**Team Roles:**

Steven Guo (yiguo1@umbc.edu)

* Advisor-Side Developer

Stephen Lin (lin14@umbc.edu)

* Advisor-Side Developer

Andrew McLamb (andrew56@umbc.edu)

* Use Cases
* Student-Side Developer
  + Organization/Management

Khadijah Wali ([kwali2@umbc.edu](mailto:kwali2@umbc.edu))

* Student-Side Developer
  + Lead Programmer

Sarah Yake (syake1@umbc.edu)

* Lead Documenter
  + README (with SlickSheet)
  + Presentation
  + Use Cases (Advisor and Student -- earlier)
  + Videos
* Initial database Setup on Student Side
* Student Side Developer
  + Email handling: scrapped because we found out that it would not work on gl

**Location of Project:**

**Project Description:**

The purpose of this project is to make scheduling advising appointments in the CNMS department easier and more streamlined for both advisors and students.

**What’s New?**

Student Side:

Much of the original code was edited to accommodate for the changes we made to the database. For example, we no longer needed students\_advising\_info, and rather linked to the pre-advising worksheet pdf. We also removed the student\_appts table from the database and just added a data field for appt\_id in student\_basic\_info because a student only ever needed to sign up for one appointment at a time, and if a student wanted another, that student would have to cancel the previous appointment beforehand. The given code also did not have complete implementation for searching and applying for appointments, so those had to be implemented from scratch. There was also a view\_appointments button on the homepage, which did not work. We removed that button and just displayed the information of the current appointment on the homepage, because we concluded that a student would not want to load a separate page just to view their appointment information. The pages handling logging in and registering a student were edited to accommodate the changes we made in the databases (for example, adding a preferred name column, as requested in the project description).

The given code also had nothing for allowing the student to edit information, despite having a button to do this, so that functionality was implemented from scratch. There was also a file called processHome.php, which detected which button was clicked and redirected accordingly. This was removed and the buttons were set up to link to the appropriate pages directly. The given code had some errors in checking for correct passwords, etc., so our team added more input validation for ensuring all data was actually entered in the appropriate fields (where required) and ensuring the student logging in or signing up for an account entered a proper email address (a umbc.edu email). On the page where a student creates an account, an ‘other’ option was added to list of selected majors (as requested in the project description), so that, when selected, the student would be redirected to a page that urged them to visit the UMBC advising website. Another feature was added to handle cases where students forget their passwords, allowing them to reset the password by clicking a link. The given code also had very scant CSS, so more styling was added to make the website more visually appealing.

Advisor Side:

All of the given code was working as it was. We did, however, change the structure of the database (ie. adding and removing columns and tables), to more efficiently handle queries and be more in-line with the client’s requests. The given code is lacking many important features, such as printable tables for the meetings, multiple ways to manage appointments, and controlling the data flow of the database (activate/deactivate I/O).

