UNIVERSITY DATABASE

PREPARED FOR

Professor Kahlid Afeez

Data Management Systems

PREPARED BY

Michelle Cheng 100696572
Evans Mosomi 100719552
Clarissa Branje 100716458
Martins Babajide 100709716

TABLE OF CONTENTS

EXECUTIVE SUMMARY

As society enters a technological revolution, one of the major challenges is the storage and access of information. Think about an educational system, what data need to be stored? Where will it be held?

With thousands of classes, professors, staff members, courses and educational pathways available, it is easy to lose track of this data. This is where a university database can be the ultimate solution. This following report will outline a framework for a university database which would be essential for any educational institution.

The 3 key features to this project include:

- University Scheme run in MySQL Workbench
- GUI run in a local Wamp server host
- Development of GUI with HTML,CSS and PHP

The following report will provide an overview of the project in its development, analysis and development stages. Sections included in this report include:

- Project Overview
- Industry and Market Analysis
- Database Design Diagrams
- Database Design Application
- Back-end Application
- Front-end Application
- Technical obstacles
- Milestones and Reporting
- Further Advancements

GitHub Link:

1. Project Overview

This project is based around a university database which allows staff and students to access information that is relevant to them. Information stored within the database includes student data, staff data, grade, courses, department, prerequisite, section numbers and more. Furthermore, various queries can be run within the database which allows for specific information to be pulled out from the data stored.

This project also utilizes various languages such as html, php, json and Rest API to implement a user friendly interface hosted by a local server. This allows for easy access to the information for all sides of the application.

2. Industry and Market Analysis

Prior to the development of the database we did a mock study to replicate the needs for a university database system. These results were collected through a google form and the results can be seen below as Table 1-4.

Table 1. Type of Staff

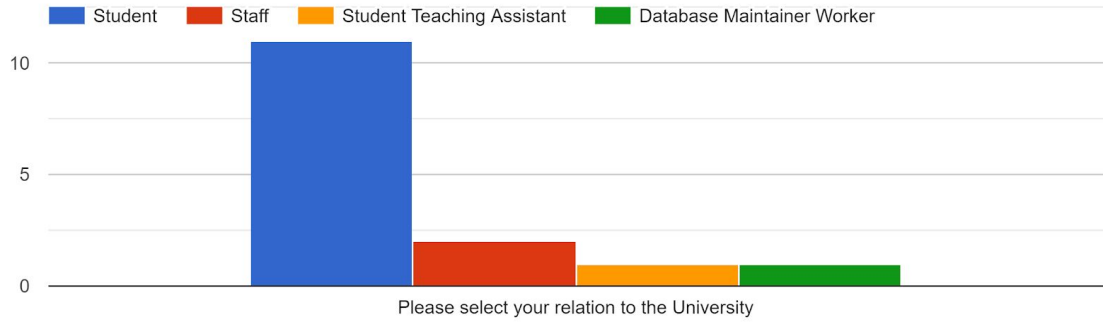


Table 2. Rate Ability of the System

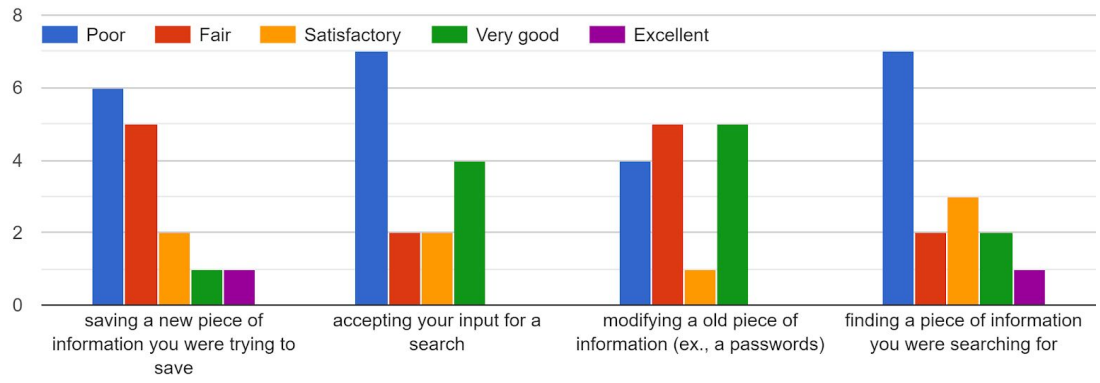


Table 3. Skill and Responsiveness of Database

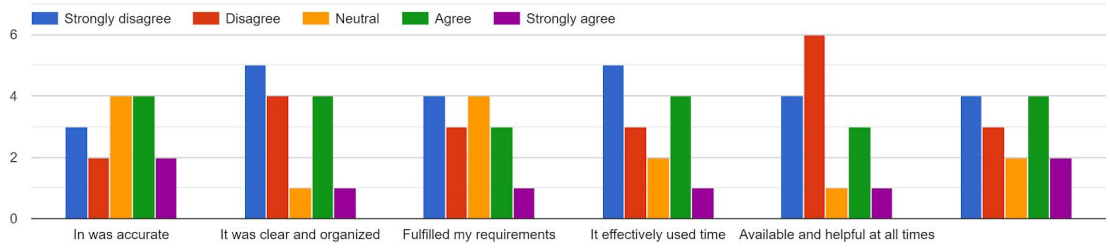
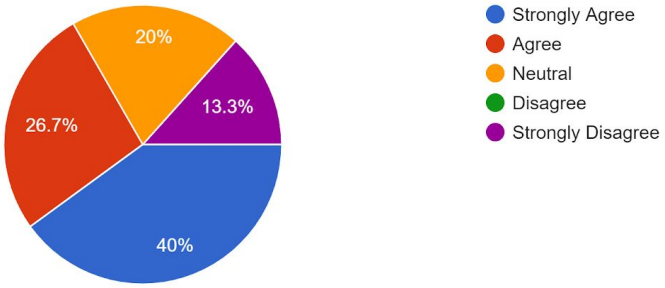


