1.Team Introduction:

* Logan Dudley
  + [logdud1@umbc.edu](mailto:logdud1@umbc.edu)
  + PHP, Authentication, Documentation
* James Gordon
  + [gordon7@umbc.edu](mailto:gordon7@umbc.edu)
  + Error Handling, Input Validation, Documentation
* Phillip Hilliard
  + [phi3@umbc.edu](mailto:phi3@umbc.edu)
  + Database Management, MySQL
* Sara Nabulsi
  + [snabuls@umbc.edu](mailto:snabuls@umbc.edu)
  + Feature Implementation, Alert Handling, Human Relations
* Connor Shaffer
  + [pb10459@umbc.edu](mailto:pb10459@umbc.edu)
  + CSS, Design, Documentation, Feature Implementation

2. Location of Project

The entire project is contained within a folder called Advising-Site-Master, When the user enters this folder in order to gain access to the websites they then need to enter the src folder. The student page can be immediately accessed by entering the student folder. In order to enter the advisor side, the user will have to access the advisor folder and then enter the view folder.

3. Project Description

This website was intended as a means to streamline the advising signup process for students and advisors alike in the College of Natural and Mathematical Sciences (CNMS). The website features a simple appointment creation and editing process, as well as a calendar for easy viewing and managing of appointments on the advisor side. Just as well, features have been provided in the case that unexpected circumstances arise for the advisors, such as editing room assignments, appointment cancellation, and a master password for advisors to be able to edit their colleagues appointments. On the student side, Students can view available appointments from a list of available days and can register for either an individual or group advising session. Students can also cancel and reschedule appointments as they see fit.

4. What was added if given old code?

The entirety of the CSS styling, including color scheme

CSS + Design, Season Over, Calendar, Advisor Password, Pre-advising Wksht, Alerts,

Editing appointments, advisor passwords

Once a student is logged in they have been given the option to fill out the pre-advising form. The form saves their answers in between sessions and is auto-populated once they reenter the page.

Added the ability to open and close the advising season. This makes it so that once the season is closed students are no longer able to log in and make appointments.

Alert features were added to both student and advisor side. Whenever a student logins into their page they are alerted to the fact if an advisor has deleted the appointment they were signed up for. When advisors login, they are alerted to the fact if the master password has been used to login into their account.

5. What was improved upon if given old code?

Authentication when users log in was improved. Before, if a user attempted to log in with incorrect credentials, all text fields were erased and an inaccurate error message popped up. Now, contents of text fields are saved between attempts, and appropriate error messages appear. Passwords were added to the advisor side. Students need to now enter both a valid student ID and email in order to login.

Viewing advisors appointments is much easier. Before, all appointments were in a vertical list, with all students in each appointment also listed vertically. Now, advisors can view their appointments through a calendar, and can optionally expand a list of students in each appointment.

Advisors have now been given the option to set the maximum number of students for a group of students. Originally selecting group advising set the maximum number of students to ten, now the limit can be set between two and forty students.

6. Database and Code Setup

#Advisor Documentation

---

## advisor/views

---

There are four main webpages that are used to navigate the the advisor site.

#### index.php

Note that `index.php` is the starting point for a new advisor since it contains the forms for registering an advisor.

Somethings to note here are that it redirects you to the meeting dashboard if you're already logged in the top.

The are error messages that are stored in the user's session.

The logic for processing the forms is in `advisor/utils/forms/registerAdvisor.php`

#### login.php

On the client side it contains a single form that lets the user input their data.

It also processes the POST data on the top and verifies if the user's email exists and is valid.

#### homepage.php

The main web dashboard for the advisor.

Contains forms for creating new meetings, and deleting meetings.

Also has various SQL queries to search for information on students.

#### logout.php

There is very little logic here, the main thing is that it clears the session variables and redirects the user to the login `login.php`

## advisor/utils

---

#### dbconfig.php

Holds a list of static variables that allows you to access the database

#### forms/createMeeting.php

Contains logic for creating meetings. Also stores errors into session variables should the inputs be incorrect. Uses raw SQL queries to process inputted values into database.

#### forms/deleteMeeting.php

Where logic for checking if a meeting exists and deleting such meeting is placed. Also stores errors into session variables should there be any errors. Uses raw SQL queries to process inputted values into database.

#### forms/registerAdvisor.php

Has logic for creating an advisor . Also stores errors into session variables should the inputs be incorrect. Uses raw SQL queries to process inputted values into database.

