Faculty of Engineering and Applied Science

SOFE 3700U: Data Management Systems

Final Report

Supervised by: Dr. Khalid A. Hafeez

MUSE

Management of University Student Enrollment (Records)

GROUP 9:

Jessica Leishman (100747155)

Daniel Gohara Kamel (100754671)

Adris Azimi (100704571)

Abida Choudhury (100700985)

Table of Contents

Introduction

This project initially began with the goal of creating a MyCampus alternative with a focus on visual navigation with effective data entry and display. As the development of MUSE began, however, it became clear that the application could better fulfil a more specific need of its primary users, professors and students, by focusing on providing useful information in a central place.

MUSE builds onto an existing application such as MyCampus, RAMSS (Ryerson), SOLUS (Queen's), or Quest (Waterloo) to present key statistics of the university enrollment record database attached to the system. The views it displays are valuable information for users such as academic performance grouped by highschool, students on academic probation, academic transcripts, teacher course sections for the year, identifying students to TA or receive academic awards, and even allows the creation of a professor and associated departments directory.

Background and Motivation

In running an institution where existing members have complicated information that links to sections of the institution, and where decisions need to be made about future members, such as universities and their staff and students. It is necessary to track and analyse information. This information can prove difficult to store, analyse, sort, or derive useful information/analytics from. For example, an admissions office would be able to use the data and make informed choices about future student admissions.

MUSE is a frontend web application that focuses on a clear navigation UI to allow its users easy access to pertinent information regarding their department, and academic career. Similarly, MUSE allows users identified as professors access to information that can be used to assess the performance of other various professors, programs, and even faculties as a whole.

While existing university management websites have simple interfaces that provide the basic information needed for professors and students, MUSE intends to integrate within these pre-existing systems and provide a more detail-rich and tailored experience when interacted with.

MUSE resolves the issue of having to use complex, by hand or out of application calculations in order to determine the overall performance of faculties, or using convoluted navigation layouts to access these statistics.

System Design

The development process for MUSE began by first creating an encompassing relational schema (Appendix C, Part A), which allowed the team to fully visualize and understand the type of database that needed to be developed. Each table in the physical schema and create

statements were defined as outlined in this schema. Similarly, the data that was created needed to meet all of the needs of the schema to present the most relevant views. The final CSS implementation and layout mirrors that which was planned during the design process.

Below, find the use cases associated with the MUSE system, and a brief description of the actions the user takes during the use case.

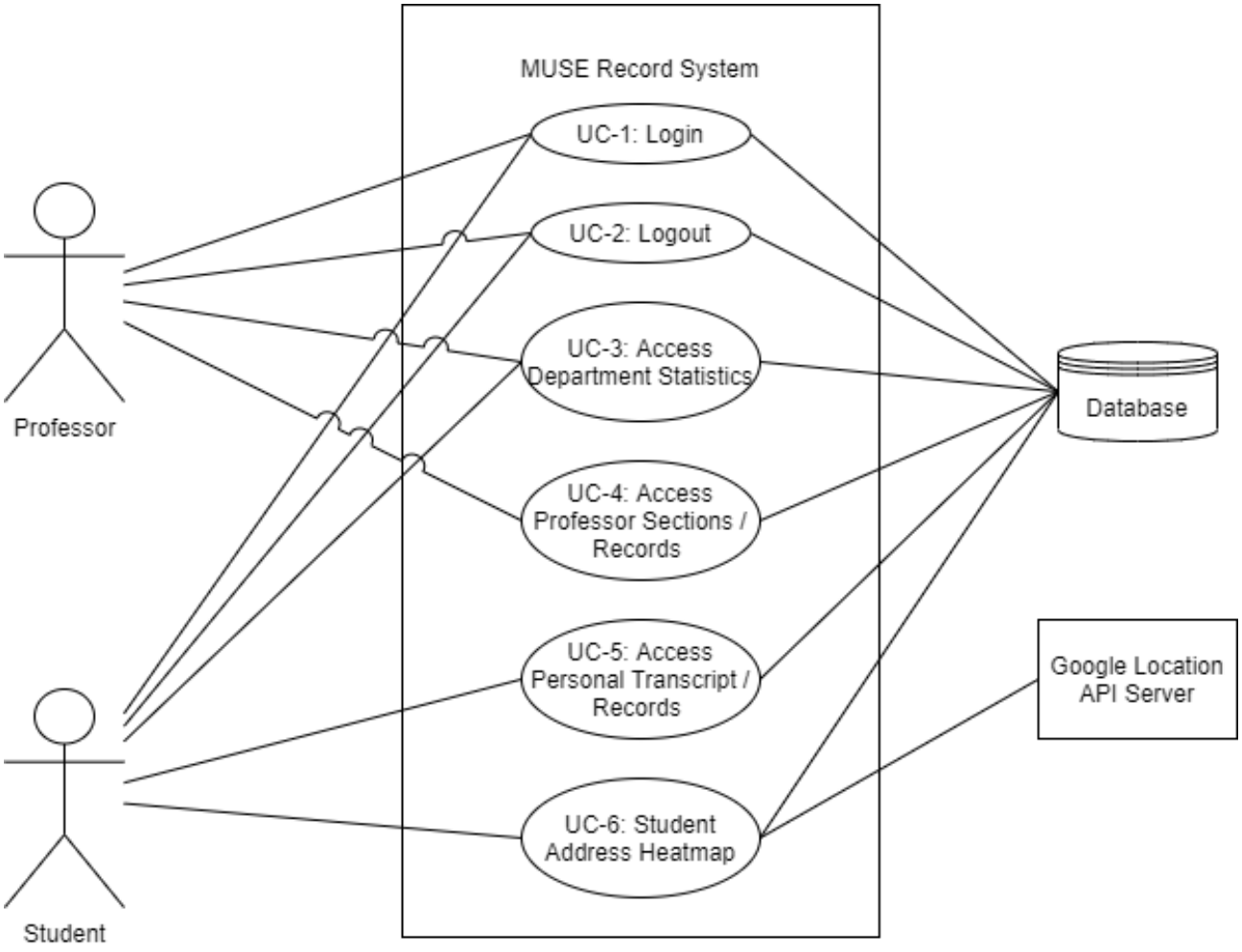


Figure 1: Use Case Diagram

Use Case Descriptions:

