Fixture Management and Notification System

H446-03

Ayokunle Toye Ikusemori

Candidate Number:5111

Centre Name: Queens' School

Centre Number:17611

CONTENTS

A.Analysis

Description of the problem

Currently, there is a system in place when there is a sports fixture at my school where an email is sent to every teacher informing them of students that will not be in lesson.

The problem with this is that regardless of whether you tutor the students you still receive an email which clutters the inbox of teachers and wastes time. Furthermore, even if the fixture is called off the email is sent regardless. After speaking to numerous teachers, I found that they were generally annoyed and wished for a solution

Outline of the project

In response to the problem I am going to create a web-based management system that will contain a database of students with their teachers along with information on their sports. Emails will be able to be sent to the correct teacher when a child has a sports fixture and will not be in lesson. Upcoming fixtures will also be displayed on the page. Users of the solution will exclusively be able edit team info and fixture info. A log in system will be implemented in order to provide basic security for the stakeholder and distinguish who will be able to access certain pages

Stakeholders

Identification of Stakeholders

The general stakeholder of this solution is the school that is generally having the issue however within the school only two areas are really affected.

- The PE Department

  The main users of my program will be the teachers in the sports department as they will be the main source of input data by updating the database with team and fixture information by adding/editing/removing teams and fixtures.

- Teachers in the school
  The teachers will receive the emails from the program letting them know when students in their class will not in lesson.

Stakeholder requirements

Requirement	Explanation
Database support	This is needed as the stakeholders are part of a school which uses databases to hold information about their students.
A log in page	This is a basic security requirement to allow only the relevant users to access the system.
Admin Accounts	This is required by the stakeholder so that certain users in the department can see and edit areas in the system while others cannot.
Fixture information can be accessed from the main page	This is required so that fixtures information can be viewed or edited.
Team and fixture information can be edited on their respective pages.	This is also required as it enables the database to be dynamically updated without the creator of the database accessing the database itself. It also enables information to be removed from the system if it is not relevant to the system anymore
Emails can be sent correctly when a fixture is upcoming	This is required as this is the problem that the stakeholder requires a solution for

Justification Of computational methods.

The main reason for using computational methods for my project is that although the problem arose due to the use of an inefficient system created by computational methods, it would be wise to use computational methods to create a solution as the problem wasn't with using computational methods, but with using an inefficient one. Computational methods provide the potential to create a system that would be greatly beneficial to the user. Computers can process substantial amounts of data at a very efficient speed unlike a system such as a paper-based system which would take a large amount of time to process and be an inconvenience to my user especially when the stakeholder requires a database which can hold vast amounts of data. With computational methods, I can display the data mentioned previously in arrays and lists which makes understanding the data easier to the stakeholder which would be beneficial as the data would be somewhat complex.

Research

interview with Head of sports Department

Q: What do you think of the emails that get sent when a student is not in school?

A: They were good initially but now they create a lot of problems for us as teachers complain that they're receiving emails for students they don't even teach

Q: What do you think of the way sports fixtures are handled as whole in the school?

A: There's really nothing special about them, we announce them, and they get put up on the notice board.

Q: What do you think could be improved about the way they are handled?

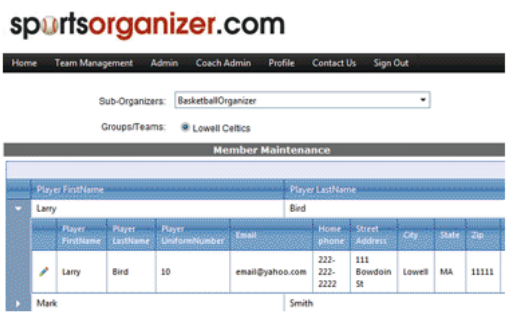
A: If we could manage them on a computer, I think it would be a lot easier as we as a department use a lot of our paper budget on printing out paper to display the fixtures. It can also get crowded on the board when there are a lot of fixtures.

Q: As I previously explained to you, I am creating a system mainly based around the email problem you previously stated, what else would you like to see?

A: If possible and if it wasn't too much trouble, could you create it online? It would be easier for the department to use and understand.

Existing Systems

Screenshot1: Main menu, options for managing a team



Screenshot 2: Managing players

An existing system I researched was one created by http://sportsorganizer.com/EGroupMain/SO/Index.aspx. It is a web-based sports management system that utilizes some features that would be useful to the users solution system based on the images displayed on the website and above. Features such as a log in system, team management and admin sections.

However, the register page on the website is not functional and it is not possible to investigate the system further and discover any other features it has. The Pictures displayed on the website however give good insight to what a good management system could look like but seem too complex to produce in the time frame given so I shall produce something simpler in order to create the solution on time for the stakeholder.

The user could also implement a system that uses an email template to send an email to whoever they select based on the information they have provided. However, this would be very time consuming to the user especially if they have a large team and have to input the names of every student in the team.

Features of Proposed Solution

- Database support
- Table showcasing upcoming fixtures
- The ability to create/edit/delete a team
- The ability to create/edit/delete a team
- The ability to send an email to the correct teacher when a fixture is approaching

Concept of my solution considering the research:

Considering what I have researched, I will create a web-based system using php ,JavaScript and html that will utilise a log in feature in order to access pages that will enable the user to manage fixtures and teams in a database with which the user will be able to send emails to the teachers who would not have certain students in their lesson due to a fixture.

However, due to the complexity potential of this solution I have set a few rules to make a functioning solution within the time given.

1. Students can only join one team- This allows me to keep my database simple as by assigning a student to one team I do not need to create additional columns for the extra teams they may be in
2. Students in a year have the same teacher. - This limits the number of teachers in my database down to five
3. Fixtures only occur at period 5 on a timetable – This again limits the number of teachers that need to be sent an email to 5
4. The timetable is a one-week timetable and not two- This ensures that the teachers stay the same again limiting the teachers being sent an email to 5

Limitations of my solution

With reference to the concept of my solution, the main limitation of my solution is that it does not fully reflect the real situation the stakeholder will experience such as student's being in multiple teams and having multiple teachers within the year group. However, given enough time the solution could become a lot more practical and when used in a real scenario would function effectively.

One of the limitations of my solution is that it is web-based and therefore stored on a server. If the server fails or malfunctions it would make the system inaccessible or difficult to use. Furthermore, if any improvements are made to the solution, the server would not be available until the solution has been updated unlike a program where the previous versions can be used while the update is being worked on.

Since my solution uses students as data realistically, I would not be able to create a solution based on 300+ students in the time given to create the solution. So, to deal with this problem I shall use a sample of students that have been created by me as I cannot use actual students in my school due to the data protection

My solution also does not validate against SQL injection which is a dangerous threat to the database holding the data of my solution. However, since the solution would be used on a LAN network the threat is

minimalised to someone in the school entering a dangerous SQL query in a text field as opposed to a someone visiting the site and entering a malicious query.

Hardware and Software Requirements

Since the system I am creating is web based the system requirements are based on the minimum requirements to run the browsers chrome and Internet explorer which are the common browsers used by my stakeholders.

Chrome:

Minimum Processor: Intel Pentium 4 or later

Hard Disk Space: 100mb

RAM: 128 mb

Windows 7, Windows 8, Windows 8.1, Windows 10 or later

Internet Explorer

Computer/processor	1 gigahertz (GHz) 32-bit (x86) or 64-bit (x64)
Operating system	• Windows 10 (32-bit or 64-bit) • Windows 8.1 Update (32-bit or 64-bit) • Windows 7 with SP1 (32-bit or 64-bit) • Windows Server 2012 R2 • Windows Server 2008 R2 with SP1 (64-bit only)
Memory	• Windows 10 (32-bit)-1 GB • Windows 10 (64-bit)-2 GB • Windows 8.1 Update (32-bit)-1 GB • Windows 8.1 Update (64-bit)-2 GB • Windows 7 with SP1 (32-bit or 64-bit)-512 MB • Windows Server 2012 R2-512 MB • Windows Server 2008 R2 with SP1 (64-bit only)-512 MB
Hard drive space	• Windows 10 (32-bit)-16 GB • Windows 10 (64-bit)-20 GB • Windows 8.1 Update (32-bit)-16 GB • Windows 8.1 Update (64-bit)-20 GB • Windows 7 with SP1 (32-bit)-70 MB • Windows 7 with SP1 (64-bit)-120 MB • Windows Server 2012 R2-32 GB • Windows Server 2008 R2 with SP1 (64-bit only)-200 MB
Drive	CD-ROM drive (if installing from a CD-ROM)
Display	Super VGA (800 x 600) or higher-resolution monitor with 256 colors

Peripherals for both systems: A keyboard, mouse and an internet connection

Success Criteria

User must be able to input data into the input boxes when necessary.	This allows the user to be able to login and use the functions of the solution. Without this criterion being successful the solution is useless
User must be able to click the buttons and the menu items	These buttons allow the user to submit forms and visit areas of the solution. Without this criterion the user again cannot use the solution.
User must be able to add a team into the database	This allows the user to update the database which is necessary when a new team is introduced.
User must be able to edit a team	This allows the user to correct any errors produced when creating a team but also allows the user to update teams when necessary to the user
User must be able to delete a team	This allows the user to remove any data that is unnecessary and prevent any mistakes caused by using old teams in fixtures
User must be able to add a fixture into the database	This allows the user to be able to create fixtures of which the details can then be used when sending an email
User must be able to edit a fixture in the database	This again allows the user to correct any errors they may have produced when adding a fixture to the database but also allows them to update a fixture when necessary. E,g, postponed fixture
User must be able to send an email	This criterion is necessary as without it being successful the stakeholder's problem is not solved.
User can use the solution without much help	This is somewhat necessary as if the user requires help with every area of the solution, the solution is not really practical and would be complex
Visual Requirements	
The interface is viewable without much strain to the eyes	This criterion is necessary as the user could be looking at the screen for long periods of time while entering large amounts of data so making sure the interface doesn't hurt the eyes is neccesary
The user interface is not intimidating and user friendly	If the interface is complex and full of boxes and information the user may have a hard time understanding it and may be deterred from using it so it is important that the interface is simple

Algorithm Requirements	
Page information is displayed when visited	This criterion is necessary as without the pages being displayed the user would not be able to input any information to the solution. E, g, button clicks, Team and fixture data etc.
User is redirected to the correct areas when they commit an action that would cause them to redirect. E.g. form submission, item on the navigation bar clicked	This criterion must be successful as it ensures that the user is able to move around the solution and use different functions e.g. add a team
The database can be connected to and affected in different ways	This criterion needs to be successful as the database holds the login information for the solution and should the database be unable to be connected to, the user will not be able to use the solution. It is also necessary as it allows the functions to operate correctly and prevent the user from encountering error messages unnecessarily.
The correct teacher is emailed when an email is sent	This criterion is necessary as without it being successful the solution would not solve the problem.

B.Design

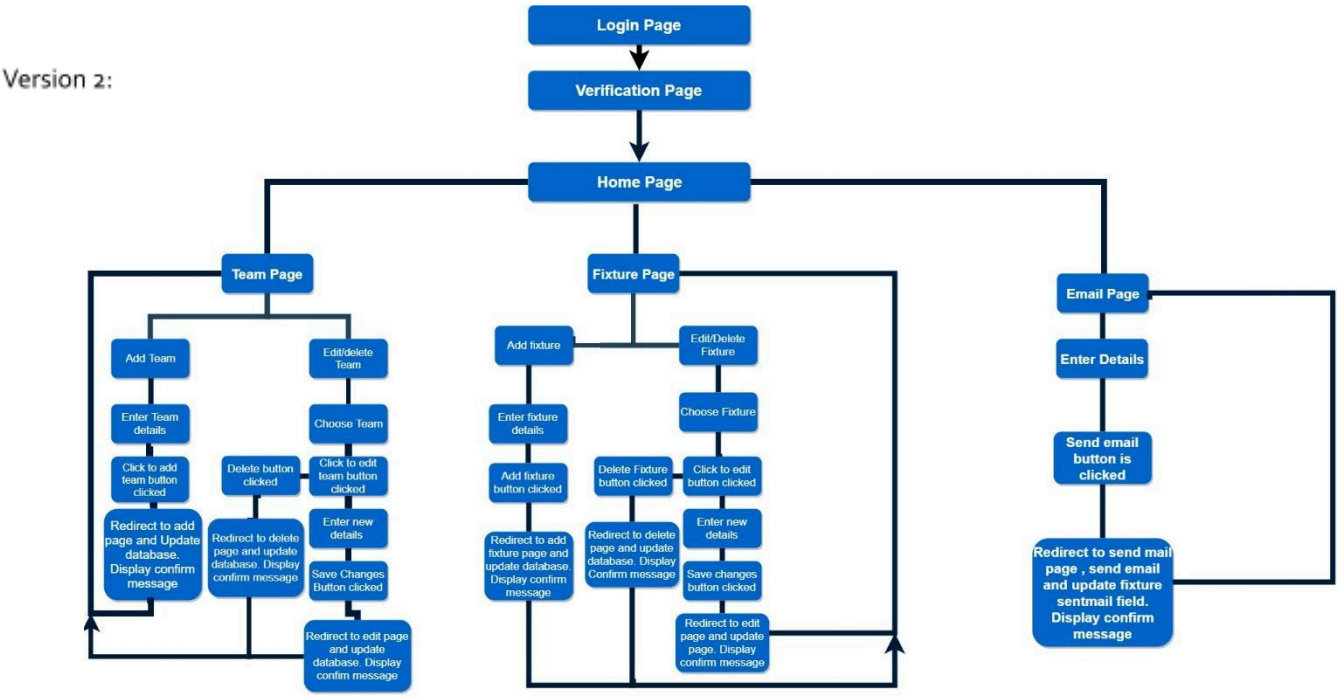
Structure of the solution

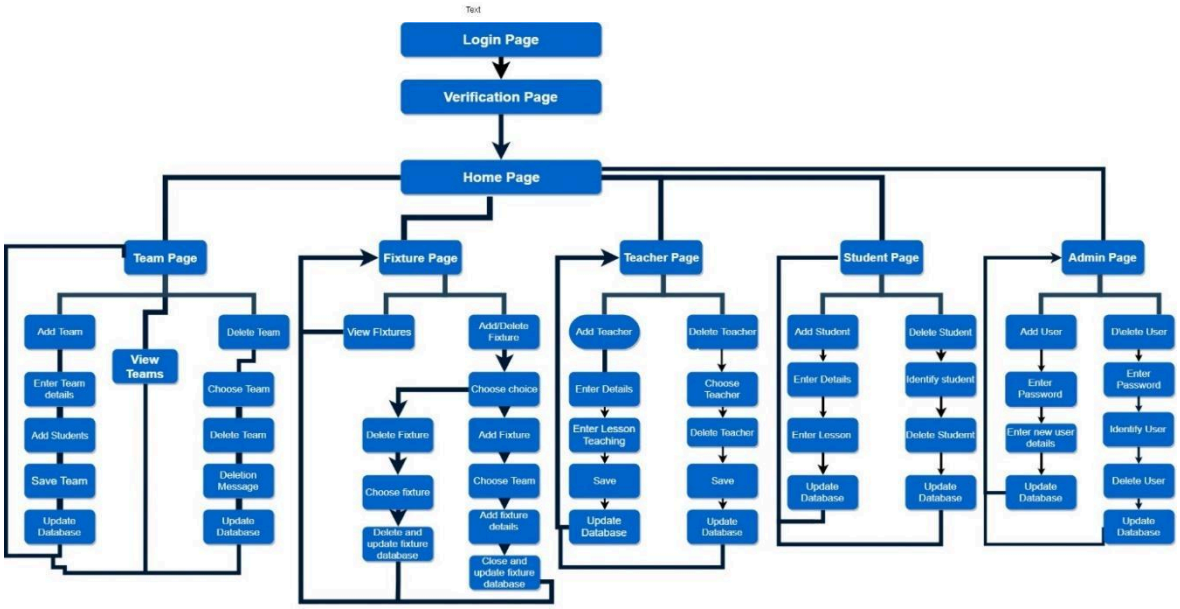
As previously stated, in order to use my solution to its full potential, student and teacher information would have to be imported to my database. However, because of the Data Protection Act I am unable to receive this information. Because of this, I will create my solution but without the student, teacher and admin page. I will use a set of students created by me instead.

This was the initial structure of the solution however, it has now been changed to reflect the removal of the previously stated pages and the updated stakeholder requirements (see page 20)

Version 1:

Version 2:


Login Page
Verification Page
Home Page
Team Page
Add Team
Enter Team details
Click to add team button clicked
Redirect to add page and Update database. Display confirm message
Edit/delete Team
Choose Team
Click to edit team button clicked
Enter new details
Save Changes Button clicked
Redirect to edit page and update database. Display confirm message
Delete button clicked
Redirect to delete page and update database. Display confirm message
Fixture Page
Add fixture
Enter fixture details
Add fixture button clicked
Redirect to add fixture page and update database. Display confirm message
Edit/Delete Fixture
Choose Fixture
Click to edit button clicked
Enter new details
Save changes button clicked
Redirect to edit page and update page. Display confirm message
Delete Fixture button clicked
Redirect to delete page and update database. Display Confirm message
Email Page
Enter Details
Send email button is clicked
Redirect to send mail page , send email and update fixture sentmail field. Display confirm message


Text
Login Page
Verification Page
Home Page
Team Page
Add Team
Enter Team details
Add Students
Save Team
Update Database
View Teams
Delete Team
Choose Team
Delete Team
Deletion Message
Update Database
Fixture Page
View Fixtures
Add/Delete Fixture
Delete Fixture
Choose fixture
Delete and update fixture database
Choose choice
Add Fixture
Choose Team
Add fixture details
Close and update fixture database
Teacher Page
Add Teacher
Enter Details
Enter Lesson Teaching
Save
Update Database
Delete Teacher
Choose Teachor
Delete Teacher
Save
Update Database
Student Page
Add Student
Enter Details
Enter Lesson
Update Database
Delete Student
Identify student
Delete Student
Update Database
Admin Page
Add User
Enter Password
Enter new user details
Update Database
Delete User
Enter Password
Identify User
Delete User
Update Database

What I will use to create my solution

Windows 10 operating system- Generally used operating system with the ability to run the programs I will need to create my solution.

XAMPP – is a program that uses multiple packages and programs in order to simulate a web server using the localhost hostname and other components.

It features:

Apache HTTP server- that supports IPV6 addresses, user and session tracking, HTTP/2 protocol support and contains embedded php scripting. These features are especially useful as it emulates the functionality of a live web server while being up to date with IP and web protocols

MariaDB or MYSQL- Provides an interface that allows the user to create and manage databases. It gives a detailed view of tables, shows relationships between the tables and allows the import of files such as excel csv files to make importing large amounts of data to the database easy. This is essential to my solution as I will need a database of various users, students, teachers, teams and fixtures in order to create it.

PHP Support – Which is especially useful to my solution as PHP is major part of webpages and without it I will not be able to make a solution as effective as the stakeholder requires. Using Php I can connect to the MariaDB and create, delete or manage the database using SQL and handle users and their sessions which is a major part of my solution.

Notepad++ - is a source code and text editor that allows for multiple files to be open and edited in a single instance. It also features syntax highlighting, auto complete suggestions and highlighting the location of things such as open and close bracket which is especially useful when there are nested loops.

Microsoft Access – is a database management system program that can be used to create databases and tables, simulate queries and view database relationships.

Google Chrome Browser – is a modern browser that complies with the latest protocols which allows me to observe if the solution works well in the current web environment. Its feature of the inspect element and inspect console allows me the individual elements on the page and see what version of my source code us being run.

Why I will use these programs to create my solution

XAMPP- Provides a way to access a web server without having to pay for one. It also gives me full control on what I can do on the server as opposed to one provided by a company or someone else. Furthermore, the multitude of extra features will make the task of creating my solution easier.

Notepad++ - will make the coding of the solution a lot easier with its features mentioned above and with the ability to open multiple files in an instance I can seamlessly move between different sections of my solution and edit them. Also requires no fees

Microsoft Access- Allows me to plan and create a database I can use for the my solution where I can make modifications without affecting the actual solution's database

Google chrome browser- Easy to use modern browser with a great interface that I can use to view how my pages would look like to the user. Also, free to install.

Decomposition of the problem

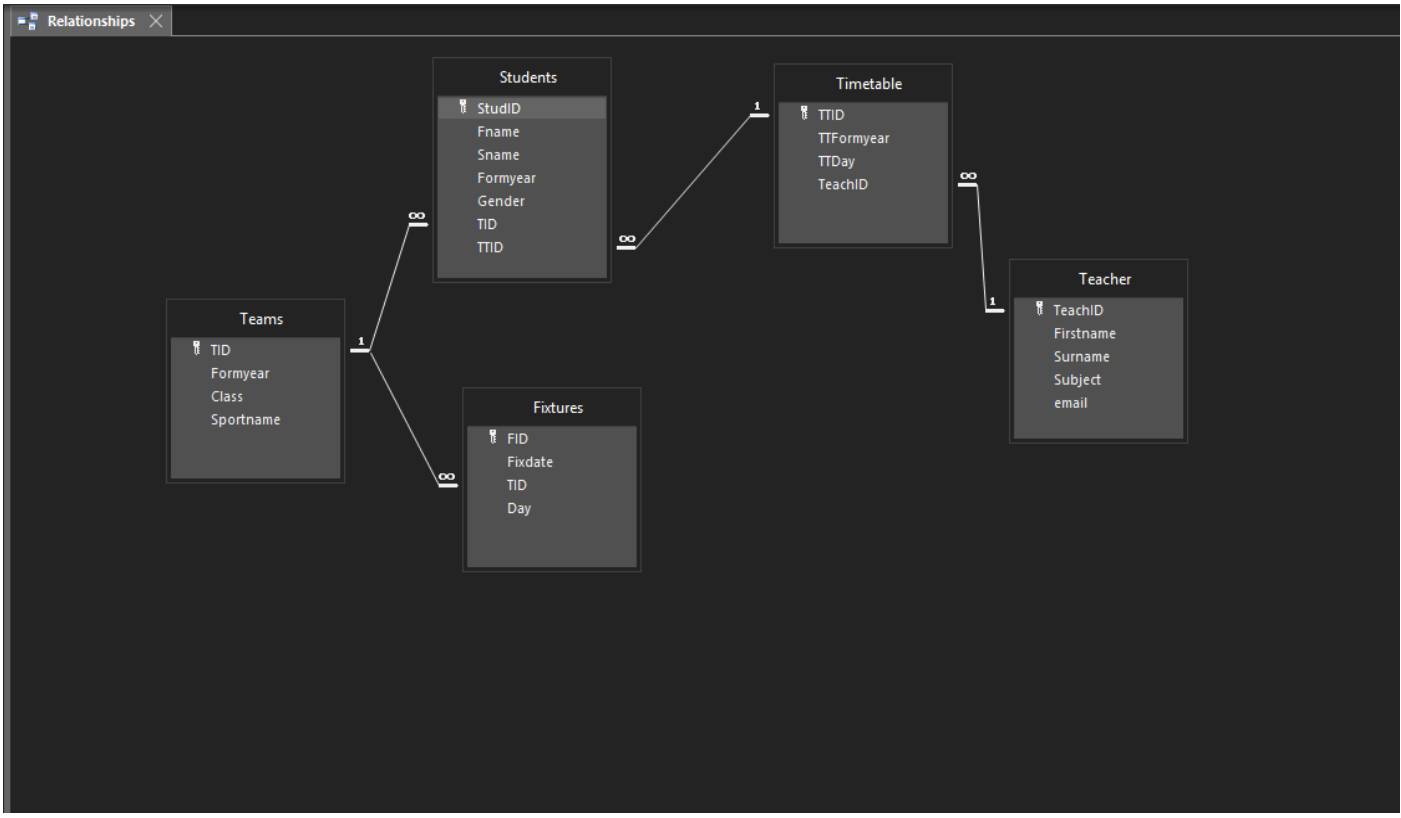
Initial concept of my solution:

The Database:

Since my solution heavily relies on a database to function, I had to plan and create a database before I could implement it into my solution. To do this, i used Microsoft's Access program. With it I could get a detailed view of tables and their relationships with reports that would show me the effectiveness of the database

I identified all the tables I would require for my solution and created a mock-up version of them. However, when creating these tables I realised I could not create my database to be in Third Normal Form. Third Normal Form is where a database is in Second Normal Form and its attributes solely depend on the key. In order to preserve the integrity of a database, having it in Third Normal Form is a major part of ensuring this. My database could not be in Third Normal Form as since I was dealing with multiple students and teams, I was unable to find a way of unique assigning a student to a team without duplicating the data when doing the same with another student. However, a database is still able to function in Second Normal Form as with my database which is still able to have partial key dependency, no duplicate attributes or many to many relationships.

