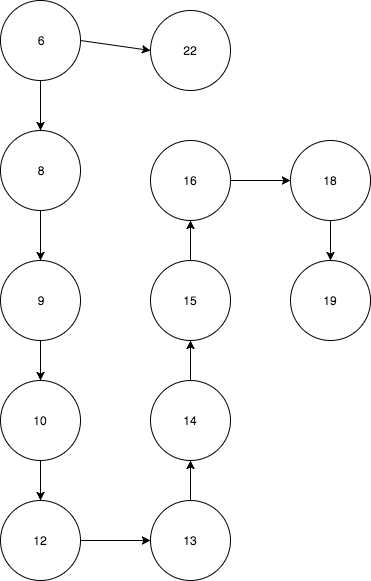
|  |  |
| --- | --- |
| **Test** | **Result** |
| No fields entered | Assignment not graded - Pass |
| No grade entered | Assignment not graded - Pass |
| No comment | Assignment Graded - Pass |
| All fields entered | Assignment Graded - Pass |

**13.3.2. Interface Testing**

**13.3.2.1. Administrative Interface Testing**

**13.3.2.1.1. Account Creation**

Shown in figure 41 is a screenshot of our account creation code, written in PHP and also described as a part of our pseudocode section. As a part of unit testing, our next step in development was to make a flow graph. This code is the same across all types of account creation, and only differs in database storage and fields unique to the type of account created.

****

**Figure 40. Gradebook Flow Chart**

**13.3.2.1.2. Account Creation Flow Graph**

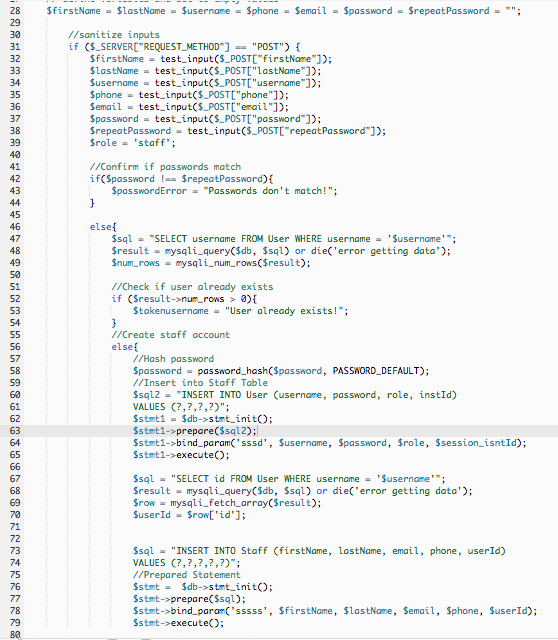
Shown in figure 42 is the flow graph generated from analysis of the account creation code above - the graph, though long, shows that our code does not have any complexities in the form of nested for loops. This graph will not be subject to change complexity-wise when creating accounts of all types, even if the code analyzed to develop this flow graph is specifically for staff creation.

**13.3.2.1.3. Account creation cyclomatic complexity/test cases**

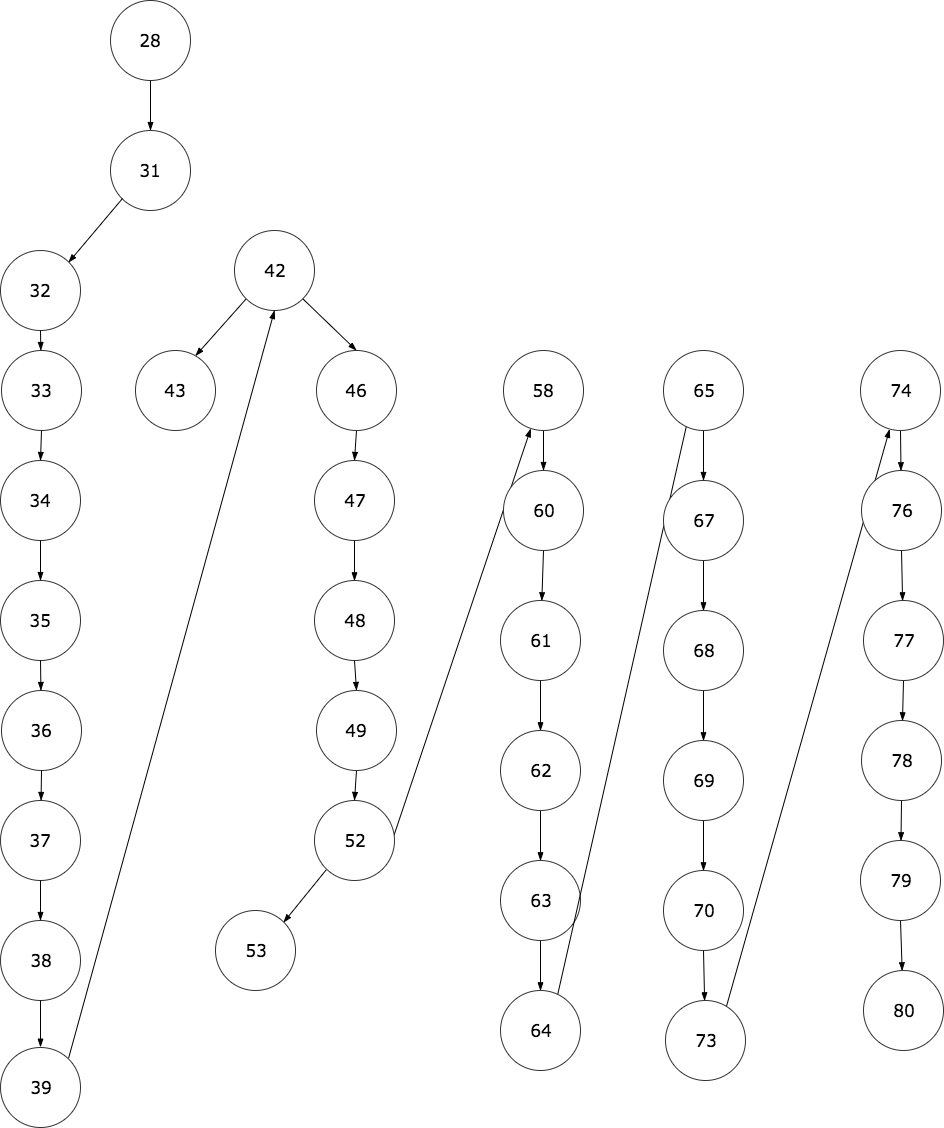
Using the flow graph from figure 42 to test account creation, cases can be generated for every outcome. Our process for doing so is first calculating the cyclomatic complexity, and then creating test cases to exercise every path in the flow graph accordingly. So, here is the work for our cyclomatic complexity calculations:

Complexity = # Edges - # Nodes + 2(# connected) = 35 - 36 + 2(4) = 7

Having a cyclomatic complexity of 5 means that there are 5 paths for our code to go through. Looking at the code, this can easily be justified. There is only one path to the end if fields are successfully entered, and otherwise accounts are not created. Our test cases, then, are the following:



**Figure 41. Account Creation Source Code**



**Figure 42. Account Creation Flow Chart**

We tested the following cases with success. Our inputs were accurately sanitized, and all possible combinations of incorrect or correct fields passed our expected outcome tests. This success can be attributed to our consistent testing and debugging before officially documenting our results as unit testing.

|  |  |
| --- | --- |
| **Test Case** | **Result** |
| No fields are entered | No account created - Pass |
| E-mail field has invalid e-mail format | No account created - Pass |
| Fields are partially complete; some are missing | No account created - Pass |
| Fields are entered, but passwords do not match | No account created - Pass |
| Attempted account already exists | No account created - Pass |
| Fields are entered with SQL injection | No injection occurs - Pass |
| Fields are entered and user does not exist | Account created - Pass |

**13.3.2.3. Staff Interface Testing**

The below tables feature the results for testing through the staff interface. All required input fields should be filled in order for tests to pass. If one is empty, the test should fail and the user should be prompted to fill out the empty field.

**13.3.2.3.1. Course Creation Testing**

|  |  |
| --- | --- |
| **Test Case** | **Result (Pass/Fail)** |
| All fields empty | No Course Created - Pass |
| Course Name Empty | No Course Created - Pass |
| Course Title Empty | No Course Created - Pass |
| All fields entered | Course Created - Pass |

**13.3.2.3.1. Announcement Creation Testing**

|  |  |
| --- | --- |
| **Test Case** | **Result** |
| All fields empty | No announcement created - Pass |
| No course selected | No announcement created - Pass |
| No announcement description | No announcement created - Pass |
| Required fields filled out | Announcement Created - Pass |
| All fields entered | Announcement Created - Pass |

**13.4. Integration Testing**

The goal of integration testing in WhiteBoard was to test and ensure functionality across combined components and interfaces, something that could be of concern once interfaces across user interfaces were combined and components such as the gradebook and calendar were integrated. Two primary and contrasting methods of integration testing usually converge into iterative or non-iterative approaches. However, incremental testing was more in-line with the rest of our development cycle, where integration of all components did not occur at the same time. From there, our decision was that of picking between a top-down incremental approach and a bottom-up incremental approach. We opted for more of a top-down approach to ensure that our most major components were functional, and tested/debugged as integration occurred throughout the development cycle of WhiteBoard - typically, there would be issues, especially when merging code developed by different team members. However, as WhiteBoard was developed, integration testing was continual, and when issues arose, we were able to fix them using the debugging techniques as documented.

**13.5. Validation Testing**

Following integration testing techniques used in WhiteBoard’s development was validation testing. The goal of using validation testing was to emphasize output relating directly to what system users see and demonstrate functionality across all components. More specifically, our hopes of validation testing were ensuring that all functional requirements are satisfactory, behavioral characteristics are as desired, performance requirements are met, and all other non-functional requirements are met.

**13.5.1. Validation Test Cases**

This section more specifically documents our validation test cases, broken down by component and type of validation test. All validation tests were conducted with our client.

**13.5.1.1. Administrator View**

**13.5.1.1.1. Functional Requirements Validation**

Functional requirements for our administrative interface were defined early in our project to allow for creation, deletion, and modification of accounts. An additional feature created for our administrative view is choosing the colors that instructors and students can view when accessing WhiteBoard based on their school. Additional functional requirements included providing administrators with a view of system metrics.

These functional requirements were tested with our client, who utilized our completed interface to create, update, and delete accounts, as well as change the colors. Our client validated the functionality of this interface with no failures.

**13.5.1.1.2. Behavioral Characteristics Validation**

Behavioral characteristics test cases for our administrative interface relate to the dynamic behavior of our system and its responses to various events. The only events possible for our administrative interface entail creation, deletion, or modification of student or instructor accounts, as well as changing colors and successfully viewing system metrics such as the number of users.

The test cases for validation of behavioral characteristics are no different from those executed for functional requirements; these test cases are not documented in this section because they are described in unit testing, and the only difference is the involvement with our client. As with the functional requirements, our behavioral characteristics validation was successful.

**13.5.1.1.3. Usability/Performance Validation**

We defined usability and performance success for our administrative interface to be the ability for our interface to load in a reasonable amount of time as defined by our client, and usable such that the client could successfully and easily navigate our interface without confusion or need for help. Usability and performance were both validated in parallel with behavioral characteristics and functional requirements, marking all validation needed for our administrative interface to be complete.

**13.5.1.2. Student View**

**13.5.1.2.1. Functional Requirements Validation**

Functional requirements for our student view were defined as such, and can be read in more detail in earlier sections of our documentation: students are provided with a central home screen from which they can access core features of WhiteBoard. These core features are access to a messaging system, whose more specific requirements are also documented, view of a calendar system, gradebook, and assignments or announcements relating to the courses a student is enrolled in.

Our client logged into the system using our test student account and validated all of these requirements by testing each in detail. As with the other functional requirements validation tests, the tests executed are the same executed in the develop team’s unit testing, and thus are not mentioned in this section of the documentation.

**13.5.1.2.2. Behavioral Characteristics Validation**

Behavioral characteristics validation for this interface are the same across the three interfaces, with a definition relating to the dynamic behavior of the system when given user input. The student interface is more complex than the administrative interface, and events for this interface can be defined as inputs into the calendar system, messaging system, gradebook, and course-related functionality.

Our client’s methodology for testing the student interface, once again, involved logging in and testing our various features. For example, for our messaging system, the user tried to view the people they could communicate with, which is students in the same class or professors teaching classes. Our system responded successfully to various inputs, and this testing was successful on the first attempt due to the development team’s continuous testing throughout the software development cycle.

**13.5.1.2.3. Usability/Performance Validation**

Usability and performance validations for the student interface are the same as the administrative interface requirements. As with the administrative interface, these were tested at the same time as with the other validation requirements, and proved to be a success.

**13.5.1.3. Instructor View**

**13.5.1.3.1. Functional Requirements Validation**

The instructor interface functional requirements are similar to student interface functional requirements, but have more extensive features in the form of being able to make modifications to the calendar, gradebook, and assignment features that students are only allowed to view. As with the other functional requirements, these can be found in more detail in the section dedicated to requirements.

Following the same process carried out to test the other two interfaces, our client validated our functional requirements successfully across all subcomponents accessible through the instructor interface. As with the other interfaces, specific test cases used can be found in our unit testing documentation, the difference being that our client did not analyze our code and instead analyzed our system from the user’s perspective.

**13.5.1.3.2. Behavioral Characteristics Validation**

Behavioral characteristics remain the same as with the two previous interfaces validated, and as stated, the instructor interface is very similar to the student interface. Core differences primarily entail the instructor’s ability to make changes to the gradebook, calendar, and course information, while another, smaller difference is that instructors have the ability to message other instructors that students would not necessarily be tied to.

    Once again, the client’s method of testing behavioral characteristics did not change across the interfaces. The system responded successfully to every test imposed by our client, and the behavioral characteristics for the instructor interface were validated.

**13.5.1.3.3. Usability/Performance Validation**

Yet again, usability and performance validation are the same as they are for the administrative interface and student interface. As expected, these tests were passed with ease - this was no surprise given that the student interface is extremely similar to that of the instructor interface.

**13.5.2. Client Involvement in Validation**

An important part of validation testing is in client involvement, where our client’s input is necessary to of course validate our system. The above test cases were executed and passed by our client to ensure successful completion, as stated above - where our development team continually validated software component internally, true validation required confirmation by our client.

**13.5.3. Validation Testing Conclusions**

As our test cases above prove, our validation testing was successful, and there were no objections to our client, who accepted our system through these validation tests. These tests will also be described in our acceptance testing section due to validation testing being an aspect of acceptance testing.

**13.6. System Testing**

There are a wide range of system tests that we first considered the necessity of and then used. This section documents the various system testing techniques used in the development cycle of WhiteBoard, as well as particular results if applicable.

**13.6.1. Recovery Testing**

The goal of recovery testing is to test a system’s capabilities for recovering from faults, and general methodologies entail forcing a system to fail and then verifying that a system successfully recovers. This section specifically breaks down various ways in which we identified WhiteBoard could shut down due to various, general categories, and our specific methodologies of testing.

**13.6.1.1. Hardware Recovery**

For WhiteBoard, there are a few ways in which our system could fail - one method is the failure of our hardware, the Raspberry Pi. This is an entirely plausible cause of failure that could easily occur from problems such as power outages or the hardware overheating. So, our testing of recovery from hardware failure is as such: unplugging and plugging the Raspberry Pi back in. Our initial results were as such: the Raspberry Pi took very little time to recover, and the time taken was a mix of the Pi’s recovery and the docker-compose build which then again took place. It required the development team to manually reconfigure the Docker containers, and so the largest concern with hardware recovery time is not with the Pi itself; rather, it is with the ability of the development team to recognize failure and reboot quickly.

**13.6.2. Security Testing**

Security testing is also touched upon in the security section of our documentation, and in this section comes more analysis and detailed description of our testing techniques used to conclude and ensure secure components of our system.

The below table shows how the system reacted to a computer user attempting to access various pages in the system. The majority of pages require login credential to gain access to, therefore if access was attempted and a login page was displayed the system passed the security test. If access to the page was granted without asking for login credentials, the system failed the test and integrity was compromised. If the attacker attempted to login without filling in proper credentials, the login failed. Many parts of our code are accessible to guest users such as css styling sheets, however these do not provide any private information useful to an attacker and therefore do not compromise security. They are available simply as source code that could possibly be used by anyone who might want them.

|  |  |
| --- | --- |
| Page Attempted to Access | Result |
| admin-create page | Login Required - Pass |
| admin-update-delete page | Login Required - Pass |
| admin page | Login Required - Pass |
| color-change page | Login Required - Pass |
| resource-monitor | Login Required - Pass |
| admin-session | Login Required - Pass |
| staff-session | Login Required - Pass |
| student-session | Login Required - Pass |
| staff-create | Login Required - Pass |
| student-create | Login Required - Pass |

**13.6.3. Stress Testing**

Stress testing ties into performance tests executed to learn the limitations of our hardware, as stress testing is focused on the functionality of a system when it is pushed to unusually high demand. As our performance testing has involved gathering data on the capabilities of components such as our database and Raspberry Pi capabilities, we did not feel the need specifically to use stress testing as a technique for WhiteBoard. The results from performance tests reinforce the notion that currently our system would be unable to handle a large number of users without any upgrades to hardware. The system is scalable, however, and if hardware components were upgraded it would be able to handle a much larger load.

**13.6.4. Performance Testing**

As with some of the other testing methods mentioned here, performance testing has an entire section dedicated to it and the results of our system in the form of plotted data. This section of documentation more generally describes how the performance testing folds into our overarching goals and motives of general system testing. As our graphs will later visualize, our performance tests helped us better understand the limitations and boundaries of the hardware on which our project needs to run.

**13.7. Alpha and Beta Testing**

Alpha and beta testing is one technique considered in the development of WhiteBoard. The decision we made regarding the possibility of alpha and beta testing was as such: we opted to use alpha testing, where we, the developers, would be able to test our software in a controlled environment, but no beta testing (where there is typically a “live release”, and we would not be present). This is simply because we already know the boundaries of our Raspberry Pi, and a beta release would not do it any favors. More issues with a beta release would be finding not only students, but also instructors and administrators to test our software with. This is a pressing issue caused by a short deadline of the end of the semester.

**13.8. Verification Testing**

This section documents our methodologies and considerations taken with regards to verification testing carried out with the goals of ensuring various algorithms and components execute as expected. Of course, other testing methods and strategies documented are a part of verification testing, especially unit testing. But, this section is aimed at describing a more general version of our verification testing needs and specific techniques.

**13.8.1. Static Verification Techniques**

There are two general categories of verification techniques falling into static or dynamic methodologies, and this section is dedicated to static verification. This entails pair programming, program inspections, and analysis tools used in the development of WhiteBoard with the goals of improving our development process by employing these techniques reported to have immense success.

**13.8.1.1. Pair Programming**

Pair programming is well-known and incredibly well-documented as a tool for success in software development. The goal of pair programming is to reap the benefits of reviews, detailed on a larger scale in another section of this document, through a small team of two people who as a result of collaboration result in better design and coding than one individual would. The development and division of development is documented as being broken down for individuals on the component-level scale, but for some components, pair programming was employed and extremely successful - this was true both between two frontend developers and the combination of one frontend developer and one backend developer. Having one frontend and one backend developer work together on the project was valuable for the sake of creating a flexible design beneficial to both teams, while having two frontend developers work together was effective so that less experienced frontend team members could gain an expedited understanding of web development and technologies.

**13.8.1.2. Program Inspections**

Program inspections are similar in spirit to other techniques that include pair programming and reviews, as their goals are to detect faults in programs by means of formal review. Code is reviewed on a line-by-line basis, and a checklist of common errors is followed. Our checklist of errors began with the inspection of code for potential data faults - often the result of initialization issues, constants, or array boundaries. Next comes inspecting for control faults, which involves conditions, loop terminations, compound statements, and case statements. Following this, our process involves checking for input and output faults, which entails making sure that every output is assigned a correct value. Following this is interface faults - this is a check for parameter numbers, types, order, structures, and shared memory. Finally, storage management faults and exceptions are more errors to check the code for. Program inspections are much more exhaustive and time-consuming than many of the other tests and testing techniques used for WhiteBoard.

Although we were not able to execute program inspections particularly frequently due to the nature of our group meetings and inability to meet for long periods of time because of differing schedules, code was reviewed on a moderate basis. Program inspections were helpful when possible, and otherwise, our other review methodologies proved to be one of the most important components of our developing team.

**13.8.1.3. Analysis Tools**

One method of software testing is the use of program analysis tools, something much less involving for team members - program analysis tools are very useful for checking control flow, data use, and interface faults. Cross-reference tables showing every variable and object, information flow analysis to identify input variables, and path analysis are only a handful of the perks associated with the use of analysis tools.

**13.8.2. Dynamic Verification Techniques**

Dynamic verification techniques used in our project are documented in other sections of this document, as the techniques used include unit testing, system testing, integration testing, performance testing, and acceptance testing among other methods.