Table 4. The university system needs an upgrade to improve performance



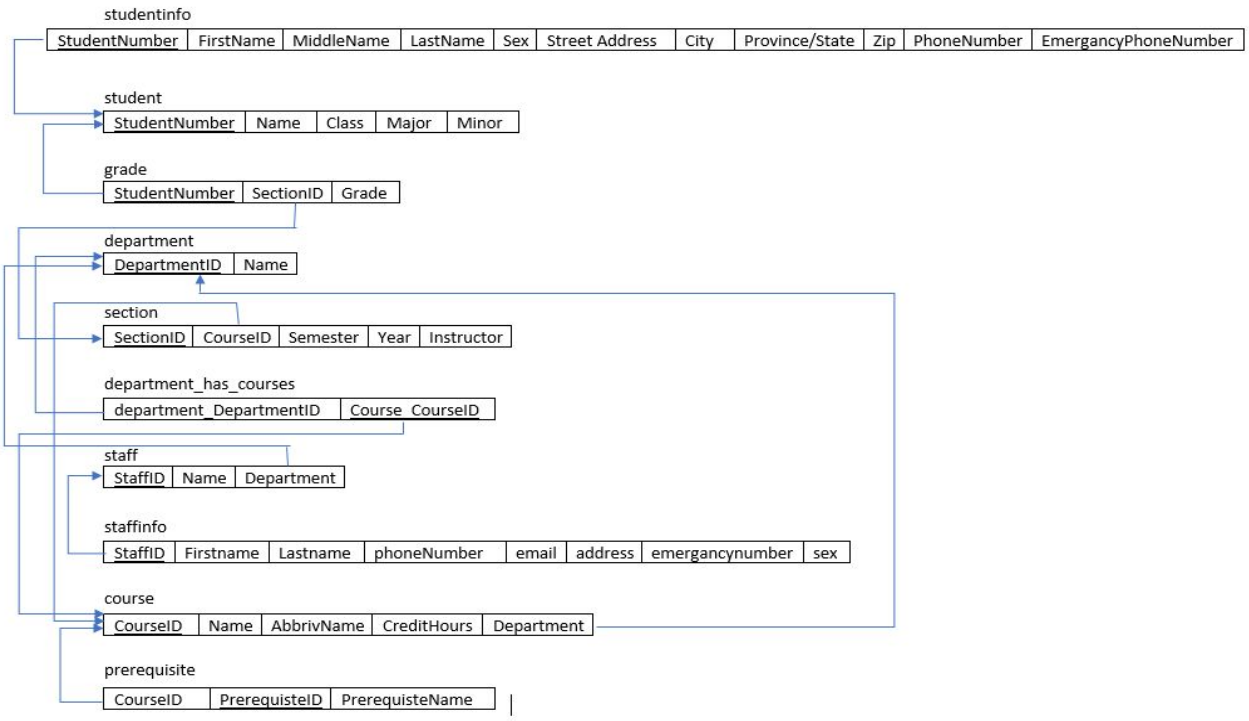
3. Database Design Diagrams

In the initial stages of our database design, we had to utilize the relational schema and ER diagram prior to creation of database in SQL workbench. These documents can be found below.

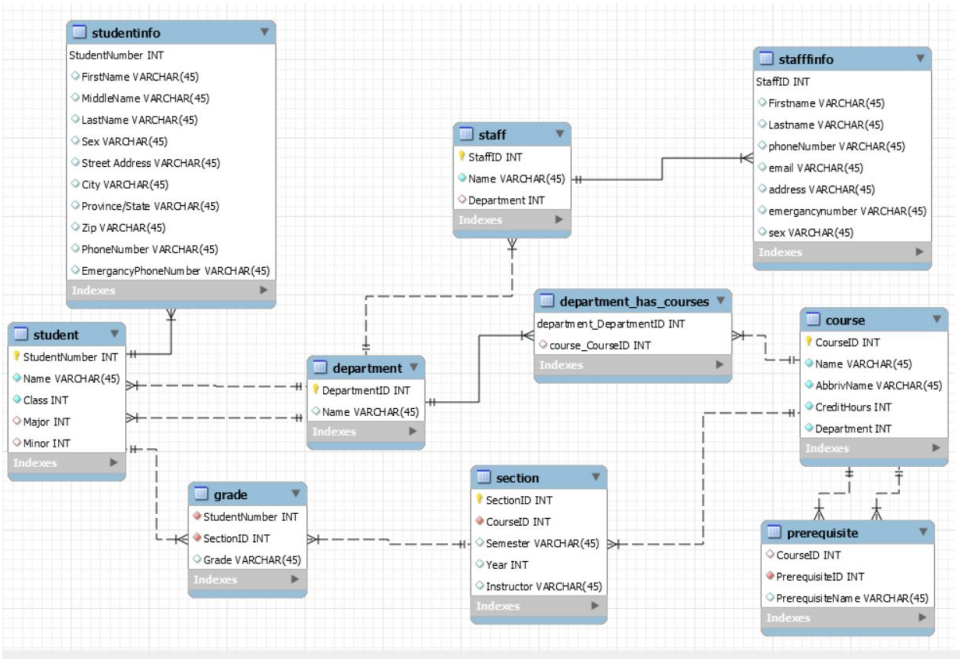
The relational scheme helped us plan out the tables we found to be necessary for the database. In addition it outlined the primary and foriegn key connections, which made creation of the database easier.

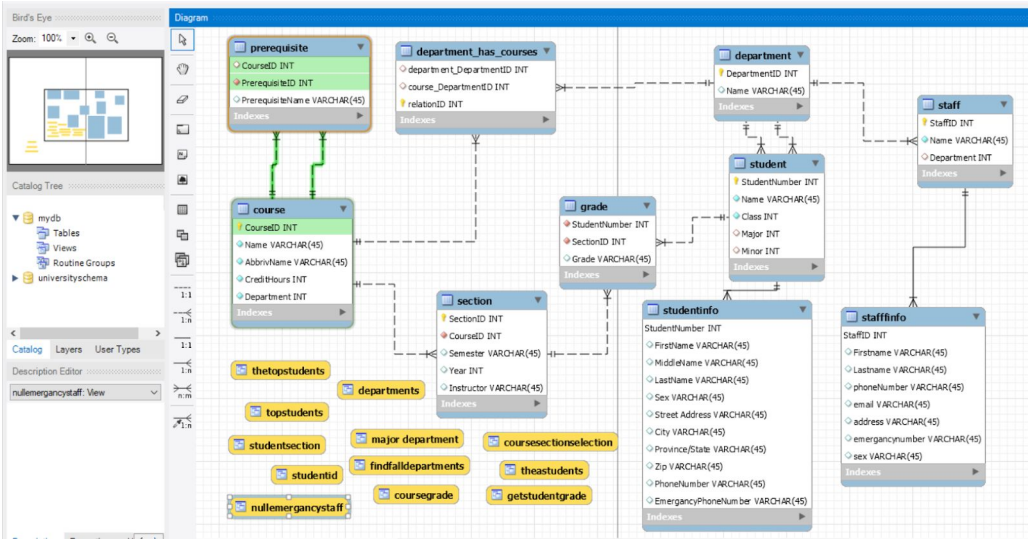
ER diagram was created in mySQL and helped to see the overall structure of the database. With it, we are able to confirm that correct connections are made and see the bigger picture of the entire application.