View of my database and its table relationships:

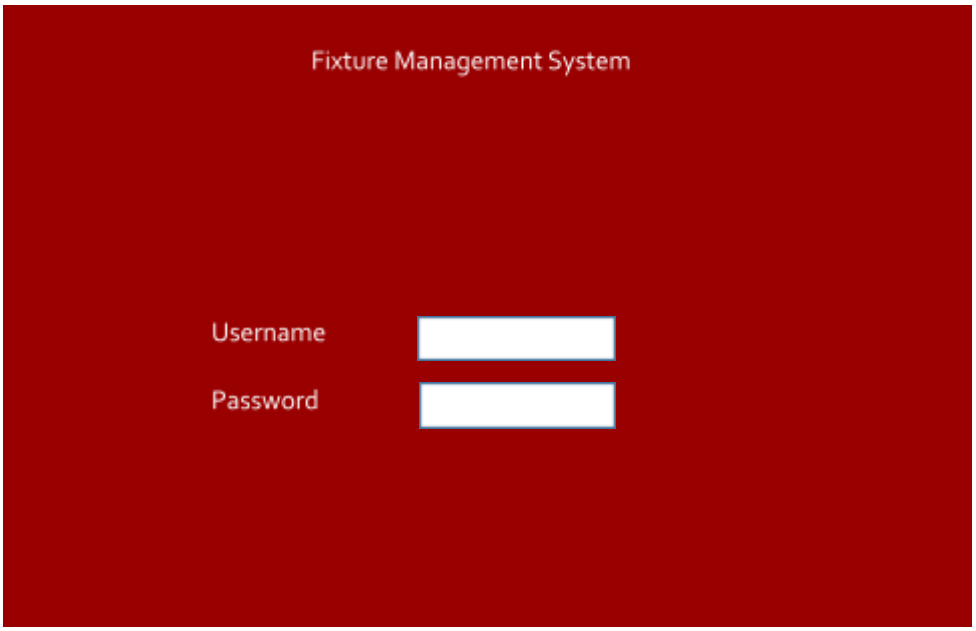


For my solution to work I would have to link all the tables together and select the email of the teacher who is teaching a year group of which the team in the Year Group is having a fixture. The students that are in the team will be sent to the teacher as they wouldn't be in lesson

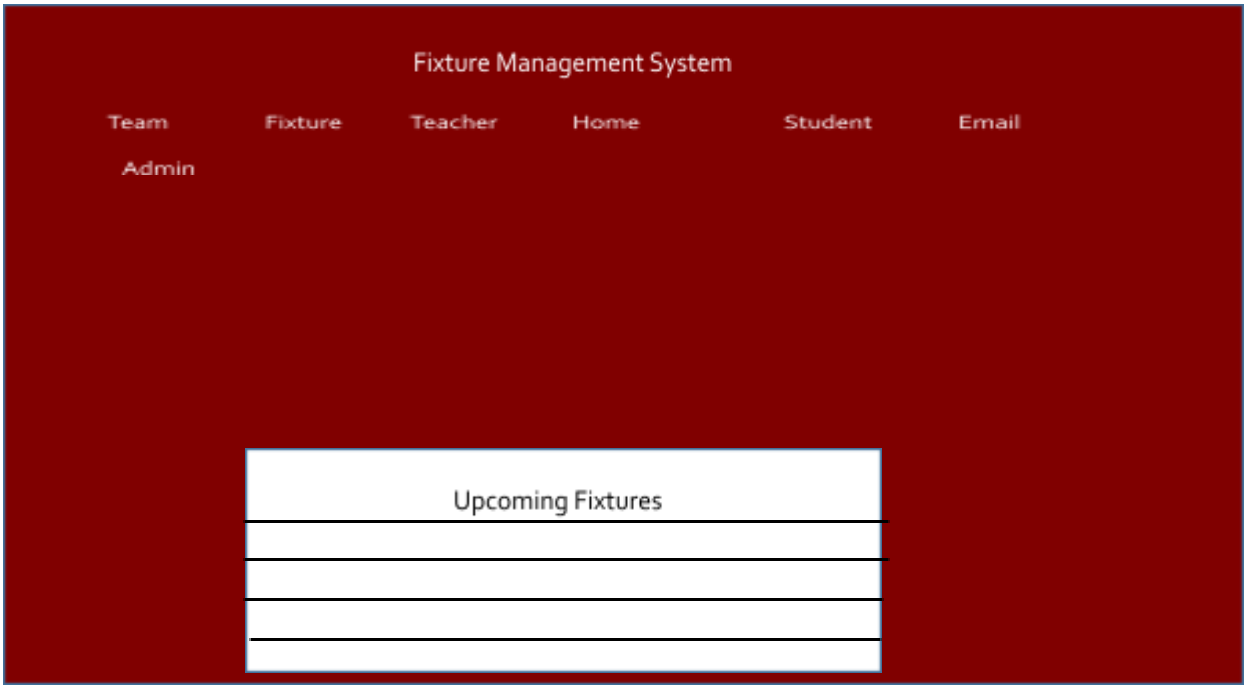
The Webpages Design

All pages will have a background colour of a burgundy to represent the colours of the school the solution will be used in. The text colour will be white to ensure the text stands out against the background colour so the user is easily able to read it.

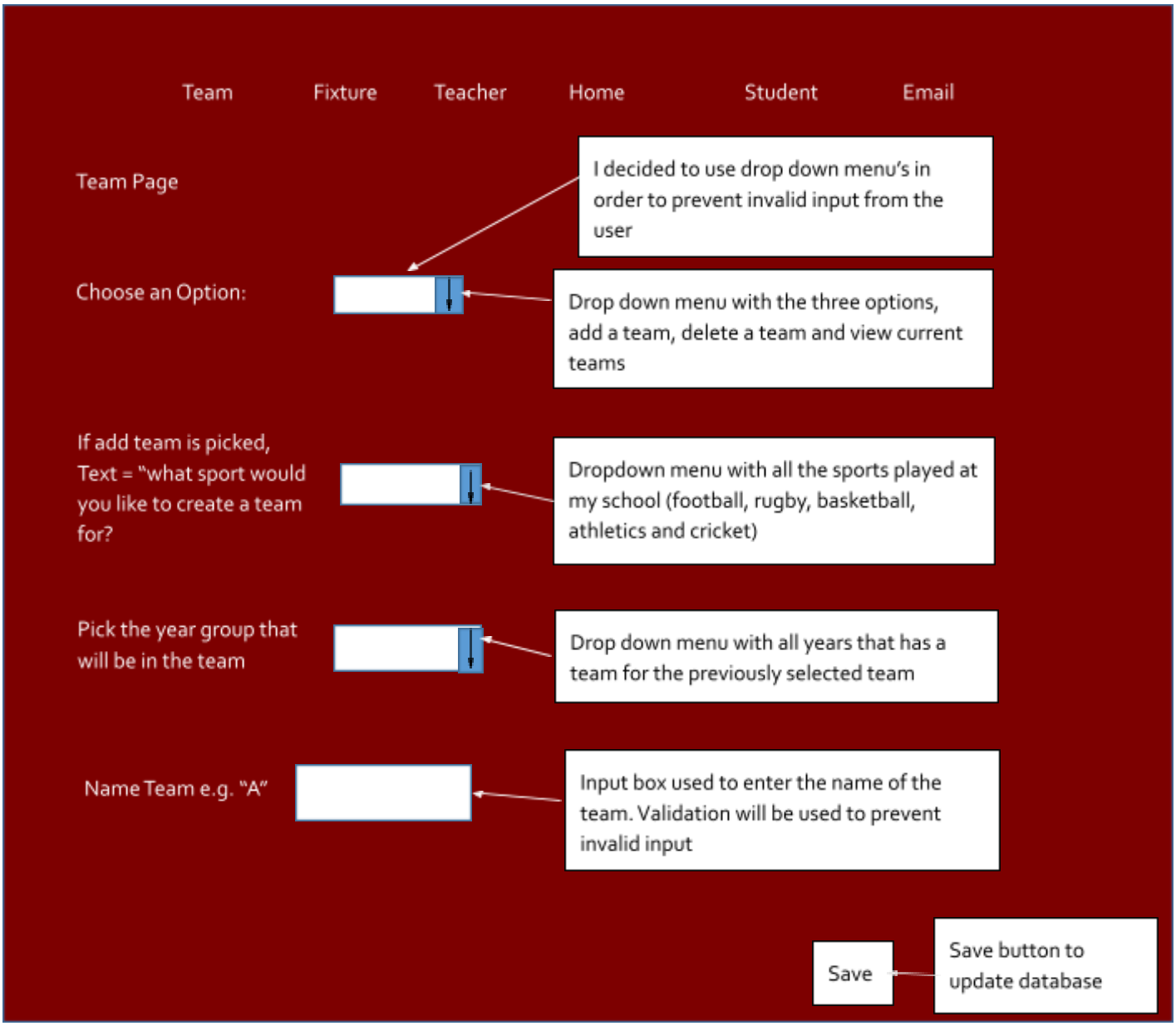
Login Page:



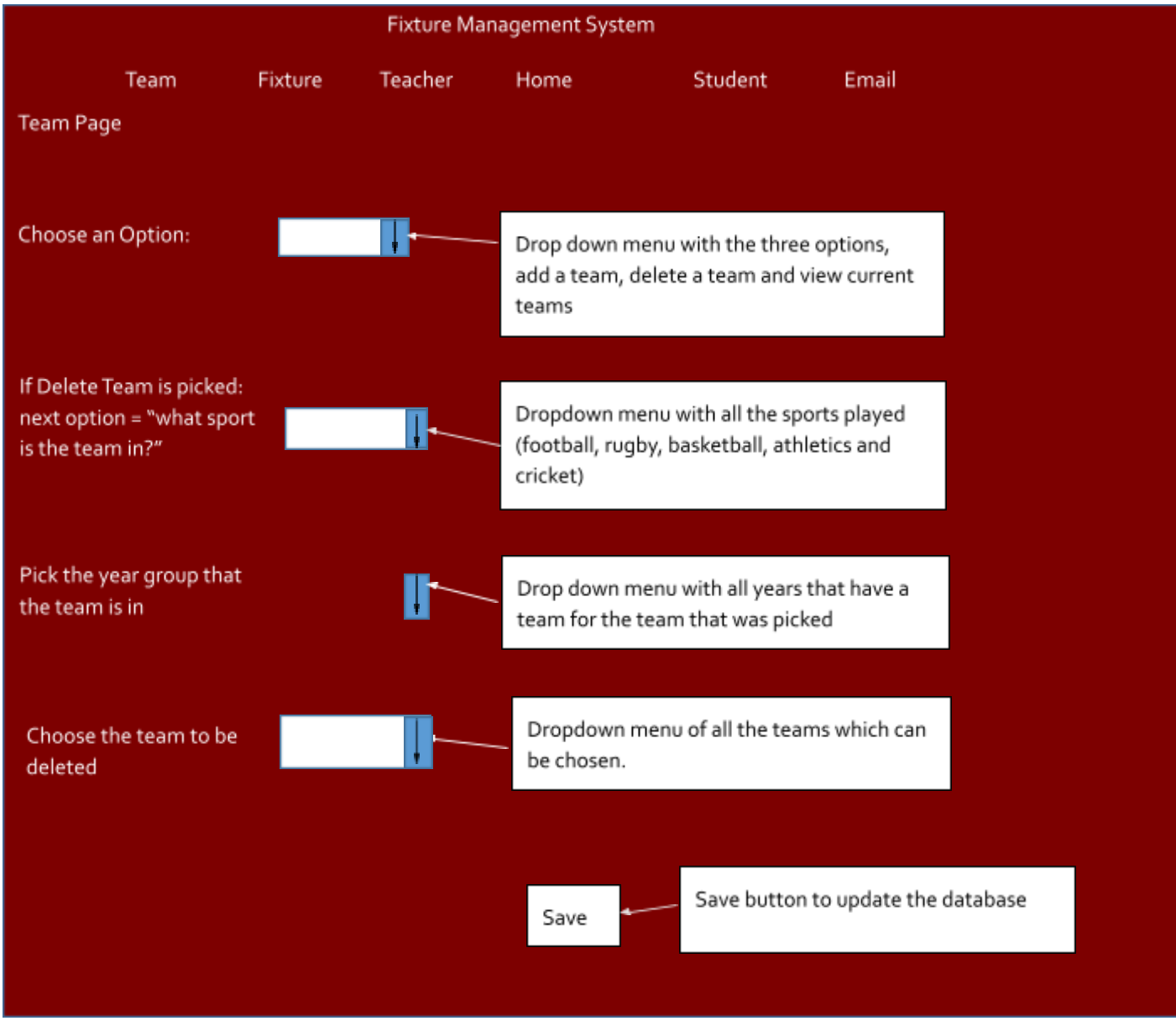
Home page

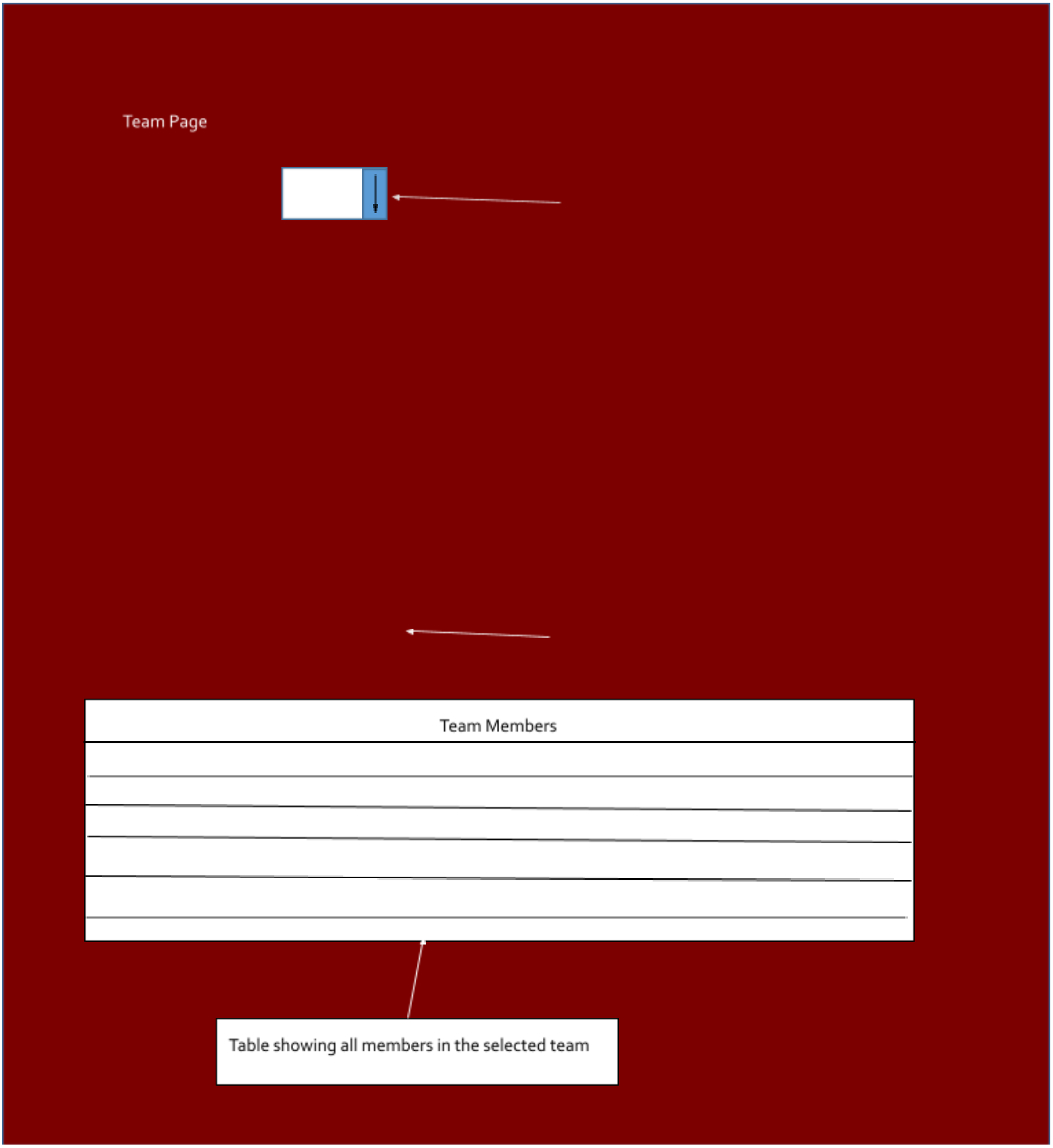

Fixture Management System
Team
Fixture
Teacher
Home
Student
Email
Admin
Upcoming Fixtures

Team Page when "create team" is chosen



Team Page when "Delete page" is chosen



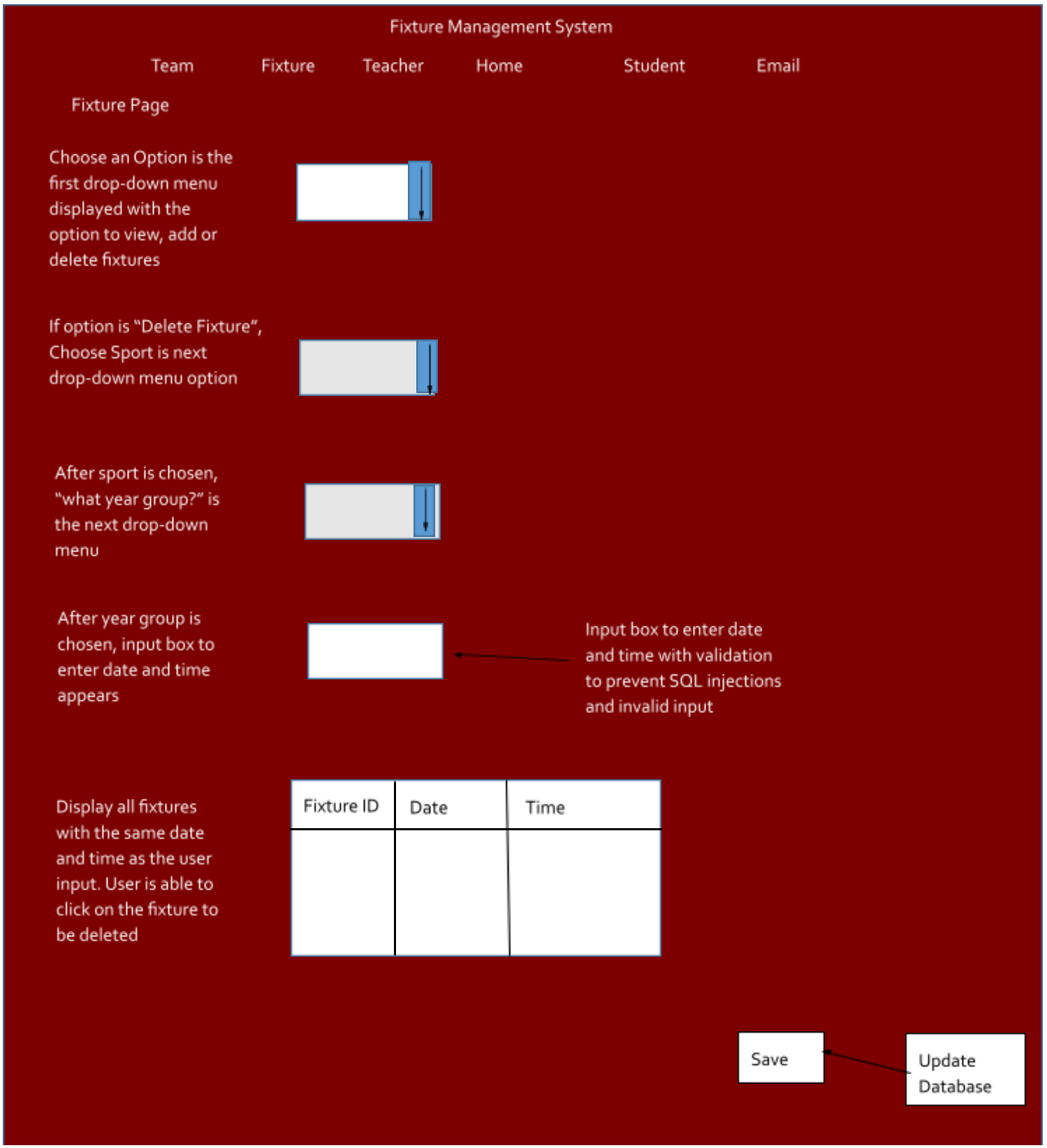


Team page when "view teams" is picked

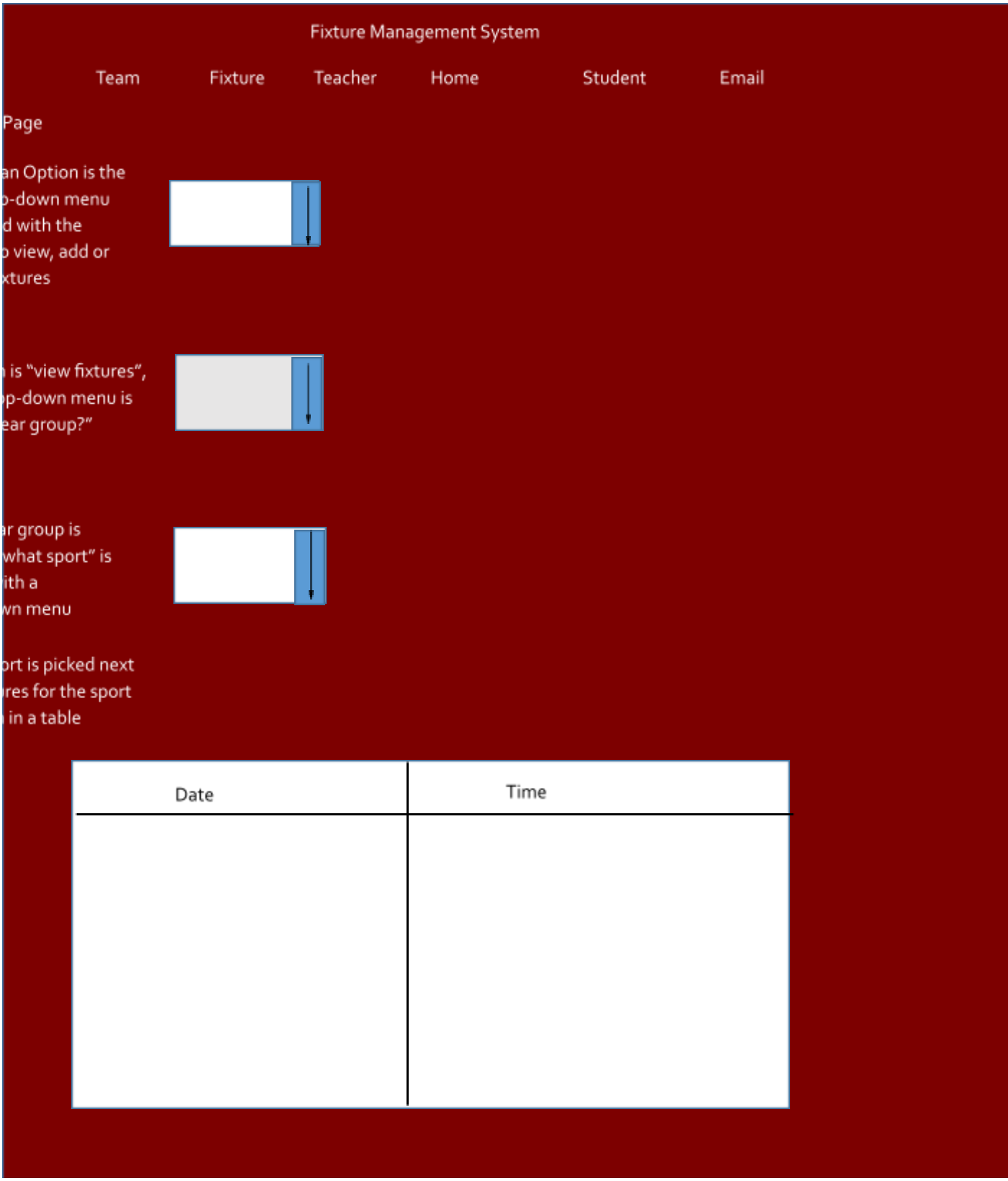
Drop down menu with the three options, add a team, delete a team and view current