**13.8.3. Defensive Programming**

Defensive programming is one simple thing to keep in mind during software development, and simply entails writing simple code to best avoid errors. More techniques in the folds of defensive programming include avoiding risky programming constructs and rewriting code difficult to read. Additionally, something important and easily neglected is the inclusion of comments for every method in our code. These techniques can be categorized more, as with other aspects to software testing, as a long-term strategy for success. This is a clear and sharp contrast to running particular tests and debugging or otherwise fixing code after the result of a case made post-development of a particular component or feature.

**13.9. Debugging Strategies**

Debugging strategies and methodologies were necessary after successful test execution to ensure that any errors could be fixed in the most efficient way possible, and this section of our project documentation describes the various techniques and thoughts we have regarding the debugging of our code. Common problems with debugging are geographically remote errors, human error, timing problems, non-errors, and temporarily disappearing errors; making sure our project sticks to our intended schedule, therefore, could greatly benefit from careful planning while developing our project and before debugging code that may have gone awry.

There are two primary methods of debugging which we considered and ultimately ended up mixing throughout development - brute force and backtracking. Backtracking is more successful, and involves manual tracing of code. However, there are many times where backtracking fails, and brute force is the superior technique. Some questions we always must keep in mind while debugging are as such: is the cause of a particular error also prevalent in another area of the program? Will fixing one error result in more errors? Could this bug have been prevented, and how? These questions are all considerations for developers at every point in WhiteBoard’s development and debugging processes.

**14. Pseudocode**

Pseudocode is one method of both planning and describing code created for WhiteBoard, and this section documents any project-related pseudocode generated for any purpose. Pseudocode is intended to be the same regardless of programming language or implementation, and has strengthened our planning methodologies to better increase the efficiency of our project development cycle.

**14.1. User Account Creation**

One of the core requirements for successful completion and delivery of WhiteBoard is that administrators must be able to create users. Below is the pseudocode showing the logic behind our implementation of this feature, done using PHP and MySQL in our project.

**Create blank username and password**

**Sanitize inputs**

**If two password inputs match and user does not exist yet**

**Hash password for security**

**Insert user into database**

**Else account is not created**

**14.2. Announcements Pseudocode**

A major aspect of the front end interface is the announcements section.  This section displays all new information regarding course announcements.  It accomplishes this task through this algorithm using PHP and MySQL.

**For every course**

**Create column and header**

**Retrieve most recent 3 announcements from course**

**For every announcement**

**Create html element with announcement information**

**Insert into column**

**14.3. Creating a Course**

Course creation is similar to creation and posting of other WhiteBoard components, but below is the pseudocode. After course creation, these courses are then an integral part of WhiteBoard, and are viewed by the students and instructors in and/or teaching the course.

**Read in text-based input**

**Sanitize inputs**

**Create course as database entry**

**14.4. Posting Assignments**

Posting assignments, a task conducted by instructors, is very similar to course creation. After assignment posting, the assignments can be viewed by students and instructors.

**Read in text-based input**

**Sanitize inputs**

**Create assignment as database entry**

**14.5. Calendar Postings**

Our calendar activities are managed by use of the Fullcalendar framework and JSON arrays of activity information. This method is detailed in the pseudocode below.

**Read in text-based input from instructor assignment**

**Sanitize inputs**

**Format text into Fullcalendar event array**

**Add event to calendar**

**15. Reviews**

Code reviews amongst the development team proved to be a valuable tool to the creation of WhiteBoard. Generally, reviews are where team members review everybody’s work - aspects such as the level of formality involved are up to the team to determine. Developers share important information and specifics about components being developed, and feedback is given and distributed across the team.

    Our weekly client meetings discussed and disclosed progress made on WhiteBoard, and evolved into reviews of our system. As well as proposing questions and describing development to our client, team members would always take the time to report on and discuss progress with one another. Our code reviews can be described as informal and casual, but extremely effective for project development - our client was able to serve as both a moderator for discussion and provide their input on system requirements and development advice, and the client’s presence motivated development team members to take the meetings seriously enough for progress to ensue. Some reviews typically involve scribes, but there was never any need for a scribe in our reviews; this is in part because we would discuss and review components being developed, and then begin development on such components by working together on them.

    Overall, the use of reviews as a software development technique for our team proved to be a massive success. Some developers had prior knowledge of web development and technologies, while others had to learn basics such as HTML to make development of WhiteBoard possible. So, more experienced developers coming together with the less experienced developers in that regards was always helpful for ensuring that every team member had a direction to take in project development. Additionally, development would often be divided on a component-level basis. For example, one team member would focus on getting the calendar feature up and running while another’s goal was to develop the messaging system component of WhiteBoard. Reviews were necessary for discussion of how these components would tie together and interface.

**16. Acceptance Testing**

Acceptance testing was completed during a client meeting on November 17th.  Bailey Henson of the development team completed the testing with our client.  Our client experienced an interactive demonstration of the product in its current state to review whether the product is meeting the requirements and expectations of the client.  The system was tested on the following criteria.

**16.1. Functional Correctness and Completeness**

This section will examine the WhiteBoard system to ensure the functional requirements for the project are being met correctly and completely.

**16.1.1. Account Creation**

The WhiteBoard system must provide the function to create the three different levels of accounts: administration, instructor and student.  The acceptance testing for this functionality consisted of an interactive demonstration for the client.  In this demonstration the client was able to create each different type of account successfully through the account creation interface.  The client confirmed that this functionality has been achieved and accepted.

**16.1.2. Login System**

An operational login system that distinguishes between the three separate account types is a requirement for the WhiteBoard product.  This aspect of the system was tested with the client for acceptance by allowing the client to use the accounts that were created in the account creation phase of acceptance testing to access secure interfaces through the login system. After testing the ability to login successfully the system was then tested for its ability to decline entrance to a request with invalid account information.  The client accepted these functional requirements as being met.

**16.1.3. Administration Accounts**

Administration accounts are responsible for creating instructor and student accounts.  An added functionality to administration accounts is the ability to change the color scheme of the WhiteBoard interface for all users associated with that administration.  The administration accounts also have access to a resource monitor for the WhiteBoard system.  These functions were tested by the client through an interactive demonstration where the client was able to create new instructor and student accounts, change the color scheme, and view the resource monitor.  The results of these functions were successful and the client confirmed they met the requirements.

**16.1.4. Instructor Accounts**

Instructor accounts are able to create courses, assignments, course content and announcements.  Instructors are also able to add students to courses and grade submitted assignments.  Through an interactive demonstration the client was able to create a new course and add students and content to it.  The client was also able to create new announcements which includes the ability to make the announcement a calendar event if so desired.  The client created a new assignment and graded an assignment submission.  All of these functions were accepted by the client to meet the requirements.

**16.1.5. Student Accounts**

Student accounts can access course content, assignments, grades and announcements.  Students must also be able to submit assignments.  These functions were tested by the client through an interactive demonstration where the client was able to access course content through the web interface.  The client was also able to view and submit an assignment then view the grade received on that submission.  Viewing announcements was completed upon accessing a student account, as these are displayed on the initial page after login.  After testing these functions the client confirmed they met the requirements for the project.

**16.1.6. Calendar**

A calendar is one of the features the client specified for the WhiteBoard project.  This calendar must be accessible by instructor and student accounts and this was displayed to the client upon accessing instructor and student accounts through the interactive demonstration.  Calendar events are created automatically for all assignments and this functionality was verified by allowing the client to view the calendar listing for the assignment created previously during the acceptance testing for the instructor accounts.  Instructor accounts may also create calendar events out of announcements if they so choose, and this functionality was demonstrated to the client as well.  The calendar feature was accepted as meeting requirements.

**16.1.7. Messaging**

A messaging system that enables communication between instructor and student accounts is another feature of WhiteBoard.  This system must restrict contact to only students and instructors that share a common course.  The client was able to interact with the messaging interface for acceptance testing.  The client sent and received messages, and the capability to restrict contacts was demonstrated for the client.  The client accepted this feature as meeting requirements.

**16.1.8. Gradebook**

A gradebook that is accessible by both student and instructor accounts.  For instructor accounts the gradebook must display the grades of all students in every course that instructor teaches.  It must also enable the instructor to update grades and comments for all assignments.  The client was first allowed to access the gradebook through an instructor account and test the system’s ability to complete these functions. The client agreed these functions of the gradebook meets the requirements.  The client tested the student gradebook interface next by accessing the gradebook through a student account to view the grades submitted while testing the instructor account.  The client verified that the student view also meets the requirements.

**16.2. Backup and Recovery**

Backup and recovery is another criteria which the WhiteBoard system was tested for acceptance by the client.  This was completed by demonstrating how quickly the system recovers after a failure.  Docker made this easy to accomplish through its compose function which saves the previous state of the system.  The client agreed that the recovery time meets the requirements of the project.

**16.3. Competitive Edge & Usability**

Competitive edge and usability were not specifically tested for.  However, these criteria were assessed by the client during the interactive demonstration used in the acceptance testing for functional correctness and completeness.  The client deemed the system to be competitive for the functionality included and the usability of the interface.  The client was very impressed with the design of the web interface and felt it was very easy to use even for someone with no experience or training.  The client agreed the system met the standards for competitiveness and usability.

**16.4. Performance**

Performance was tested during the interactive demonstration provided to the client.  While checking the functionality of the system, the client also assessed how well these functions performed in real-time.  Another aspect of acceptance testing for performance was reviewing the results of benchmark testing already performed on the system.  After reviewing this information and experiencing the system first-hand the client accepted the system as meeting performance requirements.

**16.5. Documentation**

Documentation acceptance testing was completed with the client by reviewing the current documentation with the client.  The client verified all desired information was included.  Specified information included diagrams, performance testing, system testing, requirements, scope, organizational process, risk management, pseudocode, quality management, and project progress.  The client agreed to accept the documentation as meeting requirements.

**17. Final Project Adjustments**

**17.1 Final Project Overview**

As the project entered its final milestone the overview had not changed much.  This milestone started off on time with the projected tasks left to complete.  The implementation tasks included finalizing the web interface, and the messaging, calendar and gradebook features.  Other tasks such as system testing were completed in this milestone as well and the results of these tests were added to the documentation.

**17.2 Final Project Scope**

The final scope of our project is similar to the scope originally described in the beginning sections of this report. The project functions mainly as a account and login system for educational institutions. There are three types of user accounts: administrators, students, and instructors.  Administrators are institutional staff who have the ability to make school-wide announcements, create student and instructor accounts, and create courses. Instructors have the ability to create assignments/exams, message other users, make class announcements, and more. Student accounts have similar functionality in that they can access course materials, submit assignments, view homework, and message other users. More detailed descriptions of each type of account can be seen in section 3.

Other features of Whiteboard include the Calendar, Gradebook, and Communication. A calendar is available that allows students and teachers to view current and future assignments, exams, or meetings. The gradebook allows students to view their grades in each of their classes and allows instructors to enter grades for each student in all of their classes. A messaging system allows student and instructor users to send messages back and forth. Again, more detailed descriptions for each of these features can be examined in section 3.

**17.3 Final Project Requirements**

The final requirements for our product did not end in significant change to the final WhiteBoard software. With regards especially to non-functional requirements, our plans did not change, and WhiteBoard was developed to accommodate for various requirements. The only significant change in requirements is, as stated in other areas of our documentation, the lack of SMS messaging integration in favor of a solely-internal system.

**18. Conclusion**

This document has detailed the development process for WhiteBoard, recording general project progress as well as techniques used for development. It entails changes in development patterns and implementation decisions, with benchmarking on system performance and detailed description of frontend and backend technologies. All of this has culminated in our final product, delivered to the client and confirmed through acceptance testing. Our previous beliefs were that, based on initial feasibility research, our project could be developed over the course of the fall 2017 semester, and we were able to successfully meet our goals.

**19. Future Work**

The requirements for our client have been met, but future work on WhiteBoard would consist of porting to a machine with higher user capabilities and further fleshing out and debugging features developed during the 2017 fall semester. Previously-planned features that had to be omitted due to time constraints could be touched on, and a direction the project could go as educational software could be more detailed beta testing through usage in a Western Kentucky University computer science course.

**I. Appendix**

1. **Raspberry Pi Setup**

To ensure that the development team had everything we needed to properly set up the Raspberry Pi development environment, our team purchased the CanaKit Raspberry Pi 3 Complete Starter Kit - 32 GigaByte Edition from Amazon.com. This Starter Kit includes the Raspberry Pi 3 Model B, which is one of the latest Raspberry Pi models currently. This model is roughly 10 times as quick as a first generation Raspberry Pi, and incorporates built-in WiFi and Bluetooth functionality. The Raspberry Pi 3 Model B features a Quad-Core 1.2 GHz processor and 1 gigabyte of random-access memory.

The starter kit had all the essentials to get our Raspberry Pi functional. This includes:

* 32 gigabyte MicroSD card (Class 10)
* 2.5A Micro USB power supply
* Raspberry Pi 3 case
* HDMI cable
* 2 heat sinks

The MicroSD card was preloaded with NOOBS. NOOBS (New Out of Box Software) allows for easy installation of operating systems onto the Raspberry Pi without the need to manually image the MicroSD card. If a MicroSD card is not preloaded with NOOBS, one can simply format the MicroSD card (must be at least 8 GB) as FAT, and then download and extract the files from the NOOBS zip file onto the formatted MicroSD card. More information about this procedure can be found on https://www.raspberrypi.org/documentation/installation/noobs.md.

The installation procedure of the Raspberry Pi followed as such:

**1.** The Raspberry Pi is installed into the case provided. The case comes is separated into two parts: a container and a lid.

**2.** The MicroSD card preloaded with NOOBS is inserted into the MicroSD card slot on underside of the Raspberry Pi.

**3.** A mouse, keyboard, and connection to a HDMI monitor/TV are needed for first time setup. The 2.5A power adapter is plugged into the microUSB slot on the Raspberry Pi board.

**4.** The NOOBS window will appear. From this menu, an operating system can be selected for installation. As of NOOBS v1.3.10, only Raspbian will show by default. If any other Operating System is desired, the Raspberry Pi can be connected to a network via ethernet cable or WiFi network, from which additional Operating Systems will appear to be installed. Our team installed the Raspbian OS.

**5.** From there, the Raspberry Pi automatically boots straight into the desktop interface, and the installation process is finished.

1. **GitHub Repository Setup**

For version control and collaboration, our group is using GitHub. GitHub allows us to work together on the WhiteBoard project from anywhere, while saving us the bother of uploading and managing each individual change. To be able to effectively use GitHub, Git needs to be installed on all machines cloning and committing the repository. Thankfully, Git is really simple to install:

* On Windows: The official build of Git is available to download on the Git Website, <http://git-scm.com/download/win>. The download starts automatically, and then it can be ran and installed.
* On Mac: A binary installer, like the OSX Git installer from the Git website, can be downloaded and used to easily install Git. The link is <http://git-scm.com/download/mac>
* On Linux: Basic package-management tools associated with the Linux distribution can be used to install Git. The Raspberry Pi our group uses has Raspbian installed, which is a Debian-based distribution. So we would use a terminal command like

$ sudo apt-get install git-all

A free GitHub account can then be created. From there, a repository can be created. GitHub will ask if the repository is being created from nothing, or if there is an existing repository locally. In our situation, we had an existing project that we needed to add. Through Terminal/Command Prompt, we navigate to the directory in which our project is in. We initialize the directory as a repository using the command: git init

The files in the newly created local repository can then be added with: git add .

These files then need to be committed, or in other words, need to be saved. This will prepare the files in the local repository to then be pushed to the remote repository on GitHub. To commit the files staged: git commit -m “Message about change goes here”

On the GitHub repository page, there should be a remote repository URL. Copy this URL and then insert it in: git remote add origin \*URL HERE\*

The files in the local repository can then finally be pushed to the remote repository on GitHub. Push the changes using: git push origin master

The GitHub repository we will be using for the WhiteBoard project can be found on <https://github.com/bly01854/WhiteBoard>

**C. Connecting to the Raspberry Pi**

Methods for viewing our project and connecting to the Raspberry Pi are documented in this section. A few ways in which to see functionality and code relating to our project are as such:

1. **Login using SSH key -** the Raspberry Pi can be logged into using the following information: our Pi can be reached at the IP address 74.141.208.189 on Port 22 using the username pi and password WhiteBoard. To login, there are a variety of tools and technologies one could use; for example, the command-line interface could be used, or software such as Filezilla. In the simple-docker folder found in the www directory, all start-up procedures for our website and docker setup instructions can be found.
2. **Login using test accounts -** project functionality can be tested by logging into the accounts we have created for an institution, administrator, student, and teacher account. To do so, one needs to access <http://74.141.208.189/login.php>, and the credentials needed are as such: the accounts go under an institution named Western Kentucky University. To login as an administrator, use the username “admin” and password “admin.” For the student account, the username “student” and password “student” are needed. For the instructor account, the username “teacher” and password “teacher” are necessary.

**D. Final Source Code**

This section of the appendix contains all final source code which we developed for this project; the only code excluded comes from external sources such as Bootstrap or the Fullcalendar framework. Our code is sorted by the files and directories in which they are found.

* Admin
  + admin-create.php code:

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Create Staff</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<body>

   <?php include 'header.php';

   // define variables and set to empty values

  $username = $password = $repeatPassword = "";

       //sanitize inputs

       if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

           $username = test\_input($\_POST["username"]);

           $password = test\_input($\_POST["password"]);

           $repeatPassword = test\_input($\_POST["repeatPassword"]);

           $role = 'admin';

           //Confirm if passwords match

           if($password !== $repeatPassword){

               $passwordError = "Passwords don't match!";

           }

           else{

               $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($num\_rows > 0){

                   $takenusername = "User already exists!";

               }

               //Create admin account

               else{

                   //Hash password

                   $password = password\_hash($password, PASSWORD\_DEFAULT);

                   //Insert into user Table

                   $sql = "INSERT INTO User (username, password, role)

                   VALUES (?,?,?)";

                   $stmt = $db->stmt\_init();

                   $stmt->prepare($sql);

                   $stmt->bind\_param('sss', $username, $password, $role);

                   $stmt->execute();

                   $success = "Admin account successfully created!";

               }

           }

           }

       //sanitize input

       function test\_input($data) {

           $data = trim($data);

           $data = stripslashes($data);

           $data = htmlspecialchars($data);

           return $data;

       }

   ?>

   <div>

       <div class="container">

           <div class="row">

               <div class="col-md-12">

                   <div class="row register-form">

                       <div class="col-md-8 col-md-offset-2">

                           <h2 class="text-center">Create Admin</h2>

                           <form method="post" class="custom-form" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Username </label>

                                   <input class="form-control" type="text" name="username" required="" id="username">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="pawssword-input-field">Password </label>

                                   <input class="form-control" type="password" name="password" required="" id="password">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="repeat-pawssword-input-field">Repeat Password </label>

                                   <input class="form-control" type="password" name="repeatPassword" required="" id="repeatPassword">

                                   <p class="text-danger text-center"><?php echo $passwordError; echo $takenusername; ?></p>

                               </div>

                               <p class="text-success text-center"><?php echo $success ?></p>

                               <button class="btn btn-default submit-button" type="submit" id="submit">Submit Form</button>

                           </form>

                       </div>

                   </div>

               </div>

           </div>

       </div>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Admin-update-delete.php code:

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Update/Delete Admin</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<body>

   <?php include 'header.php';?>

<div class="container">

       <h2 class="text-center">Admin List</h2>

    <div class="row">

       <div class="col-sm-6 col-sm-offset-3">

<?php

       $query = "SELECT username, id FROM User WHERE role = 'admin'";

       $result = mysqli\_query($db, $query) or die('error getting data');

       $num\_rows = mysqli\_num\_rows($result);

       //show results

       echo "$num\_rows results found";

       echo "<table class='table'>";

       echo "<tr> <th> Username </th> </tr>";

       while ($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

           $userId = $row['id'];

           $studentID = $row['studentID'];

           echo "<tr><td><a href='admin.php?adminID=" . $userId . "'>";

           echo $row['username'];

           echo "</a></td></tr>";

       }

       echo "</table>";

?>

       </div>

    </div>

</div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Admin.php code:

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Admin Info</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Profile-Edit.css">

   <link rel="stylesheet" href="../css/Profile-Edit1.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<?php

   include 'header.php';

   //is $\_Get Secure in the scenerio probably.