Relational Schema



ER Diagram


studentinfo
StudentNumber INT
FirstName VARCHAR(45)
MiddleName VARCHAR(45)
LastName VARCHAR(45)
Sex VARCHAR(45)
Street Address VARCHAR(45)
City VARCHAR(45)
Province/State VARCHAR(45)
Zip VARCHAR(45)
PhoneNumber VARCHAR(45)
EmergancyPhoneNumber VARCHAR(45)
Indexes
staff
StaffID INT
Name VARCHAR(45)
Department INT
Indexes
stafffinfo
StaffID INT
Firstname VARCHAR(45)
Lastname VARCHAR(45)
phoneNumber VARCHAR(45)
email VARCHAR(45)
address VARCHAR(45)
emergancynumber VARCHAR(45)
sex VARCHAR(45)
Indexes
department_has_courses
department_DepartmentID INT
course_CourseID INT
Indexes
course
CourseID INT
Name VARCHAR(45)
AbbrivName VARCHAR(45)
CreditHours INT
Department INT
Indexes
student
StudentNumber INT
Name VARCHAR(45)
Class INT
Major INT
Minor INT
Indexes
department
DepartmentID INT
Name VARCHAR(45)
Indexes
grade
StudentNumber INT
SectionID INT
Grade VARCHAR(45)
Indexes
section
SectionID INT
CourseID INT
Semester VARCHAR(45)
Year INT
Instructor VARCHAR(45)
Indexes
prerequisite
CourseID INT
PrerequisiteID INT
PrerequisiteName VARCHAR(45)
Indexes


Bird's Eye
Zoom: 100%
Catalog Tree
mydb
Tables
Views
Routine Groups
universityschema
Catalog
Layers
User Types
Description Editor
nullemergancystaff: View
Diagram
prerequisite
CourseID INT
PrerequisiteID INT
PrerequisiteName VARCHAR(45)
Indexes
department_has_courses
department_DepartmentID INT
course_DepartmentID INT
relationID INT
Indexes
department
DepartmentID INT
Name VARCHAR(45)
Indexes
staff
StaffID INT
Name VARCHAR(45)
Department INT
Indexes
student
StudentNumber INT
Name VARCHAR(45)
Class INT
Major INT
Minor INT
Indexes
course
CourseID INT
Name VARCHAR(45)
AbbrivName VARCHAR(45)
CreditHours INT
Department INT
Indexes
grade
StudentNumber INT
SectionID INT
Grade VARCHAR(45)
Indexes
section
SectionID INT
CourseID INT
Semester VARCHAR(45)
Year INT
Instructor VARCHAR(45)
Indexes
studentinfo
StudentNumber INT
FirstName VARCHAR(45)
MiddleName VARCHAR(45)
LastName VARCHAR(45)
Sex VARCHAR(45)
Street Address VARCHAR(45)
City VARCHAR(45)
Province/State VARCHAR(45)
Zip VARCHAR(45)
PhoneNumber VARCHAR(45)
EmergancyPhoneNumber VARCHAR(45)
Indexes
stafffinfo
StaffID INT
Firstname VARCHAR(45)
Lastname VARCHAR(45)
phoneNumber VARCHAR(45)
email VARCHAR(45)
address VARCHAR(45)
emergancynumber VARCHAR(45)
sex VARCHAR(45)
Indexes
thetopstudents
departments
topstudents
major department
coursesectionselection
studentsection
findfalldepartments
theastudents
studentid
coursegrade
getstudentgrade
nullemergancystaff

4. Database Design Application

The database was heavily influenced by the course as there were some examples of a university database used to example some certain concepts. This helped establish how certain relationships work between another and how to map out a proper design as well as develop complex queries the project can utilize. Some sample queries are utilised as followed:

Retrieving a Grade for a Student

```
Create VIEW `studentgrade` as
 SELECT DISTINCT student.Name, student.StudentNumber, grade.Grade, course.Name as Course
FROM student Join department ON student.Major = department.DepartmentID
Join department_has_courses On department.DepartmentID = department_has_courses.department_DepartmentID
Join course On department_has_courses.course_DepartmentID = course.CourseID
Join section On course.CourseID = section.CourseID
Join grade On section.SectionID = grade.SectionID
Where student.StudentNumber = '100716458';
```

```
1 •  SELECT * FROM universityschema.grade;
```

StudentNumber	SectionID	Grade
100712727	5	A
100716458	1	A
100716459	3	B
100716460	7	B
100717212	5	C+
100727272	8	B+

```
1 •  SELECT * FROM universityschema.studentgrade;
```

Name	StudentNumber	Grade	Course
Clarissa Branje	100716458	B+	Algorithm

Retrieving Prerequisite information

Upon construction for designing the database. Numerous tables were used to help keep information private and easily accessible. Having many relations between the tables deems the ability to create queries for variating instances. For example, one table designated for the list of staff and another for the list of students. This allowed the Id's of each type of user to be separated and organised. A view with a Join can be used to bring the tables together for the login instance where values from both checked for a match.