ID	Title	Description
UC-1	Login	A professor or student logs into their respective MUSE account, once logged in each user only has access to information related to their role.
UC-2	Logout	The user logs out of their account and is presented with the initial login page
UC-3	Access Department Statistics	A professor or student attempts to view the department statistics page from either the home page or navigation bar. The associated views relating to the various departments in the database is displayed
UC-4	Access Professor Sections / Records	A professor attempts to view their records page. The page is loaded with the professor's class sections for the current year, student performance in each class by highschool, students on academic probation, students with a GPA of 3.7 or higher to be eligible for an academic award, students in residence grouped by highschool, and the grades of students in the professor's past courses.
UC-5	Access Personal Transcript / Records	A student attempts to view the records page and the page is loaded with views and information relevant to students and that student in particular
UC-6	Student Address Heatmap	A user accesses the Student Address Heatmap page and is presented with a Google map view of the school with options to customize how the information appears.

The MUSE system had a primary focus of providing useful views to the user, so it was decided that the API to be implemented should allow more complicated information to be presented. It was settled on the Google Maps API to allow for student location data to be presented. Google Maps being a famous and comprehensive tool used in several widespread applications involving location. The Google Sheets API was also considered to create tables with user interaction and searching but decided against for the initial project iteration because work on the API would need to be completed before views could be implemented and the visualization aspect of the Google Maps API was deemed to be more useful.

The Google Maps API, specifically the javascript available aspects, are loaded in the HTML document using a script tag with the source typically being "https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap" with the callback parameter being the function that first loads when the API finishes loading. The MUSE application does not utilise the callback feature, This is because other functions need to asynchronously load first to Retrieve data from our local database before some API functions can be loaded.

MUSE's heatmap functionality requires full and valid addresses when communicating with the Google Maps API. Therefore, a random address generator website was used to ensure consistent and accurate sample data. Student names were sourced from a list of names which was inserted into the student table. To fill the transcript column with sample data, a small python script was created to pull already generated sample data from the database and insert them all into a text file which was inserted into the transcript column. The remaining data was generated by hand as the database needs specific but more importantly, consistently-related data for course information.

Below, Figure 2 shows the basic navigational page layout that can be followed by a user accessing the site.
Figure 3 shows the layer breakdown of the system, with each of the different pages represented as components on each of the three layers associated with the system