   $userId = $\_GET['adminID'];

   $query1 = "SELECT username FROM User WHERE id= '$userId'";

   $result1 = $db->query($query1);

   $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

   $oldUsername = $row1['username'];

   $username = "";

   //sanitize inputs

   if (isset($\_POST['update\_button'])) {

       $username = test\_input($\_POST["username"]);

       if ($username != $oldUsername){

           $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($result->num\_rows > 0){

                   $takenusername = "User already exists!";

                   exit;

               }

       }

       $sql2 = "UPDATE User SET username=? WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('sd', $username, $userId);

       $stmt1->execute();

       //submit success message

       $success = "Admin Account Updated!";

       //refresh data

       $query1 = "SELECT username FROM User WHERE id= '$userId'";

       $result1 = $db->query($query1);

       $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

       $oldUsername = $row1['username'];

   }

   else if (isset($\_POST['delete\_button'])){

       $sql2 = "DELETE FROM User WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('d', $userId);

       $stmt1->execute();

       echo '<script type="text/javascript">

          window.location.replace("admin-update-delete.php");

               </script>';

   }

   //sanitize input

   function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<body>

   <div class="container profile profile-view" id="profile">

       <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]) . "?adminID=" . $\_GET['adminID'];?>">

           <div class="row profile-row">

               <div class="col-md-12">

                   <h2 class="text-center">Admin Profile</h2>

                   <hr>

                   <div class="form-group">

                       <label class="control-label">Username </label>

                       <input class="form-control" type="text" required="" autocomplete="off" name="username" value='<?php echo $oldUsername; ?>'>

                   </div>

                   <div class="row">

                       <p class="text-danger text-center"><?php echo $takenusername  ?></p>

                       <p class="text-success text-center"><?php echo $success ?></p>

                   </div>

                   <hr>

                   <div class="row">

                       <div class="col-md-12 content-right">

                           <button class="btn btn-primary form-btn" name="update\_button" type="submit">UPDATE </button>

                           <button class="btn btn-danger form-btn" name="delete\_button" type="submit">DELETE </button>

                       </div>

                   </div>

               </div>

           </div>

       </form>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Changecolor.php code

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <?php

   include '../connection.php';

   include '../session/admin-session.php';

   if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

     $color1 = $\_POST['color1'];

     $color2 = $\_POST['color2'];

     $sql = "UPDATE Institution SET primaryColor='$color1', secondaryColor='$color2' WHERE id='$session\_isntId'";

     $result = mysqli\_query($db, $sql) or die('error getting data');

     header("Refresh:0");

   }

   ?>

 <body id="main" style="background-color: <?php echo $session\_primaryColor; ?>">

   <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top secondary">

     <a class="navbar-brand" href="interface.php">WhiteBoard Admin</a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarsExampleDefault">

       <ul class="navbar-nav mr-auto">

         <li class="nav-item active">

           <a class="nav-link" href="manage.html">Manage Accounts <span class="sr-only">(current)</span></a>

         </li>

         <li class="nav-item">

           <a class="nav-link" href="#">Change Colors</a>

         </li>

        <li class="nav-item">

           <a class="nav-link disabled" href="#">Use Statistics</a>

         </li>

       <!---  <li class="nav-item">

           <a class="nav-link disabled" href="#"></a>

         </li>-->

         <li class="nav-item">

           <a class="nav-link disabled" href="#">Settings <i class="fa fa-cog" aria-hidden="true"></i></a>

         </li>

       </ul>

     </div>

   </nav>

<div class="container">

       <div class="row justify-content-md-center">

         <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

         <div class="col whiteboard whiteboard-main gradebook">

          <p class="lead grades"> Set School Colors  </p>

           <div class="row">

             <div class="col">

               <p class="lead ingrade"> Main Color:</p>

                 <input type="color" id="color" class="colorinput" value="<?php echo $session\_primaryColor; ?>" name="color1">

               <p class="lead ingrade"> Secondary Color:</p>

                 <input type="color" id="color2" class="colorinput" value="<?php echo $session\_secondaryColor; ?>" name="color2">

             </div>

             <div class="col">

             </div>

             <button type="button" class="btn btn-outline-secondary"

         onclick="document.getElementById('main').style.backgroundColor = document.getElementById('color').value; document.getElementById('secondary').style.backgroundColor = document.getElementById('color2').value">

             Change Colors

           </button>

           <button type="submit" class="btn btn-outline-secondary" style="padding:45px;margin-left:20px">Save</button>

           </div>

         </div>

       </div>

       </form>

       <script>

        document.getElementById("test").style.color = "blue";

       </script>

   <footer id="secondary" style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Header.php code

<body>

   <div>

       <nav class="navbar navbar-default navigation-clean-button">

           <div class="container">

               <div class="navbar-header"><a class="navbar-brand navbar-link" href="interface.php">WhiteBoard Admin</a>

                   <button class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navcol-1"><span class="sr-only">Toggle navigation</span><span class="icon-bar"></span><span class="icon-bar"></span><span class="icon-bar"></span></button>

               </div>

               <div class="collapse navbar-collapse" id="navcol-1">

                   <ul class="nav navbar-nav">

                       <li role="presentation"><a href="resource-monitor.php">Statistics</a></li>

                       <!---<li role="presentation"><a href="#">Second Item</a></li>

                       <li class="dropdown"><a class="dropdown-toggle" data-toggle="dropdown" aria-expanded="false" href="#">Dropdown <span class="caret"></span></a>

                           <ul class="dropdown-menu" role="menu">

                               <li role="presentation"><a href="resource-monitor.php">Statistics</a></li>

                               <li role="presentation"><a href="#">Second Item</a></li>

                               <li role="presentation"><a href="#">Third Item</a></li> --->

                           </ul>

                       </li>

                   </ul>

                   <p class="navbar-text navbar-right actions"><a class="btn btn-default action-button" role="button" href="../logout.php">Log Out</a></p>

               </div>

           </div>

       </nav>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

* + Interface.php code

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>WhiteBoard Admin</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<body>

   <?php include 'header.php';?>

   <div>

       <div class="container">

           <div class="row">

               <div class="col-md-6 col-md-offset-3">

                   <h2 class="text-center"><?php echo $session\_institution ?> </h2>

               </div>

           </div>

       </div>

       <div class="container">

           <div class="row">

               <div class="col-md-6">

                   <h3 class="text-center">Student </h3>

                   <div class="btn-group btn-group-justified" role="group">

                       <a class="btn btn-default" role="button" href="student-create.php" id="addStudent">Add </a>

                       <a class="btn btn-default" role="button" href="student-update-delete.php" id="updateStudent">Update/Delete </a>

                   </div>

               </div>

               <div class="col-md-6">

                   <h3 class="text-center">Staff </h3>

                   <div class="btn-group btn-group-justified" role="group">

                       <a class="btn btn-default" role="button" href="staff-create.php" id="staffAdd">add </a>

                       <a class="btn btn-default" role="button" href="staff-update-delete.php" id="staffUpdate">Update/Delete </a>

                   </div>

               </div>

           </div>

       </div>

       <div class="container">

           <div class="row">

               <div class="col-md-6">

                   <h3 class="text-center">Admin </h3>

                   <div class="btn-group btn-group-justified" role="group">

                       <a class="btn btn-default" role="button" href="admin-create.php" id="addAdmin">Add </a>

                       <a class="btn btn-default" role="button" href="admin-update-delete.php" id="updateStudent">Update/Delete </a>

                   </div>

               </div>

               <div class="col-md-6">

                   <h3 class="text-center">Website / Server Status </h3>

                   <div class="btn-group btn-group-justified" role="group">

                       <a class="btn btn-default" role="button" href="changecolor.php" id="colorChange">Change Colors </a>

                       <a class="btn btn-default" role="button" href="resource-monitor.php" id="resource-monitor">Resource Monitor </a>

                   </div>

               </div>

           </div>

       </div>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Resource-monitor.php code

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Create Staff</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

</head>

<body>

   <?php include 'header.php';

   $monitor = shell\_exec('python simple\_resource\_monitor.py ');

   $monitorData = json\_decode($monitor, true);

   $number\_of\_users = count(scandir(ini\_get("session.save\_path"))) - 30;

   ?>

   <div style="padding:25px">

       <div class="container">

           <div class="row">

               <div class="col-md-12 text-center" style="padding:10px">

                   <i class="fa fa-desktop fa-5x" aria-hidden="true"></i><h4 id="cpu"> CPU Usage:  <?php echo $monitorData[0]  ?>%</h4>

               </div>

           </div>

           <div class="row" style="padding:10px">

               <div class="col-md-12 text-center">

                   <i class="fa fa-tachometer fa-5x" aria-hidden="true"></i><h4 id="ram"> Memory Usage:  <?php echo $monitorData[1]  ?>%</h4>

               </div>

           </div>

           <div class="row" style="padding:10px">

               <div class="col-md-12 text-center">

                   <i class="fa fa-hdd-o fa-5x" aria-hidden="true"></i><h4 id="disk"> Disk Usage:  <?php echo $monitorData[2]  ?>%</h4>

               </div>

           </div>

           <div class="row" style="padding:10px">

               <div class="col-md-12 text-center">

                   <i class="fa fa-users fa-5x" aria-hidden="true"></i><h4 id="users"> Active Users:  <?php echo $number\_of\_users  ?></h4>

               </div>

           </div>

       </div>

   </div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script>

   $(document).ready(function(){

       setInterval(function() {

           $('#cpu').load(window.location.href + ' #cpu');

           $('#ram').load(window.location.href + ' #ram');

           $('#disk').load(window.location.href + ' #disk');

           $('#users').load(window.location.href + ' #users');

       }, 3000);

   });

</script>

</body>

</html>

* + Simple\_resource\_monitor.py

import psutil, json

def cpu\_monitor():

   cpu\_usage = psutil.cpu\_percent(1)

   return cpu\_usage;

def memory\_monitor():

   memory\_used = psutil.virtual\_memory().available

   memory\_total = psutil.virtual\_memory().total

   memory\_available = float(memory\_used) / float(memory\_total)

   memory\_usage = round((1 - memory\_available) \* 100, 2)

   return memory\_usage

def disk\_monitor():

   disk\_percent = psutil.disk\_usage('/').percent

   return disk\_percent

def main():

   statistics=[cpu\_monitor(),memory\_monitor(), disk\_monitor()]

   print json.dumps(statistics)

if \_\_name\_\_ == "\_\_main\_\_":

   main()

* + Staff-create.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Create Staff</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<body>

   <?php include 'header.php';

   // define variables and set to empty values

   $firstName = $lastName = $username = $phone = $email = $password = $repeatPassword = "";

       //sanitize inputs

       if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

           $firstName = test\_input($\_POST["firstName"]);

           $lastName = test\_input($\_POST["lastName"]);

           $username = test\_input($\_POST["username"]);

           $phone = test\_input($\_POST["phone"]);

           $email = test\_input($\_POST["email"]);

           $password = test\_input($\_POST["password"]);

           $repeatPassword = test\_input($\_POST["repeatPassword"]);

           $role = 'staff';

           //Confirm if passwords match

           if($password !== $repeatPassword){

               $passwordError = "Passwords don't match!";

           }

           else{

               $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($result->num\_rows > 0){

                   $takenusername = "User already exists!";

               }

               //Create staff account

               else{

                   //Hash password

                   $password = password\_hash($password, PASSWORD\_DEFAULT);

                   //Insert into Staff Table

                   $sql2 = "INSERT INTO User (username, password, role, instId)

                   VALUES (?,?,?,?)";

                   $stmt1 = $db->stmt\_init();

                   $stmt1->prepare($sql2);

                   $stmt1->bind\_param('sssd', $username, $password, $role, $session\_isntId);

                   $stmt1->execute();

                   $sql = "SELECT id FROM User WHERE username = '$username'";

                   $result = mysqli\_query($db, $sql) or die('error getting data');

                   $row = mysqli\_fetch\_array($result);

                   $userId = $row['id'];

                   $sql = "INSERT INTO Staff (firstName, lastName, email, phone, userId)

                   VALUES (?,?,?,?,?)";

                   //Prepared Statement

                   $stmt =  $db->stmt\_init();

                   $stmt->prepare($sql);

                   $stmt->bind\_param('sssss', $firstName, $lastName, $email, $phone, $userId);

                   $stmt->execute();

                   $success = "Staff account successfully created!";

               }

           }

           }

       //sanitize input

       function test\_input($data) {

           $data = trim($data);

           $data = stripslashes($data);

           $data = htmlspecialchars($data);

           return $data;

       }

   ?>

   <div>

       <div class="container">

           <div class="row">

               <div class="col-md-12">

                   <div class="row register-form">

                       <div class="col-md-8 col-md-offset-2">

                           <h2 class="text-center">Create Staff</h2>

                           <form method="post" class="custom-form" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">First Name</label>

                                   <input class="form-control" type="text" name="firstName" required="" id="firstName">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Last Name</label>

                                   <input class="form-control" type="text" name="lastName" required="" id="lastName">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Username </label>

                                   <input class="form-control" type="text" name="username" required="" id="username">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Phone Number (format: x-xxx-xxx-xxxx) </label>

                                   <input class="form-control" type="tel" name="phone" required="" id="phone" pattern="^\d{1}-\d{3}-\d{3}-\d{4}$">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="email-input-field">Email </label>

                                   <input class="form-control" type="email" name="email" required="" id="email">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="pawssword-input-field">Password </label>

                                   <input class="form-control" type="password" name="password" required="" id="password">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="repeat-pawssword-input-field">Repeat Password </label>

                                   <input class="form-control" type="password" name="repeatPassword" required="" id="repeatPassword">

                                   <p class="text-danger text-center"><?php echo $passwordError; echo $takenusername; ?></p>

                               </div>

                               <p class="text-success text-center"><?php echo $success ?></p>

                               <button class="btn btn-default submit-button" type="submit" id="submit">Submit Form</button>

                           </form>

                       </div>

                   </div>

               </div>

           </div>

       </div>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Staff-update-delete.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Update/Delete Staff</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

   <style>

       .results:hover {

       background-color:LightGrey;

       cursor:pointer;

       }

   </style>

</head>

<body>

   <?php include 'header.php';

   ?>

   <div class="container">

       <h2 class="text-center">Search for Staff</h2>

    <div class="row">

       <div class="col-sm-6 col-sm-offset-3">

           <div id="imaginary\_container">

               <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

                   <div class="input-group stylish-input-group">

                       <input type="text" class="form-control"  placeholder="Search by name" name="criteria" >

                       <span class="input-group-addon">

                       <button type="submit" >

                           <span class="glyphicon glyphicon-search"></span>

                       </button>

                       </span>

               </div>

           </form>

           </div>

       </div>

    </div>

</div>

<?php

   if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

       $criteria = $\_POST['criteria'];

       $query = "SELECT \* FROM Staff WHERE (concat\_ws(' ', firstName, lastName) LIKE '%$criteria%')";

       $result = mysqli\_query($db, $query) or die('error getting data');

       $num\_rows = mysqli\_num\_rows($result);

       //show results

       echo "$num\_rows results found";

       echo "<table class='table'>";

       echo "<tr> <th> First Name </th> <th> Last Name </th> <th> Username </th> <th> Email </th> <th> Phone </th>  </tr>";

       while ($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

           $userId = $row['userId'];

           $query1 = "SELECT username FROM User WHERE id= '$userId'";

           $result1 = $db->query($query1);

           $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

           $username = $row1['username'];

           $staffID = $row['id'];

           echo "<tr class='results' data-value='$staffID'><td>";

           echo $row['firstName'];

           echo "</td><td>";

           echo $row['lastName'];

           echo "</td><td>";

           echo $username;

           echo "</td><td>";

           echo $row['email'];

           echo "</td><td>";

           echo $row['phone'];

           echo "</a></td></tr>";

       }

       echo "</table>";

   }

?>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

   <script>

   $(document).ready(function(){

       $('tr.results').click(function(){

           var value = this.getAttribute('data-value');

           window.location.assign('staff.php?staffID=' + value);

       })

   });

   </script>

</body>

</html>

* + Staff.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Staff Info</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Profile-Edit.css">

   <link rel="stylesheet" href="../css/Profile-Edit1.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<?php

   include 'header.php';

   $staffID = $\_GET['staffID'];

   $query = "SELECT \* FROM Staff WHERE id= '$staffID'";

   $result = $db->query($query);

   $row = $result->fetch\_array(MYSQLI\_ASSOC);

   $userId = $row['userId'];

   $query1 = "SELECT username FROM User WHERE id= '$userId'";

   $result1 = $db->query($query1);

   $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

   $oldUsername = $row1['username'];

   $firstName = $lastName = $username = $email = $phone = "";

   //sanitize inputs

   if (isset($\_POST['update\_button'])) {

       $firstName = test\_input($\_POST["firstName"]);

       $lastName = test\_input($\_POST["lastName"]);

       $username = test\_input($\_POST["username"]);

       $email = test\_input($\_POST["email"]);

       $phone = test\_input($\_POST["phone"]);

       if ($username != $oldUsername){

           $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($result->num\_rows > 0){

                   $takenusername = "User already exists!";

                   exit;

               }

       }

       $sql2 = "UPDATE User SET username=? WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('sd', $username, $userId);

       $stmt1->execute();

       $sql1 = "UPDATE Staff SET firstName=?, lastName=?, email=?, phone=? WHERE userId=?";

       //Prepared Statement

       $stmt =  $db->stmt\_init();

       $stmt->prepare($sql1);

       $stmt->bind\_param('ssssd', $firstName, $lastName, $email, $phone, $userId);

       $stmt->execute();

       //submit success message

       $success = "Staff Account Updated!";

       //refresh data

       $staffID = $\_GET['staffID'];

       $query = "SELECT \* FROM Staff WHERE id= '$staffID'";

       $result = $db->query($query);

       $row = $result->fetch\_array(MYSQLI\_ASSOC);

       $query1 = "SELECT username FROM User WHERE id= '$userId'";

       $result1 = $db->query($query1);

       $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

       $oldUsername = $row1['username'];

   }

   else if (isset($\_POST['delete\_button'])){

       $sql1 = "DELETE FROM Staff WHERE userId=?";

       //Prepared Statement

       $stmt =  $db->stmt\_init();

       $stmt->prepare($sql1);

       $stmt->bind\_param('d', $userId);

       $stmt->execute();

       $sql2 = "DELETE FROM User WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('d', $userId);

       $stmt1->execute();

       echo '<script type="text/javascript">

          window.location.replace("staff-update-delete.php");

               </script>';

   }

   //sanitize input

   function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<body>

   <div class="container profile profile-view" id="profile">

       <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]) . "?staffID=" . $\_GET['staffID'];?>">

           <div class="row profile-row">

               <div class="col-md-12">

                   <h2 class="text-center">Staff Profile</h2>

                   <hr>

                   <div class="row">

                       <div class="col-md-6 col-sm-12">

                           <div class="form-group">

                               <label class="control-label">First name </label>

                               <input class="form-control" type="text" required="" name="firstName" value='<?php echo $row['firstName']; ?>'>

                           </div>

                       </div>

                       <div class="col-md-6 col-sm-12">

                           <div class="form-group">

                               <label class="control-label">Last name </label>

                               <input class="form-control" type="text" required="" name="lastName" value='<?php echo $row['lastName']; ?>'>

                           </div>

                       </div>

                   </div>

                   <div class="form-group">

                       <label class="control-label">Username </label>

                       <input class="form-control" type="text" required="" autocomplete="off" name="username" value='<?php echo $oldUsername; ?>'>

                   </div>

                   <div class="form-group">

                       <label class="control-label">Email </label>

                       <input class="form-control" type="email" required="" autocomplete="off" name="email" value='<?php echo $row['email']; ?>'>

                   </div>

                   <div class="form-group">

                       <label class="control-label">Phone Number (format: x-xxx-xxx-xxxx) </label>

                       <input class="form-control" type="text" pattern="^\d{1}-\d{3}-\d{3}-\d{4}$" required="" autocomplete="off" name="phone" value='<?php echo $row['phone']; ?>'>

                   </div>

                   <div class="row">

                       <p class="text-danger text-center"><?php echo $takenusername  ?></p>

                       <p class="text-success text-center"><?php echo $success ?></p>

                   </div>

                   <hr>

                   <div class="row">

                       <div class="col-md-12 content-right">

                           <button class="btn btn-primary form-btn" name="update\_button" type="submit">UPDATE </button>

                           <button class="btn btn-danger form-btn" name="delete\_button" type="submit">DELETE </button>

                       </div>

                   </div>

               </div>

           </div>

       </form>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Student-create.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Create Student</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<body>

   <?php include 'header.php';

   // define variables and set to empty values

   $firstName = $lastName = $username = $email = $password = $repeatPassword = "";

       //sanitize inputs

       if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

           $firstName = test\_input($\_POST["firstName"]);

           $lastName = test\_input($\_POST["lastName"]);

           $username = test\_input($\_POST["username"]);

           $email = test\_input($\_POST["email"]);

           $password = test\_input($\_POST["password"]);

           $repeatPassword = test\_input($\_POST["repeatPassword"]);

           $role = 'student';

           //Confirm if passwords match

           if($password !== $repeatPassword){

               $passwordError = "Passwords don't match!";

           }

           else{

               $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($result->num\_rows > 0){

                   $takenusername = "User already exists!";

               }

               //Create student account

               else{

                   //Hash password

                   $password = password\_hash($password, PASSWORD\_DEFAULT);

                   //Insert into Student Table

                    $sql2 = "INSERT INTO User (username, password, role, instId)

                   VALUES (?,?,?,?)";

                   $stmt1 = $db->stmt\_init();

                   $stmt1->prepare($sql2);

                   $stmt1->bind\_param('sssd', $username, $password, $role, $session\_isntId);

                   $stmt1->execute();

                   $sql = "SELECT id FROM User WHERE username = '$username'";

                   $result = mysqli\_query($db, $sql) or die('error getting data');