STUDENT DOCUMENTATION

Student directory depends on CommonMethods.php:

CommonMethods.php must be altered on these lines to be used with your database:

Line 11: $rs = $this 🡪connect(“DATABASE PASSWORD”);

Line 19: $conn = @mysql\_connect( <DATABASE URL>, <USERNAME>, <PASSWORD>)

Commented lines 22 and 29 can be used for debugging

Student directory includes:

index.php

login.php

homePage.php

viewMeeting.php

logout.php

cancelMeeting.php

chooseGroupMeeting.php

chooseIndividualMeeting.php

chooseMeeting.php

deleteMeeting.php

meetingChosen.php

index.php

This is the registration page for students. It uses a form to accept basic data to be stored and used in your database.

Regex is used to validate email address input which reports an error when incorrect.

Empty fields are handled with error messages.

Email is used to validate if a student record exists in the database table, so if preexisting emails are entered an error is thrown and the information is not entered in to table.

After sql query is executed on line 122, the user is redirected to login.php

Login.php

This is the login in portal for students. It uses a form to accept an email that is compared to preexisting data in Student table.

If the text field is left unfilled or an unregistered email is entered an error is thrown and handled.

After a legitimate email is entered four session variables are defined using data from the Student table; they include a boolean to signify that the user is logged in, and three strings for email, studentID and major which are used later.

The user is redirected to homePage.php.

homePage.php

This portal is used to handle appointment processes.

Using provided links the users can navigate to:

viewMeeting.php(“View Scheduled Appointment”)

cancelMeeting.php(“Cancel Advising Appointment”)

chooseMeeting.php(“Schedule Advising Appointment”)

The user can also logout and is redirected to login.php after passing through logout.php

viewMeeting.php

This portal uses predefined session variables to access the StudentMeeting and Meeting tables to print the students currently scheduled appointment.

If the StudentMeeting query on line 15 when executed on line 18 reports now rows with the StudentID of the student currently logged in message is printed letting them know they have no scheduled appointments

logout.php

This small file destroys the current session and redirects the user to the login.php page where they can then login or redirect to index.php

cancelMeeting.php

cancelMeeting.php is responsible for allowing a user to cancel their meetings. If the student are already registered for a meeting, then it will display a confirmation to check whether they want to go through with the cancelation. If they select yes then they will be redirected to deleteMeeting.php and the meeting will be deleted. If they select “NO” then they will simply be redirected to the homepage, homepage.php.

In the event that the user does not currently have a meeting selected, cancelMeeting.php displays a notice and allows the user to go back to the home page.

chooseGroupMeeting.php

chooseGroupMeeting.php is like chooseIndividualMeeting.php, it lists all the available group meetings and allows the user to select one using a radio button. It then proceeds to add the appointment to the student Meeting table in the database and increments the number of students registered for the meeting in the Meeting table.

In the event there are no group meetings available or the user is already registered for a meeting the page will display a notice and will have a link for the user to go to the homepage.

chooseIndividualMeeting.php

chooseIndividualMeeting.php is similar to chooseGroupMeeting.php, it lists all the available individual meetings and allows the user to select one using a radio button. It then proceeds to add the appointment to the student Meeting table in the database and increments the number of students registered for the meeting in the Meeting table.

In the event, there are no individual meetings available or the user is already registered for a meeting the page will display a notice and will have a link for the user to go to the homepage.

chooseMeeting.php

This page simply consists of two buttons that allows the user to choose between registering for a group and an individual meeting.

deleteMeeting.php

This page runs the php and sql the allows a student to cancel their registration.

For both types of meetings the appointment is deleted from the studentMeeting table. Then, in the Meetings table the quantity of students registered for that meeting is decremented.

After the meeting is fully deleted the user is redirected to the home page.

meetingChosen.php

This page simply consists of the warning message shown when a user attempts to register for a advising meeting while already registered for a meeting.