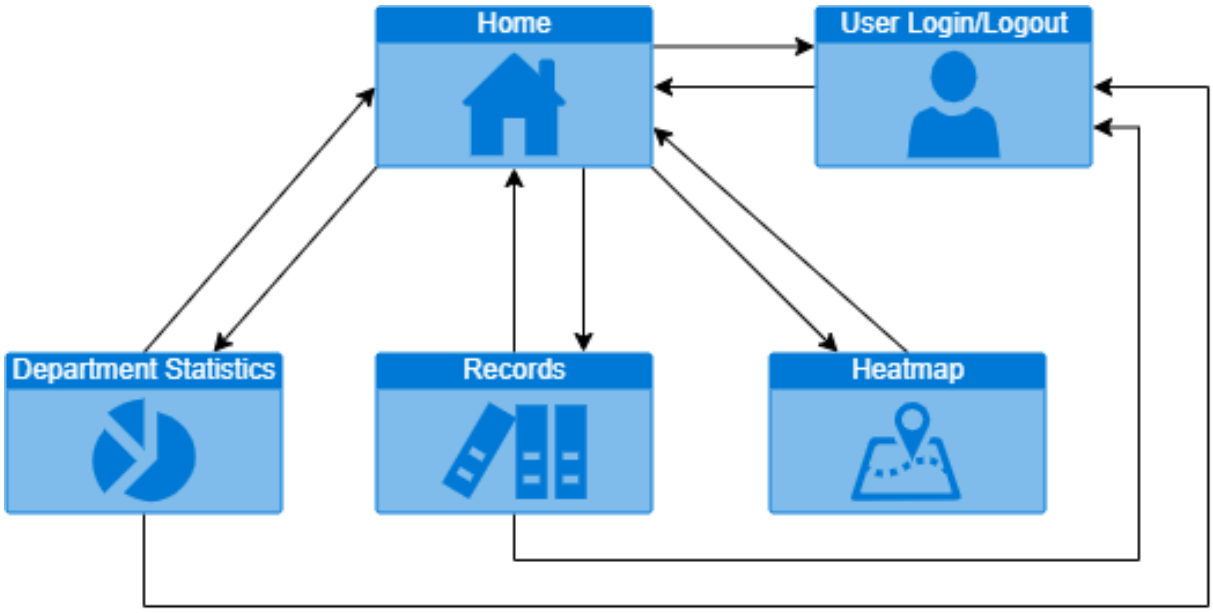


Figure 2: System flowchart for page navigation

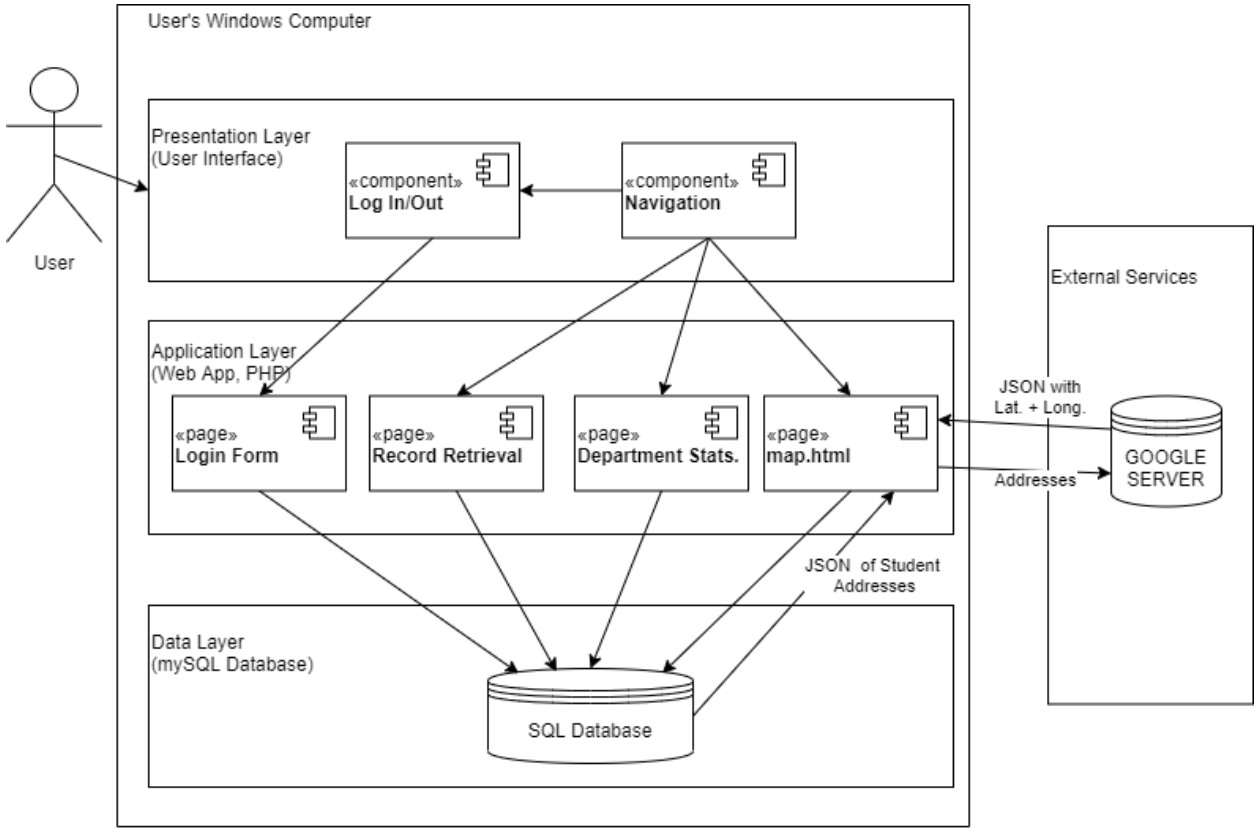


Figure 3: System layer diagram of MUSE system

Observations and Analysis

The MUSE system could be utilized in various business settings, such as its original purpose as a school system, but its performance calculations could also be useful for corporate settings in order to see how the company's various departments fare.

The MUSE system addresses specific issues found in similar types of applications such as the need for a clear navigation UI to allow users to easily access the information they need. These statistics are available on a wider department-view, or on a specific, individual personalized page to the individual currently utilizing the web application. MUSE also provides the professor users access to information that can be used to assess the performance of other professors, programs, and even faculties as a whole within the institution.

The motivation goals for MUSE were mostly met as it was intended to be a MyCampus alternative with improvements on visual navigation as well as effective data entry and display. While it no longer acts as a standalone site such as MyCampus, it instead serves as an add-on to be implemented alongside MyCampus in order to further improve the information and statistics provided. MUSE successfully meets the desired focuses of clear navigation UI, clearly presented information, and statistics presented are of use to all users who utilize the application.

Conclusion

Some challenges were encountered during the development of the MUSE system. These mainly pertained to the development of new components to meet the deliverables associated with the use of API's, however making valuable design decisions also played a big role in shaping the system.

The development of the asynchronous functions used to retrieve and load the JSON data proved challenging for team member Daniel, who tackled this functionality.
Additionally, determining and implementing views that would prove useful for both professors and students was difficult, as the team wanted to ensure there was a page that could present information for both, as well as an individual records page showing personal information tables unique to individual students and professors.

Generating realistic data also proved challenging, as each student needed to have addresses centralized to our university and a general spread of highschools to accurately reflect possible patterns that could be revealed by the MUSE system. A script was implemented to generate the transcripts for each student, however, each table grew in complexity due to the linking required to other tables so further utilizing the script was not viable. This meant the majority of the data needed to be created by hand and took a substantial amount of time.

Moving forward, some of the next steps for development could include further research into Google APIs such as Google Sheets API, to be leveraged for different visual representations of data and searchable tables. Additionally, views could be further customized to the viewing-professor's department, such that a Mechatronics professor would be presented with Mechatronics exclusive department and class statistics. This would further improve the versatility of MUSE and allow it to better serve its users.

Another improvement that could be made to the system would be introducing more classes of users, and formally separating administrator permissions from the professors. Should MUSE be integrated with other academic services that are used to track student performance, only administrators would be able to perform the actual modification of the database to prevent user misuse.