                   $row = mysqli\_fetch\_array($result);

                   $userId = $row['id'];

                   $sql1 = "INSERT INTO Student (firstName, lastName, email, userId)

                   VALUES (?,?,?,?)";

                   //Prepared Statement

                   $stmt =  $db->stmt\_init();

                   $stmt->prepare($sql1);

                   $stmt->bind\_param('ssss', $firstName, $lastName, $email, $userId);

                   $stmt->execute();

                   $success = "Student account successfully created!";

               }

           }

           }

       //sanitize input

       function test\_input($data) {

           $data = trim($data);

           $data = stripslashes($data);

           $data = htmlspecialchars($data);

           return $data;

       }

   ?>

   <div>

       <div class="container">

           <div class="row">

               <div class="col-md-12">

                   <div class="row register-form">

                       <div class="col-md-8 col-md-offset-2">

                           <h2 class="text-center">Create Student</h2>

                           <form method="post" class="custom-form" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">First Name</label>

                                   <input class="form-control" type="text" name="firstName" required="" id="firstName">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Last Name</label>

                                   <input class="form-control" type="text" name="lastName" required="" id="lastName">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="name-input-field">Username </label>

                                   <input class="form-control" type="text" name="username" required="" id="username">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="email-input-field">Email </label>

                                   <input class="form-control" type="email" name="email" required="" id="email">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="pawssword-input-field">Password </label>

                                   <input class="form-control" type="password" name="password" required="" id="password">

                               </div>

                               <div class="form-group">

                                   <label class="control-label" for="repeat-pawssword-input-field">Repeat Password </label>

                                   <input class="form-control" type="password" name="repeatPassword" required="" id="repeatPassword">

                                   <p class="text-danger text-center"><?php echo $passwordError; echo $takenusername; ?></p>

                               </div>

                               <p class="text-success text-center"><?php echo $success; ?></p>

                               <button class="btn btn-default submit-button" type="submit" id="submit">Submit Form</button>

                           </form>

                       </div>

                   </div>

               </div>

           </div>

       </div>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* + Student-update-delete.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Update/Delete Student</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

   <style>

       .results:hover {

       background-color:LightGrey;

       cursor:pointer;

       }

   </style>

</head>

<body>

   <?php include 'header.php';

   ?>

   <div class="container">

       <h2 class="text-center">Search for Student</h2>

    <div class="row">

       <div class="col-sm-6 col-sm-offset-3">

           <div id="imaginary\_container">

               <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

                   <div class="input-group stylish-input-group">

                       <input type="text" class="form-control"  placeholder="Search by name" name="criteria" >

                       <span class="input-group-addon">

                       <button type="submit" >

                           <span class="glyphicon glyphicon-search"></span>

                       </button>

                       </span>

               </div>

           </form>

           </div>

       </div>

    </div>

</div>

<?php

   if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

       $criteria = $\_POST['criteria'];

       $query = "SELECT \* FROM Student WHERE (concat\_ws(' ', firstName, lastName) LIKE '%$criteria%')";

       $result = mysqli\_query($db, $query) or die('error getting data');

       $num\_rows = mysqli\_num\_rows($result);

       //show results

       echo "$num\_rows results found";

       echo "<table class='table'>";

       echo "<tr> <th> First Name </th> <th> Last Name </th> <th> Username </th> <th> Email </th>  </tr>";

       while ($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

           $userId = $row['userId'];

           $query1 = "SELECT username FROM User WHERE id= '$userId'";

           $result1 = $db->query($query1);

           $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

           $username = $row1['username'];

           $studentID = $row['studentID'];

           echo "<tr class='results' data-value='$studentID'><td>";

           echo $row['firstName'];

           echo "</td><td>";

           echo $row['lastName'];

           echo "</td><td>";

           echo $username;

           echo "</td><td>";

           echo $row['email'];

           echo "</td></tr>";

       }

       echo "</table>";

   }

?>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

   <script>

   $(document).ready(function(){

       $('tr.results').click(function(){

           var value = this.getAttribute('data-value');

           window.location.assign('student.php?studentID=' + value);

       })

   });

   </script>

</body>

</html>

* + Student.php

<?php

include '../session/admin-session.php';

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Student Info</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="../css/Profile-Edit.css">

   <link rel="stylesheet" href="../css/Profile-Edit1.css">

   <link rel="stylesheet" href="../css/Pretty-Registration-Form.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack3.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack2.css">

   <link rel="stylesheet" href="../css/Navigation-with-Button1.css">

   <link rel="stylesheet" href="../css/PUSH---Bootstrap-Button-Pack.css">

   <link rel="stylesheet" href="../css/styles.css">

</head>

<?php

   include 'header.php';

   $studentID = $\_GET['studentID'];

   $query = "SELECT \* FROM Student WHERE studentID= '$studentID'";

   $result = $db->query($query);

   $row = $result->fetch\_array(MYSQLI\_ASSOC);

   $userId = $row['userId'];

   $query1 = "SELECT username FROM User WHERE id= '$userId'";

   $result1 = $db->query($query1);

   $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

   $oldUsername = $row1['username'];

   $firstName = $lastName = $username = $email = "";

   //sanitize inputs

   if (isset($\_POST['update\_button'])) {

       $firstName = test\_input($\_POST["firstName"]);

       $lastName = test\_input($\_POST["lastName"]);

       $username = test\_input($\_POST["username"]);

       $email = test\_input($\_POST["email"]);

       if ($username != $oldUsername){

           $sql = "SELECT username FROM User WHERE username = '$username'";

               $result = mysqli\_query($db, $sql) or die('error getting data');

               $num\_rows = mysqli\_num\_rows($result);

               //Check if user already exists

               if ($result->num\_rows > 0){

                   $takenusername = "User already exists!";

                   exit;

               }

       }

       $sql2 = "UPDATE User SET username=? WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('sd', $username, $userId);

       $stmt1->execute();

       $sql1 = "UPDATE Student SET firstName=?, lastName=?, email=? WHERE userId=?";

       //Prepared Statement

       $stmt =  $db->stmt\_init();

       $stmt->prepare($sql1);

       $stmt->bind\_param('sssd', $firstName, $lastName, $email, $userId);

       $stmt->execute();

       //submit success message

       $success = "Student Account Updated!";

       //refresh data

       $studentID = $\_GET['studentID'];

       $query = "SELECT \* FROM Student WHERE studentID= '$studentID'";

       $result = $db->query($query);

       $row = $result->fetch\_array(MYSQLI\_ASSOC);

       $query1 = "SELECT username FROM User WHERE id= '$userId'";

       $result1 = $db->query($query1);

       $row1 = $result1->fetch\_array(MYSQLI\_ASSOC);

       $oldUsername = $row1['username'];

   }

   else if (isset($\_POST['delete\_button'])){

       $sql1 = "DELETE FROM Student WHERE userId=?";

       //Prepared Statement

       $stmt =  $db->stmt\_init();

       $stmt->prepare($sql1);

       $stmt->bind\_param('d', $userId);

       $stmt->execute();

       $sql2 = "DELETE FROM User WHERE id=?";

       $stmt1 = $db->stmt\_init();

       $stmt1->prepare($sql2);

       $stmt1->bind\_param('d', $userId);

       $stmt1->execute();

       echo '<script type="text/javascript">

          window.location.replace("student-update-delete.php");

               </script>';

   }

   //sanitize input

   function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<body>

   <div class="container profile profile-view" id="profile">

       <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]) . "?studentID=" . $\_GET['studentID'];?>">

           <div class="row profile-row">

               <div class="col-md-12">

                   <h2 class="text-center">Student Profile</h2>

                   <hr>

                   <div class="row">

                       <div class="col-md-6 col-sm-12">

                           <div class="form-group">

                               <label class="control-label">First name </label>

                               <input class="form-control" type="text" required="" name="firstName" value='<?php echo $row['firstName']; ?>'>

                           </div>

                       </div>

                       <div class="col-md-6 col-sm-12">

                           <div class="form-group">

                               <label class="control-label">Last name </label>

                               <input class="form-control" type="text" required="" name="lastName" value='<?php echo $row['lastName']; ?>'>

                           </div>

                       </div>

                   </div>

                   <div class="form-group">

                       <label class="control-label">Username </label>

                       <input class="form-control" type="text" required="" autocomplete="off" name="username" value='<?php echo $oldUsername; ?>'>

                   </div>

                   <div class="form-group">

                       <label class="control-label">Email </label>

                       <input class="form-control" type="email" required="" autocomplete="off" name="email" value='<?php echo $row['email']; ?>'>

                   </div>

                   <div class="row">

                       <p class="text-danger text-center"><?php echo $takenusername  ?></p>

                       <p class="text-success text-center"><?php echo $success ?></p>

                   </div>

                   <hr>

                   <div class="row">

                       <div class="col-md-12 content-right">

                           <button class="btn btn-primary form-btn" name="update\_button" type="submit">UPDATE </button>

                           <button class="btn btn-danger form-btn" name="delete\_button" type="submit">DELETE </button>

                       </div>

                   </div>

               </div>

           </div>

       </form>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* CSS
  + Messaging.css

.container

{

    margin-top: 70px;

}

body

{

    background-color: #77abff;

}

.col

{

}

/\* Clear floats after the columns \*/

.row:after {

   content: "";

   display: table;

   clear: both;

}

.whiteboard

{

    background-color: white;

    border: 7px solid #ffcc66;

   border-radius: 10px;

   margin: 20px;

    padding: 0px;

}

.instructor

{

    text-align: right;

}

.whiteboard-main

{

    padding: 0px;

}

footer

{

    color: white;

    background-color: #333;

    padding: 20px;

    border: 7px solid #333;

   border-radius: 10px;

}

.whiteboard-iframe

{

    background-color: white;

    padding: 60px;

}

iframe

{

    border: 0;

    width: 100vh;

    height: 50vh;

}

.whiteboard-nav

{

    background-color: white;

}

.nav-elem

{

    color: black;

}

.nav-elem:hover

{

    background-color: #e6e6e6;

    color: #333;

    border: 4px solid #e6e6e6;

   border-radius: 10px;

}

.nav-curr

{

    background-color: #e6e6e6;

    color: #333;

    border: 4px solid #e6e6e6;

   border-radius: 10px;

}

/\* Chat Code \*/

.chatlogs {

    width: 660px;

    height: 360px;

    background-color: white;

    overflow-x: hidden;

    overflow-y: scroll;

}

.chat {

    display: felx;

    flex-flow: row wrap;

    align-items: flex-start;

    margin-bottom: 10px;

}

.chat .user-photo{

    width: 60px;

    height: 60px;

    background: #ccc;

    border-radius: 50%;

    overflow: hidden;

}

.chat .user-photo img{

    width: 100%;

}

.chat .chat-message {

    width: 70%;

    padding: 15px;

    margin: 5px 10px 0;

    border-radius: 10px;

    color: #fff;

    font-size: 18px;

}

.self .chat-message {

    background: #77abff;

    order: -1;

}

.contact .chat-message {

    background: #d3d3d3;

}

.chat-form {

    margin-top: 20px;

    display: flex;

    align-items: flex-start;

}

.chat-form textarea{

    background: #fff;

    width: 75%;

    height: 50px;

    border: 2px solid #d3d3d3;

    border-radius: 3px;

    resize: none;

    padding 10px;

    font-size: 18px;

}

.chat-form button {

    background: #77abff;

    padding: 5px 15px;

    font-size: 26px;

    color: #fff;

    border:none;

    margin: 0 10px;

    border-radius: 3px;

}

/\* Table code \*/

table.messagingNames {

    background-color: white;

    border: 7px solid #ffcc66;

   border-radius: 10px;

   margin: 20px;

    padding: 20px;

 width: 30%;

 text-align: left;

 border-collapse: collapse;

}

table.messagingNames td, table.messagingNames th {

 border: 1px solid #AAAAAA;

 padding: 10px 4px;

}

table.messagingNames tbody td {

 font-size: 15px;

}

table.messagingNames thead {

   border-bottom: 2px solid #444444;

    border-radius: 10px;

}

table.messagingNames thead th {

 font-size: 15px;

 font-weight: normal;

 border-left: 2px solid #D0E4F5;

}

table.messagingNames thead th:first-child {

 border-left: none;

}

table.messagingNames tfoot td {

 font-size: 14px;

}

table.messagingNames tfoot .links {

 text-align: right;

}

table.messagingNames tfoot .links a{

 display: inline-block;

 background: #1C6EA4;

 color: #FFFFFF;

 padding: 4px 16px;

 border-radius: 5px;

}

.messagingConversation

{

    padding: 20px;

}

.grades

{

    font-size: 30px;

    padding-bottom: 20px;

}

.ingrade

{

    padding-bottom: 10px;

}

.colorinput

{

    margin: 10px;

}

* + Whiteboard.css

.container

{

    margin-top: 70px;

}

body

{

    background-color: #77abff;

}

.col

{

}

/\* Clear floats after the columns \*/

.row:after {

   content: "";

   display: table;

   clear: both;

}

.whiteboard

{

    background-color: white;

    border: 7px solid #ffcc66;

   border-radius: 10px;

   margin: 20px;

    padding: 0px;

}

.instructor

{

    text-align: right;

}

.whiteboard-main

{

    padding: 0px;

}

footer

{

    color: white;

    background-color: #333;

    padding: 20px;

    border: 7px solid #333;

   border-radius: 10px;

}

.whiteboard-iframe

{

    background-color: white;

    padding: 60px;

}

iframe

{

    border: 0;

    width: 100vh;

    height: 50vh;

}

.whiteboard-nav

{

    background-color: white;

}

.nav-elem

{

    color: black;

}

.nav-elem:hover

{

    background-color: #e6e6e6;

    color: #333;

    border: 4px solid #e6e6e6;

   border-radius: 10px;

}

.nav-curr

{

    background-color: #e6e6e6;

    color: #333;

    border: 4px solid #e6e6e6;

   border-radius: 10px;

}

/\* Chat Code \*/

.chatlogs {

    width: 660px;

    height: 360px;

    background-color: white;

    overflow-x: hidden;

    overflow-y: scroll;

}

.chat {

    display: felx;

    flex-flow: row wrap;

    align-items: flex-start;

    margin-bottom: 10px;

}

.chat .user-photo{

    width: 60px;

    height: 60px;

    background: #ccc;

    border-radius: 50%;

    overflow: hidden;

}

.chat .user-photo img{

    width: 100%;

}

.chat .chat-message {

    width: 70%;

    padding: 15px;

    margin: 5px 10px 0;

    border-radius: 10px;

    color: #fff;

    font-size: 18px;

}

.self .chat-message {

    background: #77abff;

    order: -1;

}

.contact .chat-message {

    background: #d3d3d3;

}

.chat-form {

    margin-top: 20px;

    display: flex;

    align-items: flex-start;

}

.chat-form textarea{

    background: #fff;

    width: 75%;

    height: 50px;

    border: 2px solid #d3d3d3;

    border-radius: 3px;

    resize: none;

    padding 10px;

    font-size: 18px;

}

.chat-form button {

    background: #77abff;

    padding: 5px 15px;

    font-size: 26px;

    color: #fff;

    border:none;

    margin: 0 10px;

    border-radius: 3px;

}

/\* Table code \*/

table.messagingNames {

    background-color: white;

    border: 7px solid #ffcc66;

   border-radius: 10px;

   margin: 20px;

    padding: 20px;

 width: 30%;

 text-align: left;

 border-collapse: collapse;

}

table.messagingNames td, table.messagingNames th {

 border: 1px solid #AAAAAA;

 padding: 10px 4px;

}

table.messagingNames tbody td {

 font-size: 15px;

}

table.messagingNames thead {

   border-bottom: 2px solid #444444;

    border-radius: 10px;

}

table.messagingNames thead th {

 font-size: 15px;

 font-weight: normal;

 border-left: 2px solid #D0E4F5;

}

table.messagingNames thead th:first-child {

 border-left: none;

}

table.messagingNames tfoot td {

 font-size: 14px;

}

table.messagingNames tfoot .links {

 text-align: right;

}

table.messagingNames tfoot .links a{

 display: inline-block;

 background: #1C6EA4;

 color: #FFFFFF;

 padding: 4px 16px;

 border-radius: 5px;

}

.messagingConversation

{

    padding: 20px;

}

.grades

{

    font-size: 30px;

    padding-bottom: 20px;

}

.ingrade

{

    padding-bottom: 10px;

}

.colorinput

{

    margin: 10px;

}

* + Styles.css

.stylish-input-group .input-group-addon{

   background: white !important;

}

.stylish-input-group .form-control{

    border-right:0;

    box-shadow:0 0 0;

    border-color:#ccc;

}

.stylish-input-group button{

   border:0;

   background:transparent;

}

td:hover{

background-color:gray;

cursor:pointer;

}

* + Login-Form-Clean.css

.login-clean {

 background:#f1f7fc;

 padding:80px 0;

}

.login-clean form {

 max-width:320px;

 width:90%;

 margin:0 auto;

 background-color:#ffffff;

 padding:40px;

 border-radius:4px;

 color:#505e6c;

 box-shadow:1px 1px 5px rgba(0,0,0,0.1);

}

.login-clean .illustration {

 text-align:center;

 padding:0 0 20px;

 font-size:100px;

 color:rgb(119,119,119);

}

.login-clean form .form-control {

 background:#f7f9fc;

 border:none;

 border-bottom:1px solid #dfe7f1;

 border-radius:0;

 box-shadow:none;

 outline:none;

 color:inherit;

 text-indent:8px;

 height:42px;

}

.login-clean form .btn-primary {

 background:#f4476b;

 border:none;

 border-radius:4px;

 padding:11px;

 box-shadow:none;

 margin-top:26px;

 text-shadow:none;

 outline:none !important;

}

.login-clean form .btn-primary:hover, .login-clean form .btn-primary:active {

 background:#eb3b60;

}

.login-clean form .btn-primary:active {

 transform:translateY(1px);

}

.login-clean form .forgot {

 display:block;

 text-align:center;

 font-size:12px;

 color:#6f7a85;

 opacity:0.9;

 text-decoration:none;

}

.login-clean form .forgot:hover, .login-clean form .forgot:active {

 opacity:1;

 text-decoration:none;

}

* + Pretty-Registration-Form.css

.register-form form.custom-form {

 padding:55px;

 box-sizing:border-box;

 background-color:#ffffff;

 box-shadow:0 1px 3px 0 rgba(0, 0, 0, 0.1);

 font:bold 14px sans-serif;

 text-align:center;

 margin:50px;

 color:#333;

}

@media (max-width:400px) {

 .register-form form.custom-form {

   padding:55px 10px;

 }

}

.register-form .custom-form h1 {

 display:inline-block;

 color:#4c565e;

 font-size:24px;

 font-weight:bold;

 padding:0 10px 15px;

 margin-bottom:60px;

 border-bottom:2px solid rgb(108, 174, 224);

}

.register-form .custom-form .form-group {

 margin-bottom:25px;

}

.register-form .custom-form .label-column {

 text-align:right;

 color:#5F5F5F;

}

@media (max-width:768px) {

 .register-form .custom-form .label-column {

   text-align:left;

 }

}

.register-form .custom-form .input-column {

 color:#5f5f5f;

 text-align:left;

}

.register-form .custom-form .input-column input {

 color:#5f5f5f;

 box-shadow:1px 2px 4px 0 rgba(0, 0, 0, 0.08);

 padding:12px;

 border:1px solid #dbdbdb;

 border-radius:2px;

 height:42px;

}

.register-form .custom-form .dropdown .dropdown-toggle {

 background:#fff;

 border:1px solid #dbdbdb;

 box-shadow:1px 2px 4px 0 rgba(0, 0, 0, 0.08);

 color:#333;

 outline:none;

}

.register-form .custom-form .dropdown ul {

 background:#fff;

}

.register-form .custom-form .dropdown ul li a {

 background:#fff;

 color:#333;

 opacity:0.8;

}

.register-form .custom-form .dropdown ul li a:hover {

 opacity:1;

}

.register-form .custom-form .submit-button {

 border-radius:2px;

 background:#6caee0;

 color:#ffffff;

 font-weight:bold;

 box-shadow:1px 2px 4px 0 rgba(0, 0, 0, 0.08);

 padding:14px 22px;

 border:0;

 margin:30px;

 outline:none;

}

* + Profile-Edit.css

div.container.profile {

 padding:10px 20px;

}

.center {

 text-align:center;

}

.relative {

 position:relative;

}

.absolue {

 position:absolute;

}

div.profile-row {

 margin-top:55px;

}

.absolue.center {

 left:50%;

 top:50%;

 transform:translate(-50%, 0%);

}

div.alert {

 width:40%;

}

.alert-col > .alert {

 padding:10px;

 border-radius:0;

 transition:all 0.5s ease;

 opacity:0;

}

hr {

 height:2px;

}

.content-right > \* {

 float:right;

}

.form-btn {

 min-width:100px;

 margin:0 5px;

 max-width:200px;

}

div.avatar-bg {

 background:url(https://www.gravatar.com/avatar/1234566?size=200&d=mm);

 background-position:50% 50%;

 height:200px;

 width:200px;

 background-size:cover;

 border-radius:50%;

 margin-left:calc(50% - 100px);