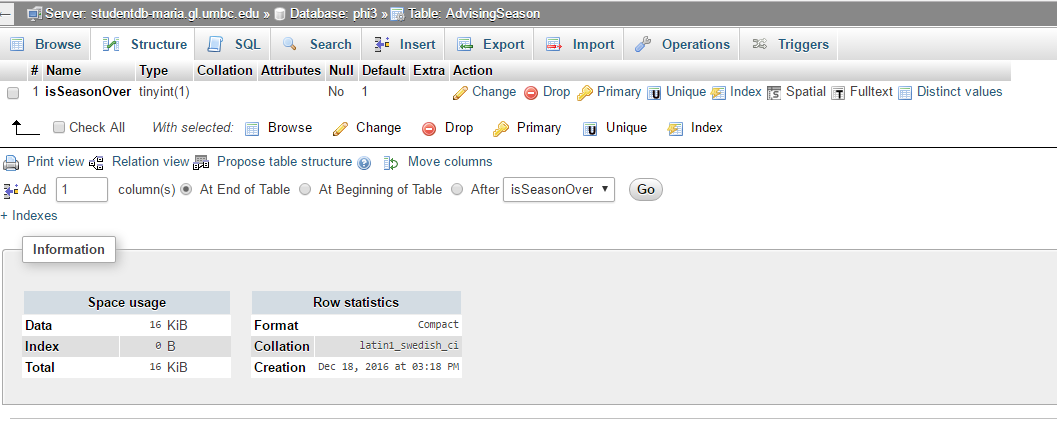
There is a link that allows the user to go back to the home page.

Database:

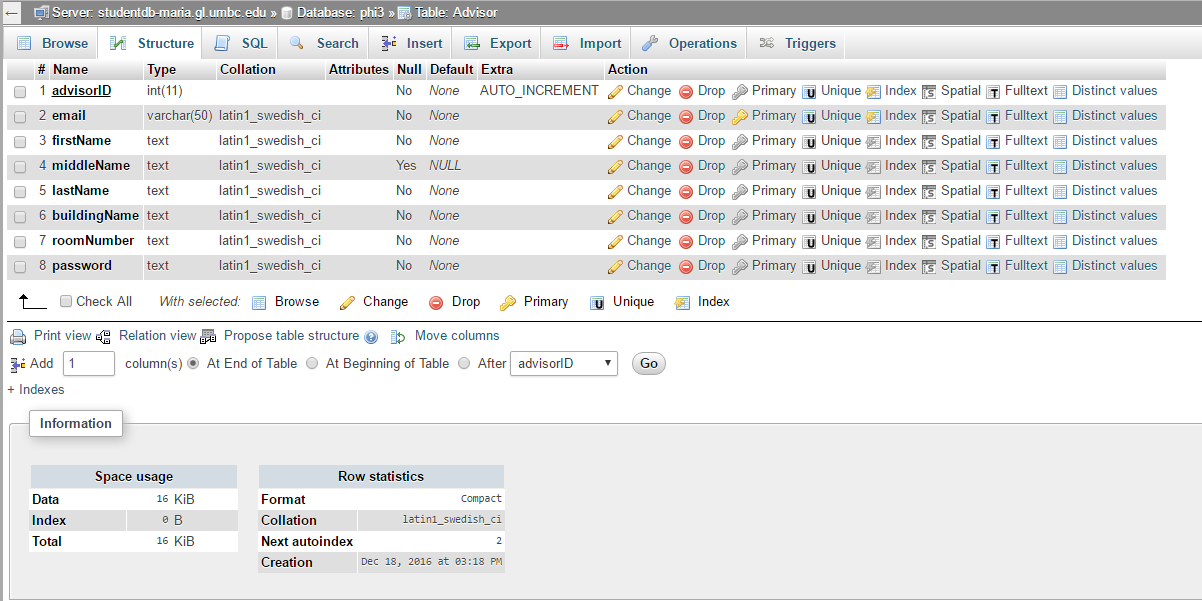
The database can be set up by running dropDB.php followed by setUpDB.php. PHP code for creating the tables in the database can be found in setUpDB.php.

Tables:

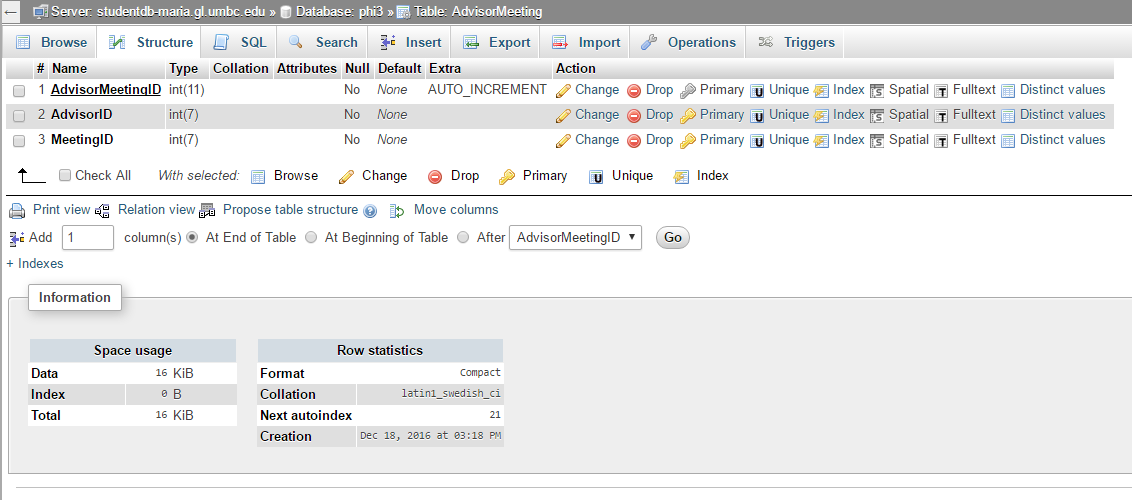
Advising Season



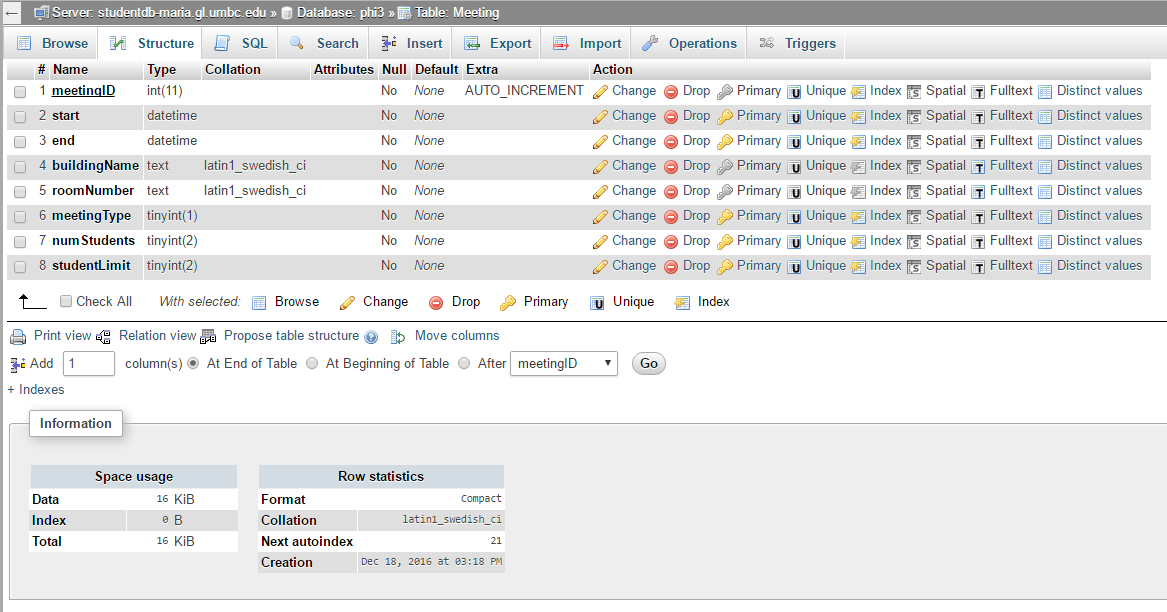
Advisor



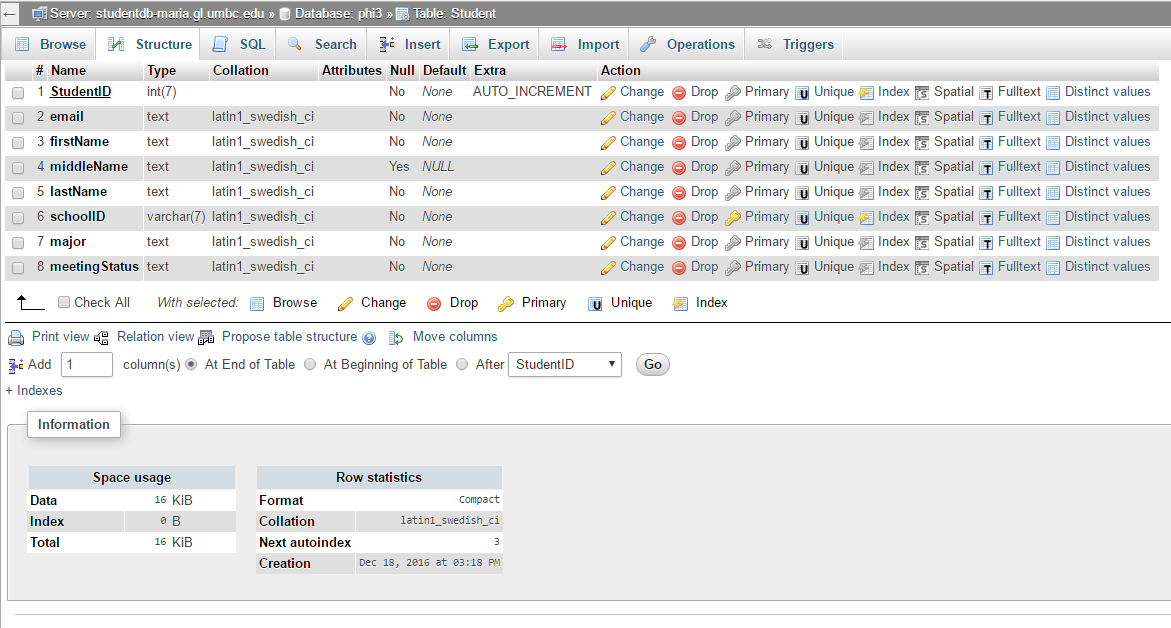
AdvisorMeeting



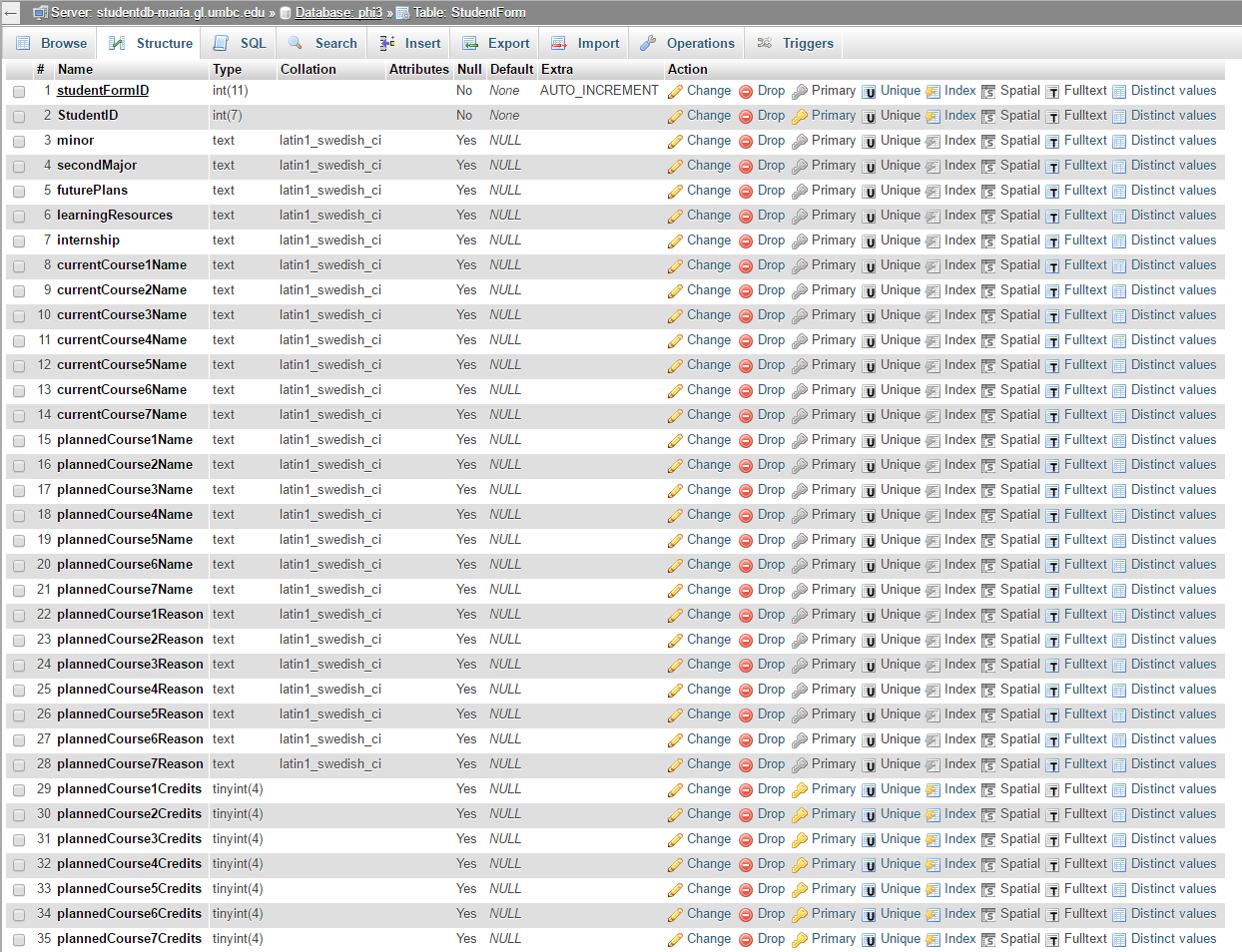
Meeting



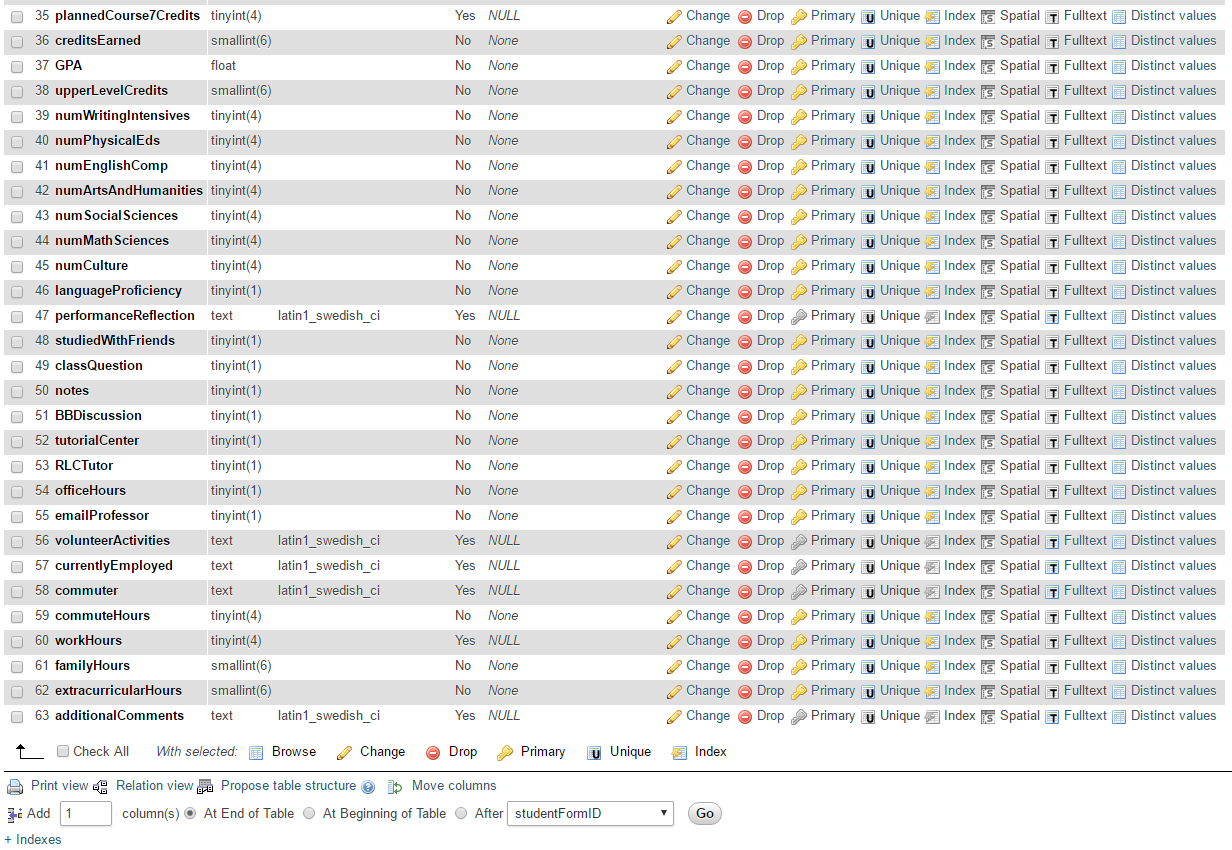
Student



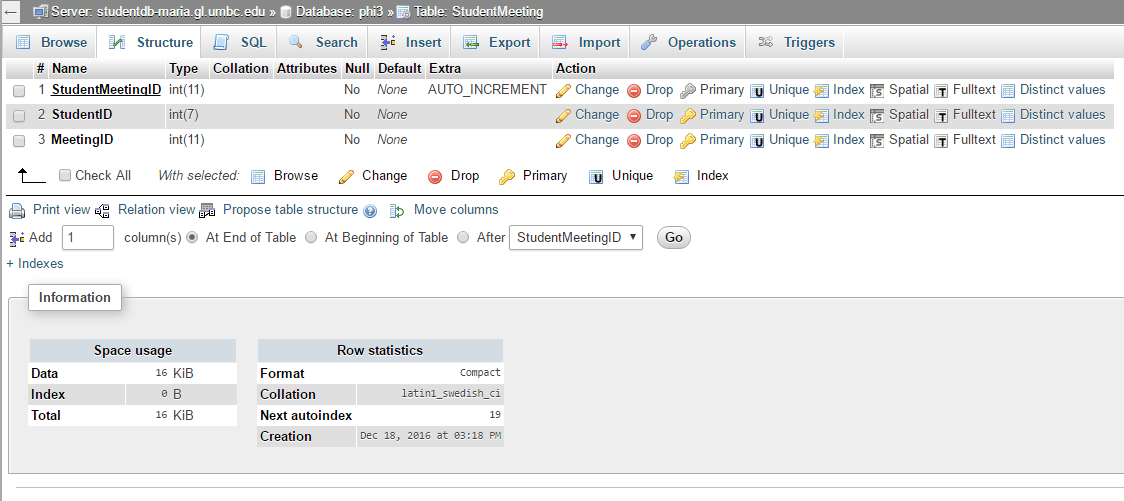
StudentForm



StudentForm (continued)



StudentMeeting



Comments:

Videos on how to use the service as provided can be found at:

<https://www.youtube.com/playlist?list=PLCzDYNvPl8px07Wu9PNevwMN3bOPmCtmx>

<https://www.youtube.com/playlist?list=PLCzDYNvPl8pyRO0VPLyZSWeQj8slngRGL>

AdvisorMeetings and StudentMeetings are junction tables between Advisor and Meeting and Student and Meeting respectively. The only things they contain are the foreign keys from the tables they join and an identifier ID.

#Advisor Documentation

---

## advisor/views

---

There are four main webpages that are used to navigate the advisor site.

