CMSC Project 1 Documentation

Computer Science Virtual Advisor

Table of Contents

1. Team Members & Contributions

2. Project Design

2.1. Front End Overview

2.2. Technical Overview

3. Server Setup

4. DB Setup

CSS Master

This project was very much a collaborative effort. While each person had specific tasks/roles to

fulfill, everyone helped out and there was overlap with all parts of the project. We utilized Git

(Bitbucket), Google Docs, and inperson

meetings to facilitate collaboration.

1

Project Design

Front End Overview

This is how the system functions from an enduser’s

perspective.

Technical Overview

This is how the system functions from a developer’s perspective.

2

Server Setup

To be able to leverage a reasonable version of PHP (as opposed to the old version on GL), we

chose to run our project on our own server. We used the WAMP server. If you already have

WAMP installed, skip to **step #5** . You may choose follow these brief instructions to install

WAMP, or use the installation instructions located on blackboard or the WAMP website.

1. Update Visual C++ Redistribute pack

2. Install WAMP (2.5 with PHP 5.5)

a. Download the 32 or 64 bit version from http://www.wampserver.com/en/

3. Map it to your primary browser while installing. Open search for Chrome/Firefox, then

open file location to get the address

4. Click the new Icon located in the system tray, select “start all services”

5. Copy the **project1.zip** file into the www folder

a. The www folder is located in the WAMP install directory, potentially

**C:\wamp\www**

6. Extract **project1.zip** inside the www folder

7. The URL for the login page of the project should now be at

**http://localhost/project1/login.php**

Database Setup

To setup the database, open phpMyAdmin (in a web browser, navigate to localhost and find

phpMyAdmin on the page), navigate to the **SQL tab** , copy and paste the following SQL query,

and click “ **Go** ”:

CREATE DATABASE IF NOT EXISTS cmsc433;

CREATE USER 'cmsc433' @ 'localhost' IDENTIFIED BY '5KybKtYM' ;

GRANT ALL PRIVILEGES ON cmsc433.\* TO 'cmsc433' @ 'localhost' WITH GRANT OPTION;

CREATE TABLE IF NOT EXISTS `cmsc433` . `users` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`name` VARCHAR ( 255 ) NOT NULL , `umbc\_id` VARCHAR ( 7 ) NOT NULL , `email` VARCHAR ( 255 ) NOT NULL ,

PRIMARY KEY ( `id` ), UNIQUE ( `umbc\_id` ));

CREATE TABLE IF NOT EXISTS `cmsc433` . `cmsc200` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`user\_id` INT UNSIGNED NOT NULL , `is\_recommendation` TINYINT NOT NULL , `201` TINYINT NOT NULL

, `202` TINYINT NOT NULL , `203` TINYINT NOT NULL , `232` TINYINT NOT NULL , `291` TINYINT NOT

NULL , `299` TINYINT NOT NULL , PRIMARY KEY ( `id` ), FOREIGN KEY ( `user\_id` ) REFERENCES

users ( `id` )) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `cmsc433` . `cmsc300` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`user\_id` INT UNSIGNED NOT NULL , `is\_recommendation` TINYINT NOT NULL , `304` TINYINT NOT NULL

, `313` TINYINT NOT NULL , `331` TINYINT NOT NULL , `341` TINYINT NOT NULL , `345` TINYINT NOT

3

NULL , `352` TINYINT NOT NULL , `391` TINYINT NOT NULL , PRIMARY KEY ( `id` ), FOREIGN KEY

( `user\_id` ) REFERENCES `users` ( `id` )) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `cmsc433` . `cmsc400` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`user\_id` INT UNSIGNED NOT NULL , `is\_recommendation` TINYINT NOT NULL , `411` TINYINT NOT NULL