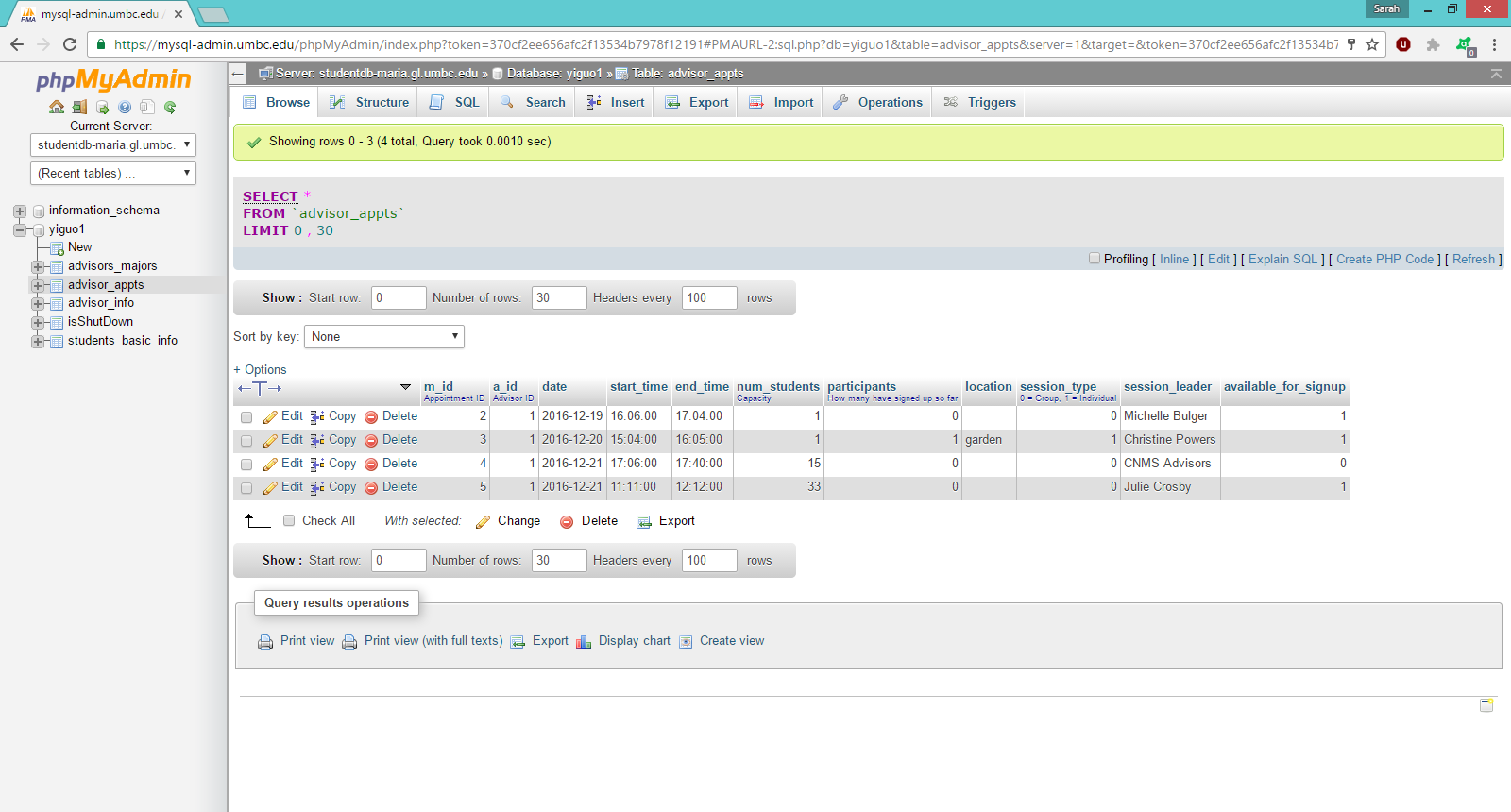
**Improvements:**

Student Side:

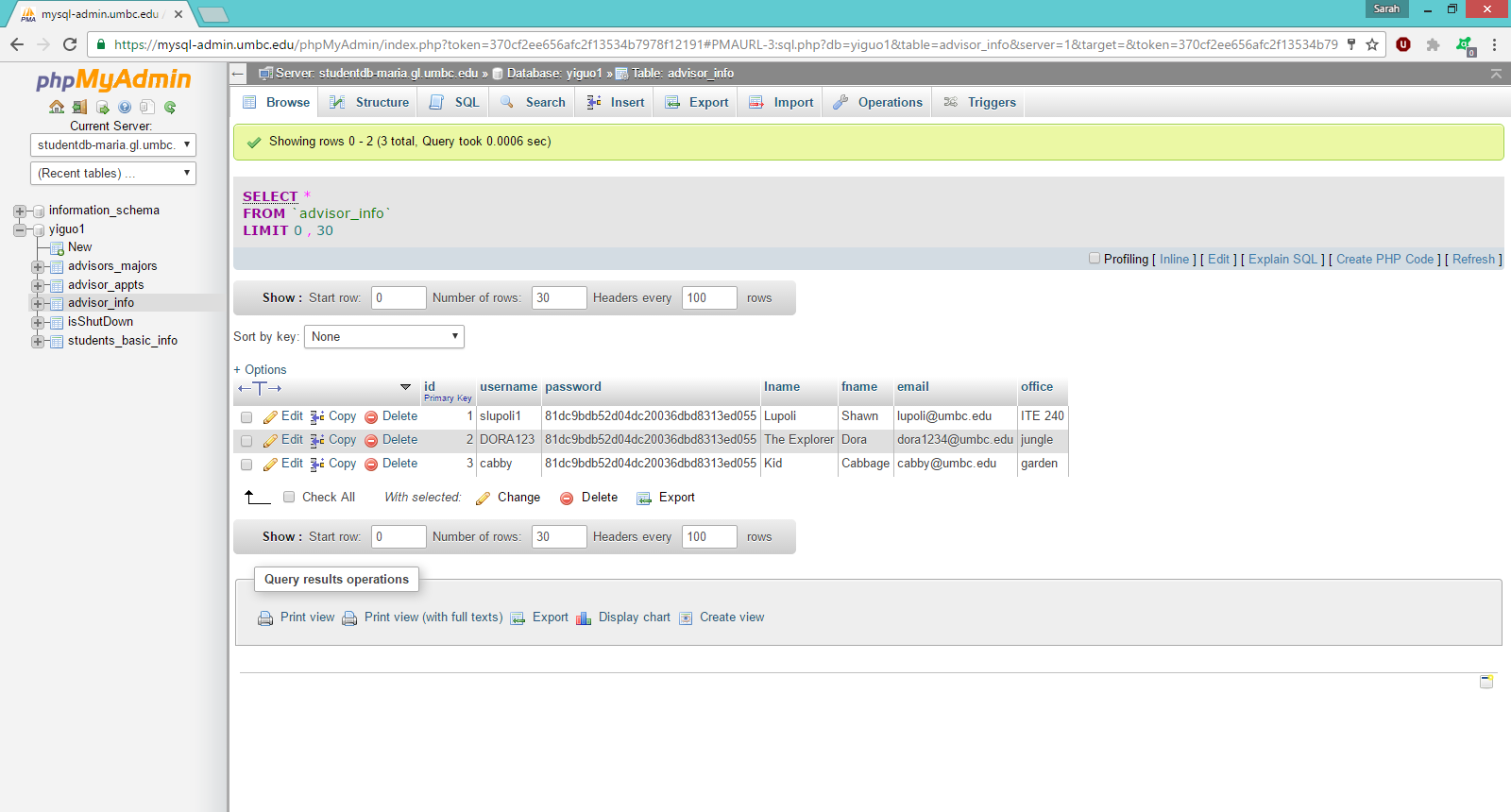
* Searching for appointments
  + Reason: This key functionality was not implemented in code given, so it was implemented from scratch.
* Applying for appointments
  + Reason: This key functionality was not implemented in code given, so it was implemented from scratch.
* Editing Information
  + Reason: This key functionality was not implemented in code given, so it was implemented from scratch.
* Displaying Information for the Students’ Scheduled Appointments on Student Homepage
  + Reason: student convenience
* Password Reset
  + Reason: convenience for advisors and students
* Back buttons on some pages
  + Reason: easier website navigation
* Styles of webpage
  + Reason: To make the website visually appealing
* Removed processHome.php and viewAppointment.php
  + Reason: These files became unnecessary in our implementation.
* Added a preferred name field
  + Reason: To be more in-line with advisor’s requests
* ‘other’ option for major to redirect students to a page urging them to go to the UMBC advising website.
  + Reason: To be more in-line with advisor’s requests
* Preventing a student from cancelling a meeting less than 12 hours before the appointment
  + Reason: This scenario would be very inconvenient for advisors with advisees who cancel meetings last second (possibly repeatedly). (If student absolutely must cancel the meeting, they must email the advisor, who has power to kick any student out of any meeting at any time.)