Fixture page when "add fixture is selected"

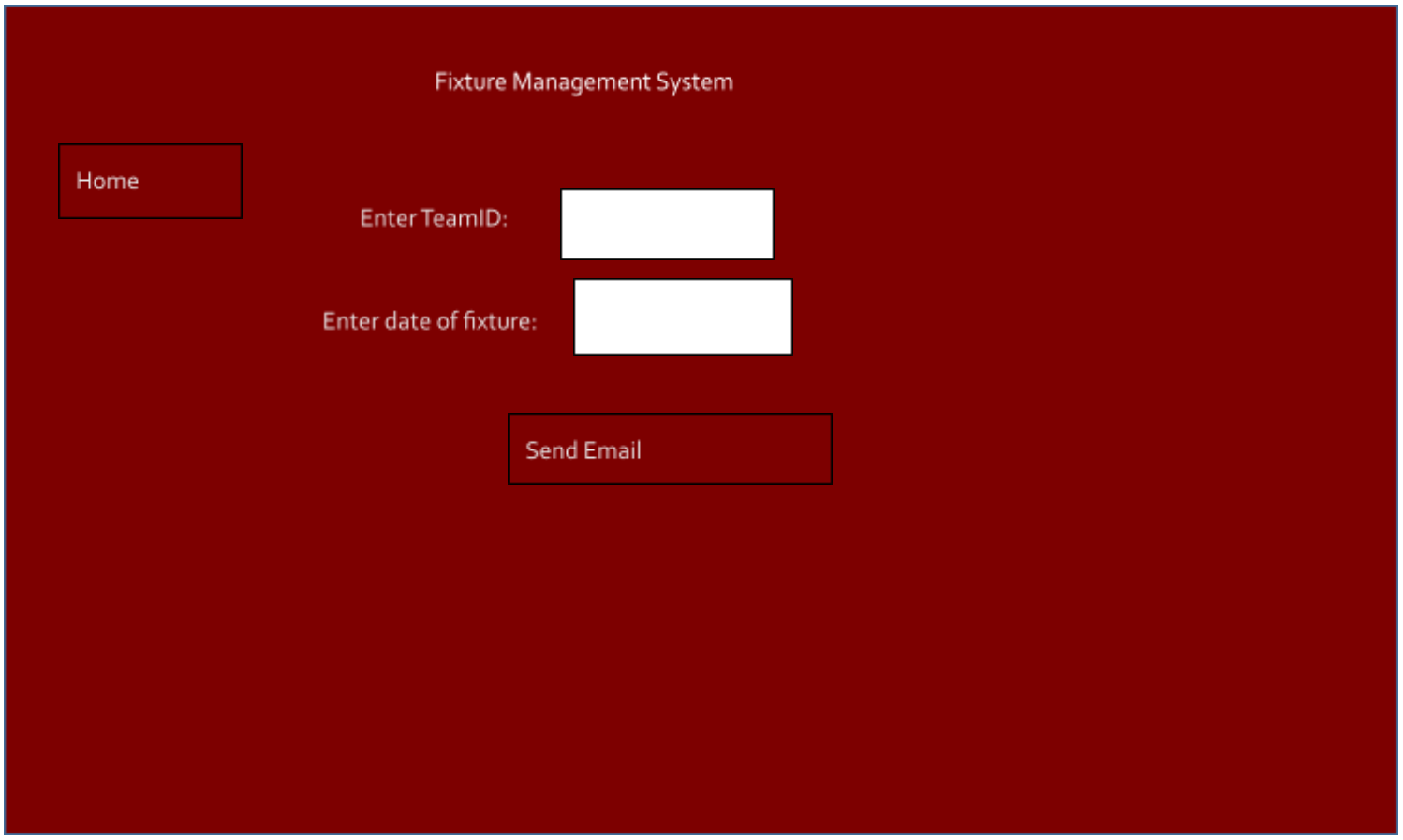
Fixture page when "delete fixture" is selected



Fixture Page when "view fixtures" is selected



Email Page:


Fixture Management System
Home
Enter TeamID:
Enter date of fixture:
Send Email

After beginning to design this solution I identified problems and usability issues of the solution that needed to be changed.

- Sequenced dropdown menu's in relation to adding/deleting/editing a fixture or team
  The inclusion of the dropdown menu was to make the program appear more appealing however, I felt that it would be time consuming to select the options and that it would be a lot easier having separate pages with the information right in front of you.

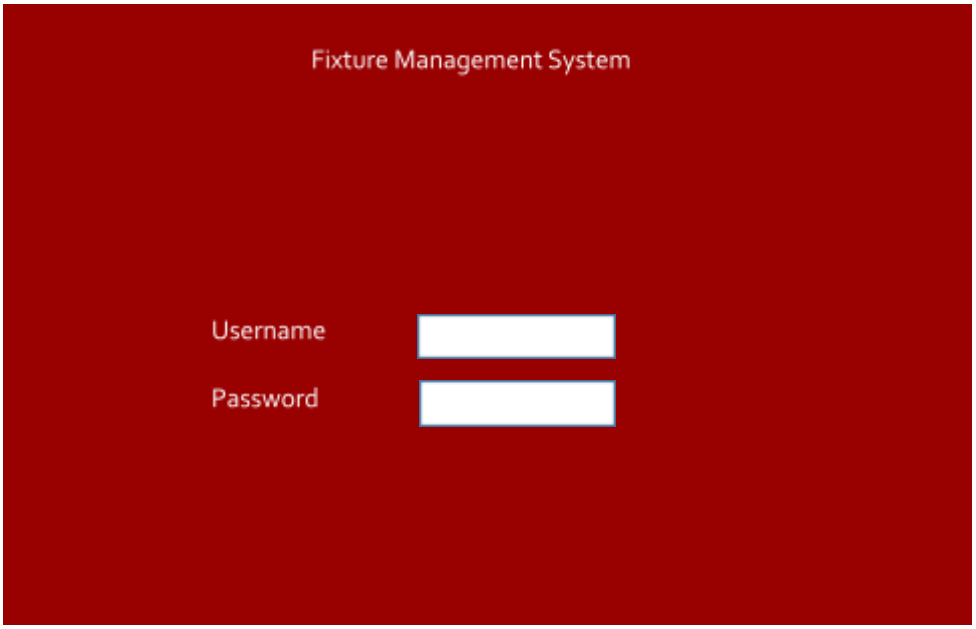
- Ability to add students
  Since the solution I intended to create would use real students, I nor the stakeholder would be able to access the information and change it. So the initial feature to add students was removed.

- The Ability to view students
  Since the feature to add students was removed the option to view students would have to be removed as the stakeholder would not be able to access the information

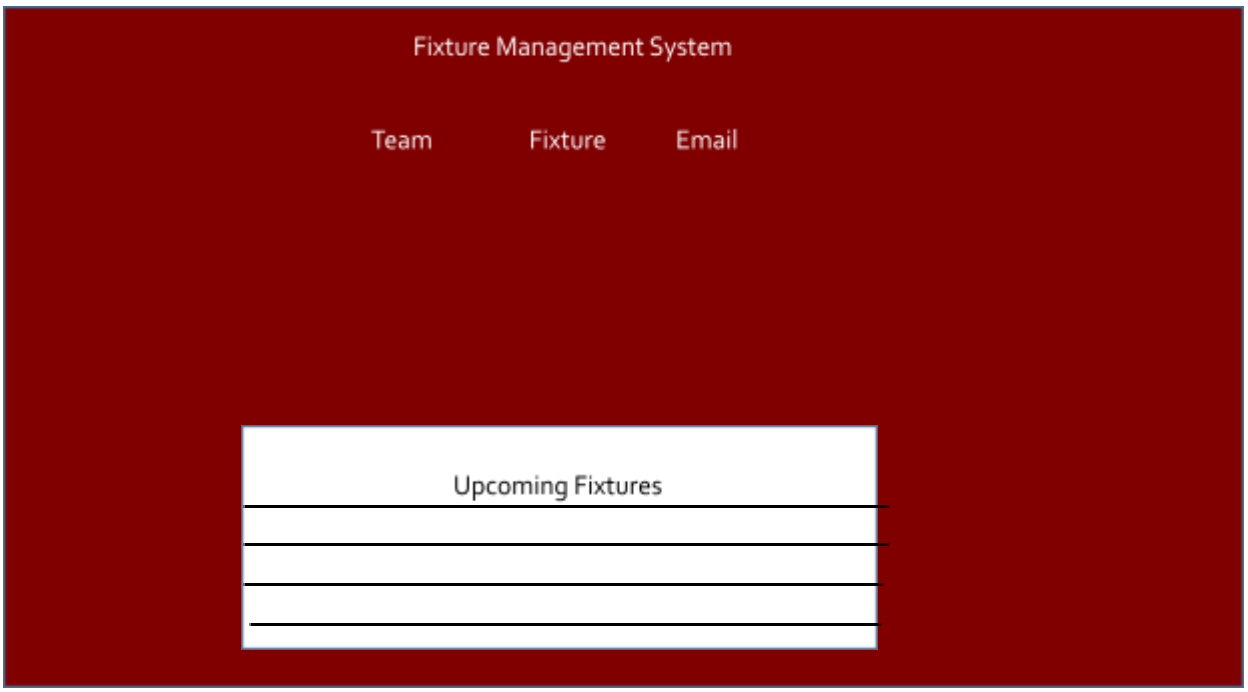
- The ability to view fixtures separately
  Since the fixtures would be shown when editing or deleting a fixture there was no need for a separate page.

Considering all of this, my concept of the solution has been updated to the designs on the next page

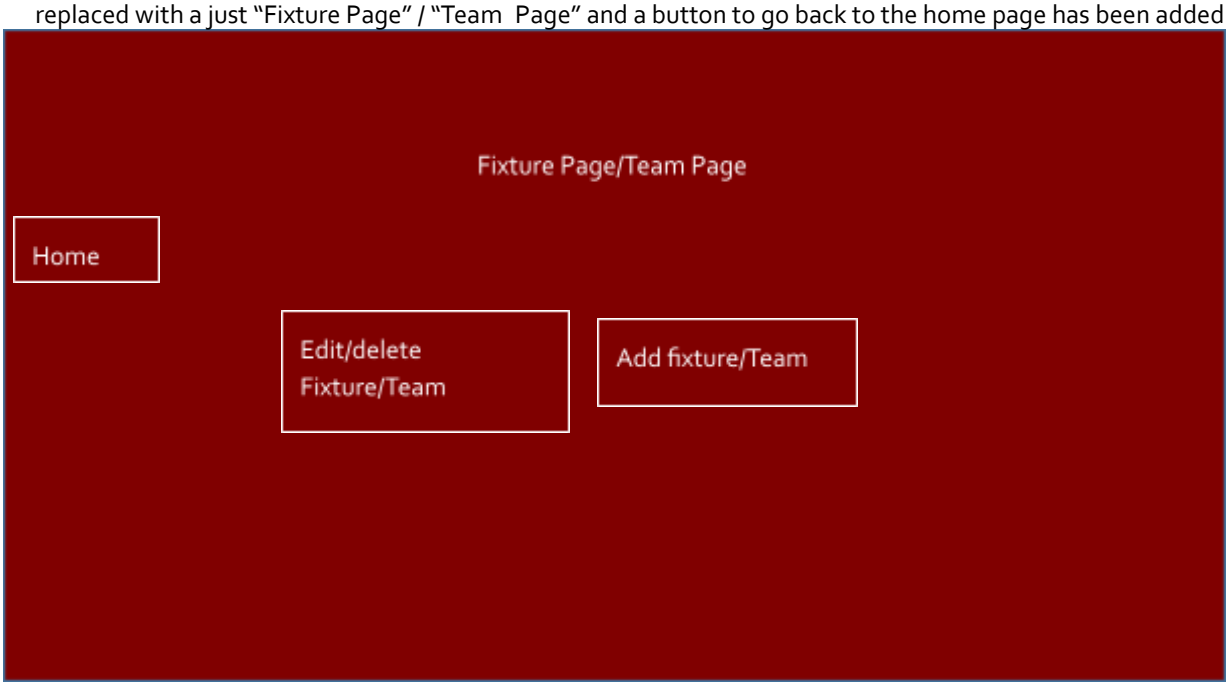
Login Page: Has been kept the same



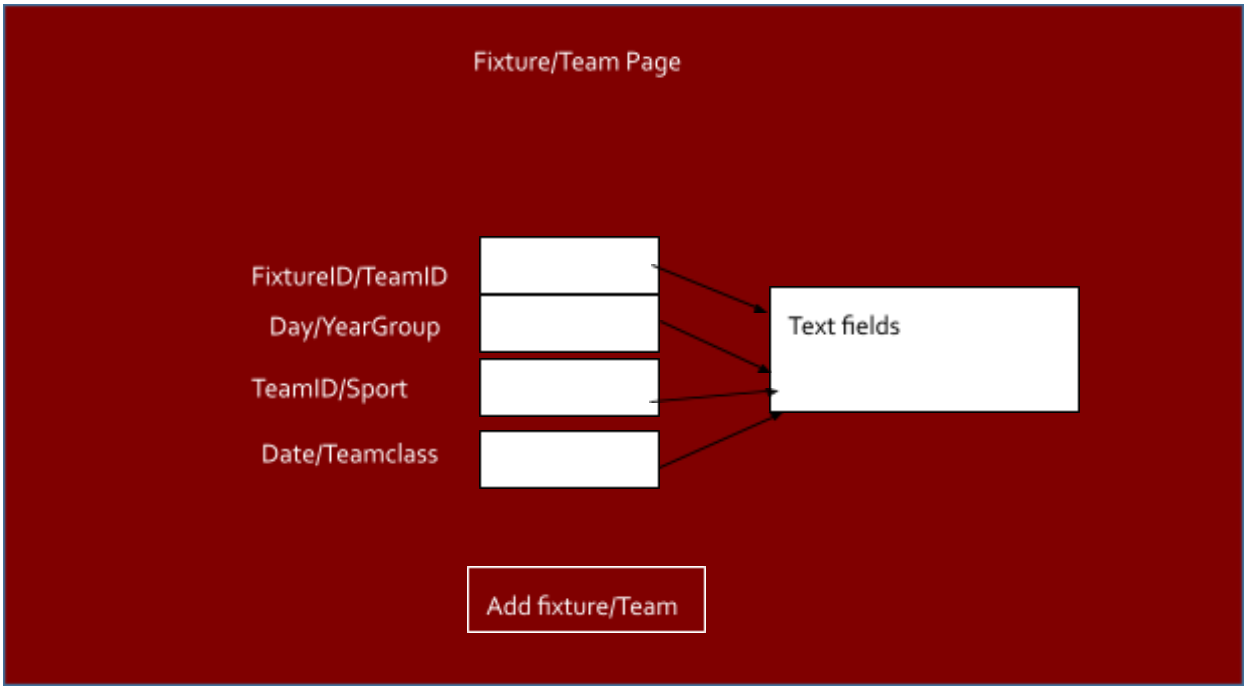
Home page: Home,Teacher and Student has been removed. Home Page was not necessary on the home page while the student and teacher buttons were removed due to the fact that the admin of the database would be the ones handling the data and not the user.



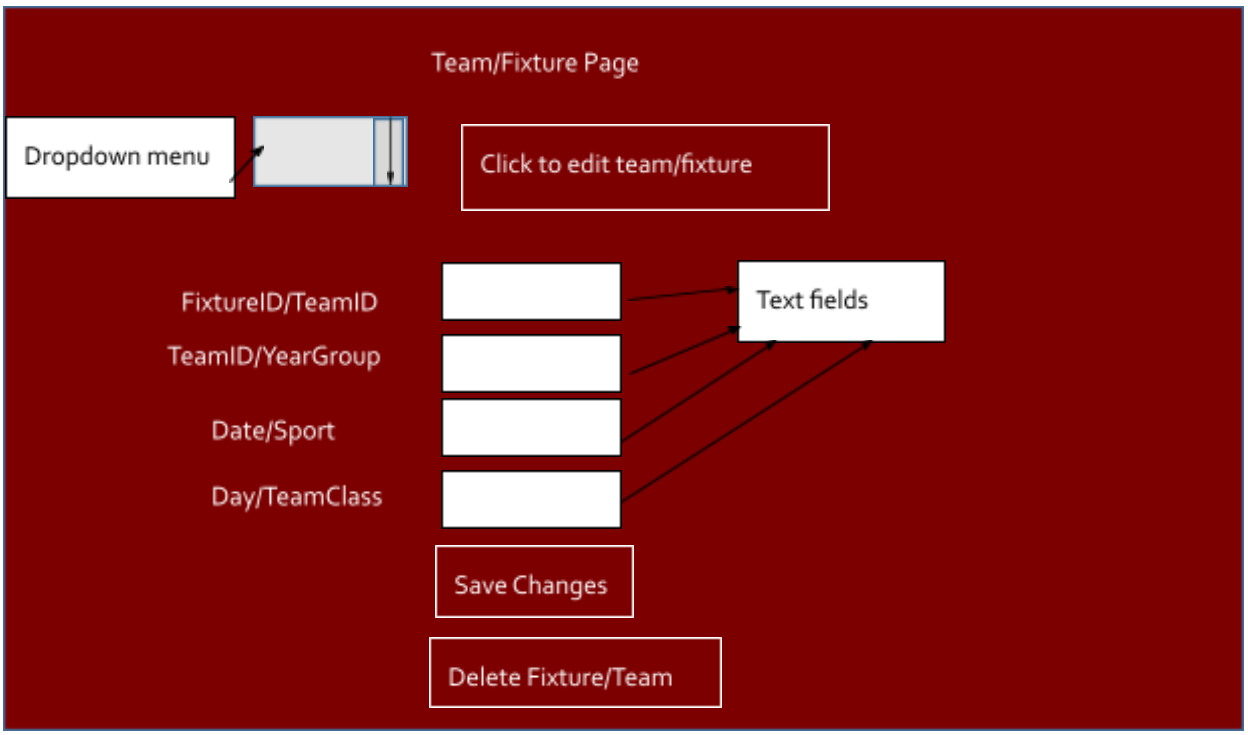
Fixture and Team Pages: Dropdown menu for the user to pick the function they wish to use has been scrapped and replacing it are two clickable buttons (signified by text with borders surrounding it) redirecting the user to appropriate pages, edit/delete a fixture/team or add a fixture/team. Fixture Management page title has been replaced with a just "Fixture Page" / "Team Page" and a button to go back to the home page has been added



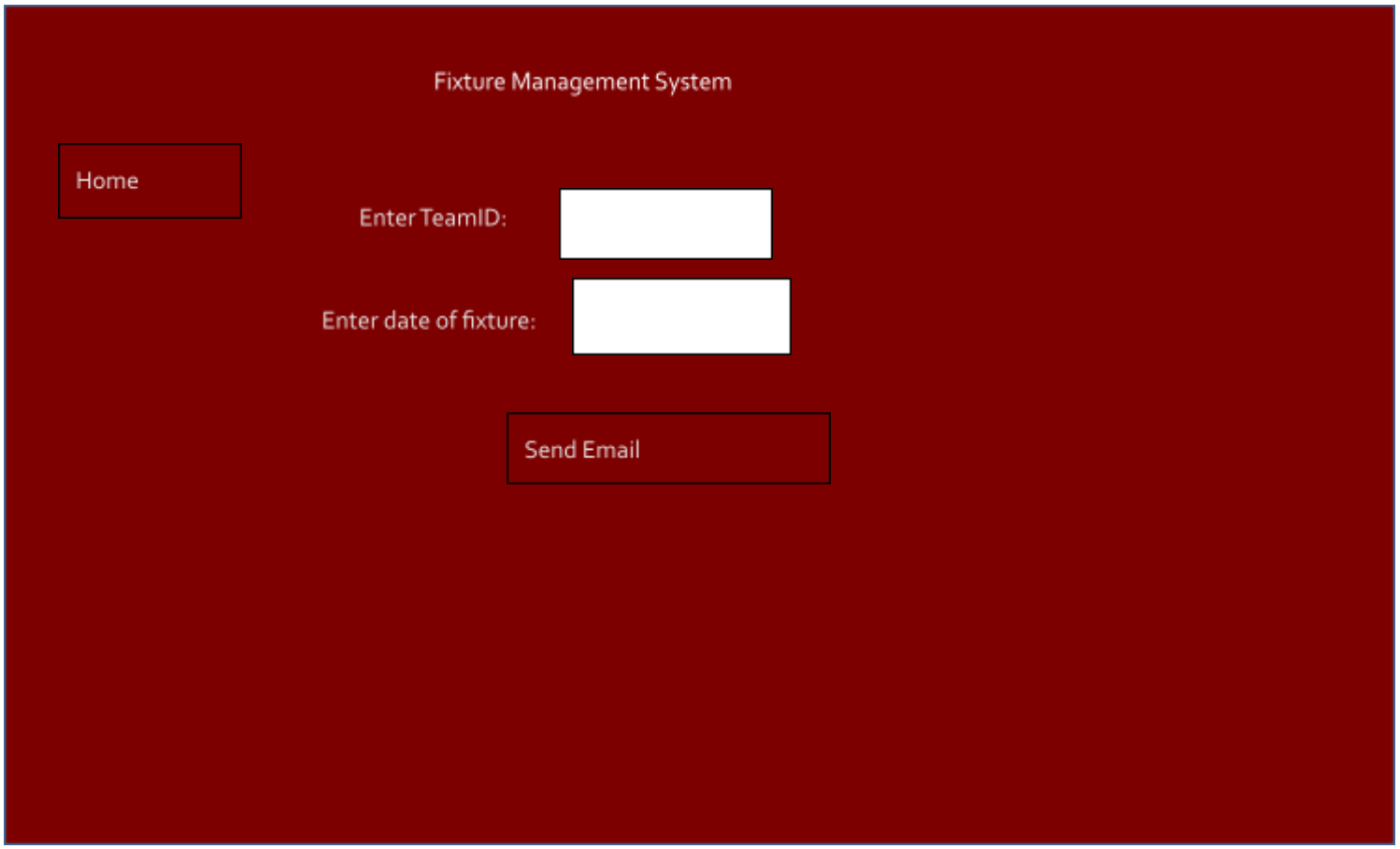
Fixture and Team Pages-Add Team/fixture: Contains various text fields that data can be entered with a button to use the information entered to create a fixture/team correctly into their tables. By clicking add fixture/team the user is redirected to a page that adds the information to the database



Fixture and Team Pages: Edit/delete a fixture/Team: Dropdown menu displaying all the TeamID's/ FixtureID's has been added. The user clicks on the button to edit team after selecting an ID. The fields are then filled with the data associated with the ID selected. After doing so by clicking save changes the user is redirected to a page that updates the database. By clicking delete team/fixture the user is redirected to a page that executes a query that deletes the record from the database



Email Page: Has also been kept the same



Inputs, Processes and Outputs

Login And Connectivity Page

The following inputs will be used for my login page:

Name		
Username	Varchar	1-10
Password	Varchar	1-20
Log-in	Button	null

The following variables will be held on the login page

Name		
User	Input box	Allows the user to input their username which can be looked at on the connectivity page
Password	Input box	Allows the user to input their password which can be also looked at on the connectivity page

The following process(es) will occur on the login page:

	Justification
Redirect to the connectivity page	This will allow the information entered by the user to be verified against the database on a separate page

The only input that will be on the connectivity page is the error message box the user would have to click if they have entered details incorrectly. Their input would then redirect them back to the login page.