Finally, the last improvement the team would like to implement moving forward would be the use of a cloud database, to enable the deployment of the system across multiple computers and for multiple users to access the database from their own local application.

References

[1] "Fake address generator," *Canada Real Random Address*. [Online]. Available: https://www.fakeaddressgenerator.com/Canada_Real_Random_Address. [Accessed: 22-Nov-2021].

[2] Google Developers. 2021. *Overview | Maps JavaScript API | Google Developers*. [online] Available at: <https://developers.google.com/maps/documentation/javascript/overview> [Accessed 24 November 2021].

[3] *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/earthquakes#maps_earthquake_markers. [Accessed: 25-Nov-2021].

[4] *Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/earthquakes#heatmaps. [Accessed: 25-Nov-2021].

[5] "Sheets API | google developers," *Google*. [Online]. Available: https://developers.google.com/sheets/api. [Accessed: 26-Nov-2021].

Appendices

Appendix A: Github Repository

Github Repository:
https://github.com/DanielKamel2001/MUSE

Github Readme:
https://github.com/DanielKamel2001/MUSE/blob/main/README.md

Appendix B: Installation Instructions

Setting Up WAMP server
Though any different web server hosting program can be used, this project used WAMP server hosting for local hosting with PHP and MySQL. If another program is used please be mindful of how you set up the application with this guide.

To use Wamp Server, download it and install it from the website: https://www.wampserver.com/en/,
or mirror sourceforge link: https://sourceforge.net/projects/wampserver/files/.

Downloading the Frontend Application from Github
The MUSE frontend web application can be accessed from the github repository linked in Appendix A, or from this link: https://github.com/DanielKamel2001/MUSE.git

First, the code must be downloaded from the github repository.

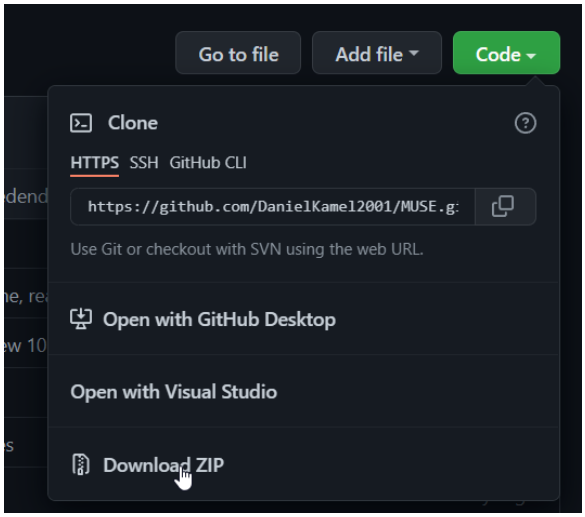


Figure 4: Download button for the repository

Unzip the archive downloaded using a service such as 7zip or Windows 10 built-in features, and copy-and-paste the contents of the MUSE-main folder into your WAMPserver www localhost folder.

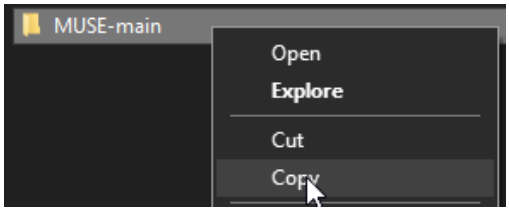


Figure 5: copying the contents into the www folder

The WAMPserver www folder can be accessed by running the Wampserver64 local server application, clicking on the tray icon, and then selecting the www directory folder.