Advisor Side:

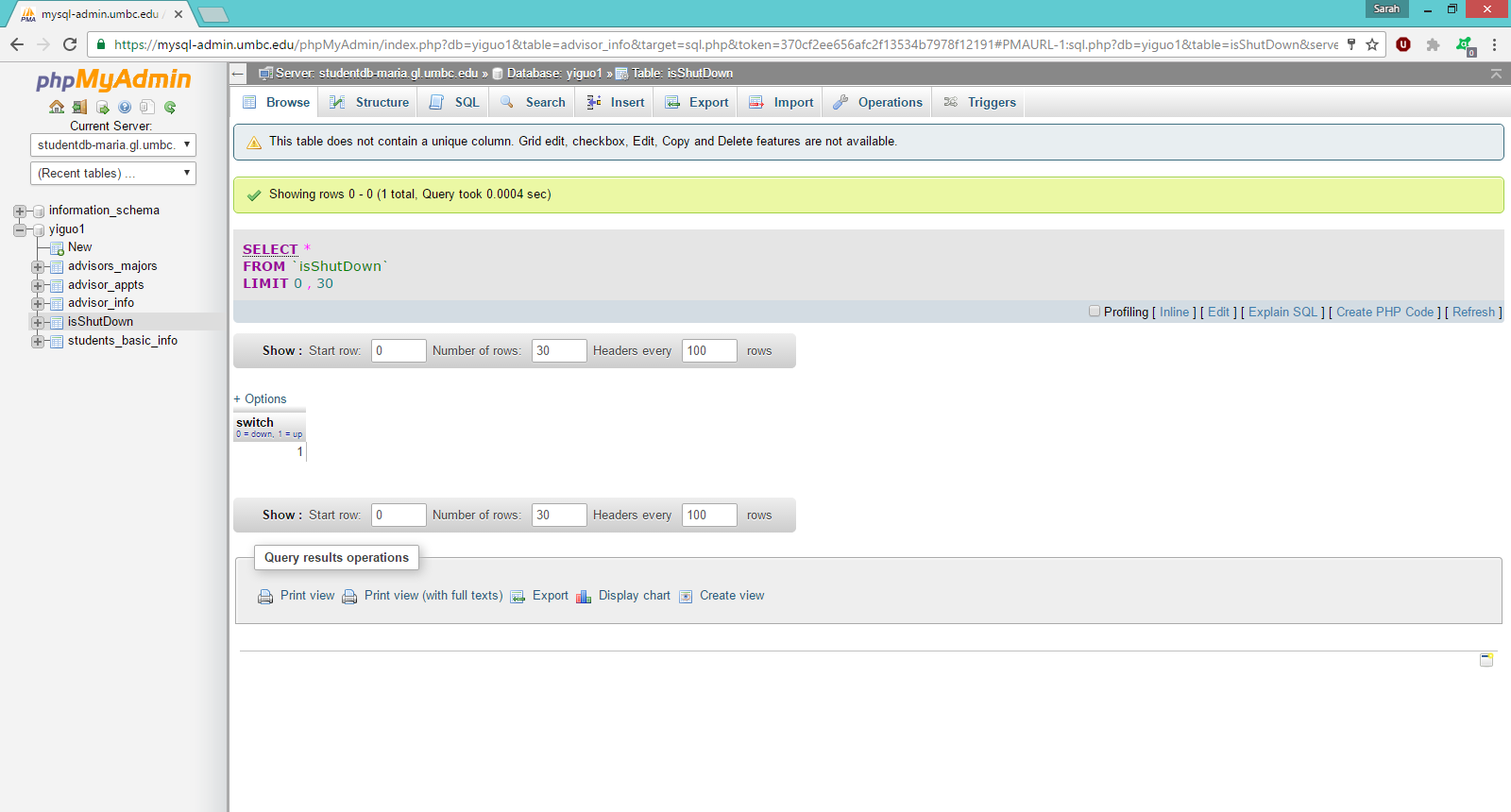
* Styling of webpages
  + Reason: Advisor ease of use, visual appeal
* Adding and removing columns and tables
  + Reason: To more efficiently handle queries and be more in-line with the client’s requests
* Printable tables for the meetings
  + Reason: Advisor Convenience
* Multiple ways to manage appointments
  + Reason: Advisor Convenience
* activate/deactivate buttons to open/close appointment creation
  + Reason: Advisor Convenience,