The following variables will be held on the connectivity page

	Type	Justification
Username variable	Varchar	This variable will hold the username from the previous page in order to compare it to the information in the database. The username could contain symbols and numbers so it being a variable that holds characters is necessary
Password variable	Varchar	This variable will hold the password from the previous page in order to compare it to the information in the database. The password could contain symbols and numbers so it being a variable that holds characters is necessary
Database information variable	Array	This variable will hold the username and password information from the database to compare it to the username and password variable

The following process will occur on the connectivity page

	Justification
Obtain the user and password values from the login page and assign them to a variable	Allows the data to be compared against the information in the database
Compare the login information in the database with the data in the variables used to hold the username and password information	Ensures the login information entered is the same as the information in the database providing security for the solution
Redirect to homepage if the information is correct	Allows the user to access the solution and its functions
Error message displayed when incorrect details are entered	Prevents the access to the solution providing security against unauthorised access

Test data for the inputs on the login page processed with the connectivity page

Name	Inputs			Outputs		
	Valid	Invalid	Extreme	Valid	Invalid	Extreme
Username	John Teacherfn 12345678	" "/blank input Johndoe123	Johndoe12(9 character input)	With valid password redirect to home page	Error message "incorrect login details"	Null
Password	20 character input	" "/blank input 21+ character input	19-characte r input	With valid username redirect to home page	Error message "incorrect login details"	null

Home And Logout Page

The following inputs will be on the homepage

Name		
Team menu item	Formatted link to act as a button	0-1 (clicked or not clicked)
Fixture menu item	Formatted link to act as a button	0-1 (clicked or not clicked)
Email menu item	Formatted link to act as a button	0-1 (clicked or not clicked)
Logout button	Button	0-1 (clicked or not clicked)

The following variables will be held on the homepage

Name		
Username variable	Varchar	Will be used to display the current user signed in
Fixture information variable	Array	Will hold the information of the upcoming fixtures that will be displayed on the page
Table storing fixture information	Table/2d array	Will display the information obtained from the fixture information variable in a organised manner.

The following process will occur on the page

	Justification
Obtaining of the fixture information	This is necessary in order to display the upcoming fixtures for the user
Redirect to the team page when the "button" is clicked	This allows the user to choose whether to use the add team, edit or delete team function of the solution
Redirect to the fixture page when the "button" is clicked	This allows the user to choose whether to use the add fixture, edit or delete function of the solution
Redirect to the email page when the "button" is clicked	Allows the user to enter the information necessary to send an email.
Redirect to the logout page when the logout button is clicked	Allows the user to logout of the solution

Test data for the inputs on the home page

Name	Inputs			Outputs		
	Valid	Invalid	Extreme	Valid	Invalid	Extreme
Team page button	Button clicked	No click	Null	On click redirect to Team page	No action performed	Null
Fixture page button	Button clicked	No click	Null	On click redirect to fixture page	No action performed	Null
Email Page button	Button clicked	No click	Null	On click redirect to the email page	No action performed	Null
Logout button	Button clicked	No click	Null	On click redirect to the logout page	No action performed	Null

Since the logout function will only be on my homepage, I have decided to show the inputs, outputs and processes of the logout page here

Inputs on the logout page

Name		
Confirm box displaying message	Button	0-1(Clicked or not clicked)

Variables on the logout page:

Name		
Username variable	Varchar	Will be used to identify the current user

Processes on the logout page

Process	
Obtainment of the username variable	Will be used to identify the user being logged out
Deletion of the session	Essentially logs out the user by removing any stored variables relating to the user identified from the username variable
Redirect to the login page	This will bring the user back to the beginning of the solution allowing them to log back in if necessary

Test data for the inputs on the logout page

Name	Inputs			Outputs		
	Valid	Invalid	Extreme	Valid	Invalid	Extreme
Confirm button	Button clicked	No click	Null	On click redirect to login page	No action performed	Null

Team Page:

Inputs that will be used on the page:

Name		
Add team "button"	Button/clickable link	0-1(Clicked or not clicked)
Edit/delete Team	Button/clickable link	0-1(Clicked or not clicked)
Home button	Button	0-1(Clicked or not clicked)

The variables that will be used on the page:

Name		
Username variable	Varchar	Will be used to identify the current user to display

Processes that will happen on the page:

Process	
Redirect to the Add Team page on button click	Allows the user to access the page they can use to update the database by adding a new record
Redirect to the Edit/delete Team page on button click	Allows the user to access the page they can use to update or delete records
Redirect to the homepage on home button click	This will bring the user back to the homepage if they do not wish to use the team page functions.

Test data for the Team page

Name	Inputs			Outputs		
	Valid	Invalid	Extreme	Valid	Invalid	Extreme
Add Team page button	Button clicked	No click	Null	On click redirect to Add Team page	No action performed	Null
Edit/delete team button	Button clicked	No click	Null	On click redirect to edit page	No action performed	Null
Home button	Button clicked	No click	Null	On click redirect to the home page	No action performed	Null

Team Page: Add Team

The inputs that will be on the page:

Name		
TeamID	Combination of characters	1-4
YearGroup	Int	1-2
TeamClass	String/text	1
Sport	String/text	1-11
Add team	Button	0-1(clicked or not clicked)
Home	Button	0-1 (clicked or not clicked)

The variables that will used on this page:

Name		
Form variable	Form	Will hold all the inputs to be obtained by the add page to insert the data into the teams table as a new record

The processes that will be used on this page

Process	
Redirect to the Add page on button click	Allows the user to access the page they can use to update the database by adding a new record
Redirect to the Edit/delete Team page on button click	Allows the user to access the page they can use to update or delete records
Redirect to the homepage on home button click	This will bring the user back to the homepage if they do not wish to use the team page functions.

Algorithms

Validation

The validation I shall be using for my solution will be two rules:

No empty fields when adding a team/fixture or editing a team/fixture:

This prevents the database being filled with irrelevant data increasing the size and making the management of the database to be difficult

Login details being correct:

This prevents somebody the user does not want accessing the solution being able to use it providing basic security for the user.

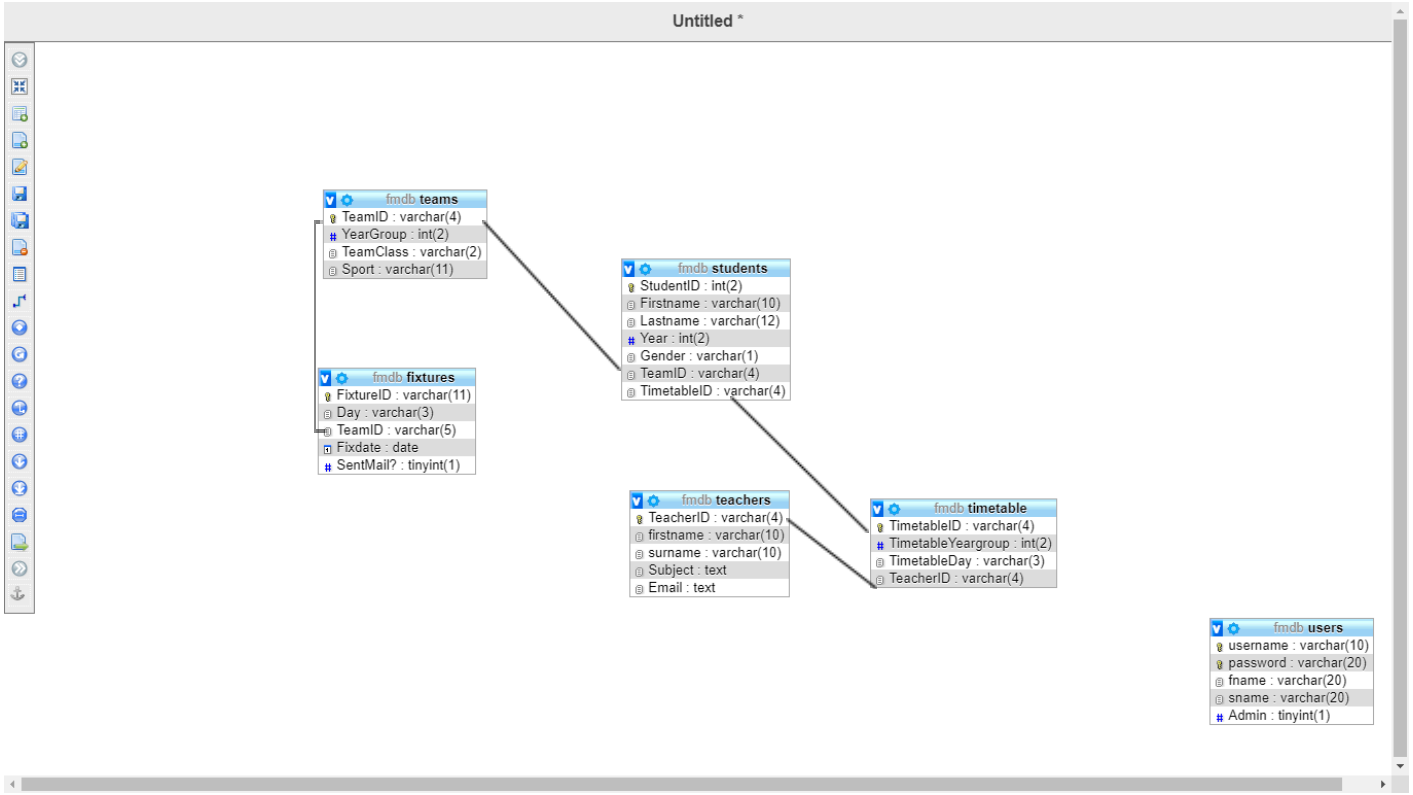
Development Method

The software development life cycle model I shall adopt is the waterfall model. Where with the requirements given to me by the stakeholder, I shall produce a solution which the stakeholder would receive once completed. I shall adopt the agile approach to the development as well as it is often the case that a stakeholder's requirements may be impossible to meet in a short time frame. By giving the stakeholder a

prototype of the solution, they are able to examine what about the solution they are happy with and what they are not. This makes it easier to make changes to the solution quicker and easier for the developer as they can focus on certain aspects of the solution and not the solution as a whole.

C.Developing the coded solution

The Database



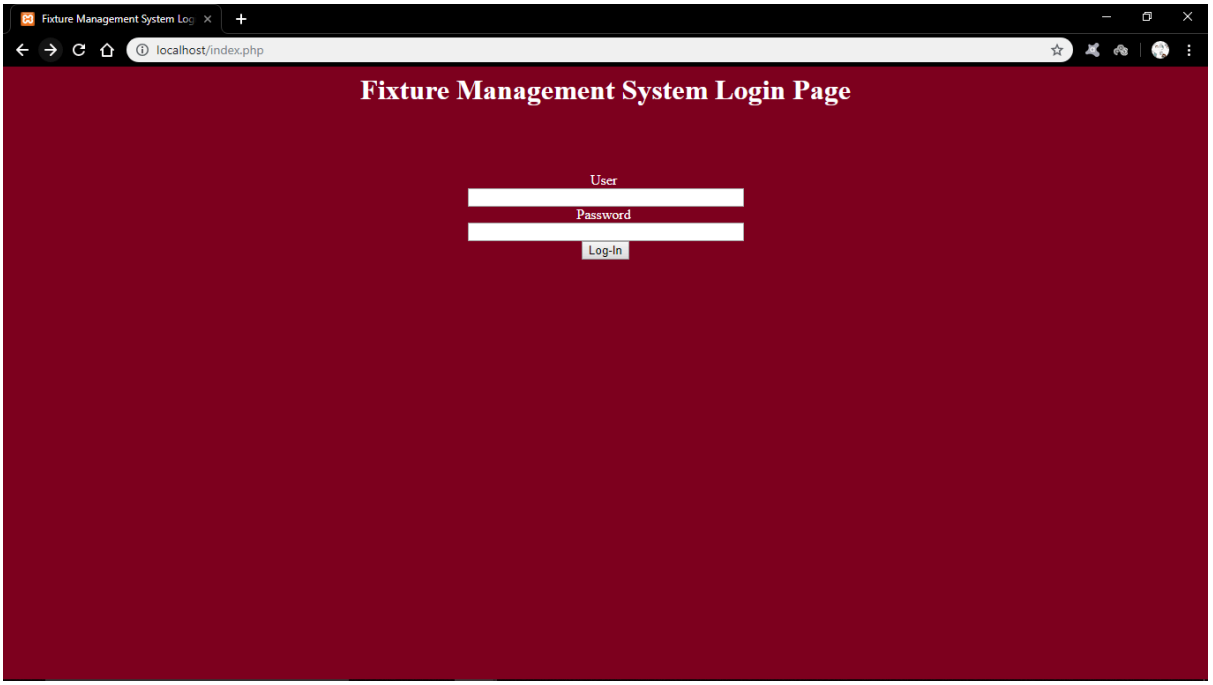
Using phpMyAdmin I was able to construct a database that reflected my design and was able to be accessed by my solution. With phpMyAdmin I was able to write sql queries to create the necessary tables and easily view/edit them. Above is the final structure of my database. In order to test the functions of my solution I populated the database with some info such as the student and teacher information as well as the username and password information.

Login And Connectivity Page(s)

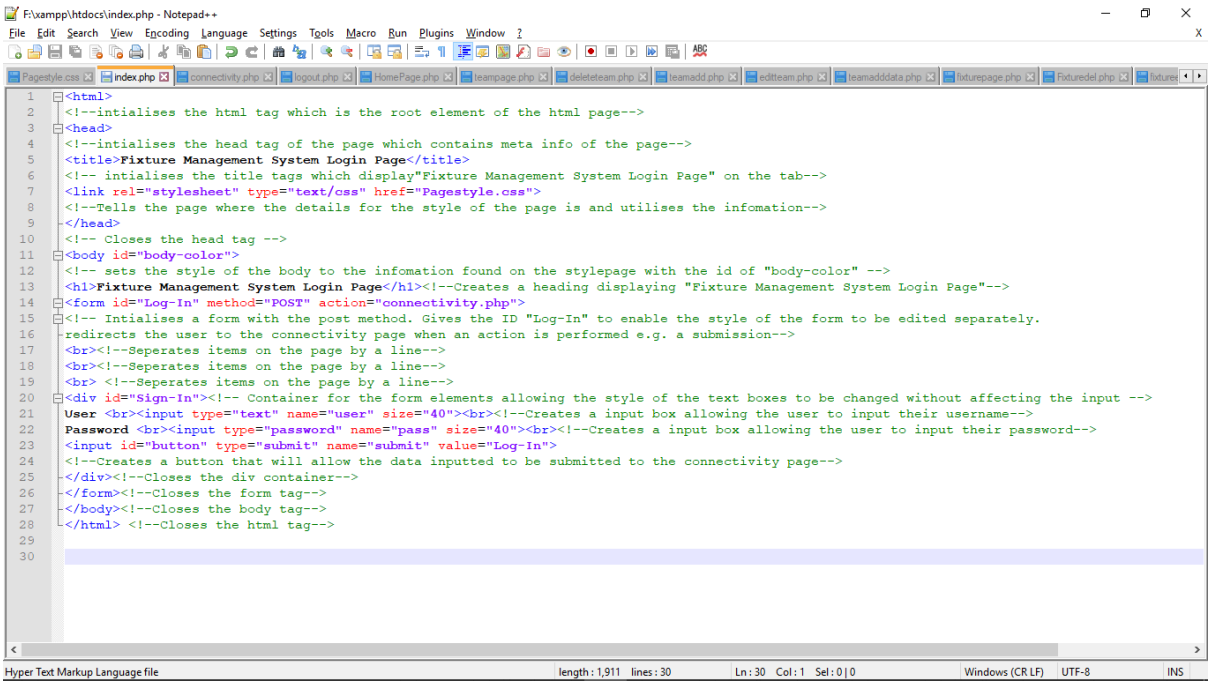
This is the first page the User will see when using the solution and since the page's use is just enter login details, a simple layout would be appropriate.

Version 1:

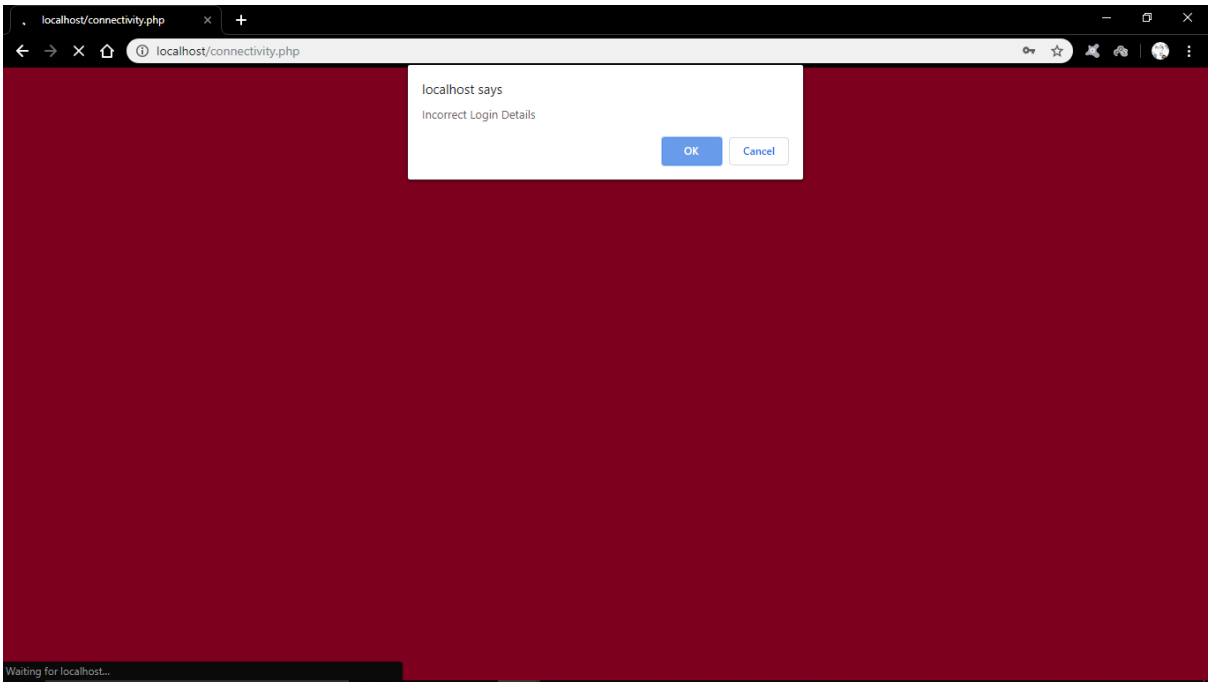
Version 2: Moved login area to center of the page and removed border around the login fields



Code for the Login page:



The connectivity page that the login page redirects to has no interface apart from when incorrect login details are entered which can be seen here:



The code for the connectivity page can be seen here:

```php
<html><!--intialises the html tag which is the root element of the html page-->
<head><!--intialises the head tag of the page which contains meta info of the page-->
<?php //intialises the php tag which allows php language to be written
 $con=mysqli_connect('localhost','root') or die("Failed to connect to MySQL: " . mysqli_error());
 //connects the user to the database address "localhost" using the username "root"
 $db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error($con)); //selects the database fmdb
 session_start(); //starts the session for user profile page
 if(!empty($_POST['user'])){ //checks if the $_POST variable contains infomation sent from the form on the index page
 $Username = $_POST['user']; $Password = $_POST['pass'];
 //assigns the username from the form on the index page to $Username and assigns the password to the $Password variable
 $sql = "SELECT * FROM users where username = '".$Username."' AND password = '".$Password."'";
 //query that selects from the users table the records where the username = $Username and the password = the $Password variable
 $query = mysqli_query($con,$sql); //executes the sql query in the variable $sql
 $row = mysqli_fetch_array($query);//sorts the infomation obtained from the sql query execution into an array variable named $row


 if(!empty($row[0]) AND !empty($row[1])){ //checks that there is infomation on the first and second index of the $row variable
   $_SESSION['user'] = $row['username'];// sets the session user name to the username obtained in $row
   header("Location: HomePage.php"); //redirects the user to the page "Homepage.php"
 }//ends the second if statement
 else{ //exeecutes the next block of code if the previous if statement conditions are not fulfilled
      echo " <script>window.confirm('Incorrect Login Details'); window.location.href='index.php'; </script>";
      //displays a confirm box to the user with the infomation "Incorrect login details" and redirects to the index page
 }//ends the else statement
 }// ends the first if statement
?> <!-- closes the php tag -->

</head><!-- closes the head tag -->
</html><!-- closes the html tag -->
```

Evaluation of the pages:

The login page:

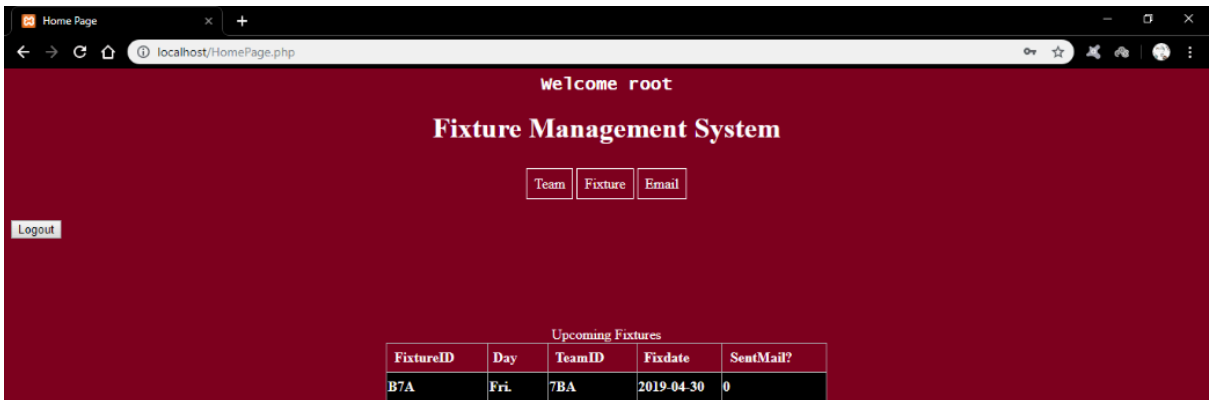
Functions as it should sending the login info that was entered over to the user and is simple enough that the user can understand.

The connectivity page:

The connectivity page function as it should obtaining the information entered by the user assigning it to a variable and comparing it with the data obtained from the database.