#### index.php

Note that `index.php` is the starting point for a new advisor since it contains the forms for registering an advisor.

Somethings to note here are that it redirects you to the meeting dashboard if you're already logged in the top.

The are error messages that are stored in the user's session.

The logic for processing the forms is in `advisor/utils/forms/registerAdvisor.php`

#### login.php

On the client side it contains a single form that lets the user input their data.

It also processes the POST data on the top and verifies if the user's email exists and is valid.

#### homepage.php

The main web dashboard for the advisor.

Contains forms for creating new meetings, and deleting meetings.

Also has various SQL queries to search for information on students.

#### logout.php

There is very little logic here, the main thing is that it clears the session variables and redirects the user to the login `login.php`

## advisor/utils

---

#### dbconfig.php

Holds a list of static variables that allows you to access the database

#### forms/createMeeting.php

Contains logic for creating meetings. Also stores errors into session variables should the inputs be incorrect. Uses raw SQL queries to process inputted values into database.

#### forms/deleteMeeting.php

Where logic for checking if a meeting exists and deleting such meeting is placed. Also stores errors into session variables should there be any errors. Uses raw SQL queries to process inputted values into database.

#### forms/registerAdvisor.php

Has logic for creating an advisor . Also stores errors into session variables should the inputs be incorrect. Uses raw SQL queries to process inputted values into database.

-- phpMyAdmin SQL Dump

-- version 4.0.10.17

-- https://www.phpmyadmin.net

--

-- Host: studentdb-maria.gl.umbc.edu

-- Generation Time: Dec 19, 2016 at 05:08 AM

-- Server version: 10.1.19-MariaDB

-- PHP Version: 5.4.44

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

--

-- Database: `phi3`

--

-- --------------------------------------------------------

--

-- Table structure for table `AdvisingSeason`

--

CREATE TABLE IF NOT EXISTS `AdvisingSeason` (

`isSeasonOver` tinyint(1) NOT NULL DEFAULT '1'

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `Advisor`

--

CREATE TABLE IF NOT EXISTS `Advisor` (

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

`email` varchar(50) NOT NULL,

`firstName` text NOT NULL,

`middleName` text,

`lastName` text NOT NULL,

`buildingName` text NOT NULL,

`roomNumber` text NOT NULL,

`password` text NOT NULL,

PRIMARY KEY (`advisorID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=2 ;

-- --------------------------------------------------------

--

-- Table structure for table `AdvisorMeeting`

--

CREATE TABLE IF NOT EXISTS `AdvisorMeeting` (

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

`AdvisorID` int(7) NOT NULL,

`MeetingID` int(7) NOT NULL,

PRIMARY KEY (`AdvisorMeetingID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=11 ;