**Database Setup:**

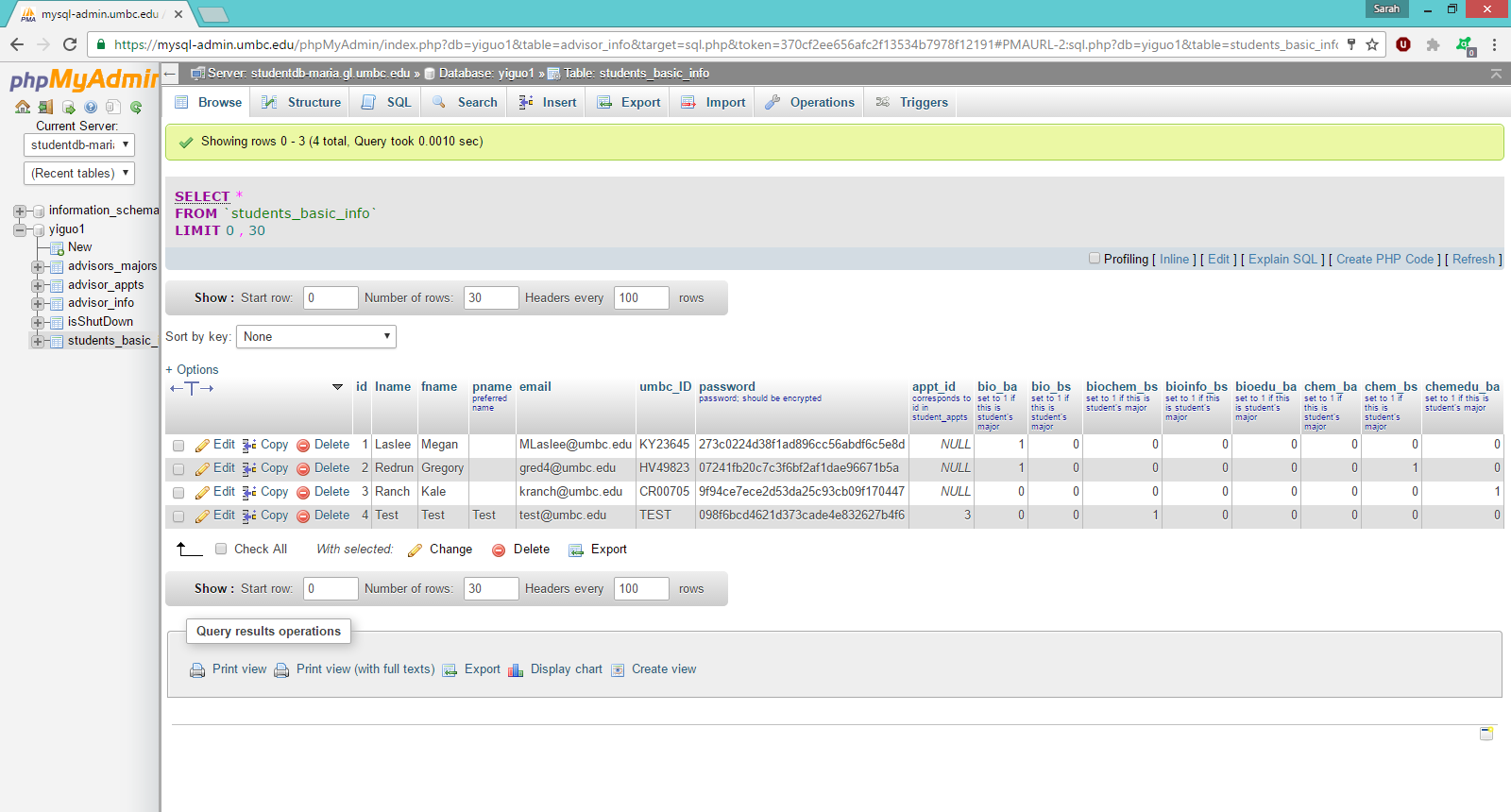
advisor\_appts Table: This table stores the unique appointment identifiers (m\_id), the id of the advisors (a\_id) that corresponds to the primary key in advisor\_info, the date, start\_time, end\_time, session\_leader, and location, availability of the meeting (available\_for\_signup), an indication of an individual/group meeting (session\_type), the number of students scheduled for the meeting (participants), and the maximum capacity students for the meeting (num\_students).