}

* + Profile-Edit1.css

div.avatar > img {

 border-radius:50%;

}

div.container.profile {

 padding:10px 20px;

}

.center {

 text-align:center;

}

.no-marging {

 margin:0 !important;

}

.relative {

 position:relative;

}

.absolue {

 position:absolute;

}

div.profile-row {

 margin-top:55px;

}

.absolue.center {

 left:50%;

 top:50%;

 transform:translate(-50%, 0%);

}

div.alert {

 width:40%;

}

.alert-col {

}

.alert-col > .alert {

 padding:10px;

 border-radius:0;

 transition:all 0.5s ease;

 opacity:0;

}

.alert-col > .alert.display {

 opacity:1;

}

hr {

 height:2px;

}

.content-right > \* {

 float:right;

}

.form-btn {

 min-width:100px;

 margin:0 5px;

 max-width:200px;

}

img.avart-img {

 max-width:200px;

}

div.avatar-bg {

 background:url(https://www.gravatar.com/avatar/1234566?size=200&d=mm);

 background-position:50% 50%;

 height:200px;

 width:200px;

 background-size:cover;

 border-radius:50%;

 margin-left:calc(50% - 100px);

}

div.avatar {

}

* Student
  + Announcement-view.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$announcement\_id = $\_GET['id'];

$sql = "SELECT \* FROM Announcement WHERE id =$announcement\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$timestamp = date("m-d-y", strtotime($row['timestamp']));

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Announcement</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px"><?php echo $row['title']; ?></h4>

               <p class="lead"><?php echo $row['body'] ?></p>

               <p class="lead">Posted: <?php echo $timestamp ?></p>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Announcements.php

<?php

include '../session/student-session.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

<?php

   include '../functions/html\_element.php';

?>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem nav-curr" href="announcements.php">Feed</a>

       <a class="btn navbar-brand nav-elem" href="inbox.php">Inbox</a>

       <a class="btn navbar-brand nav-elem" href="homework.php">Homework</a>

     </div>

   </nav>

   <?php

   $num\_of\_courses = count($courseArray);

   for ($i = 0; $i < $num\_of\_courses; $i++) {

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $header = new html\_element('h4');

      $header->set('text',$courseArray[$i]->get\_name());

      $column->inject($header);

      $courseId = $courseArray[$i]->get\_id();

      $sql = "SELECT id, title, body, timestamp FROM Announcement WHERE courseID = '$courseId' ORDER BY timestamp DESC LIMIT 3";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

        $image = new html\_element('i');

        $image->set('class', 'fa fa-bullhorn');

        $image->set('aria-hidden', 'true');

        $title = new html\_element('p');

        $title->set('class','lead');

        $title->set('text',$row['title']);

        $title->inject($image);

        $link = new html\_element('a');

        $link->set('href', 'announcement-view.php?id=' . $row['id']);

        $link->set('style', 'color:inherit; text-decoration: none');

        $link->set('target', '\_parent');

        $link->inject($title);

        $column->inject($link);

        $postDate = date("m-d-y", strtotime($row['timestamp']));

        $time = new html\_element('p');

        $time->set('text', "Posted: $postDate");

        $column->inject($time);

      }

      $row = new html\_element('div');

      $row->set('class','row');

      $row->inject($column);

      $row->output();

   }

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Chat.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$id = $\_GET['id'];

if(isset($\_POST['content'])){

   $content = htmlspecialchars($\_POST['content']);

   $sql = 'INSERT INTO message (content, reciever, creator) VALUES (?, ?, ?)';

   console\_log($sql);

   $stmt = $db->stmt\_init();

   $stmt->prepare($sql);

   $stmt->bind\_param('sdd', $content, $id, $user\_check);

   $stmt->execute();

}

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <?php

   $sql = 'SELECT \* FROM (SELECT content, creator, reciever, timestamp FROM message WHERE reciever =  ' . $user\_check . ' AND creator =' . $id . ') as FirstSet

         union (

         SELECT content, creator, reciever, timestamp FROM message WHERE reciever =' . $id . ' AND creator = ' . $user\_check . ') ORDER BY timestamp';

   $result = $db->query($sql);

   while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

     if($row['creator'] == $user\_check){

         $div = new html\_element('div');

         $div->set('class', 'row');

         $col = new html\_element('div');

         $col->set('class', 'col-md-8');

         $card = new html\_element('div');

         $card->set('class', 'card');

         $body = new html\_element('div');

         $body->set('class', 'card-body bg-primary');

         $content = new html\_element('p');

         $content->set('class', 'p-y-1 text-white pull-right');

         $content->set('text', $row['content']);

         $body->inject($content);

         $card->inject($body);

         $col->inject($card);

         $col2 = new html\_element('div');

         $col2->set('class', 'col-md-4');

         $div->inject($col2);

         $div->inject($col);

         $div->output();

     }

     else{

         $div = new html\_element('div');

         $div->set('class', 'row');

         $col = new html\_element('div');

         $col->set('class', 'col-md-8');

         $card = new html\_element('div');

         $card->set('class', 'card');

         $body = new html\_element('div');

         $body->set('class', 'card-body ');

         $content = new html\_element('p');

         $content->set('class', 'p-y-1');

         $content->set('text', $row['content']);

         $body->inject($content);

         $card->inject($body);

         $col->inject($card);

         $col2 = new html\_element('div');

         $col2->set('class', 'col-md-4');

         $div->inject($col);

         $div->inject($col2);

         $div->output();

     }

   }

     ?>

     <script>

         function scrollBottom() {window.scrollTo(0, document.body.scrollHeight);}

           if (document.addEventListener) document.addEventListener("DOMContentLoaded", scrollBottom, false)

           else if (window.attachEvent) window.attachEvent("onload", scrollBottom);

     </script>

     <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js" integrity="sha384-b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4" crossorigin="anonymous"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Content-view.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$content\_id = $\_GET['id'];

$sql = "SELECT \* FROM content WHERE id =$content\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$timestamp = date("m-d-y", strtotime($row['timestamp']));

$sql\_file = "SELECT \* FROM content\_file WHERE content\_id = $content\_id";

$result\_file = $db->query($sql\_file);

$num\_rows\_file = mysqli\_num\_rows($result\_file);

   if ($num\_rows\_file>=1){

       $row\_file = $result\_file->fetch\_array(MYSQLI\_ASSOC);

       $file = $row\_file['file'];

       }

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Content</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px"><?php echo $row['title']; ?></h4>

             <a href='../staff/uploads/<?php echo $file ?>'><?php echo $file ?> </a>

               <p class="lead"><?php echo $row['body'] ?></p>

               <p class="lead">Posted: <?php echo $timestamp ?></p>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-announcement.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem nav-curr" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, body, timestamp FROM Announcement WHERE courseID = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

        $image = new html\_element('i');

        $image->set('class', 'fa fa-bullhorn');

        $image->set('aria-hidden', 'true');

        $title = new html\_element('p');

        $title->set('class','lead');

        $title->set('text',$row['title']);

        $title->inject($image);

        $link = new html\_element('a');

        $link->set('href', 'announcement-view.php?id=' . $row['id']);

        $link->set('style', 'color:inherit; text-decoration: none');

        $link->set('target', '\_parent');

        $link->inject($title);

        $column->inject($link);

        $postDate = date("m-d-y", strtotime($row['timestamp']));

        $time = new html\_element('p');

        $time->set('text', "Posted: $postDate");

        $column->inject($time);

      }

      $row = new html\_element('div');

      $row->set('class','row');

      $row->inject($column);

      $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-assignment.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, body, timestamp, dueDate FROM Assignment WHERE courseID = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $image = new html\_element('i');

           $image->set('class', 'fa fa-pencil');

           $image->set('aria-hidden', 'true');

           $title = new html\_element('p');

           $title->set('class','lead');

           $title->set('text',$row['title']);

           $title->inject($image);

           $link = new html\_element('a');

           $link->set('href', 'assignment.php?id=' . $row['id']);

           $link->set('style', 'color:inherit; text-decoration: none');

           $link->set('target', '\_parent');

           $link->inject($title);

           $column->inject($link);

           $dueDate = date("m-d-y", strtotime($row[3]));

           $due = new html\_element('p');

           $due->set('text', "Due: $dueDate");

           $column->inject($due);

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-content.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, timestamp FROM content WHERE course\_id = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $image = new html\_element('i');

           $image->set('class', 'fa fa-sticky-note-o');

           $image->set('aria-hidden', 'true');

           $title = new html\_element('p');

           $title->set('class','lead');

           $title->set('text',$row['title']);

           $title->inject($image);

           $link = new html\_element('a');

           $link->set('href', 'content-view.php?id=' . $row['id']);

           $link->set('style', 'color:inherit; text-decoration: none');

           $link->set('target', '\_parent');

           $link->inject($title);

           $column->inject($link);

           $dueDate = date("m-d-y", strtotime($row['timestamp']));

           $due = new html\_element('p');

           $due->set('text', "Posted: $dueDate");

           $column->inject($due);

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-roster.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

$courseId = $courseArray[$location]->get\_id();

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

           <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script

     src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"

     integrity="sha256-0YPKAwZP7Mp3ALMRVB2i8GXeEndvCq3eSl/WsAl1Ryk="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js"></script>

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

   </nav>

<?php

      //create assignment object

      $student\_id\_array = array();

      $column = new html\_element('div');

      $column->set('class','col');

      $sql = "SELECT student\_id FROM Student\_Course\_Junction WHERE course\_id = '$courseId'";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

          $student\_id\_array[] = $row['student\_id'];

       }

       $student\_id\_array = implode(" OR ", $student\_id\_array);

       if ($num\_rows > 0){

       $sql = "SELECT firstName, lastName FROM Student WHERE studentID= $student\_id\_array ORDER BY lastName";

       $result = mysqli\_query($db, $sql) or die('error getting data');

       while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $name = new html\_element('p');

           $name->set('class', 'lead');

           $name->set('text', $row['firstName'] . " " . $row['lastName']);

           $column->inject($name);

       }

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

 include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

           <div class="col whiteboard whiteboard-main gradebook">

               <h3><?php echo $courseArray[$location]->get\_name() . " // " . $courseArray[$location]->get\_description(); ?></h3>

               <iframe style="width: 100%;height: 100%;position: relative;"  src="course-announcement.php?location=<?php echo $location ?>"></iframe>

           </div>

       </div>

         <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

     </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Get-contacts.php

<?php

include '../session/student-session.php';

include '../functions/console\_log.php';

$term = $\_GET['term'];

$id\_array = array();

for ($i=0; $i<count($courseArray); $i++){

    array\_push($id\_array, $courseArray[$i]->get\_id());

}

$id\_array = implode(" OR ", $id\_array);

$sql = "SELECT DISTINCT firstName, lastName, userId FROM Student JOIN Student\_Course\_Junction ON Student.studentID=Student\_Course\_Junction.student\_id WHERE course\_id = ($id\_array) AND (concat\_ws(' ', firstName, lastName) LIKE '%$term%') AND NOT userId=$user\_check";

$result = $db->query($sql);

$json = array();

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $name = $row['firstName'] . " " . $row['lastName'];

    $json[] = array(

        'value' => 'message.php?id=' . $row['userId'] . '&name=' . $name,

        'label' => $name);

}

$sql = "SELECT DISTINCT firstName, lastName, userId FROM Course JOIN Staff ON Course.staffID=Staff.id WHERE courseID = ($id\_array) AND (concat\_ws(' ', firstName, lastName) LIKE '%$term%')";

$result = $db->query($sql);

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $name = $row['firstName'] . " " . $row['lastName'];

    $json[] = array(

        'value' => 'message.php?id=' . $row['userId'] . '&name=' . $name,

        'label' => $name);

}

echo json\_encode($json);

?>

* + Get-events.php

<?php

include '../session/student-session.php';

include '../functions/console\_log.php';

$id\_array = array();

for ($i=0; $i<count($courseArray); $i++){

    array\_push($id\_array, $courseArray[$i]->get\_id());

}

$id\_array = implode(" OR ", $id\_array);

$sql = "SELECT title, start, reference\_id FROM event WHERE course\_id = $id\_array";

$result = $db->query($sql);

$json = array();

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $json[] = array(

       'title' => $row['title'],

       'start' => $row['start'],

       'url' => 'assignment.php?id=' . $row['reference\_id'],

       'allDay' => true);

}

$sql = "SELECT title, start, announcement\_id FROM event\_announcement WHERE course\_id = $id\_array";

$result = $db->query($sql);

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $json[] = array(

       'title' => $row['title'],

       'start' => $row['start'],

       'url' => 'announcement-view.php?id=' . $row['announcement\_id'],

       'color' => 'red',

       'allDay' => true);

}

echo json\_encode($json);

?>

* + Gradebook.php

<?php

include '../session/student-session.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <?php

 include 'header.php';

 include '../functions/html\_element.php';

?>

 <body style="background-color: <?php echo $session\_primaryColor; ?>" >

<div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

          <p class="lead grades"> My Grades <i class="fa fa-calculator" aria-hidden="true"></i></p>

           <div class="row">

             <div class="col" id="current">

               <?php

               //create html class objects

               $num\_of\_courses = count($courseArray);

               for ($i = 0; $i < $num\_of\_courses; $i++){

                 $sql = "SELECT SUM(grade), SUM(points) FROM Grade JOIN Submission

                         ON Grade.submissionID = Submission.id

                         JOIN Assignment ON Submission.assignmentId = Assignment.id WHERE studentID =" . $session\_studentId . " AND courseID =" . $courseArray[$i]->get\_id();

                 $result = mysqli\_query($db, $sql) or die('error getting data');

                 $row = mysqli\_fetch\_array($result, MYSQLI\_BOTH);

                 $totalGrade = $row[0];

                 $totalPoints = $row[1];

                 $class = new html\_element('p');

                 $class->set('class', 'lead ingrade');

                 $class->set('id', $i);

                 $class->set('onclick', "loadClass(this)");

                 $class->set('text',$courseArray[$i]->get\_name() . " - " . $courseArray[$i]->get\_description() . " - " . $totalGrade . " / " . $totalPoints);

                 $class->output();

               }

               /\* Original Html

               <p class="lead ingrade" id="cs360" onclick="loadClass(this)"> Software Engineering I - 60%</p>

               <p class="lead ingrade" id="cs10" onclick="loadClass(this)"> Basic Computer Literacy - 20%</p>

               <p class="lead ingrade" id="cheese" onclick="loadClass(this)"> Cheese Production - Fail</p>

               <p class="lead ingrade" id="lib" onclick="loadClass(this)"> Liberal Arts - 200%</p>

               \*/

               ?>

             </div>

             <div class="col" id="courseinfo">

              <p class="lead ingrade" id="recent"></p>

               <a id="link1" style='color:inherit; text-decoration: none'><p class="lead" id="grade1"></p></a>

               <a id="link2" style='color:inherit; text-decoration: none'><p class="lead" id="grade2"> </p></a>

               <a id="link3" style='color:inherit; text-decoration: none'><p class="lead" id="grade3"></p></a>

               <p class="lead" id="calculation"> </p>

             </div>

           </div>

         </div>

       </div>

       <script type="text/javascript">

       class Gradebook

       {

         constructor(grades)

         {

           this.grades = grades;

         }

       }

       class Grade

       {

         constructor(name, assignments, grade)

         {

           this.name = name;

           this.assignments = assignments;

           this.grade = grade;

         }

       }

       var courses = new Array();

       var grades1 = new Array();

       var grades2 = new Array();

       var grades3 = new Array();

       var titles1 = new Array();

       var titles2 = new Array();

       var titles3 = new Array();

       var id1 = new Array();

       var id2 = new Array();

       var id3 = new Array();

       <?php

       $num\_of\_courses = count($courseArray);

       for ($i = 0; $i < $num\_of\_courses; $i++){

         $sql = "SELECT grade, title, points, assignmentId FROM Grade JOIN Submission

               ON Grade.submissionID = Submission.id

               JOIN Assignment ON Submission.assignmentId = Assignment.id WHERE studentID =" . $session\_studentId . " AND courseID =" . $courseArray[$i]->get\_id() . "

               ORDER BY Grade.timestamp DESC LIMIT 3";

         $result = mysqli\_query($db, $sql) or die('error getting data');

         $row = mysqli\_fetch\_array($result, MYSQLI\_BOTH);

         if(isset($row[0])){ ?>

           grades1.push('<?php echo $row['grade'] . " / " . $row['points']; ?>');

           id1.push('<?php echo $row['assignmentId']; ?>')

           titles1.push('<?php echo $row['title']; ?>');

         <?php }

         $row = mysqli\_fetch\_array($result, MYSQLI\_BOTH);

         if(isset($row[0])){ ?>

           grades2.push('<?php echo $row['grade'] . " / " . $row['points']; ?>');

           id2.push('<?php echo $row['assignmentId']; ?>')

           titles2.push('<?php echo $row['title']; ?>');

         <?php }

         $row = mysqli\_fetch\_array($result, MYSQLI\_BOTH);

         if(isset($row[0])){ ?>

           grades3.push('<?php echo $row['grade'] . " / " . $row['points']; ?>');

           id3.push('<?php echo $row['assignmentId']; ?>')

           titles3.push('<?php echo $row['title']; ?>');

         <?php }

         ?>

         courses.push('<?php echo $courseArray[$i]->get\_name(); ?>');

       <?php } ?>

       var comments = ["1/3 of a page too short", "Nice", "", ""];

       //var need = ["","","N/A, just give up",""];

        var a = [1,2,3,4];

        var x = document.getElementById("cs360");

        //loads course information

        function loadClass(c)

        {

           recent.innerHTML = "Most recent grades for " + courses[c.id];

           if (typeof titles1[c.id] != 'undefined'){

             grade1.innerHTML = titles1[c.id] + ": " + grades1[c.id];

             document.getElementById("link1").href="assignment.php?id=" + id1[c.id];

           }

           else{

             grade1.innerHTML = "";

           }

           if (typeof titles2[c.id] != 'undefined'){

             grade2.innerHTML = titles2[c.id] + ": " + grades2[c.id];

             document.getElementById("link2").href="assignment.php?id=" + id2[c.id];

           }

           else{

             grade2.innerHTML = "";

           }

           if (typeof titles3[c.id] != 'undefined'){

             grade3.innerHTML = titles3[c.id] + ": " + grades3[c.id];

             document.getElementById("link3").href="assignment.php?id=" + id3[c.id];

           }

           else{

             grade3.innerHTML = "";

           }

           //calculation.innerHTML = "What you need on upcoming assignments to raise your grade: " + need[courses.indexOf(c.id)];

        }

        function selectClass(c) {

          c.className += " updated";

//          document.querySelectorAll(".updated")

        }

       </script>

   <footer> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Header.php

<!DOCTYPE html>

<body>

   <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">

     <a class="navbar-brand" href="index.php">WhiteBoard</a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarsExampleDefault">

       <ul class="navbar-nav mr-auto">

         <li class="nav-item">

           <a class="nav-link" href="whiteboardcalendar.php">Calendar <span class="sr-only">(current)</span></a>

         </li>

         <li class="nav-item">

           <a class="nav-link" href="gradebook.php">Grades</a>

         </li>

         <li class="nav-item">

         </li>

       </ul>

       <a class="btn btn-default action-button navbar-text navbar-right actions" role="button" href="../logout.php">Log Out</a>

     </div>

   </nav>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

* + Homework.php

<?php

include '../session/student-session.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <?php

   include '../functions/html\_element.php';

 ?>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="announcements.php">Feed</a>

       <a class="btn navbar-brand nav-elem" href="inbox.php">Inbox</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="homework.php">Homework</a>

     </div>

   </nav>

   <?php

   $num\_of\_courses = count($courseArray);

   for ($i = 0; $i < $num\_of\_courses; $i++) {

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $header = new html\_element('h4');

      $header->set('text',$courseArray[$i]->get\_name());

      $column->inject($header);

      $courseId = $courseArray[$i]->get\_id();

      $sql = "SELECT id, title, body, timestamp, dueDate FROM Assignment WHERE courseID = '$courseId' ORDER BY timestamp DESC LIMIT 3";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

        $image = new html\_element('i');

        $image->set('class', 'fa fa-pencil');

        $image->set('aria-hidden', 'true');

        $title = new html\_element('p');

        $title->set('class','lead');

        $title->set('text',$row['title']);

        $title->inject($image);

        $link = new html\_element('a');

        $link->set('target', '\_parent');

        $link->set('style', 'color:inherit; text-decoration: none');

        $link->set('href', "assignment.php?id=" . $row['id']);

        $link->inject($title);

        $column->inject($link);

        $dueDate = date("m-d-y", strtotime($row[3]));

        $due = new html\_element('p');

        $due->set('text', "Due: $dueDate");

        $column->inject($due);

      }

      $row = new html\_element('div');

      $row->set('class','row');

      $row->inject($column);

      $row->output();

   }

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Inbox.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

       <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script

     src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"

     integrity="sha256-0YPKAwZP7Mp3ALMRVB2i8GXeEndvCq3eSl/WsAl1Ryk="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js"></script>