Home Page

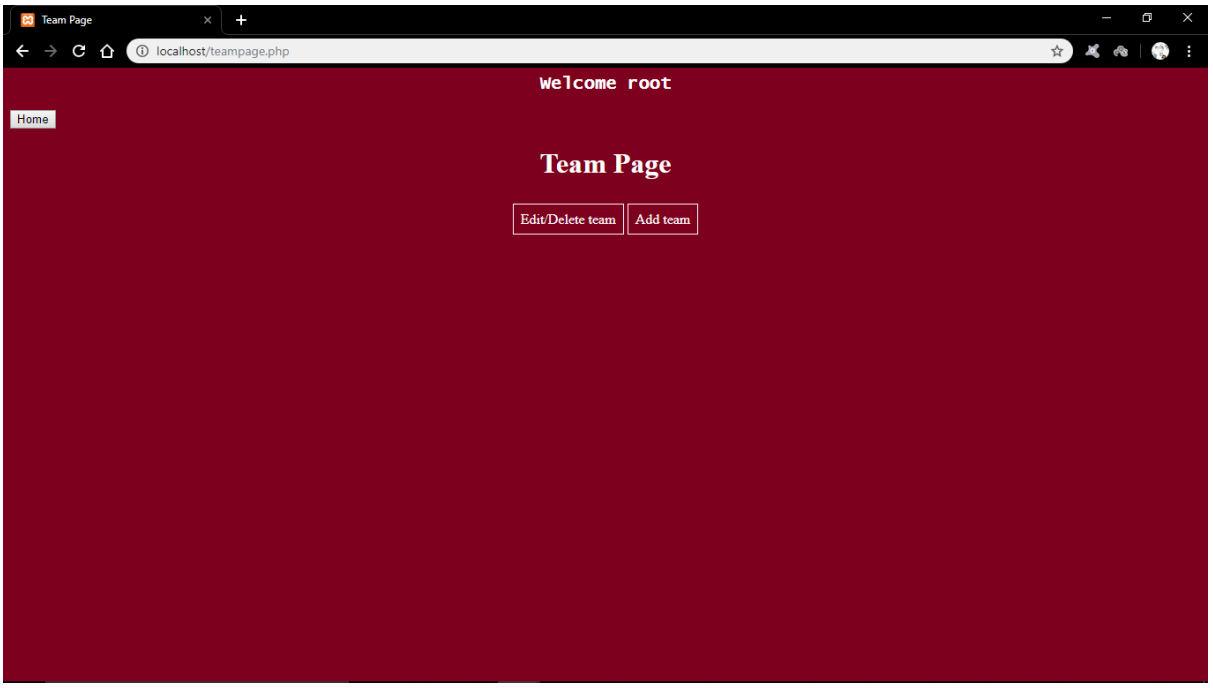
The homepage I programmed fully reflects my design ideas although I added the highlighting of the fixture table when hovering over it to allow better reading of the table as shown here:



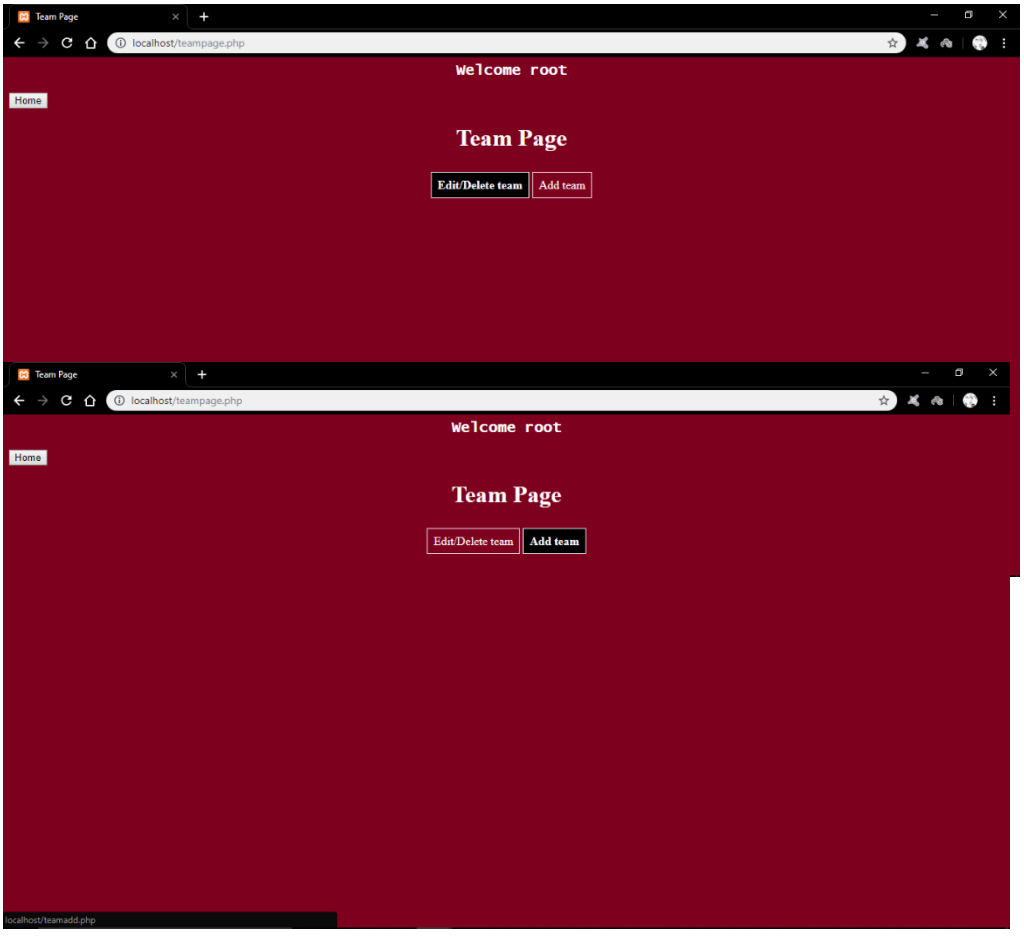
Code for the Home Page:

```php
<html><!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
session_start(); //starts the session with its variables
if(!isset($_SESSION['user'])){ //checks to see if the $_SESSION variable has any infomation in the user index
    header("Location: index.php");} //redirects the user to the index page if the index is empty
echo "<p>Welcome ".$_SESSION['user']."</p>";//displays the message "welcome" with username of the user logged in e.g. admin


$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error()); //selects the database "fmdb"
$currentdate=date('Y/m/d'); //obtains and assigns the current date in the format yyyy/mm/dd to the variable $currentdate
$obtainfixtures= "SELECT * FROM fixtures WHERE Fixdate > '".$currentdate."'";
//query that selects all the fixtures that have a date greater than the current date
$query = mysqli_query($con,$obtainfixtures) or die(mysqli_error()); //executes the query $obtainfixtures
$results = mysqli_fetch_assoc($query) or die(mysqli_error()); //obtains the infomation from $query and puts it in an associative array

$result = $con->query($obtainfixtures);//executes the query $obtainfixtures
$datas = array();//intialises $datas as a 2d array
while($data = $result->fetch_assoc()){
//intialises $data with the infomation provided from the sql query $result and loops for all the infomation obtained
   $datas[] = $data; //adds the information from $data into the $datas 2d array
} //ends the while statement

?> <!--closes the php tag-->
<head><!--intialises the head element of the page where meta info about the page is held-->
 <title>Home Page</title><!--Gives the title "Home Page" to the page that is seen on the top of a tab/window-->
 <link rel="stylesheet" type="text/css" href="Pagestyle.css"><!--sets the style of the page to the infomation found on "Pagestyle.css"-->
 </head> <!--closes the head tag -->
<body id = "body-color"> <!--sets the style of the body to the id "body-color"-->
 <h1>Fixture Management System</h1> <!--creates a heading displaying "Fixture Management System on the page-->

<ul> <!-- intialises an unordered list-->
  <li><a href="teampage.php">Team</a></li>
  <!--Adds a list item to the unordred list, a link to "teampage.php" that is labeled "team"-->
  <li><a href="fixturepage.php">Fixture</a></li>
  <!-- Adds a list item to the unordered list, a link to "fixturepage.php" that is labeled "Fixture"-->
  <li><a href="Emailpage.php">Email</a></li>
  <!-- Adds a list item to the unordered list, a link to "Emailpage.php" that is labeled "Email" -->
 </ul><!--closes the unordered list-->
  <br><!-- adds a empty line to seperate items on the page-->
 <button type="button" onclick="window.location.href='logout.php'">Logout</button>
 <!-- creates a button that redirects the user to the logout page labeled logout-->
<div style = "overflow-y:auto;">
 <!--creates a container for the table with a style that allows the table to be scrollable when it becomes too big for the page-->
<table class="center"> <!-- initialises a table and sets the table to be affected by the style infomation of the class "center"-->
 <caption>Upcoming Fixtures</caption> <!-- adds the title "Upcoming Fixtures" to the table -->
 <tr><!--defines a row in the table-->
   <th>FixtureID</th><!-- creates and defines a column to be named "FixtureID"-->
   <th>Day</th><!--creates and defines a second column named "Day"-->
   <th>TeamID</th><!-- creates and defines the third column to be named "TeamID"-->
   <th>Fixdate</th><!-- creates and defines a fourth column named "Fixdate"-->
   <th>SentMail?</th><!--creates and defines a fifth column to be named "SentMail?"-->
  </tr><!--signifies the end of the table-->

  <?php  //initialises the php tag allowing php language to be written
  foreach ($datas as $i){
  //loop that executes the next block of code that lasts the length of time it takes for the amount of infomation obtained
  // to be assigned to the variable $i
  echo "<tr>";// adds a row to the table
  echo "<td>".$i['FixtureID']."</td><td>".$i['Day']."."."</td><td>". $i['TeamID']."</td><td>".$i['Fixdate']."</td><td>".$i['SentMail?']."</td>
  //adds the FixtureID infomation obtained to the first column,"Day" infomation to the second,"TeamID" infomation to the third,
  //"Fixdate" infomation to the fourth and "SentMail?" infomation to the table
  echo "</tr>";//add a new row to the table
  }// ends the foreach statement

  ?> <!-- closes the php tag-->


 </table> <!--closes the table tag-->
 </div><!-- closes the container-->
</body><!-- closes to body tag-->
</html><!--closes the html tag-->
```

Team Page – Select Function Page:



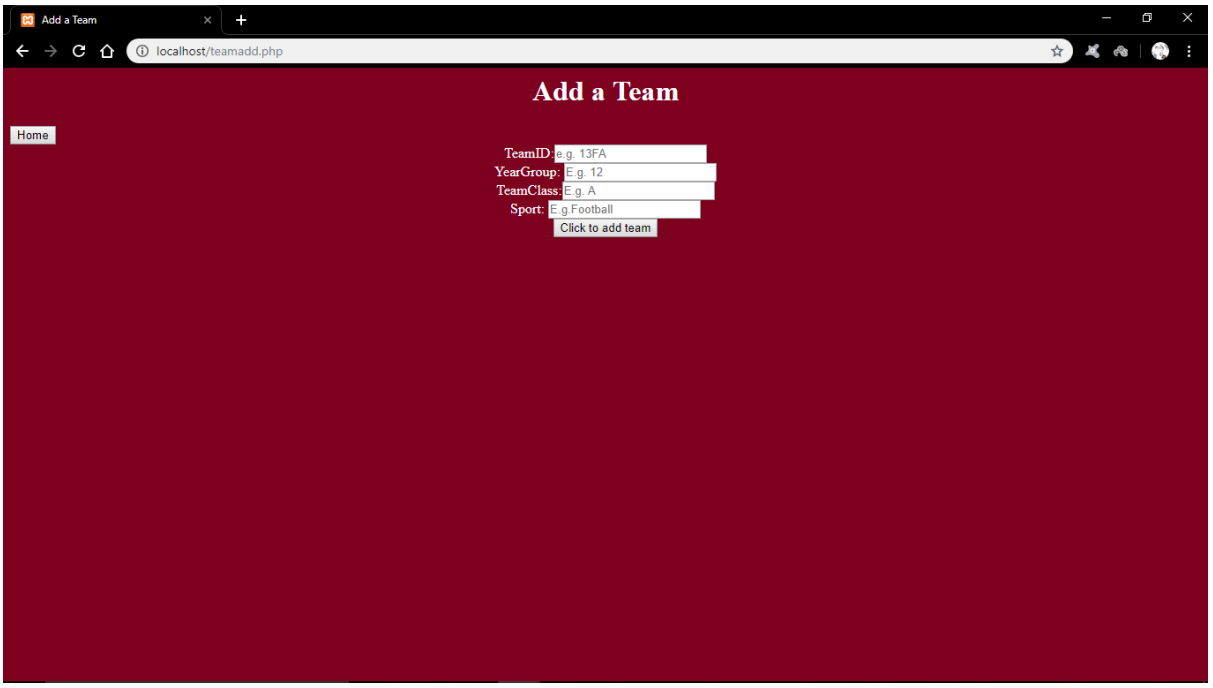
The highlighting of the text box when hovered over feature was also added to the navigation options on this page aswell which can be seen here:



Code for the team page:

```php
<html><!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
session_start(); //starts the session with its variables
echo "<p> Welcome ".$_SESSION['user']."</p>";//displays the message "welcome" with username of the user logged in e.g. admin

?>
<head><!--intialises the head tag-->

  <link rel="stylesheet" type="text/css" href="Pagestyle.css"><!--sets the style of the page to the infomation found on "Pagestyle.css"-->

</head><!-- closes the head tag-->
<button type="button" onclick="window.location.href='Homepage.php'">Home</button>
<!-- creates a button that redirects the user to "homepage.php" labeled Home-->
<body id = "body-color"><!--sets the style of the body to the id "body-color"-->
<title>Team Page</title><!--Gives the title "Team Page" to the page that is seen on the top of a tab/window-->
<h1>Team Page</h1><!-- Creates a heading that displays "Team Page" on the page-->
<ul> <!-- intialises an unordered list-->
  <li><a href="teampagedelete.php">Edit/Delete team</a></li>
  <!-- adds a list item to the unordered list, a link to "teampagedelete.php" labeled "Edit/Delete team"-->
  <li><a href="teamadd.php">Add team</a></li><!--adds a list item to the unordered list, a link to "teamadd.php" labeled Add team-->
 </ul><!-- closes the unordered list-->
</body><!--closes the body tag-->
```

Team Page: Add Team



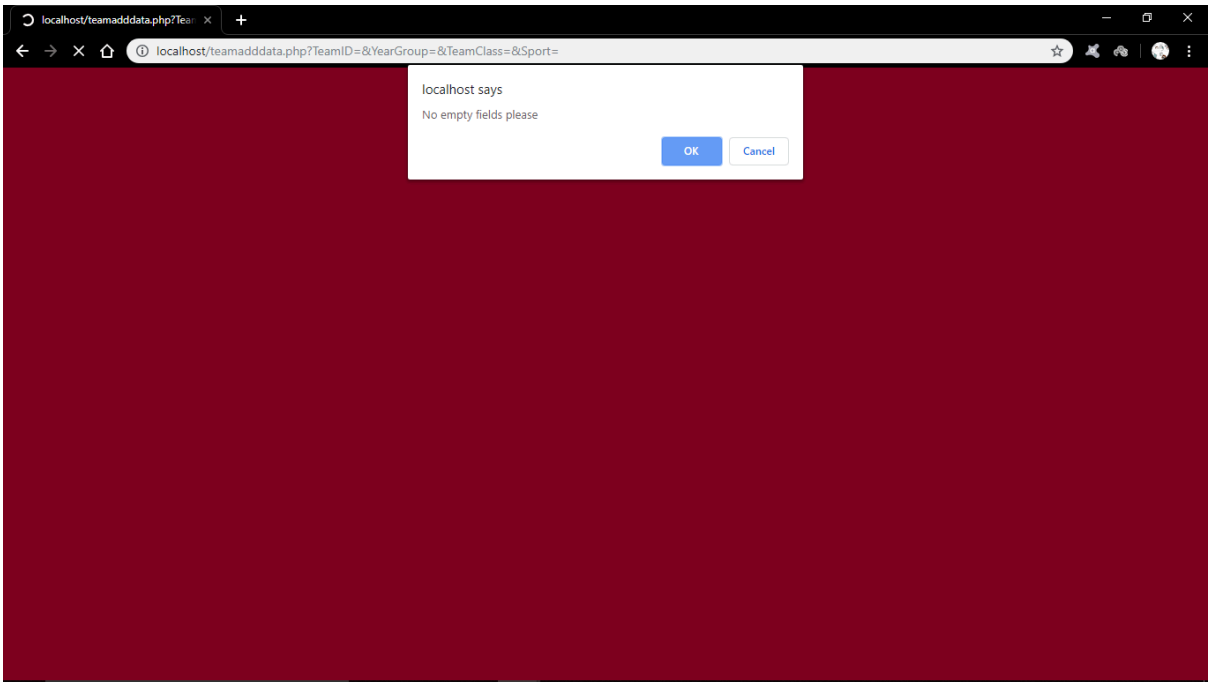
The design for the add team page of my solution is fully reflected on this page and the simple user interface criteria I identified has been followed here.

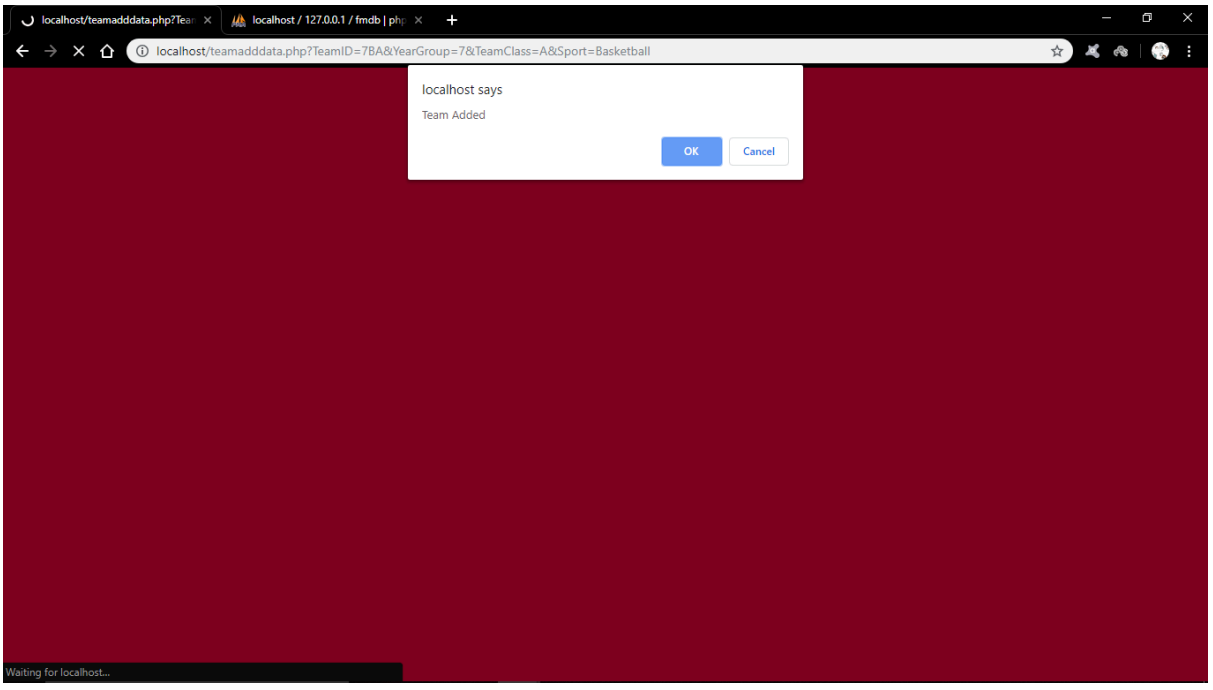
The code for Team Page: Add Team

```
<html><!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error()); //selects the database "fmdb"
?><!--closes the php tag-->
<head><!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
<title>Add a Team</title><!--Gives the title "Add a team" to the page that is seen on the top of a tab/window-->
<body id="body-color"><!-- sets the style of the body to the attribute "body-color" on the style page-->
<h1>Add a Team</h1><!-- Creates a heading that displays "Add a team" on the page-->
</head><!-- closes the head tag-->
<button type="button" onclick="window.location.href='Homepage.php'">Home</button>
<!-- creates a button that redirects the user to "homepage.php" labeled Home-->
<form method = "GET" action ="teamadddata.php">
<!-- intialises a form with the get method. Any form action such as a submission will redirect the user to "teamadddata.php"-->
TeamID:<input name="TeamID" type="text"  id ="TeamID" placeholder="e.g. 13FA"><br>
<!--creates a input box with the label TeamID and adds a new line-->
YearGroup: <input name="YearGroup" type="text" id="YearGroup"  placeholder="E.g. 12"><br>
<!--creates a input box with the label YearGroup and adds a new line-->
TeamClass:<input name="TeamClass" type="text" id="TeamClass" placeholder="E.g. A"><br>
<!--creates a input box with the label TeamClass and adds a new line-->
Sport: <input name="Sport" type="text" id="Sport" placeholder="E.g.Football"><br>
<!--creates a input box with the label Sport and adds a new line-->
<button type="submit" value="Submit">Click to add team</button><br>
<!--creates a button that submits the form with the label "Click to add team"-->
</form> <!--closes the form tag-->
</body><!--closes the body tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 2,225 lines : 29 | Ln : 29 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

The associative page that the Team Page: Add team page redirects to that processes the data and updates the database with the information only has 2 instances of a UI and that is when a team is successfully inputted into a database or not due to empty fields. This can be seen here:



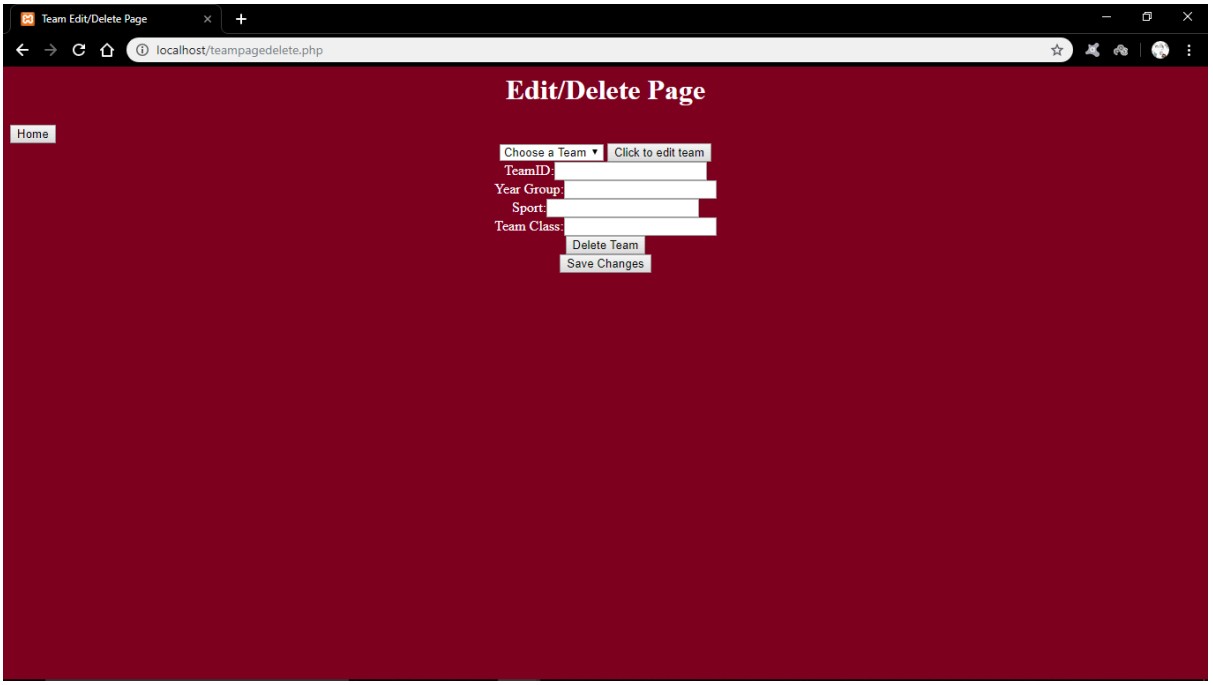


The code for this page can be seen here:

```php
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error()); //selects the database "fmdb"
if(isset($_GET['TeamID'])){ //executes the next block of code if the $_GET variable contains the index TeamID
    $form_TeamID=$_GET['TeamID']; //assigns the $form_TeamID the infomation from the TeamID index of the $_GET variable
    $form_YearGroup=$_GET['YearGroup']; //$form_YearGroup is assigned the data from the YearGroup index of the $_GET variable
    $form_Sport=$_GET['Sport'];//$form_sport is assigned the data from the sport index of the $_GET variable
    $form_TeamClass=$_GET['TeamClass'];//$form_Teamclass is assigned the data from the teamclass index of the $_GET variable

    if(!empty($form_TeamID) AND !empty($form_YearGroup) AND !empty($form_TeamClass) AND !empty($form_Sport)){
    //validation that checks no field from the field is empty and executes the next block of code if its true
        $addteam="INSERT INTO teams(TeamID,YearGroup,TeamClass,Sport) VALUES
        ('".$form_TeamID."','".$form_YearGroup."','".$form_TeamClass."','".$form_Sport."')";
        //creates a new record with the variables $form_TeamID,$form_YearGroup,$form_TeamClass and $form_Sport
        //into the appropriate columns of the table in the database
        $action = mysqli_query($con,$addteam);//executes the query infomation from $addteam
        echo "<script> window.confirm('Team Added'); window.location.href='teampage.php'; </script>";
        //displays the message "Team added" and redirects the user to "teampage.php"
        } //ends the second if statement
        else{ //executes the next block of code if the if statement conditions are not fufilled
            echo "<script> window.confirm('No empty fields please'); window.location.href='teampage.php'; </script>";
            //displays a confirm box with the message "No empty fields please" and redirects the user to "teampage.php"
        } //ends the else statement
        }// ends the first if statement
        ?> <!-- closes the php tag-->
<head> <!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css">
<!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head><!--closes the head tag-->
<body id ="body-color"> <!-- sets the style of the body to the attribute "body-color" on the style page-->
</body> <!--closes the body tag-->
</html> <!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 2,687 lines : 34 | Ln : 23 Col : 9 Sel : 0 | 0 | Windows (CR LF) | UTF-8

Team Page: Edit/Delete Page



Again, the design of this page has been closely followed and created to match

The dropdown menu contents containing the TeamID values will be shown in my testing

Code for Team Page : Edit/delete Team

```php
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());
//selects the database "fmdb"
$Teamselection= "SELECT * FROM `teams`"; // query that selects all the teams in the database
$queryrun = mysqli_query($con,$Teamselection) or die(mysqli_error()); // executes the query $Teamselection
$fetch = mysqli_fetch_array($queryrun) or die(mysqli_error());//puts the data from $queryrun into an array
$result = $con->query($Teamselection); //runs the query $Teamselection and adds the data to the $result variable
$datas = array(); //initialises $datas as an array
while($data = $result->fetch_assoc()){
//intialises $data with the infomation provided from the sql query $result and loops for all the infomation obtained
   $datas[] = $data;//adds the information from $data into the variable $datas, a 2d array
} //ends the while statement

if(isset($_GET['teams'])){ //executes the next block of code if the $_GET['teams'] variable contains data
    foreach($datas as $i){
    //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
    // to be assigned to the variable $i
        if ($i['TeamID']==$_GET['teams']){
        // executes the next block of code if the infomation in the TeamID index of $i is the same as the $_GET['teams'] variable
            $form_TeamID = $i['TeamID'];   //assigns the infomation from $i['TeamID'] to $form_TeamID
            $form_year = $i['YearGroup'];//assigns the infomation from $i['YearGroup'] to $form_year
            $form_Teamclass = $i['TeamClass'];//assigns the infomation from $i['TeamClass'] to $form_Teamclass
            $form_sport = $i['Sport'];//assigns the infomation from $i['Sport'] to $form_sport
        }//ends the second if statement
    }//ends the foreach statement
} //ends the first if statement

else{ //executes the next block of code if the conditions for the if and foreach statements arent fulfilled
    $form_TeamID = "";//intialises the $form_TeamID variable as blank
    $form_year = "";//intialises the $form_year variable as blank
    $form_Teamclass = "";//intialises the $form_Teamclass variable as blank
    $form_sport = "";//intialises the $form sport variable as blank
```

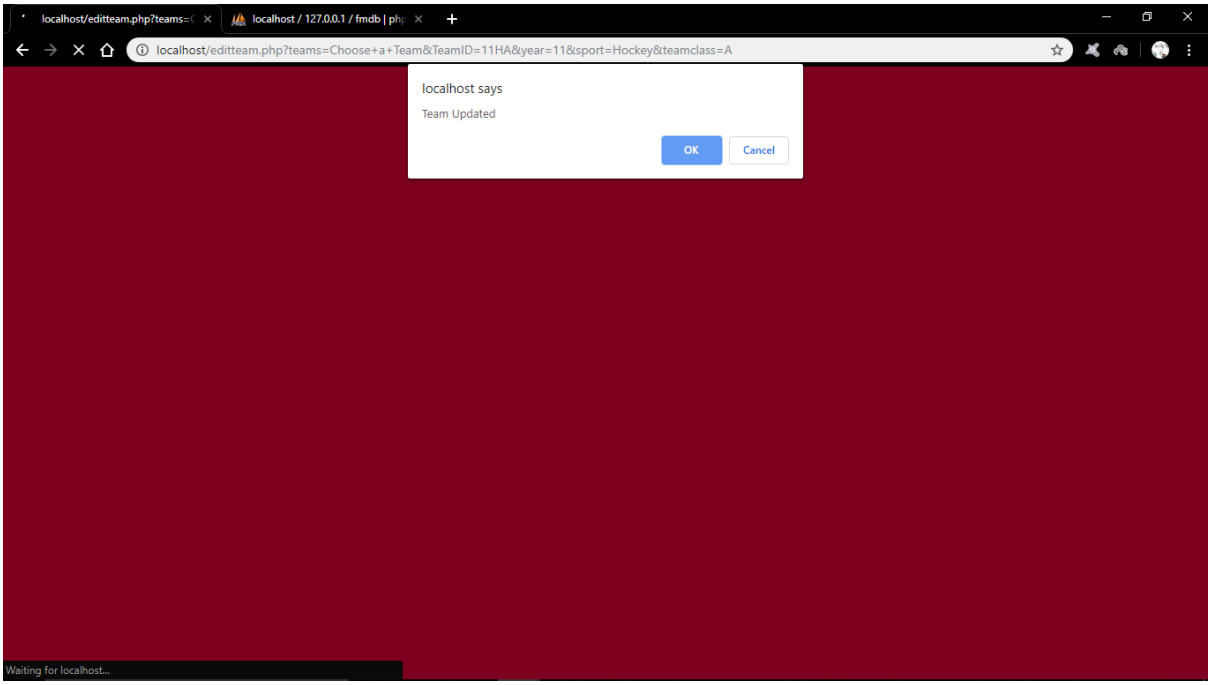
```
36
37
38  }//closes the else statement
39
40  ?><!-- closes the php tag-->
41  <head><!--intialises the head tag-->
42  <title>Team Edit/Delete Page</title> <!--Gives the title "Team Edit/Delete Page" to the page that is seen on the top of a tab/window-->
43  <link rel="stylesheet" type="text/css" href="Pagestyle.css"><!--sets the style of the page to the infomation found on "Pagestyle.css"-->
44  </head><!--closes the head tag-->
45  <h1>Edit/Delete Page</h1><!-- Creates a heading that displays "Edit/delete Page" on the page-->
46  <body id ="body-color"> <!-- sets the style of the body to the attribute "body-color" on the style page-->
47  <button type="button" onclick="window.location.href='Homepage.php'">Home</button>
48  <!-- creates a button that redirects the user to "homepage.php" labeled Home-->
49  <form method="GET"> <!-- intialises a form in the "GET" method-->
50  <select name="teams" id="teams"> <!-- creates a dropdown menu with an id of "teams"-->
51    <option selected="selected">Choose a Team</option> <!--makes the intial value "choose a team"-->
52    <?php // intialises the php tag allowing php language to be written
53      foreach($datas as $i){
54          //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
55          // to be assigned to the variable $i
56          echo "<option value=".$i['TeamID'].">".$i['TeamID']."</option>";
57          //adds all the TeamIDs that was obtained to the dropdown menu as dropdown values
58      }//ends the foreach statement
59      ?> <!-- closes the php tag-->
60  </select> <!--closes the select tag-->
61
62  <button type="submit" value="Submit">Click to edit team</button><br>
63  <!--creates a button that submits the data on the page to the same page in order to populate the input fields-->
64  TeamID:<input name="TeamID" type="text" value="<?php echo $form_TeamID;?>"><br>
65  <!--input box labeled "TeamID" that is filled with the TeamID selected from the dropdown menu when the edit button is clicked-->
66  Year Group:<input name="year" type="text" value="<?php echo $form_year;?>"><br>
67  <!--input box labeled "year" that is filled with the YearGroup relating to the TeamID when the edit button is clicked-->
```

The associative pages of the Team Page: Edit/delete team page, that updates the record with the infomation obtained from the form into the database or deletes the chosen record has only 3 instances of a UI. Two for the update of the record and one for the successful deletion of a record.

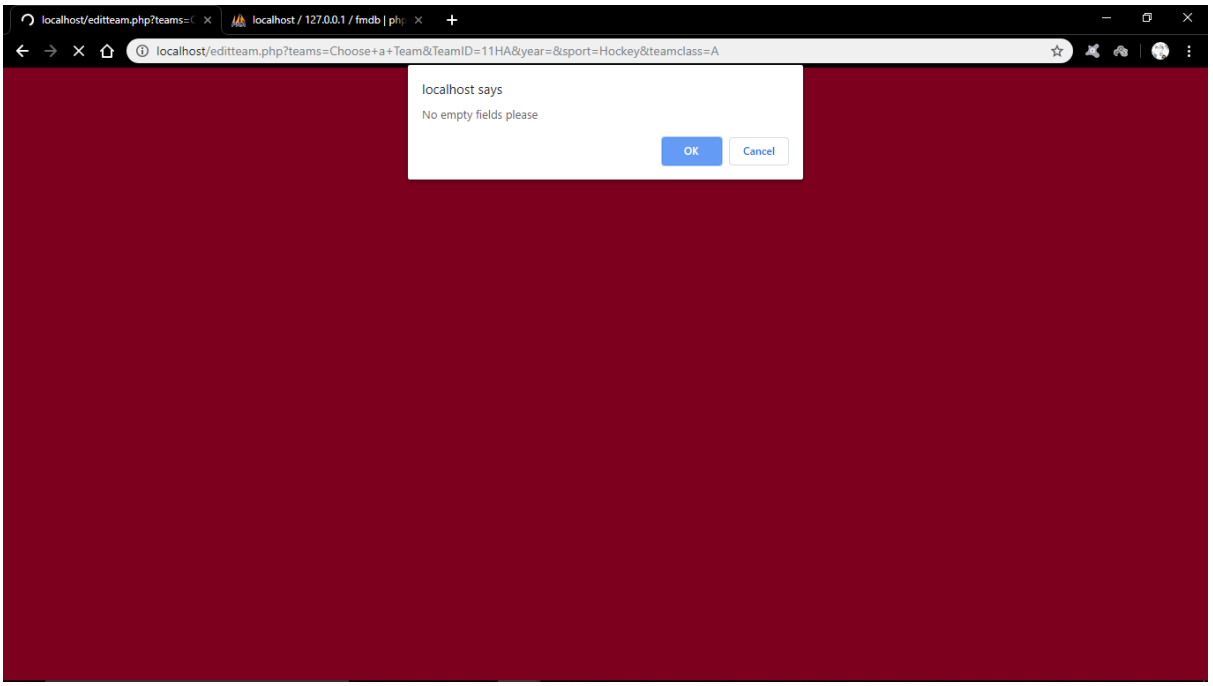
When a Team is successfully updated:

```
68  Sport:<input name="sport" type="text" value="<?php echo $form_sport;?>"><br>
69  <!--input box labeled "sport" that is filled with the Sport relating to the TeamID when the edit button is clicked-->
70  Team Class:<input name="teamclass" type="text" value="<?php echo $form_Teamclass;?>"><br>
71  <!--input box labeled "teamclass" that is filled with the TeamClass relating to the TeamID when the edit button is clicked-->
72  <button type="submit" formaction = "deleteteam.php">Delete Team</button><br>
73  <!--button that when clicked submits the form and redirects rhe user to the "deleteteam.php" page-->
74  <button type="submit" id="savechanges" formaction = "editteam.php" method="post">Save Changes</button><br>
75  <!--button that when clicked submits the form and redirects the user to the "editteam.php" page-->
76  </form> <!--closes the form tag-->
77  </body><!--closes the body tag-->
78  </html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 5,778 lines : 78 | Ln : 62 Col : 39 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS



When the team is not updated due to empty fields:

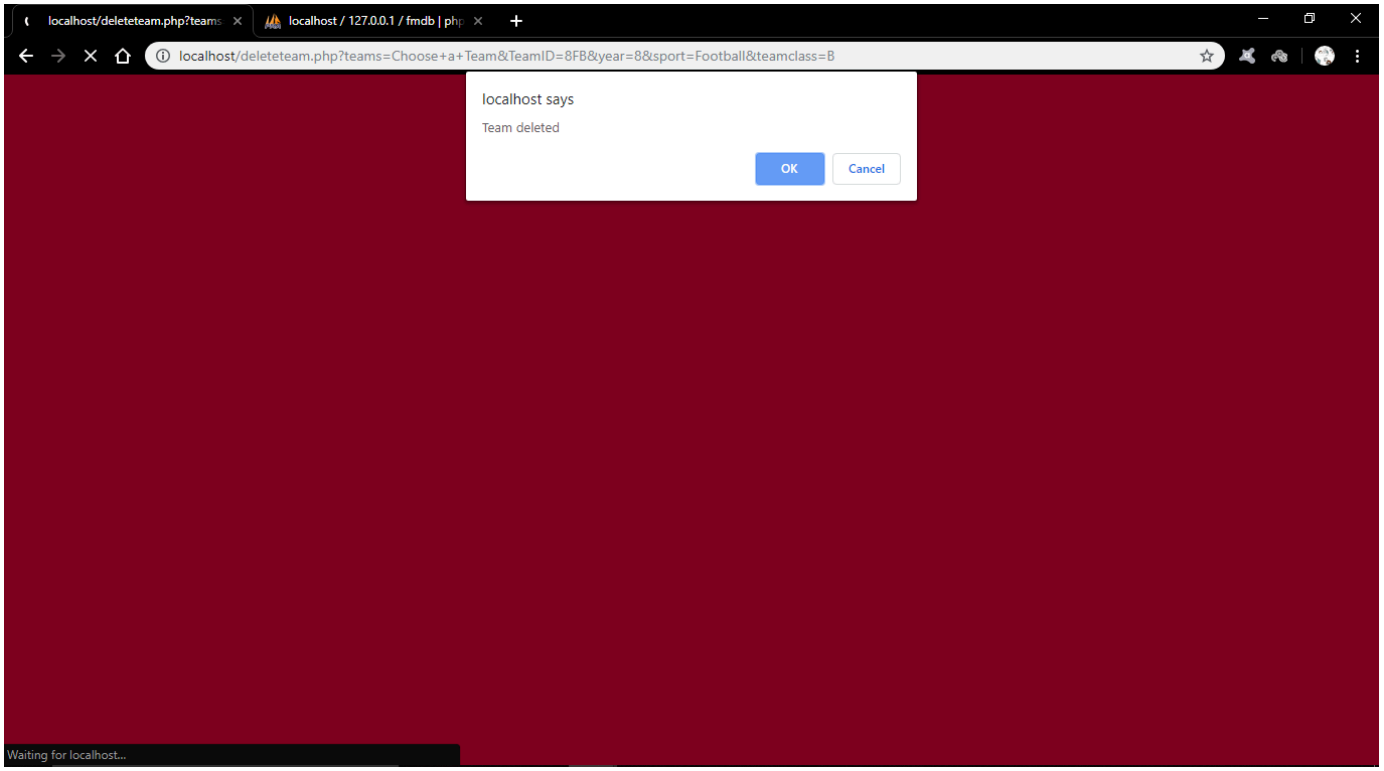


The code for the associative page called "editteam.php" is shown here:

```php
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());
//selects the database "fmdb"
if(isset($_GET['teams'])){
//if the $_GET['teams'] variable contains infomation, the next block of code is executed
$form_TeamID=$_GET['TeamID'];//$form_TeamID is assigned the data from the TeamID index of the $_GET variable
$form_year=$_GET['year'];//$form_year is assigned the data from the year index of the $_GET variable
$form_sport=$_GET['sport'];//$form_sport is assigned the data from the sport index of the $_GET variable
$form_Teamclass=$_GET['teamclass'];//$form_Teamclass is assigned the data from the teamclass index of the $_GET variable
if(!empty($form_TeamID) AND !empty($form_year) AND !empty($form_Teamclass) AND !empty($form_sport)){
//validation that checks no field from the field is empty and executes the next block of code if its true
$updateteam = "UPDATE `teams` set `TeamID` = '".$form_TeamID."', YearGroup = '".$form_year."', Sport = '".$form_sport."',
TeamClass = '".$form_Teamclass."' WHERE `TeamID` = '".$form_TeamID."'";
//sql query that updates the record with the same TeamID as the variable $form_TeamID with the variables
//$form_TeamID,$form_year,$form_sport and $form_Teamclass in their appropriate columns
$query = mysqli_query($con,$updateteam) or die(mysqli_error($con)); //executes the query infomation in $updateteam
echo "<script> window.confirm('Team Updated'); window.location.href='teampage.php'; </script>";
//displays the message "Team updated" and redirects the user to "teampage.php"
}//ends the second if statement
else{ //executes the next block of code if the if statement conditions are not fufilled
    echo "<script> window.confirm('No empty fields please'); window.location.href='teampage.php'; </script>";
    //displays the message "no empty fields please" and redirects the user to "teampage.php"
} // ends the else statement
}//ends the first if statement
?><!-- closes the php tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 2,352 lines : 29 | Ln : 29 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

The instance where a team is deleted can be seen here:

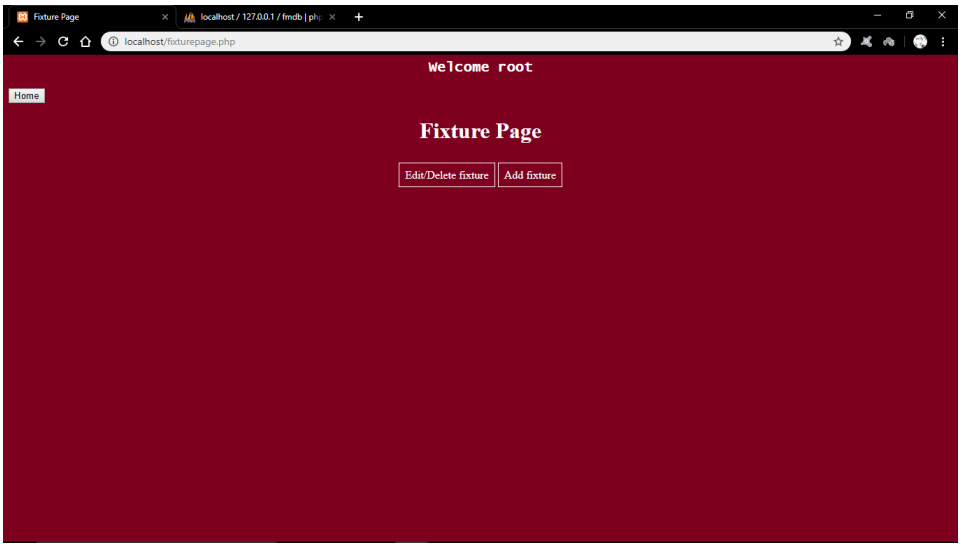


The code for the associative page called "deleteteam.php" can be seen here:

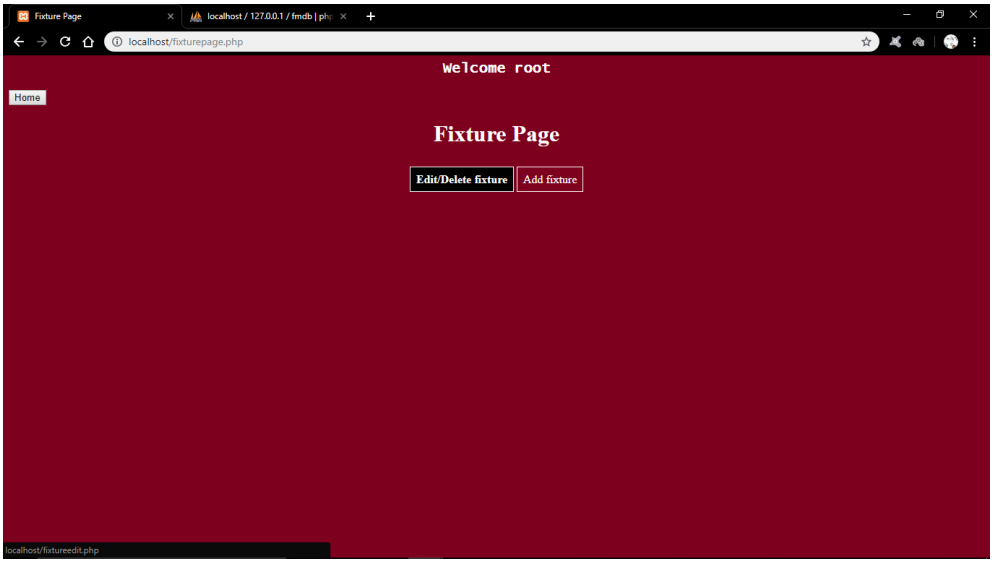
```php
<html><!--intialises the html tag of the page, the root element of the page-->
<?php  //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());//selects the database "fmdb"
$Datatodel=$_GET['TeamID']; //gets the infomation from tbe input box named "TeamID"
$DeleteTeam="DELETE from `teams` WHERE `TeamID` ='".$Datatodel."'";
//query that selects the team with the same data as the variable $datatodel and deletes it
$run = mysqli_query($con,$DeleteTeam)or die("Failed to Delete:" .mysqli_error($con)); // executes the query $DeleteTeam
echo "<script> window.confirm('Team deleted'); window.location.href='teampage.php'; </script>";
//displays the message "Team deleted"  and redirects the user to "teampage.php"
?> <!--closes the php tag -->
<head> <!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head> <!--closes the head tag-->
<h1>Delete Page</h1><!--adds a heading named Delete Page-->
<body id ="body-color"><!-- sets the style of the body to the attribute "body-color" on the style page-->
</body> <!--closes the body tag-->

```

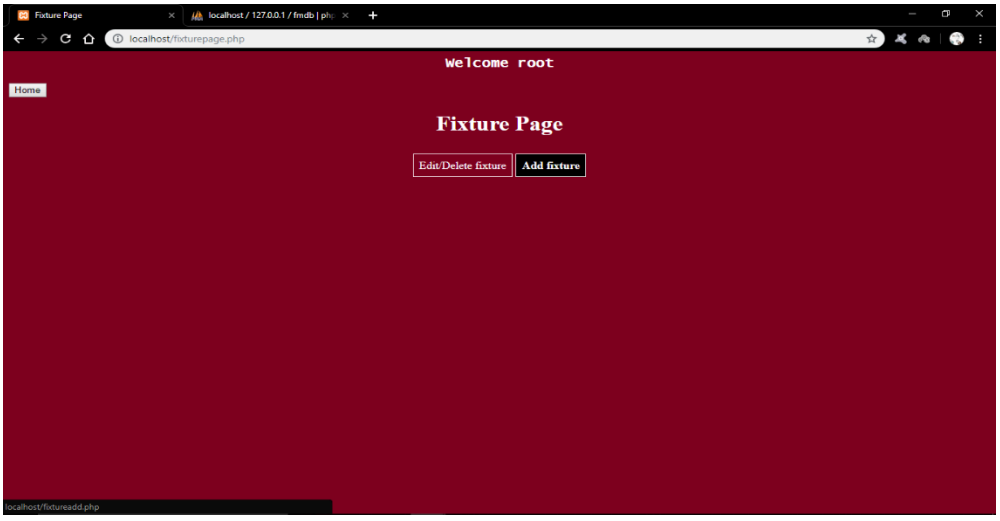
PHP Hypertext Preprocessor file | length : 1,444 lines : 19 | Ln : 12 Col : 30 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Fixture Page



The highlighting of the text box when hovered over feature was also added to the navigation options on this page aswell which can be seen here:



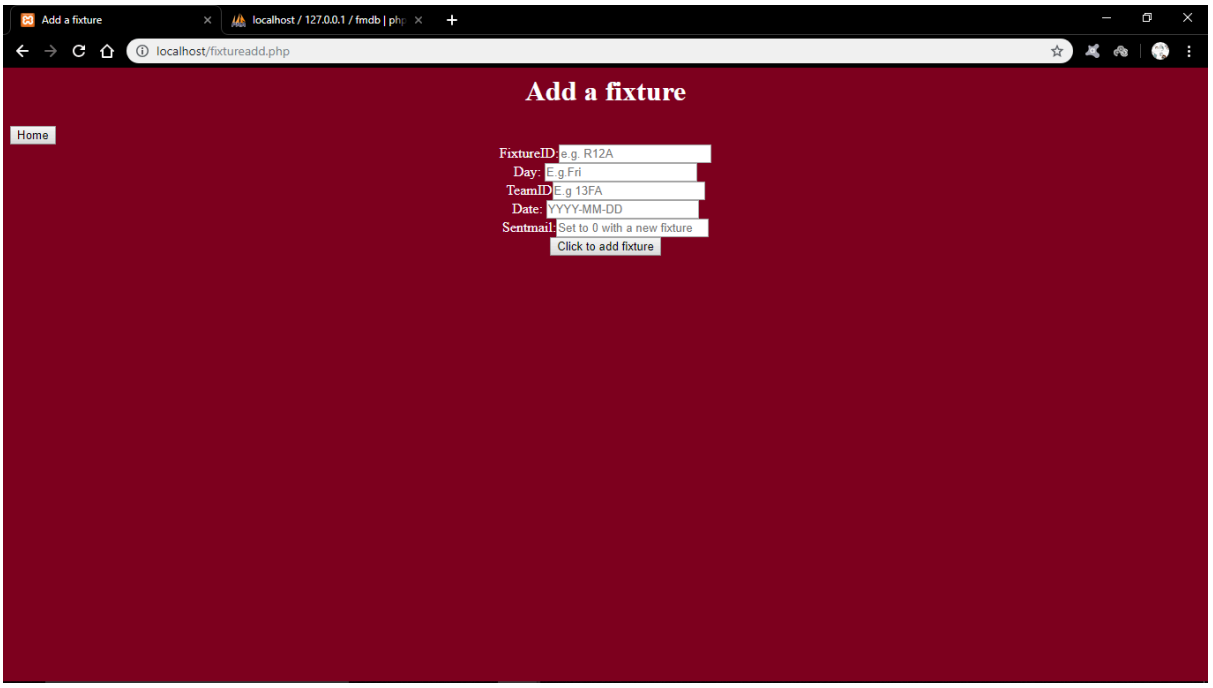


The code for the page can be seen here:

```php
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
session_start(); //starts the session with its variables
echo "<p> Welcome ".$_SESSION['user']."</p>"; //displays the message "welcome" with username of the user logged in e.g. admin
?> <!-- closes the php tag-->
<head> <!--intialises the head element of the page where meta info about the page is held-->
<title>Fixture Page</title> <!--Gives the title "Fixture Page" to the page that is seen on the top of a tab/window-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head><!-- closes the head tag-->
<body id="body-color"> <!-- sets the style of the body to the attribute "body-color" on the style page-->
<button type="button" onclick="window.location.href='Homepage.php'">Home</button>
<!-- creates a button that redirects the user to "homepage.php" labeled Home-->
<h1>Fixture Page</h1>
<!-- Creates a heading that displays "Edit/delete Page" on the page-->
<ul><!-- intialises an unordered list-->
 <li><a href="fixtureedit.php">Edit/Delete fixture</a></li>
 <!-- adds a list item to the unordered list, a link to "fixtureedit.php" labeled "Edit/Delete fixture"-->
 <li><a href="fixtureadd.php">Add fixture</a></li>
 <!--adds a list item to the unordered list, a link to "fixtureadd.php" labeled Add fixture-->
</ul><!-- closes the unordered list-->
</body><!--closes the body tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 1,586 lines : 22 | Ln : 21 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

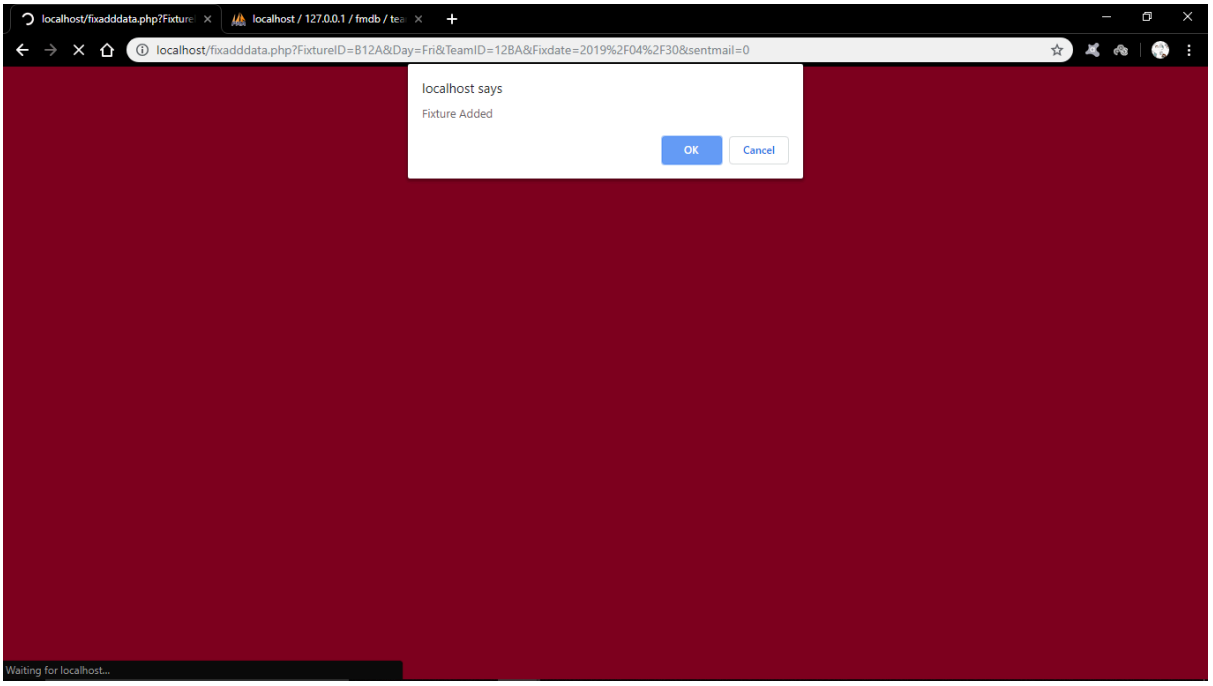
Fixture Page: Add Fixture



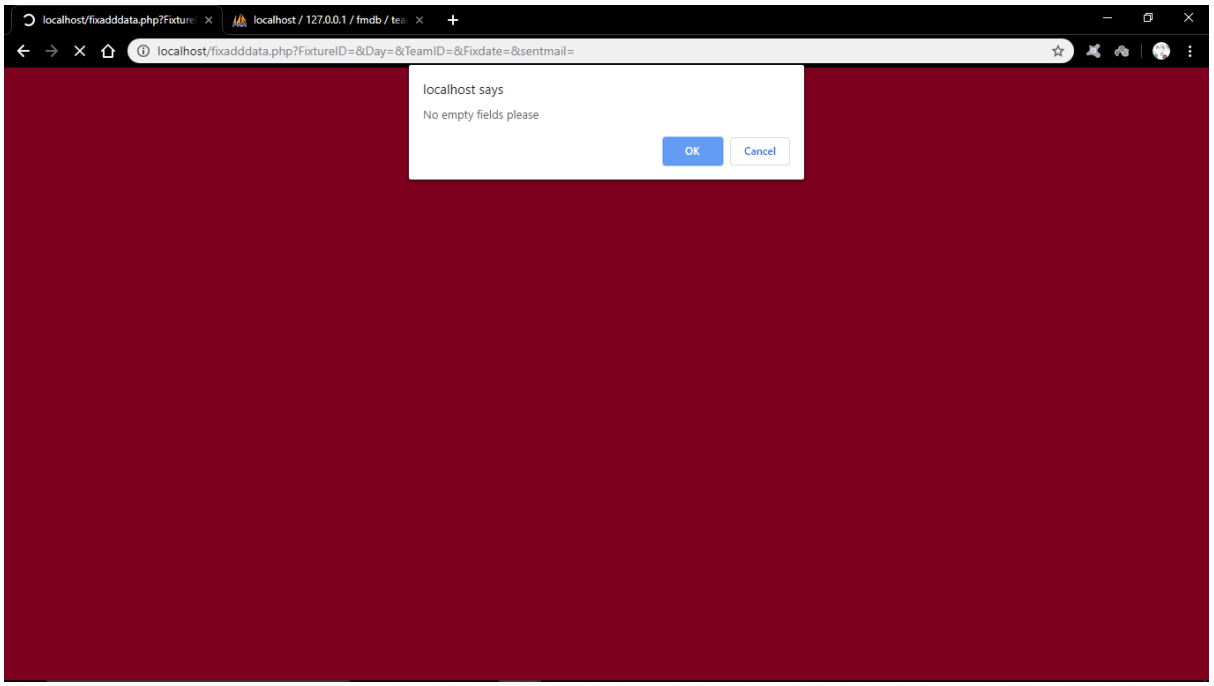
The design for the add fixture page of my solution is fully reflected on this page and the simple user interface criteria I identified has been followed here.

The associative page that the Fixture Page: Add Fixture page redirects to that processes the data and updates the database with the information only has 2 instances of a UI and that is when a Fixture is successfully inputted into a database or not due to empty fields. This can be seen on the next page:

Fixture being added:



Fixture not being added due to an empty field(s)

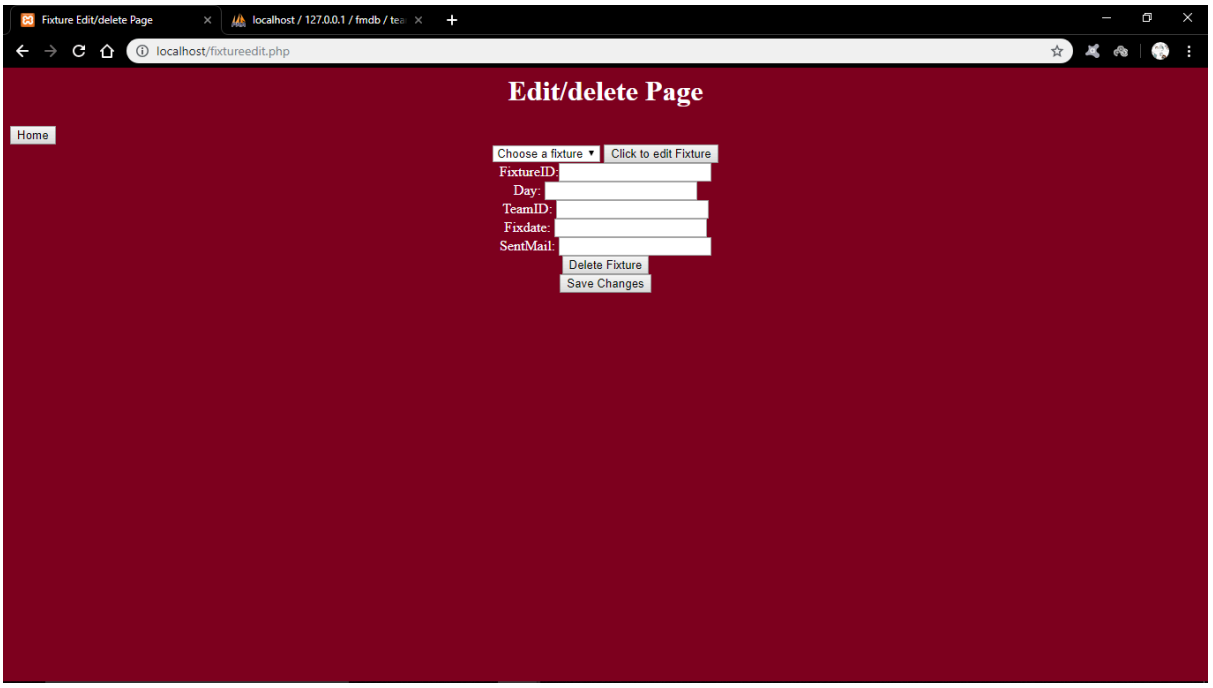


The user has to click ok/cancel to be redirected back to the previous page

The code for this assocative page named "fixtureadddata.php" can be seen here:

```php
<html><!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error()); //selects the database "fmdb"
if(isset($_GET['FixtureID'])){ //executes the next block of code if the $_GET variable contains the index FixtureID
    $form_FixtureID=$_GET['FixtureID'];//assigns the variable $form_FixtureID the infomation from the FixtureID index of the $_GET variable
    $form_Day=$_GET['Day'];//assigns the variable $form_Day the infomation from the Day index of the $_GET variable
    $form_TeamID=$_GET['TeamID'];//assigns the variable $form_TeamID the infomation from the TeamID index of the $_GET variable
    $form_Fixdate=$_GET['Fixdate'];//assigns the variable $form_Fixdate the infomation from the Fixdate index of the $_GET variable
    $form_sentmail=$_GET['sentmail'];//assigns the variable $form_sentmail the infomation from the sentmail index of the $_GET variable
    if(!empty($form_FixtureID) AND !empty($form_Day) AND !empty($form_TeamID) AND !empty($form_Fixdate) AND $form_sentmail!=""){
    //validation that checks no field from the field is empty and executes the next block of code if its true
        $addfixture="INSERT INTO fixtures(FixtureID,Day,TeamID,Fixdate,`SentMail?`) VALUES
        ('".$form_FixtureID."','".$form_Day."','".$form_TeamID."','".$form_Fixdate."','".$form_sentmail."')";
         //creates a new record with the variables $form_FixtureID,$form_Day,$form_TeamID,$form_Fixdate and $form_sentmail
        //into the appropriate columns of the table in the database
        $action = mysqli_query($con,$addfixture);//executes the query infomation from $addfixture
        echo "<script> window.confirm('Fixture Added'); window.location.href='fixturepage.php'; </script>";
        //displays the message "Fixture added" and redirects the user to "fixturepage.php"
            } //ends the second if statement
            else{ //executes the next block of code if the if statement conditions are not fufilled
                echo "<script> window.confirm('No empty fields please'); window.location.href='fixturepage.php'; </script>";
                //displays a confirm box with the message "No empty fields please" and redirects the user to "fixturepage.php"
        }//ends the else statement
            }// ends the first if statement
        ?> <!-- closes the php tag-->
<head><!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css">
<!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head><!--closes the head tag-->
<body id ="body-color"><!-- sets the style of the body to the attribute "body-color" on the style page-->
</body><!--closes the body tag-->
</html><!--closes the html tag-->
```

Fixture Page: Edit/Delete Fixture



Again, the design of this page has been closely followed and created to match

The dropdown menu contents containing the FixtureID values will be shown in my testing

The code for the page can be seen here:

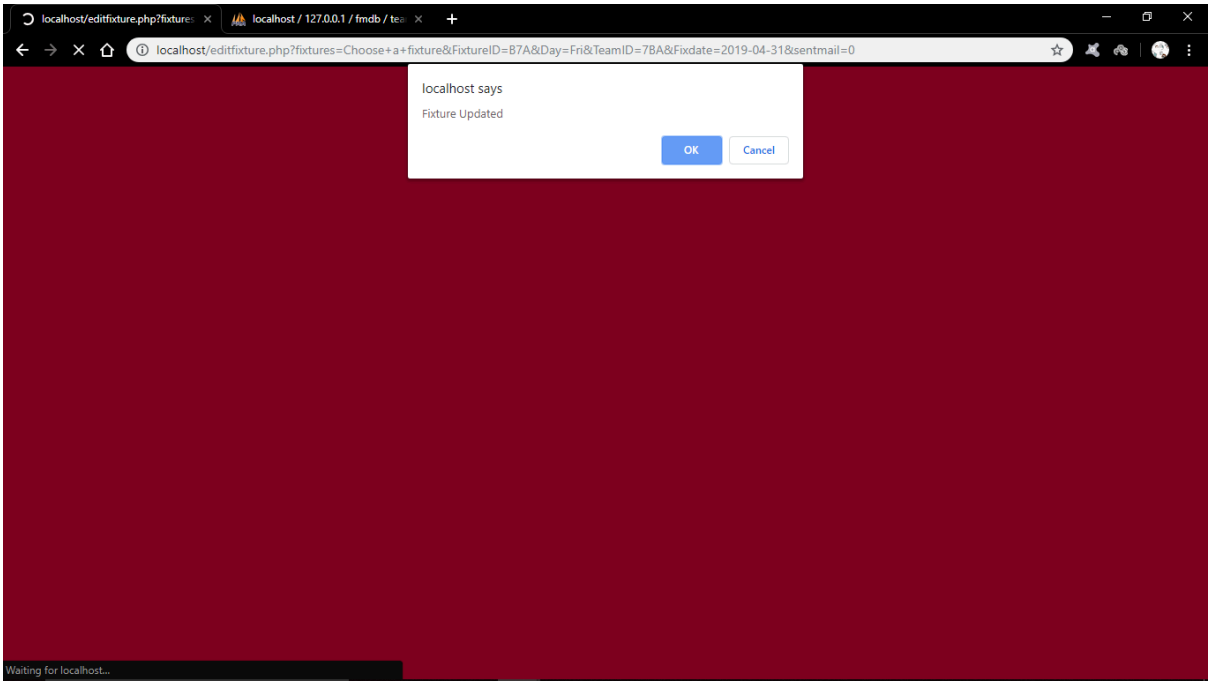
```
    $form_Fixdate = ""; //intialises the $form_Fixdate variable as blank
    $form_sentmail=""; //intialises the $form_sentmail variable as blank
    } //ends the else statement
    ?> <!--closes the php tag-->
<head><!--intialises the head element of the page where meta info about the page is held-->
<title>Fixture Edit/delete Page</title> <!--Gives the title "Fixture Page" to the page that is seen on the top of a tab/window-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head> <!--closes the head tag-->
<body id="body-color"> <!-- sets the style of the body to the attribute "body-color" on the style page-->
<h1>Edit/delete Page</h1>
<!-- Creates a heading that displays "Edit/delete Page" on the page-->
<button type="button" onclick="window.location.href='Homepage.php'">Home</button>
<!-- creates a button that redirects the user to "homepage.php" labeled Home-->
<form method="GET">
<!-- intialises a form in the "GET" method-->
<select name="fixtures" id="fixtures"> <!-- creates a dropdown menu with an id of "fixtures"-->
  <option selected="selected">Choose a fixture</option> <!--makes the intial value "choose a fixture"-->
  <?php  //intialises the php tag allowing php language to be written
    foreach($datas as $i){
        //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
        // to be assigned to the variable $i
        echo "<option value=".$i['FixtureID'].">".$i['FixtureID']."</option>";
        //adds all the FixtureIDs that was obtained as dropdown values
    }//ends the foreach statement
    ?>  <!-- closes the php tag-->
</select> <!--closes the select tag-->

<button type="submit" value="Submit">Click to edit Fixture</button><br>
<!--creates a button that submits the data on the page to the same page in order to populate the input fields-->
FixtureID:<input name="FixtureID" type="text" value="<?php echo $form_FixtureID;?>"><br>
<!--input box labeled "FixtureID" that is filled with the FixtureID selected from the dropdown menu-->
Day: <input name="Day" type="text" value="<?php echo $form_Day;?>"><br>
<!--input box labeled "Day" that is filled with the Day infomation related to the FixtureID when the edit button is clicked-->
TeamID: <input name="TeamID" type="text" value="<?php echo $form_TeamID;?>"><br>
<!--input box labeled "TeamID" that is filled with the TeamID infomation related to the FixtureID when the edit button is clicked-->
Fixdate: <input name="Fixdate" type="text" value="<?php echo $form_Fixdate;?>"><br>
<!--input box labeled "Fixdate" that is filled with the Fixdate infomation related to the FixtureID when the edit button is clicked-->
SentMail: <input name="sentmail" type="text" value="<?php echo $form_sentmail;?>"><br>
<!--input box labeled "SentMail" that is filled with the SentMail infomation related to the FixtureID when the edit button is clicked-->
<button type="submit" formaction = "fixturedel.php">Delete Fixture</button><br>
<!--button that when clicked submits the form and redirects the user to the "fixturedel.php" page-->
<button type="submit" formaction = "editfixture.php">Save Changes</button><br>
<!--button that when clicked submits the form and redirects rhe user to the "editfixture.php" page-->
</form><!--closes the form tag-->
</body><!--closes the body tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 6,071 lines : 82 | Ln : 39 Col : 5 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

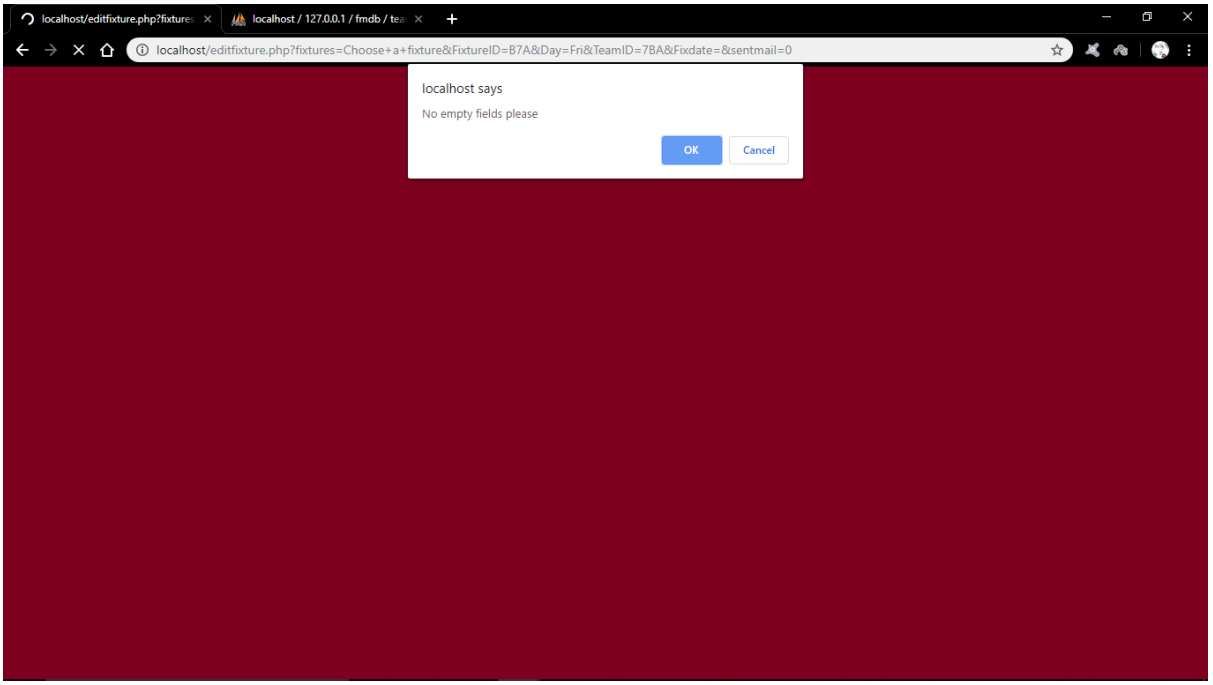
```
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());
//selects the database "fmdb"
$Fixtureselection= "SELECT * FROM `fixtures`"; //selects all the fixtures in the fixtures table
$query = mysqli_query($con,$Fixtureselection) or die(mysqli_error()); // executes the query $Fixtureselection
$fetch = mysqli_fetch_array($query) or die(mysqli_error()); //puts the data from $queryrun into an array
$result = $con->query($Fixtureselection); //runs the query $Fixtureselection and adds the data to the $result variable
$datas = array(); //initialises $datas as an array
while($data = $result->fetch_assoc()){
        //intialises $data with the infomation provided from the sql query $result and loops for all the infomation obtained
       $datas[] = $data;//adds the information from $data into the variable $datas, a 2d array
    } //ends the while statement
if(isset($_GET['fixtures'])){ //executes the next block of code if the $_GET['fixtures'] variable contains data
    foreach($datas as $i){
        //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
        // to be assigned to the variable $i
        if ($i['FixtureID']==$_GET['fixtures']){
        //executes the next block of code if the infomation in the FixtureID index of $i is the same as the $_GET['fixtures'] variable
        $form_FixtureID = $i['FixtureID']; //assigns the infomation from $i['FixtureID'] to $form_FixtureID
            $form_Day = $i['Day'];          //assigns the infomation from $i['Day'] to $form_Day
            $form_TeamID = $i['TeamID'];     //assigns the infomation from $i['TeamID'] to $form_TeamID
            $form_Fixdate = $i['Fixdate'];   //assigns the infomation from $i['Fixdate'] to $form_Fixdate
            $form_sentmail =$i['SentMail?']; //assigns the infomation from $i['SentMail?'] to $form_sentmail


        } //ends the second if statement
    } //ends the foreach statement
} //ends the first if statement

else{ //executes the next block of code if the conditions for the if statements arent fulfilled
    $form_FixtureID = ""; //intialises the $form_FixtureID variable as blank
    $form_Day = ""; //intialises the $form_Day variable as blank
    $form_TeamID= ""; //intialises the $form_TeamID variable as blank
```

The associative pages of the Fixture Page: Edit/delete fixture page, that updates the record with the information obtained from the form into the database or deletes the chosen record has only 3 instances of a UI. Two for the update of the record and one for the successful deletion of a record.