Login Join Query

```
SELECT * FROM universityschema.staff;
```

Result Grid | Filter Rows: | Edit:

StaffID	Name	Department
1	King	8
2	Anderson	8
3	James	7
4	Peter	8
5	Simone	1
6	John	5
7	Brad	4
8	Karen	5
9	Judy	2
10	Jane	3
NULL	NULL	NULL

```
SELECT * FROM universityschema.student;
```

Result Grid | Filter Rows: | Edit: | Export:

StudentNumber	Name	Class	Major	Minor
100716458	Clarissa Branje	3	1	2
100716459	Ned Nilson	2	2	7
100712727	Grace Ly	3	3	4
100716460	Greg Gregory	2	4	NULL
100717212	John Johnsten	1	3	NULL
100727272	Lisa Lucy	4	5	8
NULL	NULL	NULL	NULL	NULL

```
SELECT staff.StaffID FROM staff
        Union
        SELECT student.StudentNumber From student
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

StaffID
3
4
5
6
7
8
9
10
100712727
100716458
100716459
100716460
100717212
100727272

The database was initially planned with MySQL Workbench and then implemented with php myAdmin. The software was selected as it was easy to import and alter the database to the hosted server website.

There were more complex queries planned to be utilised to our database, but our demo website ran out of time to implement the admin stage for maintenance checks.

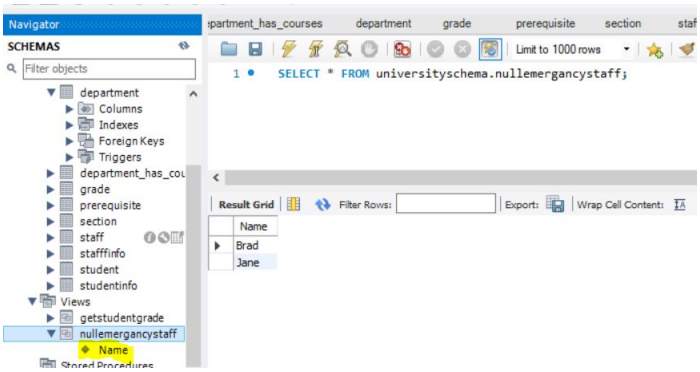
Admin Views

```
1 • SELECT * FROM universityschema.stafffinfo;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

StaffID	Firstname	Lastname	phoneNumber	email	address	emergancynum	sex
1	Kevin	King	7058989898	kk@gmail.c...	221B Baker St.	7059898989	Male
2	Andy	Anderson	7051212121	aa@gmail....	704 Hauser St.	7051111111	Male
3	Jamie	James	7051313131	JJ@gmail.c...	Apartment 5A	705131333	Male
4	Pete	Peter	7051414141	PetPet@g...	124 Conch St.	7054141414	Male
5	Sinn	Simone	7051515151	SinSim@gm...	322 Maple St.	7055151515	Male
6	Washi	John	7051616161	JW@gmail....	485 Maple Drive	7056161616	Male
7	Kayden	Brad	7051717171	KB@gmail....	698 Candlewood Lane		Male
8	Erica	Karen	7051818181	EK@gmail....	607 S. Maple St.	7058181818	Female
9	Mille	Judy	7051919191	MJ@gmail....	79 Wistful Vista	7059191919	Female
10	Sasha	Jane	7052121212	JS@gmail.c...	4222 Clinton Way		Female
11	Mary-Ann	Jill	7053131313	JAM@gmail...	4 Privet Drive	7052342345	Female
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
Create VIEW `nullemergancystaff` as
SELECT Name
FROM staff
WHERE StaffID = ANY (SELECT StaffID FROM stafffinfo WHERE emergancynumber = '');
```



5. Back-end Application

One of the major aspects to this project was developing the back-end application. This was essential in connecting the SQL University Database to the webpage so that it can be easily accessed by the user. This project uses PHP, JSON and Rest API to make the connecting to the university schema. This can be seen in the code snippet below where the connection as indicated by *$conn* is made when *mysqlu_connect* is called.

```php
<?php

$dbServername = "localhost";
$dbUsername ="root";
$dbPassword ="root";
$dbName ="universityschema";

$conn = mysqli_connect($dbServername, $dbUsername, $dbPassword, $dbName);

if(!$conn){
    die("Connection failed: ".mysqli_connect_error());
}

echo "connected sucessfully";
?>
```

The back-end is also responsible for setting up corresponding actions for when the user were to login, register or search for information within the database. As seen in the code snippet below, when LOGIN is initialized the field is first checked for blank entry in which an alert will be sent to the user. Otherwise, *query1* and *query2* are run which can be used to compare all Student ID and Staff ID with what was entered.

```php
if(isset($_LOGIN['login'])){
    $loginID=$_LOGIN['id'];

    if($loginID==''){
        echo "Alert('Please enter the id')";
        exit();
    }


    $query1="SELECT student.StudentNumber From student ";
    $query2="SELECT staff.StaffID FROM staff";

    $result1=mysql_query($query1);
    $result2=mysql_query($query2);
```

If a match is found the user is then directed to the corresponding page weather it be *studnet.php* or *staff.php*. This is demonstrated below.

```
while($row=mysql_fetch_assoc($result1)){
    if($row==$loginID){
        header( 'Location: C:\wamp64\www\datafinal\student.php'
            ) ;
    }
}

while($row=mysql_fetch_assoc($result2)){
        if($row==$loginID){
            header( 'Location: C:\wamp64\www\datafinal\staff.php
                ' ) ;
        }
    }
```

Another key aspect to our back-end code development was the use of queries to fetch and display data . On the *studnet.php*, *staff.php* and *course.php* pages distinct queries were made and referenced in order to display corresponding information. The queries entries can be seen below. By including the initial connection file in the include_once statement we were able to connect our code to the university schema and get the $conn connection. This is used as an argument in *mysqli_query* function which executes each query within the php.

Queries within Student.php

```
<?php
    include_once 'C:\wamp64\www\datafinal\dbh.inc.php';

    $query="SELECT * FROM student WHERE StudentNumber='100716458';";
    $query2="SELECT * FROM universityschema.studentgrade;";

    $result=mysqli_query($conn,$query);
    $result2=mysqli_query($conn,$query2);
?>
```

Queries within Staff.php

```php
<?php
    include_once 'C:\wamp64\www\datafinal\dbh.inc.php';

    $query1="SELECT *
                FROM universityschema.staffinfo
                WHERE StaffID = '5';
                ";

    $query2="Select  course
    .Name , section.Semester
                FROM
                section Join course on section.CourseID =
                    course.CourseID
                Join
                staff On section.Instructor = staff.Name
                Where staff.StaffID ='5';
                ";

    $query3="Select grade.StudentNumber, grade.Grade, course.Name
                FROM
                section Join course on section.CourseID =
                    course.CourseID
                Join
                staff On section.Instructor = staff.Name
                Join grade on grade.SectionID = section.SectionID
                Where staff.StaffID ='5';
                ;
                ";