   <script>

   $(document).ready(function() {

   $("#autoComplete").autocomplete({

     source: "get-contacts.php",

     select: function(event, ui){

       window.top.location.href = ui.item.value;

     }

       });

   });

       </script>

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="announcements.php">Feed</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="inbox.php">Inbox</a>

       <a class="btn navbar-brand nav-elem" href="homework.php">Homework</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <button class="btn btn-outline-secondary" style="margin-left:45px"  data-toggle='modal' data-target="#myModal">Create Message</button>

     </div>

   </nav>

 <!-- Modal -->

 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

   <div class="modal-dialog" role="document">

     <div class="modal-content">

       <div class="modal-header">

         <h5 class="modal-title" id="exampleModalLabel">Create Message</h5>

         <button type="button" class="close" data-dismiss="modal" aria-label="Close">

           <span aria-hidden="true">&times;</span>

         </button>

       </div>

       <div class="modal-body">

         <p>Send to who?</p>

         <input id='autoComplete'/>

       </div>

       <div class="modal-footer">

         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

       </div>

     </div>

   </div>

 </div>

 <?php

 $sql = 'SELECT SUBSTRING(content, 1, 45) AS content, creator, timestamp FROM message WHERE reciever =' . $user\_check . ' GROUP BY(creator) DESC';

 $result = $db->query($sql);

 while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

   //get creator name

   $creator\_sql = 'SELECT role FROM User WHERE id=' . $row['creator'];

   $creator\_result = $db->query($creator\_sql);

   $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

   if ($creator\_row['role'] == 'staff'){

     $creator\_sql = 'SELECT firstName, lastName FROM Staff WHERE userId=' . $row['creator'];

     $creator\_result = $db->query($creator\_sql);

     $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

     $name = $creator\_row['firstName'] . ' ' . $creator\_row['lastName'];

   }

   else{

     $creator\_sql = 'SELECT firstName, lastName FROM Student WHERE userId=' . $row['creator'];

     $creator\_result = $db->query($creator\_sql);

     $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

     $name = $creator\_row['firstName'] . ' ' . $creator\_row['lastName'];

   }

   $content = $row['content'];

   $creator = $row['creator'];

   if (strlen($content) > 40){

     $content = $content . "...";

   }

   //generate HTML

   $row\_div = new html\_element('div');

   $row\_div->set('class', 'row');

   $col = new html\_element('div');

   $col->set('class', 'col');

   $link = new html\_element('a');

   $link->set('style', 'color:inherit; text-decoration: none');

   $link->set('href', 'message.php?id=' . $creator . '&name=' . $name);

   $link->set('target', '\_parent');

   $title = new html\_element('p');

   $title->set('class', 'lead');

   $title->set('text', $content);

   $image = new html\_element('i');

   $image->set('class', 'fa fa-envelope');

   $image->set('aria-hidden', 'true');

   $title->inject($image);

   $link->inject($title);

   $col->inject($link);

   $row\_div->inject($col);

   $col = new html\_element('div');

   $col->set('class', 'col');

   $title = new html\_element('p');

   $title->set('class', 'instructor');

   $title->set('text', $name);

   $col->inject($title);

   $row\_div->inject($col);

   $row\_div->output();

 }

 ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Index.php

<?php

include '../session/student-session.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <?php

 include 'header.php';

 include '../functions/html\_element.php';

?>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main">

           <iframe src="announcements.php" style="width: 100%;position: relative;"></iframe>

         </div>

         <div  class="col col col-md-3 whiteboard">

           <p class="lead"> Classes</p>

           <?php

             $i=0;

             while($i < count($courseArray)){

               $course = $courseArray[$i];

               $staffID = $course->get\_staffId();

               $staffSql = "SELECT firstName, lastName FROM Staff WHERE id= '$staffID'";

               $staffResult = mysqli\_query($db, $staffSql) or die('error getting data');

               $staffRow = mysqli\_fetch\_array($staffResult, MYSQLI\_ASSOC);

               $staffName = $staffRow['firstName'] . " " . $staffRow['lastName'];

               $inner\_p\_tag = new html\_element('p');

               $inner\_p\_tag->set('class','instructor');

               $inner\_p\_tag->set('text',$staffName);

               $outer\_p\_tag = new html\_element('p');

               $outer\_p\_tag->set('class','lead');

               $outer\_p\_tag->set('text',$course->get\_name());

               $outer\_p\_tag->inject($inner\_p\_tag);

               $link = new html\_element('a');

               $link->set('href', 'course.php?location=' . $i);

               $link->set('style', 'color:inherit; text-decoration: none');

               $link->inject($outer\_p\_tag);

               $link->output();

               $i++;

             }

           ?>

         </div>

       </div>

   <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Message.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

 <?php

 include 'header.php';

     $id = $\_GET['id'];

   $name = $\_GET['name'];

 ?>

   <div class='container'>

   <div class="row justify-content-md-center"><div class="col whiteboard whiteboard-main" style="height: 300px ;position: relative;">

   <h3 class="text-center"><?php echo $name ?></h3>

   <iframe src="chat.php?id=<?php echo $id ?>" seamless='seamless' id='iframe' style="width: 100%;height: 85%;position: relative;"></iframe>

   </div>

     <div class="container whiteboard">

             <form id="confirmation" method="post" enctype="multipart/form-data" >

              <div class="form-group assgn">

                 <textarea class="form-control" rows="3" name="content" placeholder="Enter Text Here" id="textarea" required></textarea>

               </div>

                 <input type="submit" class="btn btn-primary assgn pull-right" name="input">

             </form>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js" integrity="sha384-b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4" crossorigin="anonymous"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

    <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("form#confirmation").submit(function(e){

   var formData = new FormData(this);

   $.ajax({

       url: "chat.php?id=<?php echo $id ?>",

       type: 'POST',

       data: formData,

       success: function (data) {

           document.getElementById('iframe').contentDocument.location.reload(true);

           $("form#confirmation").trigger('reset');

       },

       cache: false,

       contentType: false,

       processData: false

   });

   return false;

});

   </script>

 </body>

</html>

* + Submission-creation.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

if(isset($\_POST['comments'])){

 $comments = htmlspecialchars($\_POST['comments']);

 $graded = 0;

 $assignment\_id = $\_POST['assignment'];

 $target\_dir = "uploads/";

 $target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);

 $query = "INSERT INTO Submission (studentID, comments, assignmentID, graded) VALUES (?, ?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($query);

 $stmt->bind\_param('dsdi', $session\_studentId, $comments, $assignment\_id, $graded);

 $stmt->execute();

 console\_log($target\_file);

 $file = $\_FILES['fileToUpload']['name'];

 $file\_size = $\_FILES['fileToUpload']['size'];

 $file\_type = $\_FILES['fileToUpload']['type'];

 $submission\_id = $db->insert\_id;

 move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file);

 $sql="INSERT INTO submission\_file(file, type, size, submission\_id) VALUES(?, ?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssds', $file, $file\_type, $file\_size, $submission\_id);

 $stmt->execute();

}

// cleans input

function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

* + Whiteboardcalendar.php

<?php

include '../session/student-session.php';

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

  <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

  <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

   <link rel='stylesheet' href='../fullcalendar/fullcalendar.css' />

   <script src='../fullcalendar/lib/jquery.min.js'></script>

   <script src='../fullcalendar/lib/moment.min.js'></script>

   <script src='../fullcalendar/fullcalendar.js'></script>

<!--<script src='http://fullcalendar.io/js/fullcalendar-2.1.1/lib/jquery.min.js'></script>

<script src="http://fullcalendar.io/js/fullcalendar-2.1.1/lib/jquery-ui.custom.min.js"></script>

   <script src='http://fullcalendar.io/js/fullcalendar-2.1.1/lib/moment.min.js'></script>

<script src='http://fullcalendar.io/js/fullcalendar-2.1.1/fullcalendar.min.js'></script>-->

   <script>

   $(document).ready(function() {

       $('#calendar').fullCalendar({

         events: 'get-events.php'

       });

   });

      </script>

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

   <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">

     <a class="navbar-brand" href="index.php">WhiteBoard</a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarsExampleDefault">

       <ul class="navbar-nav mr-auto">

         <li class="nav-item active">

           <a class="nav-link" href="whiteboardcalendar.php">Calendar <span class="sr-only">(current)</span></a>

         </li>

         <li class="nav-item">

           <a class="nav-link disabled" href="gradebook.php">Grades</a>

         </li>

         <li class="nav-item">

         </li>

       </ul>

       <a class="btn btn-default action-button navbar-text navbar-right actions" role="button" href="../logout.php">Log Out</a>

     </div>

   </nav>

     <div class="container">

       <div id='calendar' class="whiteboard">

       </div>

   <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

  <!-- <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>-->

  <!-- <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>-->

   <script src="../fullcalendar/popper.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* Functions
  + Console\_log.php

<?php

function console\_log( $data ){

 echo '<script>';

 echo 'console.log('. json\_encode( $data ) .')';

 echo '</script>';

}

?>

* + Html\_element.php

<?php

// retrieved from https://davidwalsh.name/create-html-elements-php-htmlelement-class

class html\_element

{

    /\* vars \*/

    var $type;

    var $attributes;

    var $self\_closers;

    /\* constructor \*/

    function html\_element($type,$self\_closers = array('input','img','hr','br','meta','link'))

    {

        $this->type = strtolower($type);

        $this->self\_closers = $self\_closers;

    }

    /\* get \*/

    function get($attribute)

    {

        return $this->attributes[$attribute];

    }

    /\* set -- array or key,value \*/

    function set($attribute,$value = '')

    {

        if(!is\_array($attribute))

        {

            $this->attributes[$attribute] = $value;

        }

        else

        {

            $this->attributes = array\_merge($this->attributes,$attribute);

        }

    }

    function remove($att)

    {

        if(isset($this->attributes[$att]))

        {

            unset($this->attributes[$att]);

        }

    }

    function clear()

    {

        $this->attributes = array();

    }

    function inject($object)

    {

        if(@get\_class($object) == \_\_class\_\_)

        {

            $this->attributes['text'].= $object->build();

        }

    }

    function build()

    {

        //start

        $build = '<'.$this->type;

        //add attributes

        if(count($this->attributes))

        {

            foreach($this->attributes as $key=>$value)

            {

                if($key != 'text') { $build.= ' '.$key.'="'.$value.'"'; }

            }

        }

        //closing

        if(!in\_array($this->type,$this->self\_closers))

        {

            $build.= '>'.$this->attributes['text'].'</'.$this->type.'>';

        }

        else

        {

            $build.= ' />';

        }

        //return it

        return $build;

    }

    function output()

    {

        echo $this->build();

    }

}

?>

* Objects
  + Course.php

<?php

   class course {

       private $id;

       private $name;

       private $description;

       private $staffId;

       public function set\_id($new\_id){

           $this->id = $new\_id;

       }

       public function get\_id() {

           return $this->id;

       }

       public function set\_name($new\_name){

           $this->name = $new\_name;

       }

       public function get\_name(){

           return $this->name;

       }

       public function set\_description($new\_description){

           $this->description = $new\_description;

       }

       public function get\_description(){

           return $this->description;

       }

       public function set\_staffId($new\_staffId){

           $this->staffId = $new\_staffId;

       }

       public function get\_staffId(){

           return $this->staffId;

       }

       public function \_\_get($id){

           return $this->$id;

       }

       public function \_\_isset($id)  {

           return isset($this->$i);

       }

   }

* Session
  + Admin-session.php

<?php

session\_start();

   include '../connection.php';

   $user\_check=$\_SESSION['login\_user'];

   $sql = "SELECT username, role, instId FROM User WHERE id = '$user\_check'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $login\_session = $row['username'];

   $session\_role=$row['role'];

   $session\_isntId = $row['instId'];

   $sql = "SELECT name, primaryColor, secondaryColor FROM Institution WHERE id = '$session\_isntId'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $session\_institution = $row['name'];

   $session\_primaryColor = $row['primaryColor'];

   $session\_secondaryColor = $row['secondaryColor'];

   if(!isset($login\_session)){

       mysqli\_close($db);

       header( 'Location: ../login.php' );

   }

    if ($session\_role!=='admin') {

       header( 'Location: ../login.php' );

   }

?>

* + Student-session.php

<?php

session\_start();

   include '../connection.php';

   include '../objects/course.php';

   $user\_check=$\_SESSION['login\_user'];

   //get user details

   $sql = "SELECT username, role, instId FROM User WHERE id = '$user\_check'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $login\_session = $row['username'];

   $session\_role=$row['role'];

   $session\_instId = $row['instId'];

   //get institution details

   $sql = "SELECT name, primaryColor, secondaryColor FROM Institution WHERE id = '$session\_instId'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $session\_institution = $row['name'];

   $session\_primaryColor=$row['primaryColor'];

   $session\_secondaryColor = $row['secondaryColor'];

   //get student details

   $sql = "SELECT studentID, firstName, lastName, email FROM Student WHERE userId = '$user\_check'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $session\_studentId=$row['studentID'];

   $session\_firstName=$row['firstName'];

   $session\_lastName=$row['lastName'];

   $session\_email=$row['email'];

   //get student courses

   $sql = "SELECT \* FROM Student\_Course\_Junction WHERE student\_id = '$session\_studentId'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $courseArray = array();

   $i = 0;

   while ($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

       $courseId = $row['course\_id'];

       $courseSql = "SELECT \* FROM Course WHERE courseID= '$courseId'";

       $courseResult = mysqli\_query($db, $courseSql) or die('error getting data');

       $courseRow = mysqli\_fetch\_array($courseResult, MYSQLI\_ASSOC);

       $course = new course;

       $course->set\_id($courseRow['courseID']);

       $course->set\_name($courseRow['courseName']);

       $course->set\_description($courseRow['description']);

       $course->set\_staffId($courseRow['staffID']);

       // add to session array

       $courseArray[$i] = $course;

       $i++;

   }

   // redirect if session is lost

   if(!isset($login\_session)){

       mysqli\_close($db);

       header( 'Location: ../login.php' );

   }

   // redirect if role is not student

    if ($session\_role!=='student') {

       header( 'Location: ../login.php' );

   }

?>

* + Staff-session.php

<?php

session\_start();

   include '../connection.php';

   include '../objects/course.php';

   $user\_check=$\_SESSION['login\_user'];

   //get user details

   $sql = "SELECT username, role, instId FROM User WHERE id = '$user\_check'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $login\_session = $row['username'];

   $session\_role=$row['role'];

   $session\_instId = $row['instId'];

   //get institution details

   $sql = "SELECT name, primaryColor, secondaryColor FROM Institution WHERE id = '$session\_instId'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $session\_institution = $row['name'];

   $session\_primaryColor=$row['primaryColor'];

   $session\_secondaryColor = $row['secondaryColor'];

   //get staff details

   $sql = "SELECT \* FROM Staff WHERE userId = '$user\_check'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);

   $session\_staffId=$row['id'];

   $session\_firstName=$row['firstName'];

   $session\_lastName=$row['lastName'];

   $session\_email=$row['email'];

   //get staff courses

   $sql = "SELECT \* FROM Course WHERE staffID = '$session\_staffId'";

   $result = mysqli\_query($db, $sql) or die('error getting data');

   $courseArray = array();

   $i = 0;

   while ($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC)){

       $course = new course;

       $course->set\_id($row['courseID']);

       $course->set\_name($row['courseName']);

       $course->set\_description($row['description']);

       $course->set\_staffId($row['staffID']);

       // add to session array

       $courseArray[$i] = $course;

       $i++;

   }

   // redirect if session is lost

   if(!isset($login\_session)){

       mysqli\_close($db);

       header( 'Location: ../login.php' );

   }

   // redirect if role is not student

    if ($session\_role!=='staff') {

       header( 'Location: ../login.php' );

   }

?>

* Staff
  + Announcement-view.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$announcement\_id = $\_GET['id'];

$sql = "SELECT \* FROM Announcement WHERE id =$announcement\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$timestamp = date("m-d-y", strtotime($row['timestamp']));

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Announcement</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px"><?php echo $row['title']; ?></h4>

               <p class="lead"><?php echo $row['body'] ?></p>

               <p class="lead">Posted: <?php echo $timestamp ?></p>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Assignments.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="feed.php">Feed</a>

       <a class="btn navbar-brand nav-elem" href="inboxs.php">Inbox</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="assignments.php">Assignments</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <a class="btn btn-outline-secondary" style="margin-left:45px" href="create-assignment.php" target="\_parent">Create Assignment</a>

     </div>

   </nav>

               <?php

               $num\_of\_courses = count($courseArray);

               for ($i = 0; $i < $num\_of\_courses; $i++) {

                  //create assignment object

                  $column = new html\_element('div');

                  $column->set('class','col');

                  $header = new html\_element('h4');

                  $header->set('text',$courseArray[$i]->get\_name());

                  $column->inject($header);

                  $courseId = $courseArray[$i]->get\_id();

                  $sql = "SELECT id, title, body, timestamp, dueDate FROM Assignment WHERE courseID = '$courseId' ORDER BY timestamp DESC LIMIT 3";

                  $result = mysqli\_query($db, $sql) or die('error getting data');

                  $num\_rows = mysqli\_num\_rows($result);

                  while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

                    $image = new html\_element('i');

                    $image->set('class', 'fa fa-pencil');

                    $image->set('aria-hidden', 'true');

                    $title = new html\_element('p');

                    $title->set('class','lead');

                    $title->set('text',$row['title']);

                    $title->inject($image);

                    $link = new html\_element('a');

                    $link->set('href', 'course-submission.php?id=' . $row['id']);

                    $link->set('style', 'color:inherit; text-decoration: none');

                    $link->set('target', '\_parent');

                    $link->inject($title);

                    $column->inject($link);

                    $dueDate = date("m-d-y", strtotime($row[3]));

                    $due = new html\_element('p');

                    $due->set('text', "Due: $dueDate");

                    $column->inject($due);

                  }

                  $row = new html\_element('div');

                  $row->set('class','row');

                  $row->inject($column);

                  $row->output();

               }

               ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Chat.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$id = $\_GET['id'];

if(isset($\_POST['content'])){

   $content = htmlspecialchars($\_POST['content']);

   $sql = 'INSERT INTO message (content, reciever, creator) VALUES (?, ?, ?)';

   console\_log($sql);

   $stmt = $db->stmt\_init();

   $stmt->prepare($sql);

   $stmt->bind\_param('sdd', $content, $id, $user\_check);

   $stmt->execute();

}

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <?php

   $sql = 'SELECT \* FROM (SELECT content, creator, reciever, timestamp FROM message WHERE reciever =  ' . $user\_check . ' AND creator =' . $id . ') as FirstSet

         union (

         SELECT content, creator, reciever, timestamp FROM message WHERE reciever =' . $id . ' AND creator = ' . $user\_check . ') ORDER BY timestamp';

   $result = $db->query($sql);

   while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

     if($row['creator'] == $user\_check){

         $div = new html\_element('div');

         $div->set('class', 'row');

         $col = new html\_element('div');

         $col->set('class', 'col-md-8');

         $card = new html\_element('div');

         $card->set('class', 'card');

         $body = new html\_element('div');

         $body->set('class', 'card-body bg-primary');

         $content = new html\_element('p');

         $content->set('class', 'p-y-1 text-white pull-right');

         $content->set('text', $row['content']);

         $body->inject($content);

         $card->inject($body);

         $col->inject($card);

         $col2 = new html\_element('div');

         $col2->set('class', 'col-md-4');

         $div->inject($col2);

         $div->inject($col);

         $div->output();

     }

     else{

         $div = new html\_element('div');

         $div->set('class', 'row');

         $col = new html\_element('div');

         $col->set('class', 'col-md-8');

         $card = new html\_element('div');

         $card->set('class', 'card');

         $body = new html\_element('div');

         $body->set('class', 'card-body ');

         $content = new html\_element('p');

         $content->set('class', 'p-y-1');

         $content->set('text', $row['content']);

         $body->inject($content);

         $card->inject($body);

         $col->inject($card);

         $col2 = new html\_element('div');

         $col2->set('class', 'col-md-4');

         $div->inject($col);

         $div->inject($col2);

         $div->output();

     }

   }

     ?>

     <script>

         function scrollBottom() {window.scrollTo(0, document.body.scrollHeight);}

           if (document.addEventListener) document.addEventListener("DOMContentLoaded", scrollBottom, false)

           else if (window.attachEvent) window.attachEvent("onload", scrollBottom);

     </script>

     <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js" integrity="sha384-b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4" crossorigin="anonymous"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Content-upload.php

<?php

include '../session/student-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

if(isset($\_POST['title'])){

 $body = htmlspecialchars($\_POST['body']);

 $course\_id = $\_POST['course'];

 $title = test\_input($\_POST['title']);

 $target\_dir = "uploads/";

 $target\_file = $target\_dir . basename($\_FILES["fileToUpload"]["name"]);

 $query = "INSERT INTO content (title, body, course\_id) VALUES (?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($query);

 $stmt->bind\_param('ssd', $title, $body, $course\_id);

 $stmt->execute();

 console\_log($target\_file);