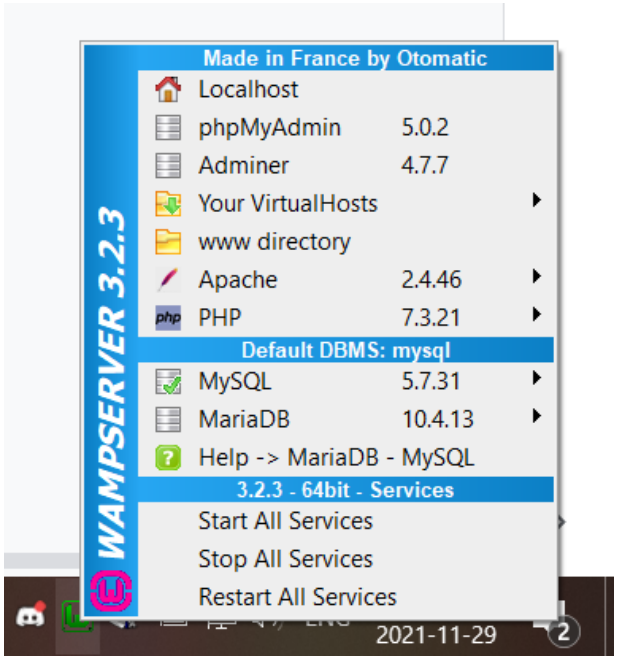


Figure 6: Finding the www folder

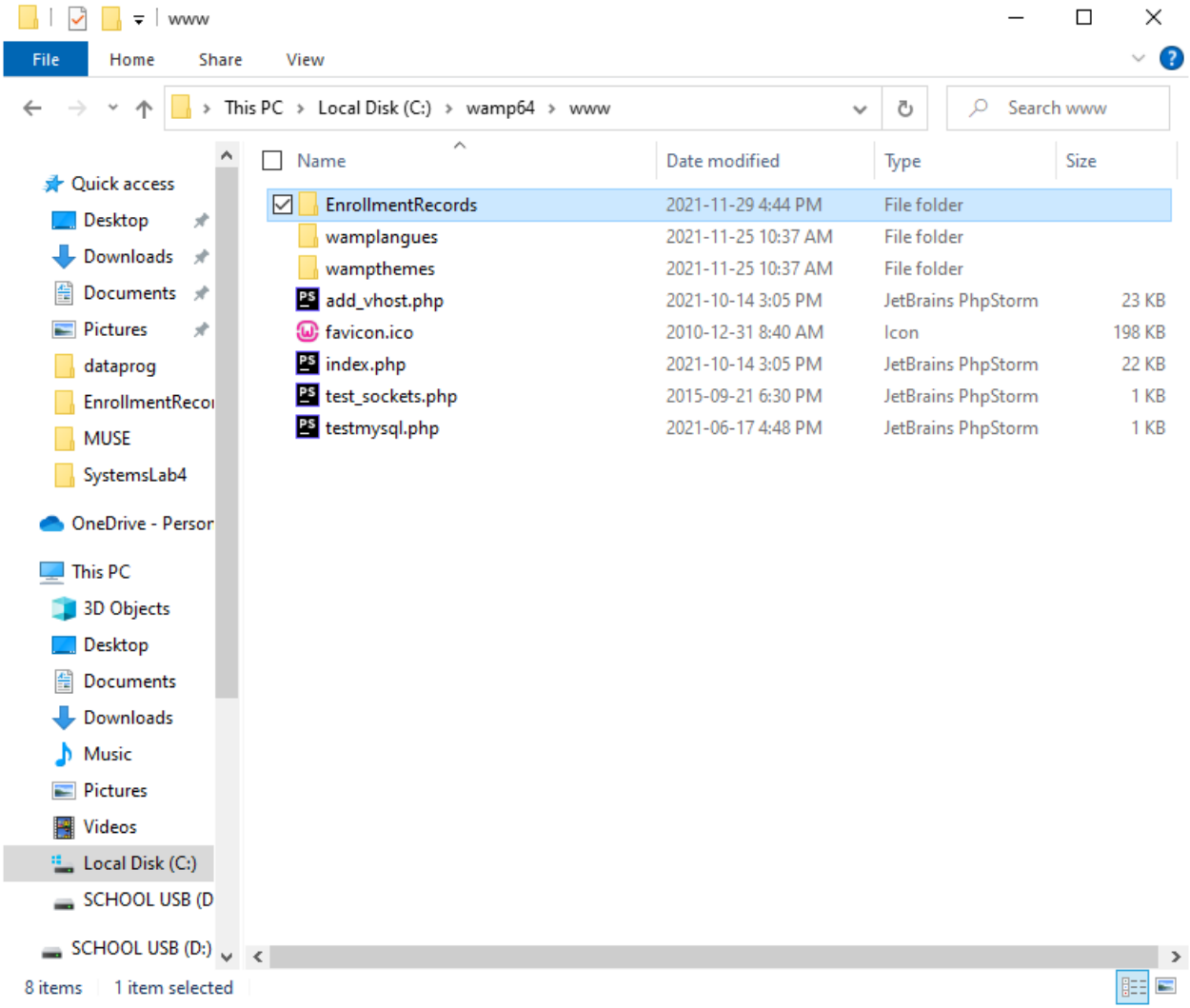


Figure 7: www folder with program inside EnrollmentRecords folder

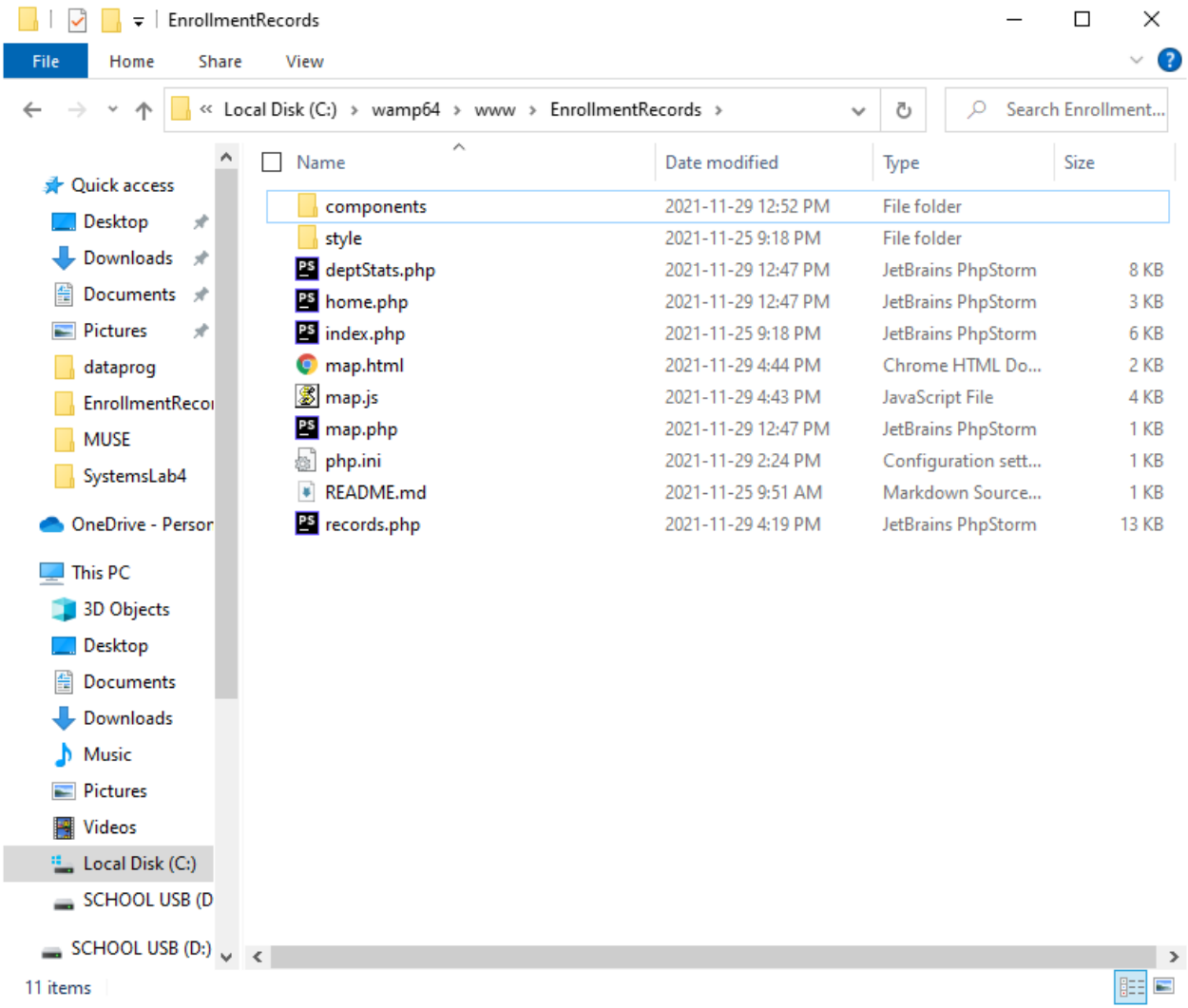


Figure 8: Front End application files inside EnrollmentRecords folder

Setting up the sample database

Included in the git repository is a file titled DatabaseExport.sql, this file is a series of MySQL statements that was created by using WAMPSERVERs phpMyAdmin feature. The database can be created by using the import feature of phpMyAdmin or by running the file in MySQL Workbench.

Modifying the database configuration instructions (php.ini)

Also included in the project zip obtained from the Github repository is a file called php.ini. This must be kept in the main project folder. Please modify this file to include the configuration details of the location of your local database.

Running the Project

To run the application start the wamp server instance and navigate to the localhost in your browser with the path inside your www folder, see the example in Figure 9