    $result1=mysqli_query($conn,$query1);
    $result2=mysqli_query($conn,$query2);
    $result3=mysqli_query($conn,$query3);

?>
```

Queries within Course.php

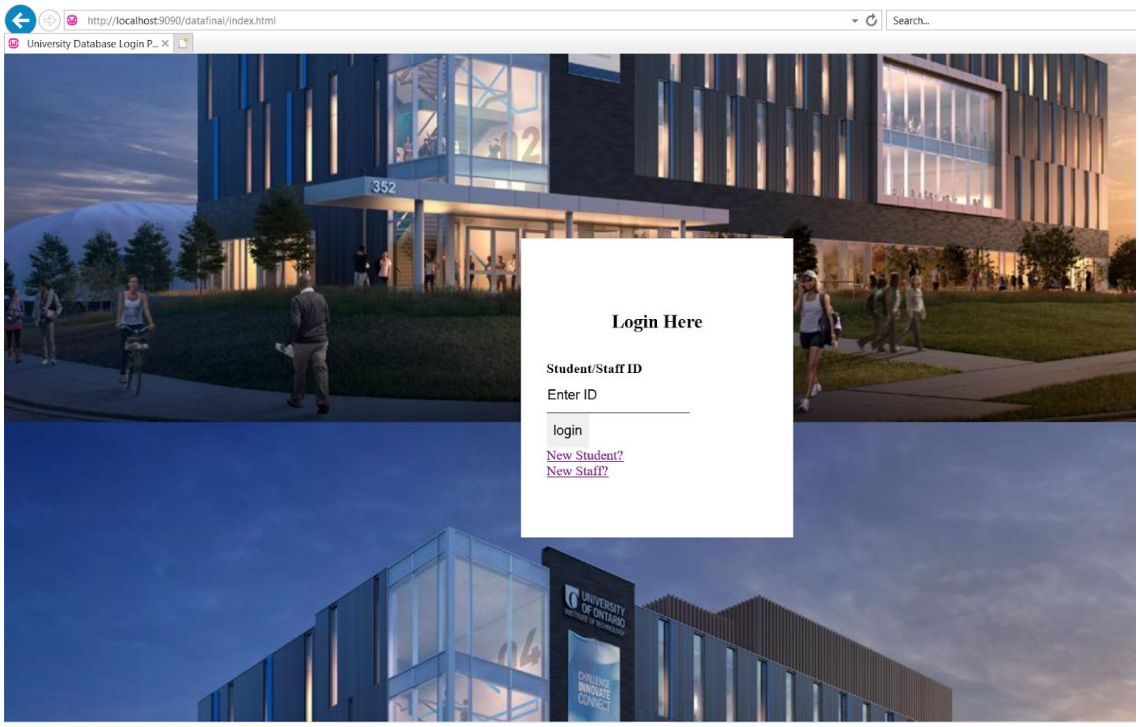
```php
<?php
    include_once 'C:\wamp64\www\datafinal\dbh.inc.php';

    $query1=    "SELECT *
                FROM department;
                ";

    $query2=    "SELECT *
                FROM course;
                ";

    $query3="SELECT Name, PrerequisiteName, CreditHours
                FROM course JOIN prerequisite on course.CourseID = prerequisite.CourseID;
                ";

    $query4 = "SELECT Name, Semester, Instructor, Year
                    FROM section JOIN course on section.CourseID = course.CourseID;";