--

-- Dumping data for table `AdvisorMeeting`

--

-- --------------------------------------------------------

--

-- Table structure for table `Meeting`

--

CREATE TABLE IF NOT EXISTS `Meeting` (

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

`start` datetime NOT NULL,

`end` datetime NOT NULL,

`buildingName` text NOT NULL,

`roomNumber` text NOT NULL,

`meetingType` tinyint(1) NOT NULL,

`numStudents` tinyint(2) NOT NULL,

`studentLimit` tinyint(2) NOT NULL,

PRIMARY KEY (`meetingID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=11 ;

-- --------------------------------------------------------

--

-- Table structure for table `Student`

--

CREATE TABLE IF NOT EXISTS `Student` (

`StudentID` int(7) NOT NULL AUTO\_INCREMENT,

`email` text NOT NULL,

`firstName` text NOT NULL,

`middleName` text,

`lastName` text NOT NULL,

`schoolID` varchar(7) NOT NULL,

`major` text NOT NULL,

`meetingStatus` text NOT NULL,

PRIMARY KEY (`StudentID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=3 ;

-- --------------------------------------------------------

--

-- Table structure for table `StudentForm`

--

CREATE TABLE IF NOT EXISTS `StudentForm` (

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

`StudentID` int(7) NOT NULL,

`minor` text,

`secondMajor` text,

`futurePlans` text,

`learningResources` text,

`internship` text,

`currentCourse1Name` text,

`currentCourse2Name` text,

`currentCourse3Name` text,

`currentCourse4Name` text,

`currentCourse5Name` text,

`currentCourse6Name` text,

`currentCourse7Name` text,

`plannedCourse1Name` text,

`plannedCourse2Name` text,

`plannedCourse3Name` text,

`plannedCourse4Name` text,

`plannedCourse5Name` text,

`plannedCourse6Name` text,

`plannedCourse7Name` text,

`plannedCourse1Reason` text,

`plannedCourse2Reason` text,

`plannedCourse3Reason` text,

`plannedCourse4Reason` text,

`plannedCourse5Reason` text,

`plannedCourse6Reason` text,

`plannedCourse7Reason` text,

`plannedCourse1Credits` tinyint(4) DEFAULT NULL,

`plannedCourse2Credits` tinyint(4) DEFAULT NULL,

`plannedCourse3Credits` tinyint(4) DEFAULT NULL,

`plannedCourse4Credits` tinyint(4) DEFAULT NULL,

`plannedCourse5Credits` tinyint(4) DEFAULT NULL,

`plannedCourse6Credits` tinyint(4) DEFAULT NULL,

`plannedCourse7Credits` tinyint(4) DEFAULT NULL,

`creditsEarned` smallint(6) NOT NULL,

`GPA` float NOT NULL,

`upperLevelCredits` smallint(6) NOT NULL,

`numWritingIntensives` tinyint(4) NOT NULL,

`numPhysicalEds` tinyint(4) NOT NULL,

`numEnglishComp` tinyint(4) NOT NULL,

`numArtsAndHumanities` tinyint(4) NOT NULL,

`numSocialSciences` tinyint(4) NOT NULL,

`numMathSciences` tinyint(4) NOT NULL,

`numCulture` tinyint(4) NOT NULL,

`languageProficiency` tinyint(1) NOT NULL,

`performanceReflection` text,

`studiedWithFriends` tinyint(1) NOT NULL,

`classQuestion` tinyint(1) NOT NULL,

`notes` tinyint(1) NOT NULL,

`BBDiscussion` tinyint(1) NOT NULL,

`tutorialCenter` tinyint(1) NOT NULL,

`RLCTutor` tinyint(1) NOT NULL,

`officeHours` tinyint(1) NOT NULL,

`emailProfessor` tinyint(1) NOT NULL,

`volunteerActivities` text,

`currentlyEmployed` text,

`commuter` text,

`commuteHours` tinyint(4) NOT NULL,

`workHours` tinyint(4) DEFAULT NULL,

`familyHours` smallint(6) NOT NULL,

`extracurricularHours` smallint(6) NOT NULL,

`additionalComments` text,

PRIMARY KEY (`studentFormID`),

UNIQUE KEY `StudentID` (`StudentID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=5 ;

-- --------------------------------------------------------

--

-- Table structure for table `StudentMeeting`

--

CREATE TABLE IF NOT EXISTS `StudentMeeting` (

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

`StudentID` int(7) NOT NULL,

`MeetingID` int(11) NOT NULL,

PRIMARY KEY (`StudentMeetingID`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=11 ;

7. Languages Used

We used PHP, JavaScript, and MySQL to complete this project. PHP was used on almost every page, JavaScript was used for authentication and validation, and MySQL let us interact with the database.