, `421` TINYINT NOT NULL , `426` TINYINT NOT NULL , `427` TINYINT NOT NULL , `431` TINYINT NOT

NULL , `433` TINYINT NOT NULL , `435` TINYINT NOT NULL , `436` TINYINT NOT NULL , `437`

TINYINT NOT NULL , `441` TINYINT NOT NULL , `442` TINYINT NOT NULL , `443` TINYINT NOT NULL ,

`444` TINYINT NOT NULL , `446` TINYINT NOT NULL , `447` TINYINT NOT NULL , `448` TINYINT NOT

NULL , `451` TINYINT NOT NULL , `452` TINYINT NOT NULL , `453` TINYINT NOT NULL , `455`

TINYINT NOT NULL , `456` TINYINT NOT NULL , `457` TINYINT NOT NULL , `461` TINYINT NOT NULL ,

`465` TINYINT NOT NULL , `466` TINYINT NOT NULL , `471` TINYINT NOT NULL , `473` TINYINT NOT

NULL , `475` TINYINT NOT NULL , `476` TINYINT NOT NULL , `477` TINYINT NOT NULL , `478`

TINYINT NOT NULL , `479` TINYINT NOT NULL , `481` TINYINT NOT NULL , `483` TINYINT NOT NULL ,

`484` TINYINT NOT NULL , `486` TINYINT NOT NULL , `487` TINYINT NOT NULL , `491` TINYINT NOT

NULL , `493` TINYINT NOT NULL , `495` TINYINT NOT NULL , `498` TINYINT NOT NULL , `499`

TINYINT NOT NULL , PRIMARY KEY ( `id` ), FOREIGN KEY ( `user\_id` ) REFERENCES `users` ( `id` ))

ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `cmsc433` . `math` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`user\_id` INT UNSIGNED NOT NULL , `is\_recommendation` TINYINT NOT NULL , `151` TINYINT NOT

NULL , `152` TINYINT NOT NULL , `221` TINYINT NOT NULL , PRIMARY KEY ( `id` ), FOREIGN KEY

( `user\_id` ) REFERENCES `users` ( `id` )) ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `cmsc433` . `science` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT ,

`user\_id` INT UNSIGNED NOT NULL , `is\_recommendation` TINYINT NOT NULL , `chem101` TINYINT NOT

NULL , `chem102` TINYINT NOT NULL , `chem102L` TINYINT NOT NULL , `biol141` TINYINT NOT NULL ,

`biol142` TINYINT NOT NULL , `phys121` TINYINT NOT NULL , `phys122` TINYINT NOT NULL ,

`phys122L` TINYINT NOT NULL , `ges110` TINYINT NOT NULL , `ges286` TINYINT NOT NULL , `sci100`

TINYINT NOT NULL , PRIMARY KEY ( `id` ), FOREIGN KEY ( `user\_id` ) REFERENCES `users` ( `id` ))

ENGINE = InnoDB;

CREATE TABLE `statistics` ( `id` INT UNSIGNED NOT NULL AUTO\_INCREMENT , `user\_id` INT UNSIGNED

NOT NULL , `is\_recommendation` TINYINT NOT NULL , `355` TINYINT NOT NULL , PRIMARY KEY ( `id` ),

FOREIGN KEY ( `user\_id` ) REFERENCES `users` ( `id` )) ENGINE = InnoDB;

If you need delete/uninstall the database:

1. Click on the database named **“cmsc433”** in the side navigation bar

2. Click on **“Operations”** in the top navigation bar

3. Under "Remove database" select " Drop the database (DROP) "

4. Select OK

To remove the user, run the following command in the SQL tab:

**DROP USER 'cmsc433'@'localhost';**

4

Database Design

The database is set up into 7 different tables.These tables are the **users, cmsc 200,**

**cmsc 300, cmsc 400, math, science, and statistics** . The users table is responsible for storing

the logged in user’s information. It has a unique primary key as its first column. Because this key

can never be repeated and auto increments, this column can be used to uniquely identify a

user. Its other columns are the user’s full name, UMBC ID, and UMBC email. UMBC ID is also

set to be unique because there should never be a user that has the same UMBC ID.

The other 6 tables are set up almost identically. The reason for splitting up classes into 6

different tables was it helps keep the data segregated, and makes it very easy to query by.

Because our front end logically splits classes up into these sections when the user is filling out

the form, it also made sense to do the same in the backend.

They all have a first column labeled `id` that is just like the user’s `id`: it’s a primary,

unique, auto incrementing key that can uniquely identify each entry.

Next each of these tables has a `user\_id` which is set up as a foreign key. As explained,

the `id` in the users table is a primary key, so we set the `user\_id` to have a foreign key

constraint to this primary key. This FORCES each entry in these tables to reference a valid user

id.

Next, each class table has an `is\_recommended` column. Currently this is not being

used; however, its a boolean that is supposed to distinguish whether the data in the row is

classes the user has already taken, or classes that are being recommended. Our current

implementation, however, is to always leave it as false (0) to represent just the classes the user

has already taken. Currently, that’s all we need because the recommendation list can always be

regenerated

with their taken classes. The rest of the columns following these 3 are all TINYINT

NOT NULL columns to represent each class respective to each table category. Our reason for

doing this is it forces the developer to have data on EVERY class; no class can be forgotten. It

also makes reading whether a user has taken a class or not very easy simply

query for the

class and if the value is 1, they have; if its 0, they have not.