advisor\_info Table: This table stores a unique primary key corresponding to each advisor (id), the advisor’s username (username), md5 sum of the advisor’s password (password), and the advisor’s first name (fname), last name (lname), email address (email), and office (office)



isShutDown Table: Just a simple “switch” to turn enable or disable the scheduling of new appointments.



students\_basic\_info Table: This table stores a unique primary key corresponding to each student (id), the students’ first names (fname), last names (lname), preferred names (pname), email addresses (email), UMBC ids (umbc\_ID), md5 sums of the students’ passwords, and the students’ appointment ids (appt\_id), as well as an indication of the students’ major(s) of the majors specified by CNMS (see table).

**Database Dump:**

If you are using phpMyAdmin, just login and click on the database you want to use. Then click on the “SQL” button on the top of the page and copy/paste the code below into the text-box.

CREATE TABLE IF NOT EXISTS `students\_basic\_info` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`lname` text NOT NULL,

`fname` text NOT NULL,

`pname` text COMMENT 'preferred name',

`email` text NOT NULL,

`umbc\_ID` varchar(7) NOT NULL,

`password` text NOT NULL COMMENT 'password; should be encrypted',

`appt\_id` int(11) DEFAULT NULL COMMENT 'corresponds to id in student\_appts',

`bio\_ba` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`bio\_bs` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`biochem\_bs` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`bioinfo\_bs` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`bioedu\_ba` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`chem\_ba` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`chem\_bs` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

`chemedu\_ba` tinyint(1) NOT NULL DEFAULT '0' COMMENT 'set to 1 if this is student''s major',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=4 ;

CREATE TABLE IF NOT EXISTS `isShutDown` (

`switch` tinyint(1) NOT NULL COMMENT '0 = shut down, 1 = not shut down'

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `isShutDown` (`switch`) Values(1);

CREATE TABLE IF NOT EXISTS `advisor\_info` (

`id` int(7) NOT NULL AUTO\_INCREMENT COMMENT 'Primary Key',

`username` varchar(12) NOT NULL,

`password` text NOT NULL,

`lname` text NOT NULL,

`fname` text NOT NULL,

`email` text NOT NULL,

`office` text NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=4 ;

CREATE TABLE IF NOT EXISTS `advisor\_appts` (

`m\_id` int(10) NOT NULL AUTO\_INCREMENT COMMENT 'Appointment ID',

`a\_id` int(11) NOT NULL COMMENT 'Advisor ID',

`date` date NOT NULL,

`start\_time` time NOT NULL,

`end\_time` time NOT NULL,

`num\_students` int(3) unsigned NOT NULL COMMENT 'Capacity',

`participants` int(10) unsigned NOT NULL COMMENT 'How many have signed up so far',

`location` text,

`session\_type` tinyint(1) NOT NULL COMMENT '0 = Group, 1 = Individual',

`session\_leader` text NOT NULL,

`available\_for\_signup` tinyint(1) NOT NULL,

PRIMARY KEY (`m\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=6 ;

CREATE TABLE IF NOT EXISTS `advisors\_majors` (

`id` int(10) NOT NULL AUTO\_INCREMENT COMMENT 'Primary Key',

`bsci\_BA` tinyint(1) NOT NULL,

`bsci\_BS` tinyint(1) NOT NULL,

`bchem\_BS` tinyint(1) NOT NULL,

`binf\_BS` tinyint(1) NOT NULL,

`bsciEd\_BA` tinyint(1) NOT NULL,

`chem\_BA` tinyint(1) NOT NULL,

`chem\_BS` tinyint(1) NOT NULL,

`chemEd\_BA` tinyint(1) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=12 ;

INSERT INTO `advisors\_majors` (`id`, `bsci\_BA`, `bsci\_BS`, `bchem\_BS`, `binf\_BS`, `bsciEd\_BA`, `chem\_BA`, `chem\_BS`, `chemEd\_BA`) VALUES

(1, 0, 0, 0, 0, 0, 1, 1, 1),

(2, 0, 1, 0, 0, 0, 0, 0, 0),

(3, 0, 1, 0, 0, 0, 0, 0, 0);

**Languages Used:**

* PHP
* JavaScript
* SQL