    $result1=mysqli_query($conn,$query1);
    $result2=mysqli_query($conn,$query2);
    $result3=mysqli_query($conn,$query3);
    $result4=mysqli_query($conn,$query4);
?>
```

-JSON STUFF-

6. Front-end Application

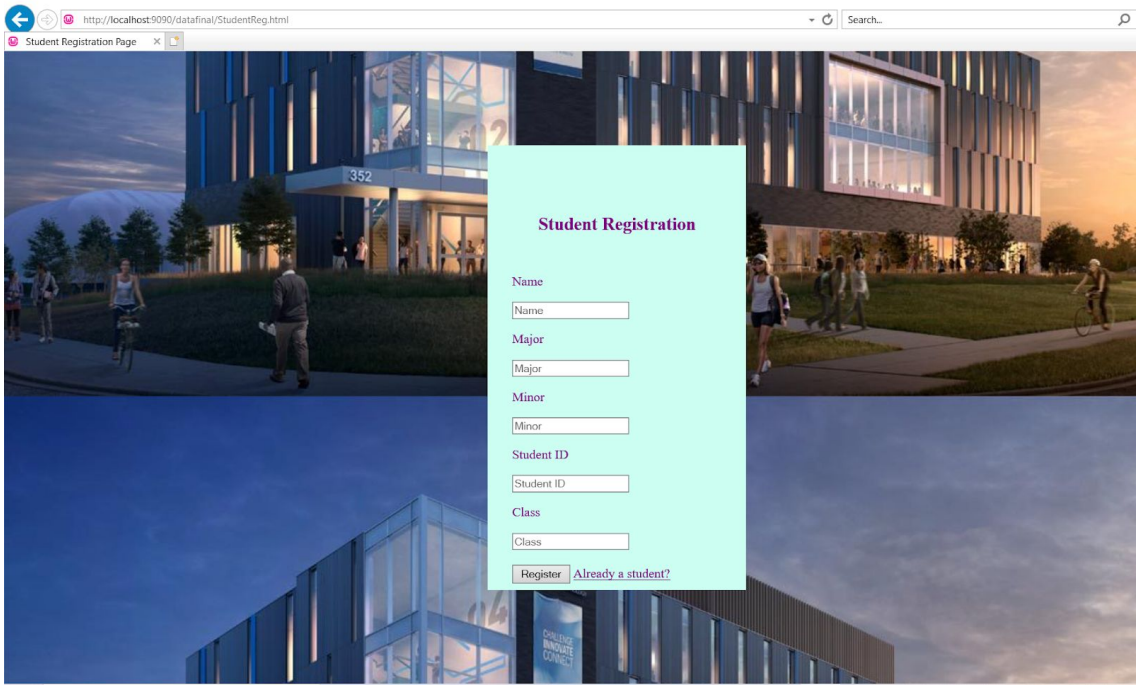
When looking at front-end execution of our program we relied on html and css code to obtain the desired aesthetics for our webpage. Example screenshots to our application are included below.

Index.html



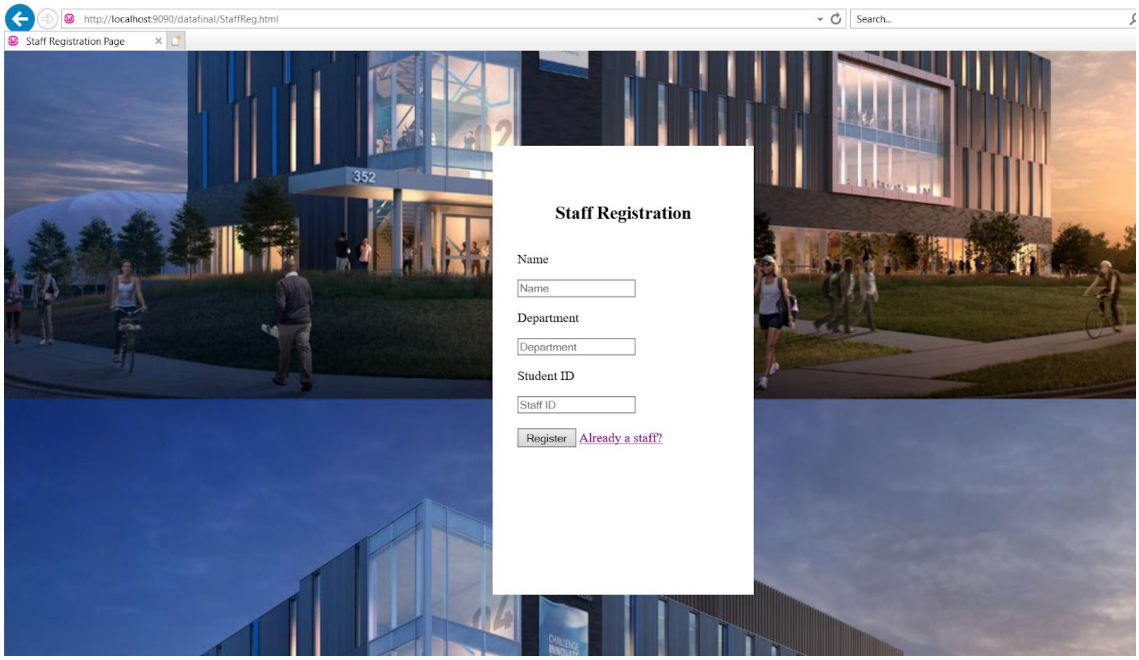
This is the homepage of our website where student/staff members of the university input their distinct ID and if the ID is registered in the database then the website prompts the user to view their records and information in another page. In contrast, if the user ID isn't in the database then the system prompts the user to register as either a new student or new staff.

Student Registration Page



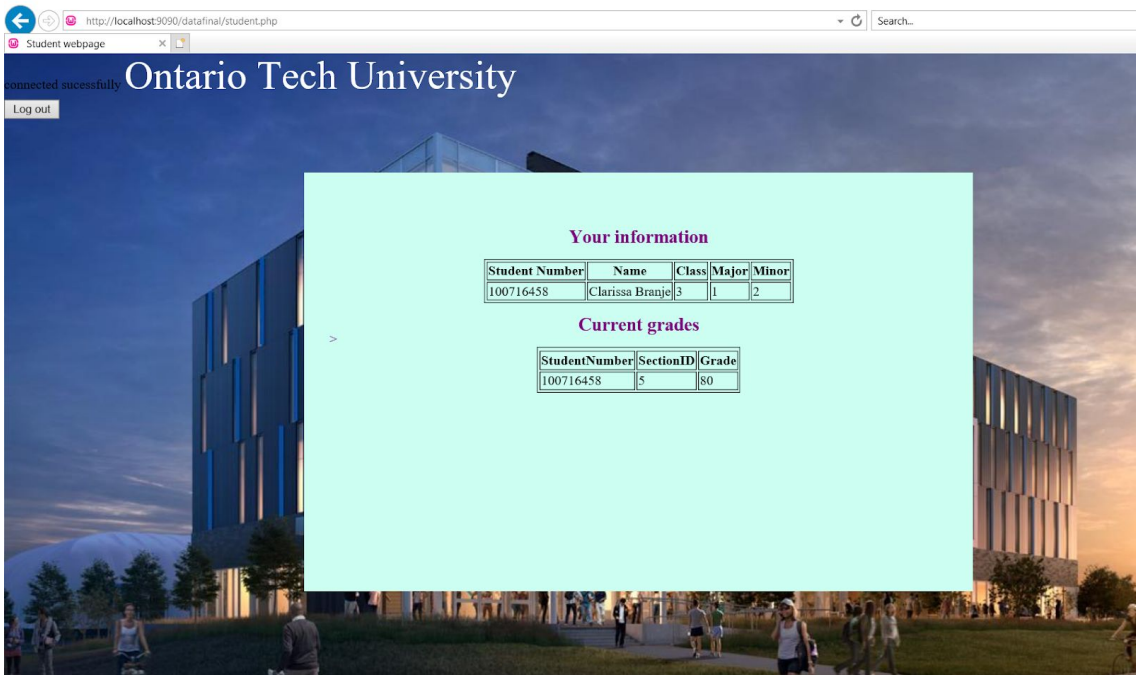
This is the student registration page where the student sets up his personal information and after registering the user is prompted to login and his information would be added to the database and displayed.

Staff Registration Page



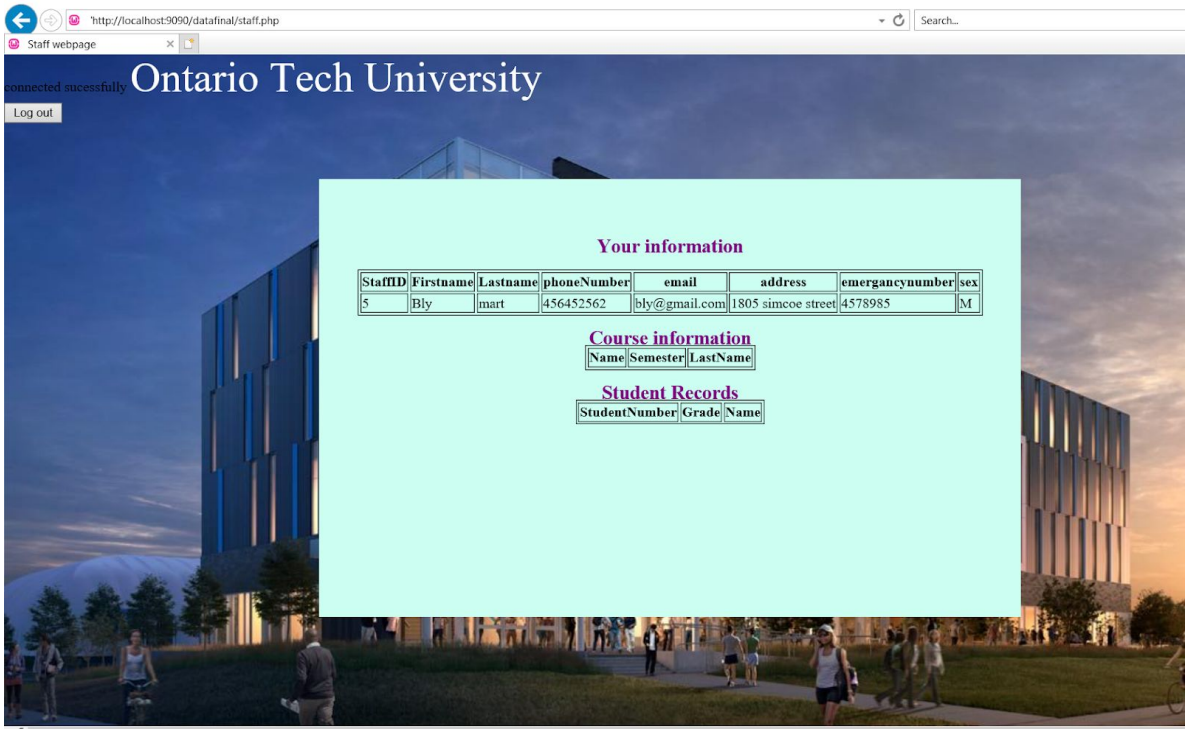
This is the page where staff members can register to be a member of the university and the user information would be added to the database and a new page prompts the staff member to login and he/she would be able to view their informations as well as the students information of the section or course they teach and the students registered under them.

Student Page



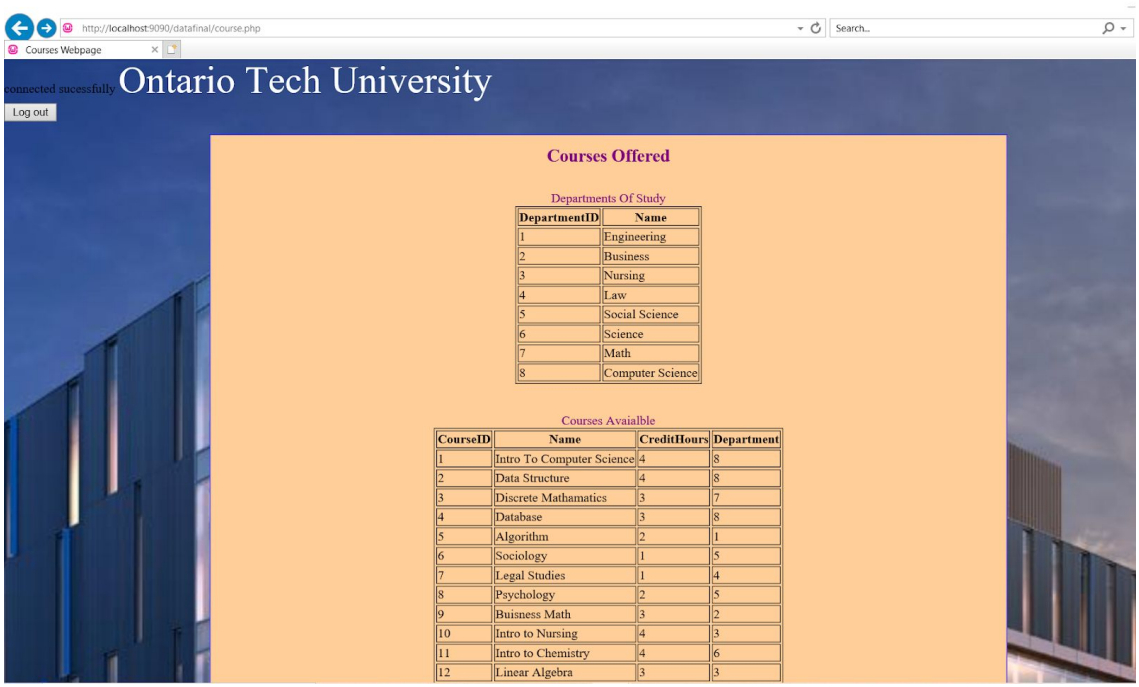
The student page of a student who is registered in the university and this page displays the students information and also the student current grades of a particular section the student is registered in.