The update of a fixture:



The unsuccessful update of a fixture due to empty fields:



:

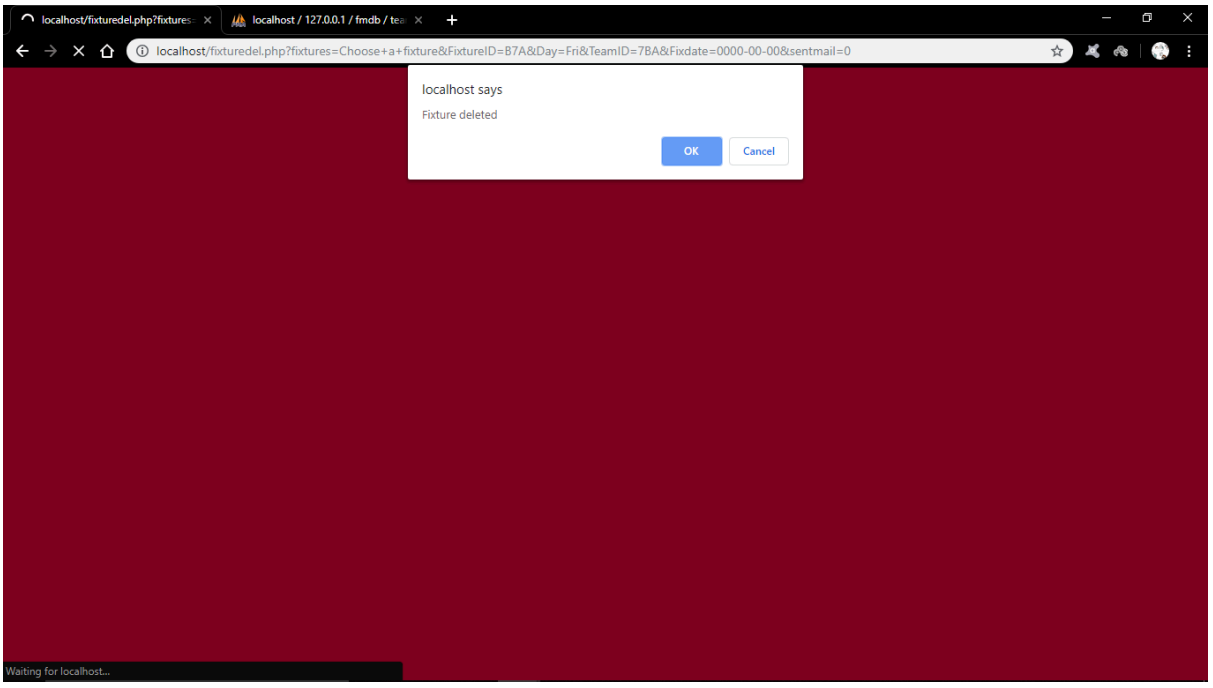
The user has to click ok/cancel to be redirected back to the previous page

The code for the update of a fixture on the associative page named "editfixture.php" can be seen here:

```php
<html><!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());
//selects the database "fmdb"
if(isset($_GET['fixtures'])){
//executes the next block of code if the $_GET['fixtures'] variable contains data
$form_FixtureID=$_GET['FixtureID']; //$form_FixtureID is assigned the data from the FixtureID index of the $_GET variable
$form_Day=$_GET['Day']; //$form_Day is assigned the data from the Day index of the $_GET variable
$form_TeamID=$_GET['TeamID']; //$form_TeamID is assigned the data from the TeamID index of the $_GET variable
$form_Fixdate=$_GET['Fixdate']; //$form_Fixdate is assigned the data from the Fixdate index of the $_GET variable
$form_sentmail=$_GET['sentmail']; //$form_sentmail is assigned the data from the sentmail index of the $_GET variable

if(!empty($form_FixtureID) AND !empty($form_Day) AND !empty($form_TeamID) AND !empty($form_Fixdate) AND $form_sentmail!=""){
//validation that checks no field from the field is empty and executes the next block of code if its true
$updatefixtures = "UPDATE `fixtures` set `FixtureID` = '".$form_FixtureID."', Day = '".$form_Day."',
TeamID = '".$form_TeamID."', Fixdate = '".$form_Fixdate."', `SentMail?`= '".$form_sentmail."' WHERE `FixtureID` = '".$form_FixtureID."'";
//sql query that updates the record with the same FixtureID as the variable $form_FixtureID with the variables
//$form_FixtureID,$form_Day,$form_TeamID , $form_Fixdate and $form_sentmail in their appropriate columns
$query = mysqli_query($con,$updatefixtures) or die(mysqli_error($con));
//executes the query infomation in $updatefixtures
echo "<script> window.confirm('Fixture Updated'); window.location.href='fixturepage.php'; </script>";
//displays the message "Fixture updated" and redirects the user to "fixturepage.php"
} //ends the second if statement
else{ //executes the next block of code if the if statement conditions are not fufilled
    echo "<script> window.confirm('No empty fields please'); window.location.href='fixturepage.php'; </script>";
    //displays the message "No empty fields" and redirects the user to "fixturepage.php"
}   //ends the else statement
} //ends the first if statement
?><!-- closes the php tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 2,590 lines : 32 | Ln : 13 Col : 35 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Successful Deletion of a fixture:



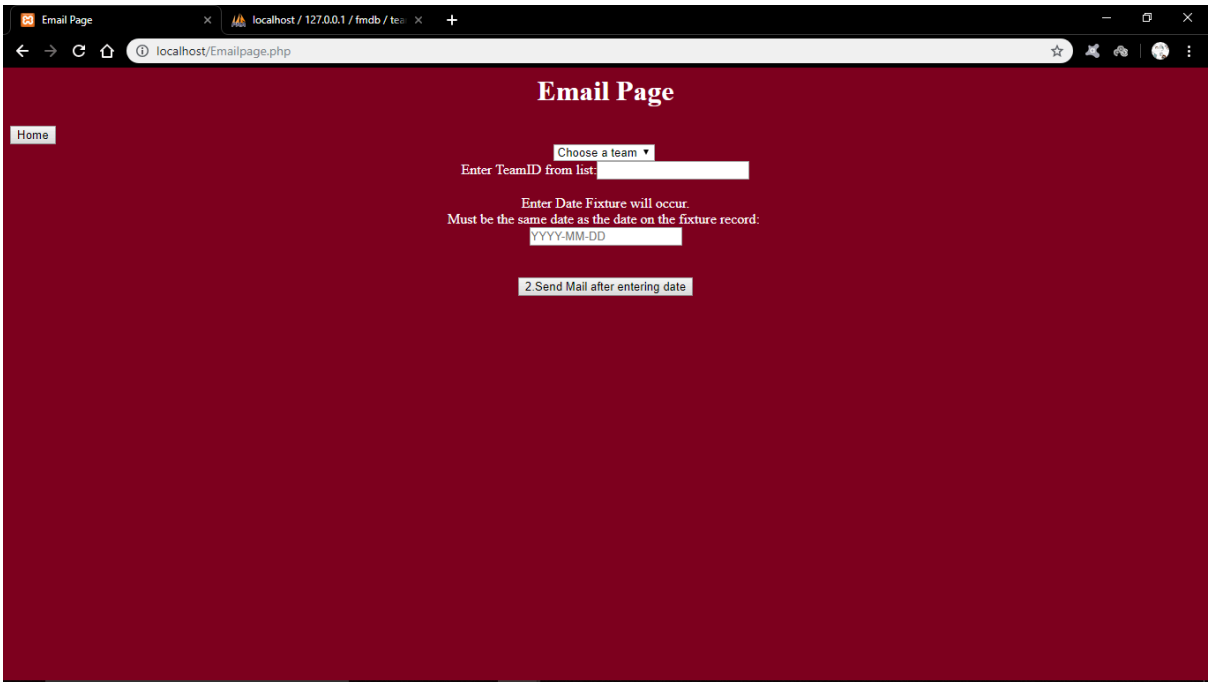
The user has to click ok/cancel to be redirected back to the previous page

The code for the associative page calle"fixturedel.php" can be seen here:

```php
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());
//selects the database "fmdb"
$Datatodel=$_GET['FixtureID'];  //gets the infomation from tbe input box named "FixtureID"
$deletefixture="DELETE from fixtures WHERE FixtureID ='".$Datatodel."'";
//query that selects the team with the same data as the variable $datadel and deletes it
$run = mysqli_query($con,$deletefixture)or die("Failed to Delete:" .mysqli_error($con));
//executes the query $deletefixture
echo "<script> window.confirm('Fixture deleted'); window.location.href='fixturepage.php'; </script>";
//displays an a confirm box with the message "Fixture deleted" and redirects the user to "fixturepage.php"
?><!--closes the php tag-->

<head> <!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
</head><!--closes the head tag-->
<h1>Delete Page</h1><!--adds a heading named Delete Page-->
<body id ="body-color"><!-- sets the style of the body to the attribute "body-color" on the style page-->
</body><!--closes the body tag-->

```

PHP Hypertext Preprocessor file | length : 1,414 lines : 21 | Ln : 20 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

Email Page



The design for the Email page of my solution is fully reflected on this page and the simple user interface criteria I identified has been followed here

The dropdown menu values being the TeamID's will be shown in my testing

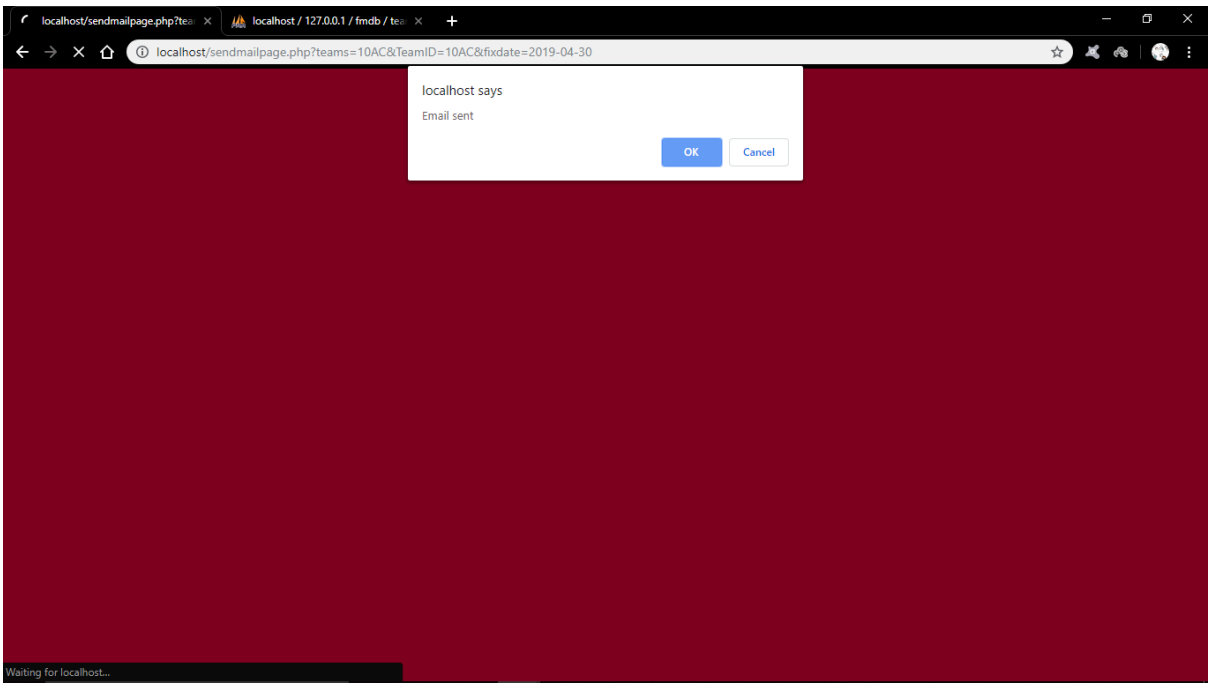
The code for the page can be seen on the next page:

```php
<html> <!--intialises the html tag of the page, the root element of the page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error());//selects the database "fmdb"
$selectteams= "SELECT * FROM `teams`"; //selects all the teams in the teams table
$query = mysqli_query($con,$selectteams) or die(mysqli_error()); // executes the query $selectteams
$fetch = mysqli_fetch_array($query) or die(mysqli_error());//puts the data from $selectteams into an array
$result = $con->query($selectteams);//runs the query $selectteams and adds the data to the $result variable
$datas = array();//initialises $datas as an array
while($data = $result->fetch_assoc()){
    //intialises $data with the infomation provided from the sql query $result and loops for all the infomation obtained
    $datas[] = $data;//adds the information from $data into the variable $datas, a 2d array
    }   //ends the while statement
    if(isset($_GET['teams'])){ //executes the next block of code if the $_GET['teams'] variable contains data
    foreach($datas as $i){
        //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
        // to be assigned to the variable $i
        if ($i['TeamID']==$_GET['teams']){
        // executes the next block of code if the infomation in the TeamID index of $i is the same as the $_GET['teams'] variable
            $form_TeamID = $i['TeamID'];//assigns the infomation from $i['TeamID'] to $form_TeamID
    }//ends the second if statement
    }//ends the foreach statement
    }//ends the first if statement
    else{ //executes the next line of code if the conditions for the if and foreach statements arent fulfilled
        $form_TeamID = ""; //intialises the $form_TeamID variable as blank
    }//closes the else statement
    ?><!-- closes the php tag-->
<head> <!--intialises the head tag-->
<title>Email Page</title><!--Gives the title "Email Page" to the page that is seen on the top of a tab/window-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
<body id = "body-color">  <!-- sets the style of the body to the attribute "body-color" on the style page-->
<h1>Email Page</h1><!-- Creates a heading that displays "Email Page" on the page-->
<button type="button" onclick="window.location.href='Homepage.php'">Home</button>
<!-- creates a button that redirects the user to "homepage.php" labeled Home-->
<br> <!--adds a new line to seperate items on a page-->
<form action="window.location.href='sendmailpage.php'" method="GET">
<!-- intialises a form in the "GET" method with actions being committed on the page such as submitting a form
sending the user to "sendmailpage.php"-->
<select name="teams" id="teams"><!-- creates a dropdown menu with an id of "teams"-->
  <option selected="selected">Choose a team</option><!--makes the intial value "choose a team"-->
  <?php // intialises the php tag allowing php language to be written
    foreach($datas as $i){
        //loop that executes the next block of code which lasts the length of time it takes for the amount of infomation obtained
        // to be assigned to the variable $i
        echo "<option value=".$i['TeamID'].">".$i['TeamID']."</option>";
        //adds all the TeamIDs that was obtained to the dropdown menu as dropdown values
    }//ends the foreach statement
    ?> <!-- closes the php tag-->
</select> <!--closes the select tag-->
<br> <!--adds a new line to seperate items on a page-->
Enter TeamID from list:<input name="teams" type="text" id="teams"></input><br>
<!--input box labeled "TeamID" to input the TeamID the user is sending an email about-->
<br><!--adds a new line to seperate items on a page-->
Enter Date Fixture will occur.<br>
Must be the same date as the date on the fixture record:
<br><!--adds a new line to seperate items on a page-->
<input name ="fixdate" type="text" id="fixdate" placeholder="YYYY-MM-DD"></input><br>
<!--input box with an id "fixdate" to input the Date the fixture is taking place-->
<br><!--adds a new line to seperate items on a page-->
<br><!--adds a new line to seperate items on a page-->
<button type ="submit" formaction="sendmailpage.php">2.Send Mail after entering date</button>
<!-- button that submits the form and redirects the user to -->
</form><!--closes the form tag-->
</body><!--closes the body tag-->
</head><!--closes the head tag-->
</html><!--closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 4,741 lines : 67 | Ln : 67 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

The associative page of the Email page , where the data from the email page is processed and used to send an email. has one instance of a UI shown on the next page.

When an email is sent:



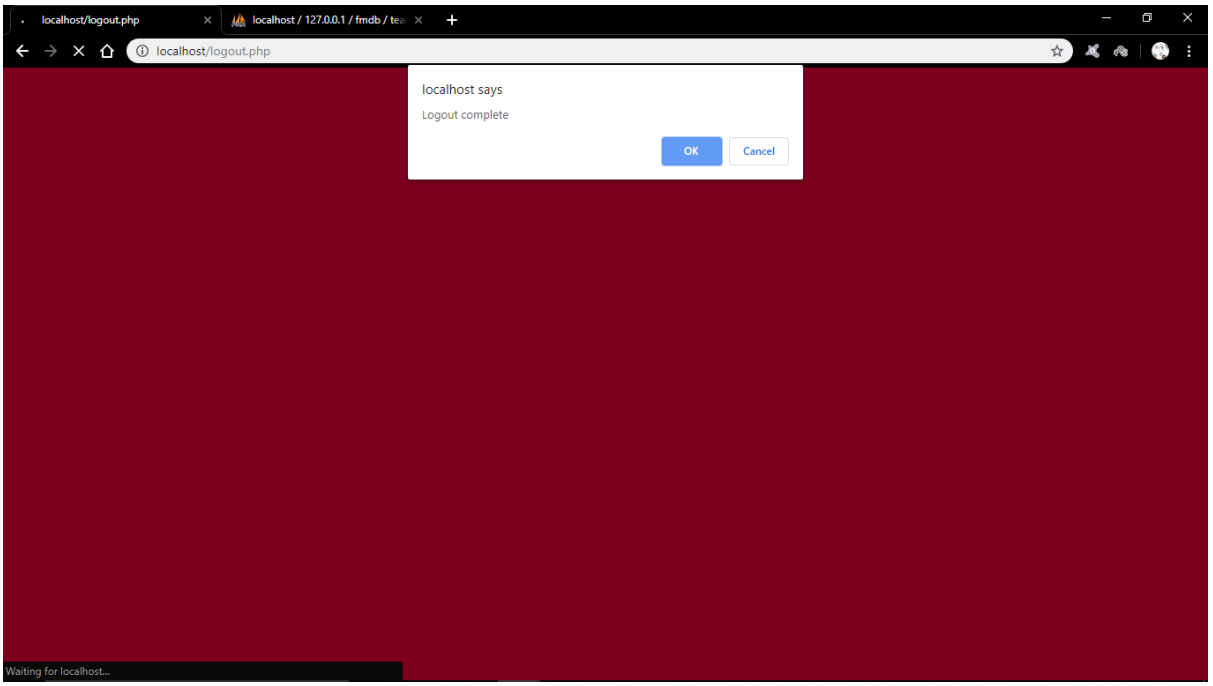
The user has to click ok/cancel to be redirected back to the previous page

The code for the associative page named sendemailpage.php can be seen here:

```
<html> <!--intialises the html tag of the page, the root element of the page-->
<head> <!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"> <!--sets the style of the page to the infomation found on "Pagestyle.css"-->
<body id = "body-color"> <!-- sets the style of the body to the attribute "body-color" on the style page-->
<?php //intialises the php tag allowing php language to be written
$con=mysqli_connect('localhost','root')or die("Failed to connect to MySQL: " . mysqli_error());
//connects the user to the database address "localhost" using the username "root"
$db=mysqli_select_db($con,'fmdb') or die("Failed to connect to MySQL: " . mysqli_error()); //selects the database "fmdb"
if(isset($_GET['teams'])){
//if the $_GET['teams'] variable contains infomation, the next block of code is executed
    $form_TeamID=$_GET['TeamID']; //assigns the $form_TeamID the infomation from the TeamID index of the $_GET variable
    $form_fixdate=$_GET['fixdate']; //assigns the $form_fixdate the infomation from the fixdate index of the $_GET variable
    $emailselection= "SELECT Email
    FROM teachers
    INNER JOIN timetable on teachers.TeacherID=Timetable.TeacherID
    INNER JOIN students on timetable.TimetableID=students.TimetableID
    INNER JOIN teams on students.TeamID=teams.TeamID
    WHERE teams.TeamID = '".$form_TeamID."'";
    // query that selects the email from teachers that are linked to the TeamID that is the same as
    // the variable $form_TeamID after joining the tables together
    $runquery=mysqli_query($con,$emailselection) or die(mysqli_error($con));
    //executes the query infomation in the variable $emailselection
    $results = mysqli_fetch_assoc($runquery) or die(mysqli_error($con));
    // puts the infomation from the query execution into an associative array
    $studentselection= "SELECT Firstname,Lastname
    FROM students
    WHERE students.TeamID='".$form_TeamID."'";
    //query to select the firstname and lastname of those that are in the team with the TeamID as the variable $form_TeamID
    $runquery2=mysqli_query($con,$studentselection) or die(mysqli_error($con));
    //executes the query infomation from the variable $studentselection
    $results2 = mysqli_fetch_assoc($runquery2) or die(mysqli_error($con));
    //puts the data obtained from the query of $studentselection in a associative array
    $datas = array(); //intialises $datas as an array
    while($data = $runquery2->fetch_assoc()){
    //intialises $data with the infomation provided from the sql query $runquery2//and loops for all the infomation obtained
```

LOGOUT PAGE

The logout page also has one instance of a user interface which is when the logout process has been completed shown here:



The code for the logout page("logout.php") can be seen here:

```php
<html><!--intialises the html element of the page, the root element of the page -->
<?php //intialises the php tag to allow php language be written
session_start();//sets the session in case it has not been previously set
session_destroy();//destroys all $_SESSION variables
?><!--closes the php tag -->
<head><!--intialises the head tag-->
<link rel="stylesheet" type="text/css" href="Pagestyle.css"><!--sets the style of the page to the infomation found on "Pagestyle.css"-->
<body id = "body-color"><!--sets the style of the page to the infomation found on the body-color attribute on the style page -->
<script language="javascript"><!--intialises the script tag with javascript being identified as the language being used -->
  javascript:window.history.forward(1);//adds 1 page  to the browser history to prevent the user visiting the home page again
</script><!-- closes the script tag-->
<?php //intialises the php tag to allow php language be written
echo "<script> window.confirm('Logout complete'); window.location.href='index.php'; </script>";
//displays a confirm box with the infomation "logout complete" and redirects the user to the index/login page.
?><!-- closes the php tag -->
</body><!-- closes the body tag-->
</head><!--closes the head tag-->
</html><!-- closes the html tag-->
```

PHP Hypertext Preprocessor file | length : 1,318 lines : 18 | Ln : 17 Col : 34 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

D.Evaluation

Test Plan

What went well

The solution I created for my stakeholder ultimately serves its purpose to the

Improvements that could be made

Throughout the creation of my solution, I have realised that my solution could be improved in many different areas and aspects. These include:

SQL Injection Prevention:

This is a major threat to the database of my solution