 $file = $\_FILES['fileToUpload']['name'];

 $file\_size = $\_FILES['fileToUpload']['size'];

 $file\_type = $\_FILES['fileToUpload']['type'];

 $content\_id = $db->insert\_id;

 move\_uploaded\_file($\_FILES["fileToUpload"]["tmp\_name"], $target\_file);

 $sql="INSERT INTO content\_file(file, type, size, content\_id) VALUES(?, ?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssds', $file, $file\_type, $file\_size, $content\_id);

 $stmt->execute();

}

// cleans input

function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

* + Content-view.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$content\_id = $\_GET['id'];

$sql = "SELECT \* FROM content WHERE id =$content\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$timestamp = date("m-d-y", strtotime($row['timestamp']));

$sql\_file = "SELECT \* FROM content\_file WHERE content\_id = $content\_id";

$result\_file = $db->query($sql\_file);

$num\_rows\_file = mysqli\_num\_rows($result\_file);

   if ($num\_rows\_file>=1){

       $row\_file = $result\_file->fetch\_array(MYSQLI\_ASSOC);

       $file = $row\_file['file'];

       console\_log($file);

       }

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Content</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px"><?php echo $row['title']; ?></h4>

             <a href='uploads/<?php echo $file ?>'><?php echo $file ?> </a>

               <p class="lead"><?php echo $row['body'] ?></p>

               <p class="lead">Posted: <?php echo $timestamp ?></p>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-add-student.php

<?php

include '../session/staff-session.php';

include '../functions/console\_log.php';

if(isset($\_POST['id'])){

 //messages to give back to the user

 $id = $\_POST['id'];

 $course\_id = $\_POST['course\_id'];

 $sql = "INSERT INTO Student\_Course\_Junction (student\_id, course\_id) VALUES (?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('dd', $id, $course\_id);

 $stmt->execute();

}

$term = $\_GET['term'];

$sql = "SELECT DISTINCT firstName, lastName, studentID FROM Student WHERE (concat\_ws(' ', firstName, lastName) LIKE '%$term%')";

$result = $db->query($sql);

$json = array();

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $name = $row['firstName'] . " " . $row['lastName'];

    $json[] = array(

        'value' => $row['studentID'],

        'label' => $name);

}

echo json\_encode($json);

?>

* + Course-announcement.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem nav-curr" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <a class="btn btn-outline-secondary" style="margin-left:45px" href="create-announcement.php" target="\_parent">Create Announcement</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, body, timestamp FROM Announcement WHERE courseID = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

        $image = new html\_element('i');

        $image->set('class', 'fa fa-bullhorn');

        $image->set('aria-hidden', 'true');

        $title = new html\_element('p');

        $title->set('class','lead');

        $title->set('text',$row['title']);

        $title->inject($image);

        $link = new html\_element('a');

        $link->set('href', 'announcement-view.php?id=' . $row['id']);

        $link->set('style', 'color:inherit; text-decoration: none');

        $link->set('target', '\_parent');

        $link->inject($title);

        $column->inject($link);

        $postDate = date("m-d-y", strtotime($row['timestamp']));

        $time = new html\_element('p');

        $time->set('text', "Posted: $postDate");

        $column->inject($time);

      }

      $row = new html\_element('div');

      $row->set('class','row');

      $row->inject($column);

      $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-assignment.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <a class="btn btn-outline-secondary" style="margin-left:45px" href="create-assignment.php" target="\_parent">Create Assignment</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, body, timestamp, dueDate FROM Assignment WHERE courseID = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $image = new html\_element('i');

           $image->set('class', 'fa fa-pencil');

           $image->set('aria-hidden', 'true');

           $title = new html\_element('p');

           $title->set('class','lead');

           $title->set('text',$row['title']);

           $title->inject($image);

           $link = new html\_element('a');

           $link->set('href', 'course-submission.php?id=' . $row['id']);

           $link->set('style', 'color:inherit; text-decoration: none');

           $link->set('target', '\_parent');

           $link->inject($title);

           $column->inject($link);

           $dueDate = date("m-d-y", strtotime($row[3]));

           $due = new html\_element('p');

           $due->set('text', "Due: $dueDate");

           $column->inject($due);

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-content-creation.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$location = $\_GET['location'];

$course\_id = $courseArray[$location]->get\_id();

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Content Upload</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px">Upload Content</h4>

               <form id="confirmation" method="post" enctype="multipart/form-data" >

               <div class="form-group assgn">

                   <label for="exampleFormControlFile1">Content Title</label>

                   <input type="text" class="form-control" name="title" required>

                 </div>

               <div class="form-group assgn">

                   <label for="exampleFormControlFile1">Upload the Content here</label>

                   <input type="file" class="form-control-file" name="fileToUpload" id="fileToUpload">

                 </div>

              <div class="form-group assgn">

                 <label for="exampleFormControlTextarea1">Body</label>

                 <textarea class="form-control" rows="3" name="body"></textarea>

               </div>

               <input  type="hidden" name="course" value="<?php echo $course\_id ?>">

                 <input type="submit" class="btn btn-primary assgn" name="input">

             </form>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

   <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("form#confirmation").submit(function(){

   var formData = new FormData(this);

   $.ajax({

       url: "content-upload.php",

       type: 'POST',

       data: formData,

       success: function (data) {

           alert("Content uploaded sucessfully!");

           window.location.replace(history.back());

       },

       cache: false,

       contentType: false,

       processData: false

   });

   return false;

});

   </script>

 </body>

</html>

* + Course-content.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <a class="btn btn-outline-secondary" style="margin-left:45px" href="course-content-creation.php?location=<?php echo $location ?>" target="\_parent">Add content</a>

     </div>

   </nav>

<?php

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $courseId = $courseArray[$location]->get\_id();

      $sql = "SELECT id, title, timestamp FROM content WHERE course\_id = '$courseId' ORDER BY timestamp DESC";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $image = new html\_element('i');

           $image->set('class', 'fa fa-sticky-note-o');

           $image->set('aria-hidden', 'true');

           $title = new html\_element('p');

           $title->set('class','lead');

           $title->set('text',$row['title']);

           $title->inject($image);

           $link = new html\_element('a');

           $link->set('href', 'content-view.php?id=' . $row['id']);

           $link->set('style', 'color:inherit; text-decoration: none');

           $link->set('target', '\_parent');

           $link->inject($title);

           $column->inject($link);

           $dueDate = date("m-d-y", strtotime($row['timestamp']));

           $due = new html\_element('p');

           $due->set('text', "Posted: $dueDate");

           $column->inject($due);

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-grade-post.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

if(isset($\_POST['grade'])){

   $submission\_id = $\_POST['submission'];

   $grade = $\_POST['grade'];

   $comments = htmlspecialchars($\_POST['comments']);

   $query = "INSERT INTO Grade (grade, comment, submissionID) VALUES (?, ?, ?)";

   $stmt = $db->stmt\_init();

   $stmt->prepare($query);

   $stmt->bind\_param('dsd', $grade, $comments, $submission\_id);

   $stmt->execute();

   $sql = "UPDATE Submission SET graded=1 WHERE id=$submission\_id";

   $result = $db->query($sql);

}

?>

* + Course-grade.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$submission\_id = $\_GET['id'];

$assignment\_id = $\_GET['assignment\_id'];

//query for submission information

$sql = "SELECT \* FROM Submission WHERE id =$submission\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$timestamp = date("m-d-y h:i A", strtotime($row['timestamp']));

$comments = $row['comments'];

//query for file information

$sql = "SELECT file FROM submission\_file WHERE submission\_id = $submission\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$file = $row['file'];

$upload = "";

if ($file != "")

$upload = '../student/uploads/' . $file;

//query for assignment information

$sql = "SELECT points FROM Assignment WHERE id = $assignment\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$points = $row['points'];

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Assignment</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px">Student Submission:</h4>

               <a class="lead" href='<?php echo $upload ?>'><?php echo $file ?></a>

               <p class="lead">Student Comment: <?php echo $comments ?></p>

               <p class='lead'>Submission time: <?php echo $timestamp ?></p>

               <form id="confirmation" method="post" enctype="multipart/form-data" >

               <div class="form-group assgn">

                   <label for="exampleFormControlFile1">Grade (Out of <?php echo $points ?> )</label>

                   <input type="number" class="form-control" name="grade" id="grade" required>

                 </div>

              <div class="form-group assgn">

                 <label for="exampleFormControlTextarea1">Comments</label>

                 <textarea class="form-control" rows="3" name="comments"></textarea>

               </div>

               <input  type="hidden" name="submission" value="<?php echo $submission\_id; ?>">

                 <input type="submit" class="btn btn-primary assgn" name="input">

             </form>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

   <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("form#confirmation").submit(function(){

   var formData = new FormData(this);

   $.ajax({

       url: "course-grade-post.php",

       type: 'POST',

       data: formData,

       success: function (data) {

           alert("Submission Graded!");

           window.location.replace('course-submission.php?id=<?php echo $assignment\_id ?>');

       },

       cache: false,

       contentType: false,

       processData: false

   });

   return false;

});

   </script>

 </body>

</html>

* + Course-roster.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

$courseId = $courseArray[$location]->get\_id();

if(isset($\_POST['id'])){

 //messages to give back to the user

 $id = $\_POST['id'];

 $course\_id = $\_POST['course\_id'];

 $sql = "INSERT INTO Student\_Course\_Junction (student\_id, course\_id) VALUES (?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('dd', $id, $course\_id);

 $stmt->execute();

}

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

           <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script

     src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"

     integrity="sha256-0YPKAwZP7Mp3ALMRVB2i8GXeEndvCq3eSl/WsAl1Ryk="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js"></script>

   <script>

   $(document).ready(function() {

   $("#autoComplete").autocomplete({

     source: "course-add-student.php",

     select: function(event, ui){

       $.post( "course-add-student.php", { id: ui.item.value, course\_id: <?php echo $courseId ?>  })

         .done(function() {

           window.location.reload();

       });

     }

       });

   });

       </script>

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="course-announcement.php?location=<?php echo $location ?>">Announcements</a>

       <a class="btn navbar-brand nav-elem" href="course-assignment.php?location=<?php echo $location ?>">Assignments</a>

       <a class="btn navbar-brand nav-elem" href="course-content.php?location=<?php echo $location ?>">Content</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="course-roster.php?location=<?php echo $location ?>">Roster</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <button class="btn btn-outline-secondary" style="margin-left:45px"  data-toggle='modal' data-target="#myModal">Add Student</button>

     </div>

   </nav>

     <!-- Modal -->

 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

   <div class="modal-dialog" role="document">

     <div class="modal-content">

       <div class="modal-header">

         <h5 class="modal-title" id="exampleModalLabel">Add student</h5>

         <button type="button" class="close" data-dismiss="modal" aria-label="Close">

           <span aria-hidden="true">&times;</span>

         </button>

       </div>

       <div class="modal-body">

         <p>Add who?</p>

         <input id='autoComplete' name='student'/>

       </div>

       <div class="modal-footer">

         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

       </div>

     </div>

   </div>

 </div>

<?php

      //create assignment object

      $student\_id\_array = array();

      $column = new html\_element('div');

      $column->set('class','col');

      $sql = "SELECT student\_id FROM Student\_Course\_Junction WHERE course\_id = '$courseId'";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

          $student\_id\_array[] = $row['student\_id'];

       }

       $student\_id\_array = implode(" OR ", $student\_id\_array);

       if ($num\_rows > 0){

       $sql = "SELECT firstName, lastName FROM Student WHERE studentID= $student\_id\_array ORDER BY lastName";

       $result = mysqli\_query($db, $sql) or die('error getting data');

       while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

           $name = new html\_element('p');

           $name->set('class', 'lead');

           $name->set('text', $row['firstName'] . " " . $row['lastName']);

           $column->inject($name);

       }

       }

       $row = new html\_element('div');

       $row->set('class','row');

       $row->inject($column);

       $row->output();

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-submission.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

$assignment\_id = $\_GET['id'];

$sql = "SELECT \* FROM Assignment WHERE id =$assignment\_id";

$result = $db->query($sql);

$row = $result->fetch\_array(MYSQLI\_ASSOC);

$dueDate = date("m-d-y", strtotime($row['dueDate']));

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Assignment</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <h4 style="margin-bottom:10px"><?php echo $row['title']; ?></h4>

               <p class="lead"><?php echo $row['body'] ?></p>

               <p class="lead">Due date: <?php echo $dueDate ?></p>

               <h4 style="margin-top:10px">Ungraded Submissions:</h4>

               <?php

               $query = "SELECT \* FROM Submission WHERE assignmentID = $assignment\_id AND graded = 0";

               $result\_submission = $db->query($query);

               while($row\_submission = $result\_submission->fetch\_array(MYSQLI\_ASSOC)){

                   $studentId = $row\_submission['studentID'];

                   $sql ="SELECT firstName, lastName FROM Student WHERE studentID = $studentId";

                   $result = $db->query($sql);

                   $row = $result->fetch\_array(MYSQLI\_ASSOC);

                   $name = $row['firstName'] . " " . $row['lastName'];

                   $submission\_id = $row\_submission['id'];

                   //create HTML

                   $p = new html\_element('p');

                   $p->set('class', 'lead');

                   $p->set('text', $name);

                   $link = new html\_element('a');

                   $link->set('href', 'course-grade.php?id=' . $submission\_id . '&assignment\_id=' . $assignment\_id);

                   $link->set('style', 'color:inherit; text-decoration: none');

                   $link->inject($p);

                   $link->output();

               }

               ?>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course-view.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

 include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

           <div class="col whiteboard whiteboard-main gradebook">

               <h2 class="text-center">Your Courses</h2>

               <?php

               $num\_of\_courses = count($courseArray);

               for ($i = 0; $i < $num\_of\_courses; $i++) {

                  //create assignment object

                  $column = new html\_element('div');

                  $column->set('class','col');

                  $link = new html\_element('a');

                  $link->set('href', 'course.php?location=' . $i);

                  $link->set('style', 'color:inherit; text-decoration: none');

                  $header = new html\_element('h4');

                  $header->set('text',$courseArray[$i]->get\_name());

                  $link->inject($header);

                  $column->inject($link);

                  $description = new html\_element('p');

                  $description->set('class','lead');

                  $description->set('text',$courseArray[$i]->get\_description());

                  $column->inject($description);

                  $row = new html\_element('div');

                  $row->set('class','row');

                  $row->inject($column);

                  $row->output();

               }

               ?>

               <a href="create-course.php"><i class="fa fa-plus" aria-hidden="true"></i>   Add Course</a>

           </div>

       </div>

         <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

     </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Course.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

$location = $\_GET['location'];

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

 include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

           <div class="col whiteboard whiteboard-main gradebook">

               <h3><?php echo $courseArray[$location]->get\_name() . " // " . $courseArray[$location]->get\_description(); ?></h3>

               <iframe style="width: 100%;height: 100%;position: relative;"  src="course-announcement.php?location=<?php echo $location ?>"></iframe>

           </div>

       </div>

         <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

     </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Create-announcement.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

if(isset($\_POST['title'])){

 //messages to give back to the user

 $feedback="";

 $title = test\_input($\_POST['title']);

 $description = htmlspecialchars($\_POST['description']);

 $course = $\_POST['course'];

 $sql = "SELECT courseID FROM Course WHERE courseName ='$course'";

 $result = $db->query($sql);

 $row = $result->fetch\_array(MYSQLI\_ASSOC);

 $courseId = $row['courseID'];

 $sql = "INSERT INTO Announcement (title, body, courseID) VALUES (?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssd', $title, $description, $courseId);

 if ($stmt->execute() == TRUE){

   $feedback = "Assignment Created!";

 }

 else{

   $feedback = "Assignment Creation Failed!";

 }

 if($\_POST['checkbox'] == 'on'){

   $date = date('Y-m-d',strtotime($\_POST['date']));

   $last\_id = $db->insert\_id;

   $sql = "INSERT INTO event\_announcement (title, start, announcement\_id, course\_id) VALUES (?, ?, ?, ?)";

   $stmt = $db->stmt\_init();

   $stmt->prepare($sql);

   $stmt->bind\_param('ssdd', $title, $date, $last\_id, $courseId);

   $stmt->execute();

 }

}

// cleans input

function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>Create Announcement</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <p class="lead"> Create New Announcement </p>

             <form id="confirmation">

               <input class="form-control assgn" type="text" name="title" id="title" placeholder="Announcement Title" required>

               <select class="form-control assgn" name="course" required>

                 <option value="" disabled selected>Select your option</option>

                 <?php

                 for ($i = 0; $i<count($courseArray); $i++){

                   $option = new html\_element('option');

                   $option->set('text', $courseArray[$i]->get\_name());

                   $option->output();

                 }

                 ?>

               </select>

              <div class="form-group assgn">

                 <label for="exampleFormControlTextarea1">Announcement Description</label>

                 <textarea class="form-control" rows="3" name="description" form="confirmation" required></textarea>

               </div>

                 <div class="form-check assgn">

                   <label class="form-check-label">

                     <input class="form-check-input" type="checkbox" id="eventBox" name='checkbox' value="on">

                         Make an event?

                   </label>

                 </div>

                 <input class="form-control assgn" type="date" min="2017-04-01" id='date' name="date" disabled>

                 <input type="submit" class="btn btn-primary assgn" >

             </form>

         </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

   <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("#confirmation").submit(function(event){

   // Prevent default posting of form - put here to work in case of errors

   event.preventDefault();

   // Abort any pending request

   if (request) {

       request.abort();

   }

   // setup some local variables

   var $form = $(this);

   // Let's select and cache all the fields

   var $inputs = $form.find("input, select, button, textarea");

   // Serialize the data in the form

   var serializedData = $form.serialize();

   // Let's disable the inputs for the duration of the Ajax request.

   // Note: we disable elements AFTER the form data has been serialized.

   // Disabled form elements will not be serialized.

   $inputs.prop("disabled", true);

   // Fire off the request to /form.php

   request = $.ajax({

       url: "create-announcement.php",

       type: "post",

       data: serializedData

   });

   // Callback handler that will be called on success

   request.done(function (response, textStatus, jqXHR){

       // Alert the user of success

       alert("Announcement Successfully Posted!");

       window.location.replace(history.back());

   });

   // Callback handler that will be called on failure

   request.fail(function (jqXHR, textStatus, errorThrown){

       // Log the error to the console

       console.error(

           "The following error occurred: "+

           textStatus, errorThrown

       );

       alert("Announcement failed to post!");

   });

   // Callback handler that will be called regardless

   // if the request failed or succeeded

   request.always(function () {

       // Reenable the inputs

       $inputs.prop("disabled", false);

   });

});

//for checkbox

document.getElementById('eventBox').onchange = function() {

   document.getElementById('date').disabled = !this.checked;

   document.getElementById('date').required = this.checked;

};

   </script>

 </body>

</html>

* + Create-assignment.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

if(isset($\_POST['title'])){

 //messages to give back to the user

 $feedback="";

 $title = test\_input($\_POST['title']);

 $description = htmlspecialchars($\_POST['description']);

 $points = $\_POST['points'];

 $course = $\_POST['course'];

 $date = date('Y-m-d',strtotime($\_POST['date']));

 $sql = "SELECT courseID FROM Course WHERE courseName ='$course'";

 $result = $db->query($sql);

 $row = $result->fetch\_array(MYSQLI\_ASSOC);

 $courseId = $row['courseID'];

 $sql = "INSERT INTO Assignment (title, body, points, courseID, dueDate) VALUES (?, ?, ?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssdds', $title, $description, $points, $courseId, $date);

 if ($stmt->execute() == TRUE){

   $feedback = "Assignment Created!";

 }

 else{

   $feedback = "Assignment Creation Failed!";

 }

 $last\_id = $db->insert\_id;

 console\_log($last\_id);

 $sql = "INSERT INTO event (title, start, reference\_id, course\_id) VALUES (?, ?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssdd', $title, $date, $last\_id, $courseId);

 $stmt->execute();

}

// cleans input

function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

//echo "<script type='text/javascript'>alert(". $feedback . ")</script>";

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <p class="lead"> Create New Assignment </p>

             <form id="confirmation">

               <input class="form-control assgn" type="text" name="title" placeholder="Assignment name" required>

               <select class="form-control assgn" name="course">

                 <option>Select a class</option>

                 <?php

                 for ($i = 0; $i<count($courseArray); $i++){

                   $option = new html\_element('option');

                   $option->set('text', $courseArray[$i]->get\_name());

                   $option->output();

                 }

                 ?>

               </select>

               <input class="form-control assgn" type="date" min="2017-04-01" name="date" required>

               <input class="form-control assgn" type="number" name="points" min="0" placeholder="Points Worth" required>

              <div class="form-group assgn">

                 <label for="exampleFormControlTextarea1">Description of this assignment</label>

                 <textarea class="form-control" rows="3" name="description" form="confirmation"></textarea>

               </div>

                 <input type="submit" class="btn btn-primary assgn" >

             </form>

         </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

   <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("#confirmation").submit(function(event){

   // Prevent default posting of form - put here to work in case of errors

   event.preventDefault();

   // Abort any pending request

   if (request) {

       request.abort();

   }

   // setup some local variables

   var $form = $(this);

   // Let's select and cache all the fields

   var $inputs = $form.find("input, select, button, textarea");

   // Serialize the data in the form

   var serializedData = $form.serialize();

   // Let's disable the inputs for the duration of the Ajax request.