Staff Page



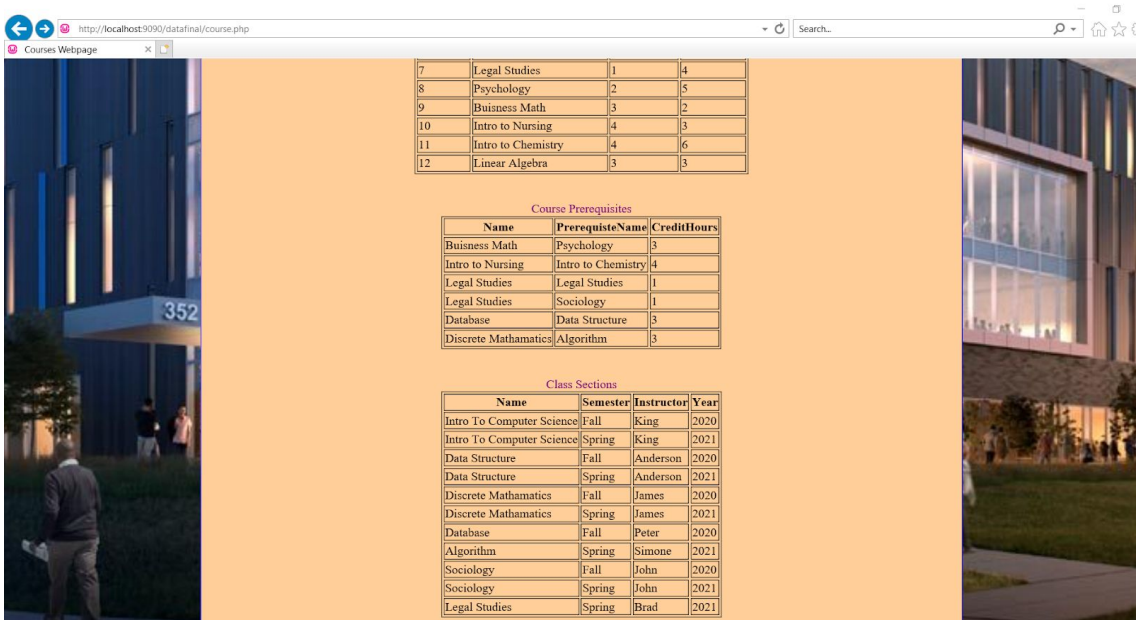
This information provided is an example of a staff logging in to the university system. The staff isn't registered in any course and thus doesn't have any student records to keep track of.

Course Page



DepartmentID	Name
1	Engineering
2	Business
3	Nursing
4	Law
5	Social Science
6	Science
7	Math
8	Computer Science

CourseID	Name	CreditHours	Department
1	Intro To Computer Science	4	8
2	Data Structure	4	8
3	Discrete Mathamatics	3	7
4	Database	3	8
5	Algorithm	2	1
6	Sociology	1	5
7	Legal Studies	1	4
8	Psychology	2	5
9	Buisness Math	3	2
10	Intro to Nursing	4	3
11	Intro to Chemistry	4	6
12	Linear Algebra	3	3



7	Legal Studies	1	4
8	Psychology	2	5
9	Buisness Math	3	2
10	Intro to Nursing	4	3
11	Intro to Chemistry	4	6
12	Linear Algebra	3	3

Name	PrerequisteName	CreditHours
Buisness Math	Psychology	3
Intro to Nursing	Intro to Chemistry	4
Legal Studies	Legal Studies	1
Legal Studies	Sociology	1
Database	Data Structure	3
Discrete Mathamatics	Algorithm	3

Name	Semester	Instructor	Year
Intro To Computer Science	Fall	King	2020
Intro To Computer Science	Spring	King	2021
Data Structure	Fall	Anderson	2020
Data Structure	Spring	Anderson	2021
Discrete Mathamatics	Fall	James	2020
Discrete Mathamatics	Spring	James	2021
Database	Fall	Peter	2020
Algorithm	Spring	Simone	2021
Sociology	Fall	John	2020
Sociology	Spring	John	2021
Legal Studies	Spring	Brad	2021

In order to display the database the information from the database we also used a table system which matched that of the sql database when a certain query is run. A code snippet has been included below to demonstrate. This particular segment is responsible for showing the Departments table in the *course.php* page.