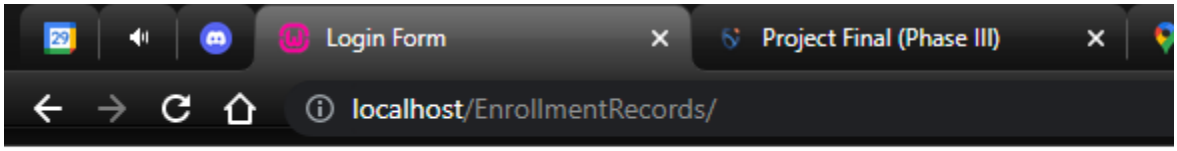


Figure 9. Example on navigating to the site in from the localhost

Also below are sample login information to help show off the views of the website:
Studentlogin:
ID Number: 1, Password: 20020724
Staff login:
ID Number: 15, Password: 19570810

Adding Personal API Key

In order to view the student heatmap feature of the application, the Google Maps API was used. An API key is needed to use the API. The key identifies which user is requesting the APIservice and bills them accordingly. A key can be created by creating an account at "https://console.cloud.google.com/google/maps-apis/". When using the API the key is included in the URL of the request as seen in Figure 10.

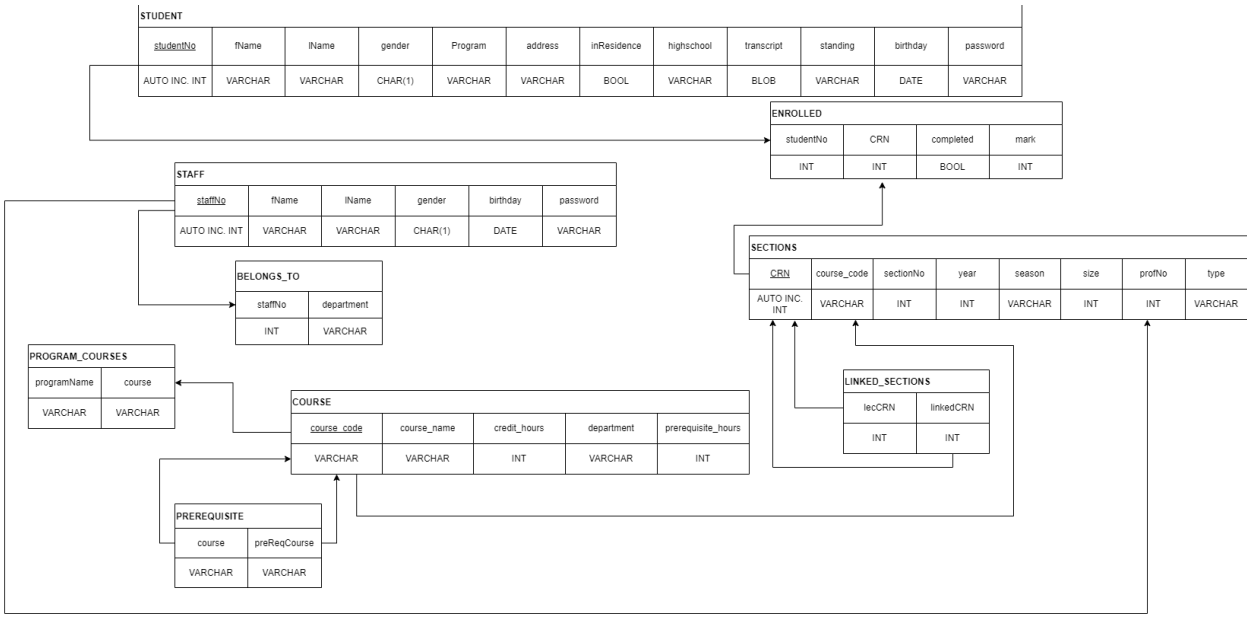
```
<script src="https://maps.googleapis.com/maps/api/js?key=YOURAPIKEY&libraries=visualization"></script>
```

Figure 10: an example of using the google maps API with an API key

Other parameters can be passed through in order to request other information or services from the Google API, notably geocoding which is retrieving longitude and latitude information from an address.

