**CSCE 4523 Database Management Systems**

**Homework 1**

**Due: Wednesday, April 21 at 11:59pm**

**By: Kayla Samuel Jackson Hernandez and Julio Nolasco**

**Objectives**

The objective of this project to implement a web frontend that manages a database. To do this we give the user functions that allows them to change the database in various ways. For example, being able to add an item to a table.

**Approach**

For this project we went with C++ and php for the languages. For the host server we went with turing since we are familiar with it. For our SQL database we followed the example that was provided. For the teams table we made TeamId the primary key. For the games table GameId is the primary key. In the result table we indicated that GameId, TeamId1, and TeamId2 are foreign keys to the other two tables. When it comes to the DBMS, we used MySQL found on turing along with OBDC to allow for interaction between the database and webpage. For the link to our homepages, we used the same one that was on the read me file but changed USERNAME to our correct usernames. To implement this code, we started by taking the sample code that was provided and figured out how to get it to run. Once we did that, we ran some test to make sure we can view our sites by using the link provided in the read me file. Once we were able to view each other’s sites. We moved on to understanding the different files what they do, and how they work together. To make sure the functions we create work we first created our new database and populated it with quick inserts. We moved on to altering the insert file and changed it to add a game. Before testing it, we changed the hello file to act as a menu that would lead the user to these different functions. To test the function, we opened the hello page which led us to the insert game function. We entered the right value and got the desired result. To double check this we manually entered the database to ensure the values we entered where there. With that first function done we moved on to the other functions.

**Add A Team:**

* Adding a team prompts the user to enter University Name, Nickname, and Rank. All these values are of type text in the php file, this makes it easy to convert it to a string that gets passed to the database. The TeamId get incremented by one every time a new team is added. When shown to the user they are in ascending order according to rank.

**Add A Game:**

* Adding a game prompts the user to enter Rank1, Rank2, Location, and Date. Like in the add team function the GameId gets incremented by one every time one gets created. When entering the values, you will only be able to enter numerical values for Rank1 and Rank2. We provided a correct format for Date to the side so that the user knows how to enter the values.

**Add the Result Of a Game:**

**View All Teams:**

* View all teams will automatically show the user all the teams along with all their attributes. These attributes are: TeamId, University Name, Nickname, and Rank. All attributes found in the Teams table.

**View All Results by Team:**

**View Results by Date:**

How did you implement this? What languages? Strategies? Design?

What format did you choose for your record? What size and order of fields?

What delimiter? What total record size? Show a sample record.

What did you use the config file for? Show the config file.

Did you use sample code?

How did you handle overflow file (methods)?

**Results**

Homepages URL:

Julio - <http://www.csce.uark.edu/~jrnolasc/project_cpp/hello.html>

Kayla - <http://www.csce.uark.edu/~ksh006/project_cpp/hello.html>

The functions worked as we intended. We can add teams, games, and results using the web front end.

**Work Split:**

Julio

Functions 1,2, and 4

Kayla

Functions 3,5,6.

**Testing**

To test the program, we started by making sure the tables we created worked and could be altered within MySQL. When we knew our database worked, we could then check to see if our functions did as they were intended in the database. If we added a team through the website, we would check MySQL to ensure that team was there. If we wanted to see all teams it should have reflected that same result MySQL. On top of that we made sure that we could view our websites to ensure we followed the right steps to make it accessible. We were able to avoid duplicate Ids since we take the TeamId and GameId out of the users’ hands. We can add games and see all the games that exist after creating a new one. We can add results to the able and give an error if the game does not exist. We can view all teams along with all their attributes. We can see the results for a given team and see who won. We can also view the results for a given date and show a winner.

Graphical user interface, text, application, email

Description automatically generated

The home page for the database with links to all

Graphical user interface, application

Description automatically generated

The first function that allows the user to enter a team.

Graphical user interface, text, application, email

Description automatically generated

Function 2 that lets the user add a game along with the results of the addition of game 6 in Sky Stadium.

Graphical user interface

Description automatically generated with low confidence

Function 2 that lets users enter the result of a game.

Graphical user interface, text, application

Description automatically generated

Function 4 that lets the user see all the teams and their attributes.

Graphical user interface, text, application

Description automatically generated

Function 5 that lets the user enter a university name to find the results for that team and indicates a winner.

Text

Description automatically generated

Function 6 that lets the user enter a date to see all the game results for that date.