```
<h1>Courses Offered</h1>
<center><label for="Departments">Departments Of Study</label></center>
<table align="center" border="1px" stype="width:600px"; line-height:30px>
<t>
    <th> DepartmentID </th>
    <th> Name </th>

</t>
</table>
    <?php
    while($row=mysqli_fetch_assoc($result1)){
    }
    ?>
    <tr>
        <td><?php echo $row['DepartmentID']; ?></td>
        <td><?php echo $row['Name']; ?></td>
    </tr>
<br>
```

7. Technical Obstacles

One of the major technical obstacles faced by our team was setting up a local host server. Amongst the 3 servers SQLserver, Whampp and Xampp we found most success with Whampp which is why it is the server we agreed upon. However, each server posed problems along the way when connecting to database and web page display.

Due to the remote nature of the project the transferring of files and ensuring all group members have the same database access posed an issue. Github helped to relieve this but could have been used better to ensure efficient pull/push from files.

Midway through the project database was erased unknowingly. Luckily recreation was not hard as documentation was kept before and this was restored shortly after.

8. Milestones and Reporting

Total estimation of man hours: 188

Milestone	Tasks	Reporting	Hrs	Date
1 - Analysis				
1.1	Analysis and design stage, gather data and create system mockup	Team	30	13/10/2020
1.2	Architecture design	Team	8	15/10/2020
1.3	Project Plan	Report	10	17/10/2020
2 - Development				
2.1	Create database	Team	10	4/11/2020
2.2	Import data into database	Team	5	5/11/2020
2.4	Development of webpage	Team	20	21/11/2020
2.5	Integration of database with web page	Team	25	22/11/2020
2.6	Phase II	Report	15	7/11/2020
3 - Testing				
3.1	Testing web page application	Team	25	23/11/2020
3.2	Testing database application	Team	25	23/11/2020
3.4	Phase III	Report	15	26/11/2020

9. Further Advancements

What was accomplished in this project is only a framework for what can be created when looking at a university database. As mentioned earlier, due to the complex nature of the information related to a university the proper implementation of a functioning database is critical. For one, it creates security for students and staff when looking at restricting data access for people unrelated. It is also needed to ensure ease when organizing courses for the semester, preparing course maps for departments and much more.

Key advancements to project include:

- **Connect mySQL to the server first before creating the webpage:**

  This allows you to ensure that you are connected to a reliable and functioning server and also this is important in writing your code to ensure that you are connected to the right host when calling your database. Ideally connect to a wamp or xampp server.

- **Incorporate a course registration for students on the course page:**

  This allows students to directly register for courses that they require through the website. This improves reusability and efficiency for users.

- **Make stored procedures to quicken the runtime:**

  Improves efficiency and organization for the program. Allows for easier advancements and additions to be made in the future.

- **Create more detailed tables:**

  Expand the data fields to collect more information. This can be for current tables and as well for future creation of new tables. Examples for students may include separating between undergraduate,master and PHD students.

- **Create an admin user to manage more input and output info:**

  Administrator side would be able to edit both students and staff for the university. They should also have ultimate access to courses, departments and can control through the website for easy organization for the university.

- **Give grade change permissions to the Professor:**

  Professors should be able to update grades for students within their course and section. This increases interaction with the platform and improves efficiency for professors.

- **Expand server through a network to carry over many computers and extend availability:**

  This helps to make work more efficient and productive, as computer operations wouldn't be one-sided and over-dependency on one system has consequences (e.g loss of files or crashing of systems), and so if access is given to multiple users, work would be done faster and there is more flexibility and reduced dependency.