Appendix C: Revised Phase II Report

Part A: Schema Diagram



Part B: Generated Data CSV Files

Github folder of CSV files:
Create SQL Script with Import data queries:

Part C: Views

View 1: Computes a join of at least three tables

1. *Select all students who have specific prof (course CRN) from a certain highschool and display their grades sorted by course*

```
SELECT STUDENT.highschool, STUDENT.studentNo, ENROLLED.mark,
ENROLLED.CRN, SECTIONS.course_code, SECTIONS.profNo
FROM STUDENT
JOIN ENROLLED on STUDENT.studentNo = ENROLLED.studentNo
JOIN SECTIONS on ENROLLED.CRN = SECTIONS.CRN
JOIN STAFF on SECTIONS.profNo = STAFF.staffNo
WHERE ENROLLED.mark is not NULL
ORDER BY SECTIONS.course_code ASC, STUDENT.studentNo ASC,
STUDENT.highschool;
```

View 2: Uses nested queries with the ANY or ALL operator and uses a GROUP BY clause

2. *Find students who got A- and higher to be peer tutors/TA for course*

```
SELECT STUDENT.studentNo, STUDENT.fName, STUDENT.lname,
enrolled.mark, enrolled.CRN, SECTIONS.course_code
FROM STUDENT, ENROLLED
JOIN SECTIONS ON Enrolled.CRN = SECTIONS.CRN
```

```
WHERE Student.studentNo = ALL
(SELECT studentNo
FROM ENROLLED
GROUP BY mark >= 80);
```

View 3: A correlated nested query

3. *All sections that belong to courses managed by the sofe department*

```
SELECT *
FROM sections
WHERE course_code IN
  (SELECT course.course_code
  FROM course
  WHERE course.department = 'SOFE');
```

View 4: Uses a FULL JOIN

4. *Finding names of professors from a specific department*
   a.
```
SELECT staffNo, fName, lName
FROM STAFF
FULL JOIN BELONGS_TO
ON STAFF.staffNo = BELONGS_TO.staffNo;
```

FULL JOIN IS NOT VALID OPERATION IN MYSQL
Workaround: Use left join union right join

   b.
```
SELECT Staff.fName, Staff.lName, BELONGS_TO.department
FROM STAFF
LEFT JOIN BELONGS_TO
ON STAFF.staffNo = BELONGS_TO.staffNo
UNION
SELECT Staff.fName, Staff.lName, BELONGS_TO.department
FROM STAFF
RIGHT JOIN BELONGS_TO
ON STAFF.staffNo = BELONGS_TO.staffNo;
```

View 5: Uses nested queries with any of the set operations UNION, EXCEPT, or INTERSECT

5. *Select all students in residence on academic probation (not in clear standing)*
   a.
```
SELECT fName, lName, inResidence
FROM STUDENT
WHERE Program = "Software Engineering"
EXCEPT
SELECT fName, lName, inResidence

FROM STUDENT
WHERE academicStanding = "Clear";
```

INTERCEPT, EXCEPT ARE NOT VALID OPERATION IN MYSQL
Workaround: Use NOT IN instead of EXCEPT

b. SELECT fName, lName, inResidence, Program, academicStanding
FROM STUDENT
WHERE Program = 'Software Engineering' and studentNo NOT IN (
SELECT studentNo
FROM STUDENT
WHERE academicStanding = "Clear");

6. (a) Display Viewing student's transcript, OR (b) display professor's sections for current calendar year.

a. SELECT SECTIONS.course_code, ENROLLED.mark, SECTIONS.season, SECTIONS.year, SECTIONS.type
FROM student
JOIN ENROLLED on STUDENT.studentNo = ENROLLED.studentNo
JOIN SECTIONS on ENROLLED.CRN = SECTIONS.CRN
WHERE Enrolled.studentNo = ".$_SESSION["sessionID"]." ORDER BY SECTIONS.year ASC, sections.season;

b. SELECT course_code, CRN, sectionNo, season, year, type, size
FROM SECTIONS
WHERE profNo = ". $_SESSION["sessionID"]." AND year = YEAR(CURRENT_TIMESTAMP)
ORDER BY course_code ASC, CRN ASC, sectionNo ASC;

7. Students selected by GPA, if their average is about an 80 (3.7), eligible for an award

SELECT *
FROM(SELECT s.studentNo, fName, lName, avg(e.mark) as studentAvg
FROM enrolled as e
LEFT JOIN student AS s ON s.studentNo = e.studentNo
GROUP BY e.studentNo) AS averages WHERE averages.studentAvg >= 80;

8. Sort students by high school to see if we need more residences (high number of non-local schools but moderate residence numbers indicates off campus housing)

SELECT STUDENT.highschool, count(STUDENT.inResidence)
FROM STUDENT
WHERE STUDENT.inResidence = true
GROUP BY highschool ASC;