   // Note: we disable elements AFTER the form data has been serialized.

   // Disabled form elements will not be serialized.

   $inputs.prop("disabled", true);

   // Fire off the request to /form.php

   request = $.ajax({

       url: "create-assignment.php",

       type: "post",

       data: serializedData

   });

   // Callback handler that will be called on success

   request.done(function (response, textStatus, jqXHR){

       // Alert the user of success

       alert("Assignment Successfully Created!");

       window.location.replace(history.back());

   });

   // Callback handler that will be called on failure

   request.fail(function (jqXHR, textStatus, errorThrown){

       // Log the error to the console

       console.error(

           "The following error occurred: "+

           textStatus, errorThrown

       );

   });

});

   </script>

 </body>

</html>

* + Create-course.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

include '../functions/console\_log.php';

if(isset($\_POST['title'])){

 //messages to give back to the user

 $feedback="";

 $title = test\_input($\_POST['title']);

 $description = test\_input($\_POST['description']);

 $sql = "INSERT INTO Course (courseName, description, staffID) VALUES (?, ?, ?)";

 $stmt = $db->stmt\_init();

 $stmt->prepare($sql);

 $stmt->bind\_param('ssd', $title, $description, $session\_staffId);

 $stmt->execute();

}

// cleans input

function test\_input($data) {

       $data = trim($data);

       $data = stripslashes($data);

       $data = htmlspecialchars($data);

       return $data;

   }

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main gradebook">

             <p class="lead"> Create New Course </p>

             <form id="confirmation">

               <input class="form-control assgn" type="text" name="title" placeholder="Course Title e.g. PSY 100" required>

               <input class="form-control assgn" type="text"  name="description" placeholder="Course Name e.g. Intro to Psychology" required>

                 <input type="submit" class="btn btn-primary assgn" >

             </form>

         </div>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

   <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("#confirmation").submit(function(event){

   // Prevent default posting of form - put here to work in case of errors

   event.preventDefault();

   // Abort any pending request

   if (request) {

       request.abort();

   }

   // setup some local variables

   var $form = $(this);

   // Let's select and cache all the fields

   var $inputs = $form.find("input, select, button, textarea");

   // Serialize the data in the form

   var serializedData = $form.serialize();

   // Let's disable the inputs for the duration of the Ajax request.

   // Note: we disable elements AFTER the form data has been serialized.

   // Disabled form elements will not be serialized.

   $inputs.prop("disabled", true);

   // Fire off the request to /form.php

   request = $.ajax({

       url: "create-course.php",

       type: "post",

       data: serializedData

   });

   // Callback handler that will be called on success

   request.done(function (response, textStatus, jqXHR){

       // Alert the user of success

       alert("Course Successfully Created!");

       window.location.replace('course-view.php');

   });

   // Callback handler that will be called on failure

   request.fail(function (jqXHR, textStatus, errorThrown){

       // Log the error to the console

       console.error(

           "The following error occurred: "+

           textStatus, errorThrown

       );

   });

   // Callback handler that will be called regardless

   // if the request failed or succeeded

   request.always(function () {

       // Reenable the inputs

       $inputs.prop("disabled", false);

   });

});

   </script>

 </body>

</html>

* + Feed.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem nav-curr" href="feed.php">Feed</a>

       <a class="btn navbar-brand nav-elem" href="inboxs.php">Inbox</a>

       <a class="btn navbar-brand nav-elem" href="assignments.php">Assignments</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <a class="btn btn-outline-secondary" style="margin-left:35px" href="create-announcement.php" target="\_parent">Create Announcement</a>

     </div>

   </nav>

<?php

   $num\_of\_courses = count($courseArray);

   for ($i = 0; $i < $num\_of\_courses; $i++) {

      //create assignment object

      $column = new html\_element('div');

      $column->set('class','col');

      $header = new html\_element('h4');

      $header->set('text',$courseArray[$i]->get\_name());

      $column->inject($header);

      $courseId = $courseArray[$i]->get\_id();

      $sql = "SELECT id, title, body, timestamp FROM Announcement WHERE courseID = '$courseId' ORDER BY timestamp DESC LIMIT 3";

      $result = mysqli\_query($db, $sql) or die('error getting data');

      $num\_rows = mysqli\_num\_rows($result);

      while ($row = mysqli\_fetch\_array($result, MYSQLI\_BOTH)){

        $image = new html\_element('i');

        $image->set('class', 'fa fa-bullhorn');

        $image->set('aria-hidden', 'true');

        $title = new html\_element('p');

        $title->set('class','lead');

        $title->set('text',$row['title']);

        $link = new html\_element('a');

        $link->set('href', 'announcement-view.php?id=' . $row['id']);

        $link->set('style', 'color:inherit; text-decoration: none');

        $link->set('target', '\_parent');

        $link->inject($title);

        $column->inject($link);

        $postDate = date("m-d-y", strtotime($row['timestamp']));

        $time = new html\_element('p');

        $time->set('text', "Posted: $postDate");

        $column->inject($time);

      }

      $row = new html\_element('div');

      $row->set('class','row');

      $row->inject($column);

      $row->output();

   }

   ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Get-contacts.php

<?php

include '../session/staff-session.php';

include '../functions/console\_log.php';

$term = $\_GET['term'];

$id\_array = array();

for ($i=0; $i<count($courseArray); $i++){

    array\_push($id\_array, $courseArray[$i]->get\_id());

}

$id\_array = implode(" OR ", $id\_array);

$sql = "SELECT DISTINCT firstName, lastName, userId FROM Student JOIN Student\_Course\_Junction ON Student.studentID=Student\_Course\_Junction.student\_id WHERE course\_id = ($id\_array) AND (concat\_ws(' ', firstName, lastName) LIKE '%$term%')";

$result = $db->query($sql);

$json = array();

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $name = $row['firstName'] . " " . $row['lastName'];

    $json[] = array(

        'value' => 'message.php?id=' . $row['userId'] . '&name=' . $name,

        'label' => $name);

}

$sql = "SELECT DISTINCT firstName, lastName, userId FROM Staff WHERE (concat\_ws(' ', firstName, lastName) LIKE '%$term%') AND NOT userId=$user\_check";

$result = $db->query($sql);

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $name = $row['firstName'] . " " . $row['lastName'];

    $json[] = array(

        'value' => 'message.php?id=' . $row['userId'] . '&name=' . $name,

        'label' => $name);

}

echo json\_encode($json);

?>

* + Get-events.php

<?php

include '../session/staff-session.php';

include '../functions/console\_log.php';

$id\_array = array();

for ($i=0; $i<count($courseArray); $i++){

    array\_push($id\_array, $courseArray[$i]->get\_id());

}

$id\_array = implode(" OR ", $id\_array);

$sql = "SELECT title, start, reference\_id FROM event WHERE course\_id = $id\_array";

$result = $db->query($sql);

$json = array();

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $json[] = array(

       'title' => $row['title'],

       'start' => $row['start'],

       'url' => 'course-submission.php?id=' . $row['reference\_id'],

       'allDay' => true);

}

$sql = "SELECT title, start, announcement\_id FROM event\_announcement WHERE course\_id = $id\_array";

$result = $db->query($sql);

while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

    $json[] = array(

       'title' => $row['title'],

       'start' => $row['start'],

       'url' => 'announcement-view.php?id=' . $row['announcement\_id'],

       'color' => 'red',

       'allDay' => true);

}

echo json\_encode($json);

?>

* + Header.php

<!DOCTYPE html>

<body>

   <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">

     <a class="navbar-brand" href="index.php">WhiteBoard</a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarsExampleDefault">

       <ul class="navbar-nav mr-auto">

         <li class="nav-item">

           <a class="nav-link" href="whiteboardcalendar.php">Calendar <span class="sr-only">(current)</span></a>

         </li>

         <li class="nav-item">

           <a class="nav-link" href="course-view.php">Courses</a>

         </li>

         <li class="nav-item">

         </li>

       </ul>

       <a class="btn btn-default action-button navbar-text navbar-right actions" role="button" href="../logout.php">Log Out</a>

     </div>

   </nav>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

* + Inboxs.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

       <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script

     src="https://code.jquery.com/ui/1.12.0/jquery-ui.js"

     integrity="sha256-0YPKAwZP7Mp3ALMRVB2i8GXeEndvCq3eSl/WsAl1Ryk="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js"></script>

   <script>

   $(document).ready(function() {

   $("#autoComplete").autocomplete({

     source: "get-contacts.php",

     select: function(event, ui){

       window.top.location.href = ui.item.value;

     }

       });

   });

       </script>

 </head>

 <body class="whiteboard-iframe">

     <nav class="navbar navbar-expand-md fixed-top whiteboard-nav">

      <div class="btn-group btn-group-justified">

       <a class="btn navbar-brand nav-elem" href="feed.php">Feed</a>

       <a class="btn navbar-brand nav-elem nav-curr" href="inboxs.php">Inbox</a>

       <a class="btn navbar-brand nav-elem" href="assignments.php">Assignments</a>

     </div>

     <div class="btn-group btn-group-right float-right">

       <button class="btn btn-outline-secondary" style="margin-left:45px"  data-toggle='modal' data-target="#myModal">Create Message</button>

     </div>

   </nav>

 <!-- Modal -->

 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

   <div class="modal-dialog" role="document">

     <div class="modal-content">

       <div class="modal-header">

         <h5 class="modal-title" id="exampleModalLabel">Create Message</h5>

         <button type="button" class="close" data-dismiss="modal" aria-label="Close">

           <span aria-hidden="true">&times;</span>

         </button>

       </div>

       <div class="modal-body">

         <p>Send to who?</p>

         <input id='autoComplete'/>

       </div>

       <div class="modal-footer">

         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

       </div>

     </div>

   </div>

 </div>

 <?php

 $sql = 'SELECT SUBSTRING(content, 1, 45) AS content, creator, timestamp FROM message WHERE reciever =' . $user\_check . ' GROUP BY(creator) DESC';

 $result = $db->query($sql);

 while($row = $result->fetch\_array(MYSQLI\_ASSOC)){

   //get creator name

   $creator\_sql = 'SELECT role FROM User WHERE id=' . $row['creator'];

   $creator\_result = $db->query($creator\_sql);

   $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

   if ($creator\_row['role'] == 'staff'){

     $creator\_sql = 'SELECT firstName, lastName FROM Staff WHERE userId=' . $row['creator'];

     $creator\_result = $db->query($creator\_sql);

     $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

     $name = $creator\_row['firstName'] . ' ' . $creator\_row['lastName'];

   }

   else{

     $creator\_sql = 'SELECT firstName, lastName FROM Student WHERE userId=' . $row['creator'];

     $creator\_result = $db->query($creator\_sql);

     $creator\_row = $creator\_result->fetch\_array(MYSQLI\_ASSOC);

     $name = $creator\_row['firstName'] . ' ' . $creator\_row['lastName'];

   }

   $content = $row['content'];

   $creator = $row['creator'];

   if (strlen($content) > 40){

     $content = $content . "...";

   }

   //generate HTML

   $row\_div = new html\_element('div');

   $row\_div->set('class', 'row');

   $col = new html\_element('div');

   $col->set('class', 'col');

   $link = new html\_element('a');

   $link->set('style', 'color:inherit; text-decoration: none');

   $link->set('href', 'message.php?id=' . $creator . '&name=' . $name);

   $link->set('target', '\_parent');

   $title = new html\_element('p');

   $title->set('class', 'lead');

   $title->set('text', $content);

   $image = new html\_element('i');

   $image->set('class', 'fa fa-envelope');

   $image->set('aria-hidden', 'true');

   $title->inject($image);

   $link->inject($title);

   $col->inject($link);

   $row\_div->inject($col);

   $col = new html\_element('div');

   $col->set('class', 'col');

   $title = new html\_element('p');

   $title->set('class', 'instructor');

   $title->set('text', $name);

   $col->inject($title);

   $row\_div->inject($col);

   $row\_div->output();

 }

 ?>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Index.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

<?php

 include 'header.php';

?>

     <div class="container">

       <div class="row justify-content-md-center">

         <div class="col whiteboard whiteboard-main">

           <iframe  src="feed.php" style="width: 100%; position: relative;"></iframe>

         </div>

         <div  class="col col col-lg-3 whiteboard">

           <p class="lead"> My Courses</p>

           <?php

           for ($i=0; $i<count($courseArray); $i++){

             $ptag = new html\_element('p');

             $ptag->set('class', 'lead');

             $ptag->set('text', $courseArray[$i]->get\_name());

             $link = new html\_element('a');

             $link->set('href', 'course.php?location=' . $i);

             $link->set('style', 'color:inherit; text-decoration: none');

             $link->inject($ptag);

             $link->output();

           }

           ?>

         </div>

       </div>

   <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>

   <script src="../../../../assets/js/vendor/popper.min.js"></script>

   <script src="js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* + Message.php

<?php

include '../session/staff-session.php';

include '../functions/html\_element.php';

?>

<!DOCTYPE html>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

   <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

   <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

 <?php

 include 'header.php';

     $id = $\_GET['id'];

   $name = $\_GET['name'];

 ?>

   <div class='container'>

   <div class="row justify-content-md-center"><div class="col whiteboard whiteboard-main" style="height: 300px ;position: relative;">

   <h3 class="text-center"><?php echo $name ?></h3>

   <iframe src="chat.php?id=<?php echo $id ?>"style="width: 100%;height: 85%;position: relative;" id='iframe'></iframe>

   </div>

     <div class="container whiteboard">

             <form id="confirmation" method="post" enctype="multipart/form-data" >

              <div class="form-group assgn">

                 <textarea class="form-control" rows="3" name="content" placeholder="Enter Text Here" id="textarea" required></textarea>

               </div>

                 <input type="submit" class="btn btn-primary assgn pull-right" name="input">

             </form>

       </div>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

   <script

     src="https://code.jquery.com/jquery-3.2.1.js"

     integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

     crossorigin="anonymous"></script>

   <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>');</script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js" integrity="sha384-b/U6ypiBEHpOf/4+1nzFpr53nxSS+GLCkfwBdFNTxtclqqenISfwAzpKaMNFNmj4" crossorigin="anonymous"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

    <script type="text/javascript">

     // Variable to hold request

var request;

// Bind to the submit event of our form

$("form#confirmation").submit(function(e){

   var formData = new FormData(this);

   $.ajax({

       url: "chat.php?id=<?php echo $id ?>",

       type: 'POST',

       data: formData,

       success: function (data) {

           document.getElementById('iframe').contentDocument.location.reload(true);

           $("form#confirmation").trigger('reset');

       },

       cache: false,

       contentType: false,

       processData: false

   });

   return false;

});

   </script>

 </body>

</html>

* + Whiteboardcalendar.php

<?php

include '../session/staff-session.php';

?>

<html lang="en">

 <head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

   <meta name="description" content="">

   <meta name="author" content="">

   <link rel="icon" href="../../../../favicon.ico">

   <title>WhiteBoard</title>

   <!-- Bootstrap core CSS -->

  <link href="../css/bootstrap.min.css" rel="stylesheet">

   <!-- Custom styles for this template -->

  <link href="../css/whiteboard.css" rel="stylesheet">

   <link rel="stylesheet" href="../font-awesome-4.7.0/css/font-awesome.min.css">

   <link rel='stylesheet' href='../fullcalendar/fullcalendar.css' />

   <script src='../fullcalendar/lib/jquery.min.js'></script>

   <script src='../fullcalendar/lib/moment.min.js'></script>

   <script src='../fullcalendar/fullcalendar.js'></script>

<!--<script src='http://fullcalendar.io/js/fullcalendar-2.1.1/lib/jquery.min.js'></script>

<script src="http://fullcalendar.io/js/fullcalendar-2.1.1/lib/jquery-ui.custom.min.js"></script>

   <script src='http://fullcalendar.io/js/fullcalendar-2.1.1/lib/moment.min.js'></script>

<script src='http://fullcalendar.io/js/fullcalendar-2.1.1/fullcalendar.min.js'></script>-->

   <script>

   $(document).ready(function() {

       $('#calendar').fullCalendar({

           events: 'get-events.php'

       });

   });

      </script>

 </head>

 <body style="background-color: <?php echo $session\_primaryColor; ?>">

   <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">

     <a class="navbar-brand" href="index.php">WhiteBoard</a>

     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-expanded="false" aria-label="Toggle navigation">

       <span class="navbar-toggler-icon"></span>

     </button>

     <div class="collapse navbar-collapse" id="navbarsExampleDefault">

       <ul class="navbar-nav mr-auto">

         <li class="nav-item active">

           <a class="nav-link" href="whiteboardcalendar.php">Calendar <span class="sr-only">(current)</span></a>

         </li>

         <li class="nav-item">

           <a class="nav-link" href="course-view.php">Courses</a>

         </li>

         <li class="nav-item">

         </li>

       </ul>

       <a class="btn btn-default action-button navbar-text navbar-right actions" role="button" href="../logout.php">Log Out</a>

     </div>

   </nav>

     <div class="container">

       <div id='calendar' class="whiteboard">

       </div>

   <footer style="background-color: <?php echo $session\_secondaryColor; ?>"> <p>  </p> </footer>

   </div>

   <!-- Bootstrap core JavaScript

   ================================================== -->

   <!-- Placed at the end of the document so the pages load faster -->

  <!-- <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>-->

  <!-- <script>window.jQuery || document.write('<script src="../../../../assets/js/vendor/jquery.min.js"><\/script>')</script>-->

   <script src="../fullcalendar/popper.js"></script>

   <script src="../js/bootstrap.min.js"></script>

   <!-- IE10 viewport hack for Surface/desktop Windows 8 bug -->

   <script src="../../../../assets/js/ie10-viewport-bug-workaround.js"></script>

 </body>

</html>

* Connection.php

<?php

   $servername = getenv('IP');

   $username = getenv('C9\_USER');

   $password = "";

   $database = "whiteboard";

   $dbport = 3306;

   //Create Connection

   $db = new mysqli($servername, $username, $password, $database, $dbport);

   //Check Connection

   if ($db->connect\_error) {

       die("Connection failed: " . $db->connect\_error);

   }

   //echo "Connected Successfully (".$db->host\_info.")";

* Login.php

<?php

session\_start();

include "connection.php";

if (isset($\_SESSION['login\_user'])){

       if ($\_SESSION['role'] == 'admin'){

           header( 'Location: /admin/interface.php' );

       }

       else if ($\_SESSION['role'] == 'student'){

           header( 'Location: /student/index.php');

       }

   }

   $error = ""; //declare error variable

   if($\_SERVER["REQUEST\_METHOD"] == "POST") {

       //information sent from form

       $username = mysqli\_real\_escape\_string($db,$\_POST['username']);

       $password = mysqli\_real\_escape\_string($db,$\_POST['password']);

       $sql = "SELECT \* FROM User WHERE username = '$username'";

       $result = mysqli\_query($db, $sql) or die('error getting data');

       $row = mysqli\_fetch\_array($result,MYSQLI\_ASSOC);

       $hash = $row['password'];

       $id = $row['id'];

       $role = $row['role'];

       $num\_rows = mysqli\_num\_rows($result);

       if ($num\_rows == 0){

           $error = "Incorrect username or password";

       }

       else {

           if ((password\_verify($password, $hash))) {

               $\_SESSION['login\_user']=$id;

               $\_SESSION['role']=$role;

               if ($\_SESSION['role'] == 'admin'){

                   header( 'Location: /admin/interface.php' );

               }

               else if ($\_SESSION['role'] == 'student'){

                   header( 'Location: /student/index.php');

               }

               else if ($\_SESSION['role'] == 'staff'){

                   header( 'Location: /staff/index.php');

               }

           }

           else{

               $error= "Incorrect username or password";

           }

       }

   }

?>

<!DOCTYPE html>

<html>

<head>

   <meta charset="utf-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>WhiteBoard Login</title>

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap-theme.min.css">

   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css">

   <link rel="stylesheet" href="css/Login-Form-Clean.css">

   <link rel="stylesheet" href="css/styles.css">

</head>

<body>

   <div class="login-clean">

       <form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]);?>">

           <h2 class="sr-only">Login Form</h2>

           <h2 class='text-center'>WhiteBoard</h2>

           <div class="illustration"><i class="icon ion-ios-browsers-outline"></i></div>

           <p class="text-danger text-center"><?php echo $error;?></p>

           <div class="form-group">

               <input class="form-control" type="text" name="username" required="" placeholder="Username">

           </div>

           <div class="form-group">

               <input class="form-control" type="password" name="password" required="" placeholder="Password">

           </div>

           <div class="form-group">

               <button class="btn btn-primary btn-block" type="submit" style="background-color:#505e6c;">Log In</button>

           </div></form>

   </div>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

   <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>

</body>

</html>

* Logout.php

<?php

   session\_start();

   if(session\_destroy()){

       header('Location: login.php');

   }

?>