9. Calculate student's average to determine if they are eligible for Co-op

SELECT AVG(mark) AS GPA FROM Enrolled
JOIN STUDENT on ENROLLED.studentNo = STUDENT.studentNo
JOIN SECTIONS on ENROLLED.CRN = SECTIONS.CRN

WHERE STUDENT.studentNo = <viewing student no>;

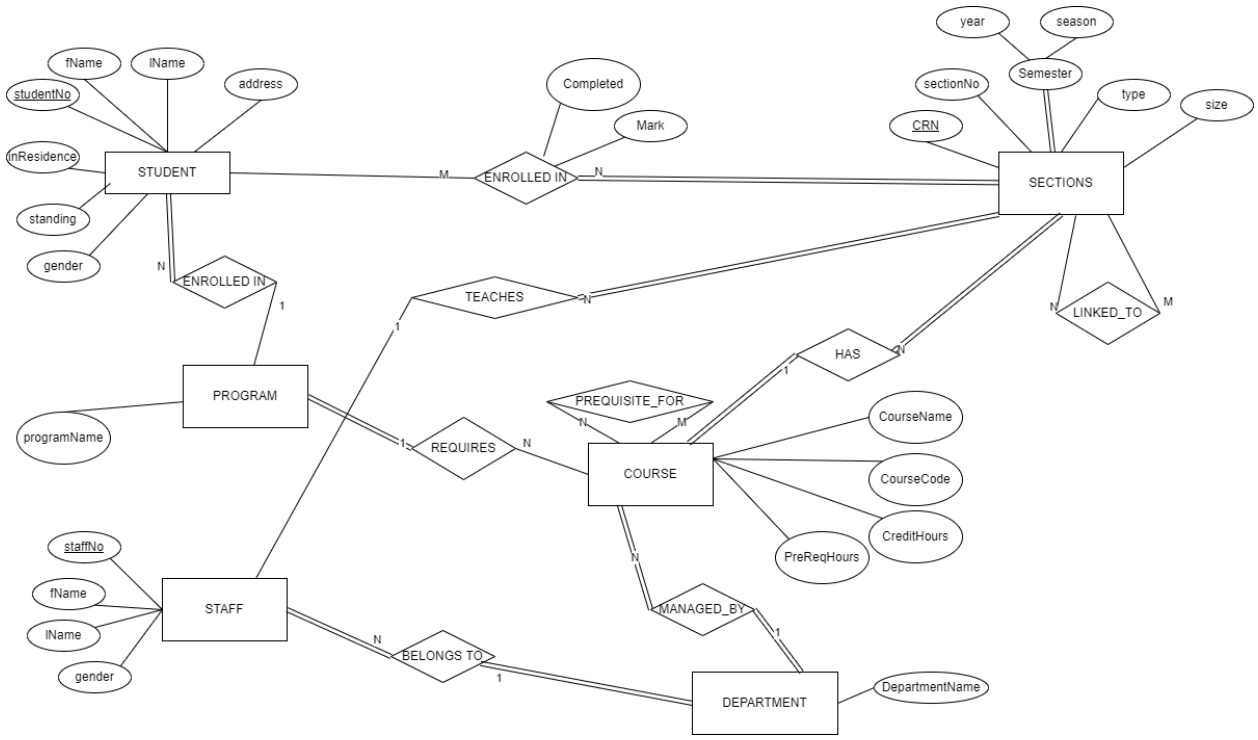
if (GPA >= 67){
YES

```
}
else{
        NO
}
```

10. Calculate number of students in clear academic standing by program

```
SELECT STUDENT.Program, COUNT(*)
FROM STUDENT
WHERE student.academicStanding='Clear'
GROUP BY Program;
```

Part D: Entity-Relationship Diagram



Appendix D: Frontend Application Screenshots

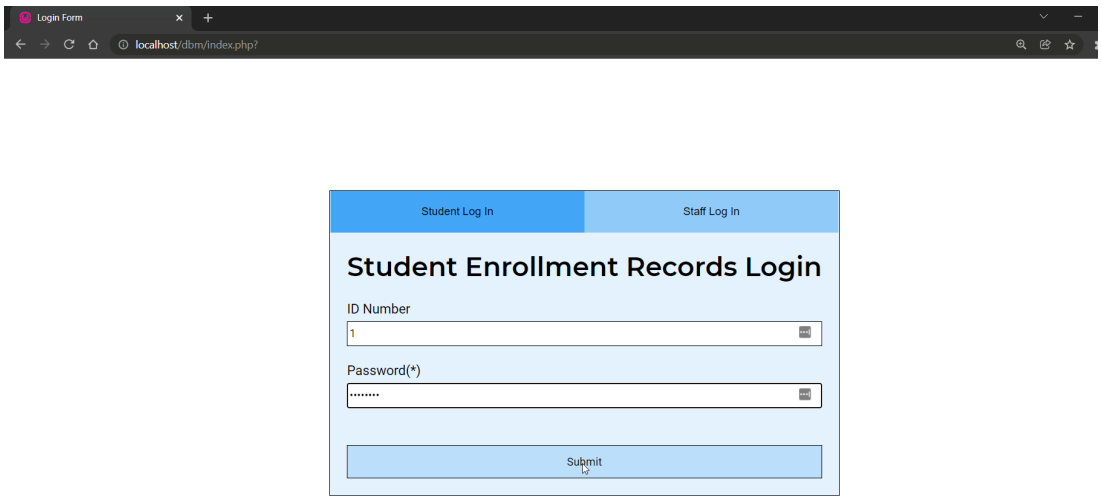


Figure 11: Student Login page

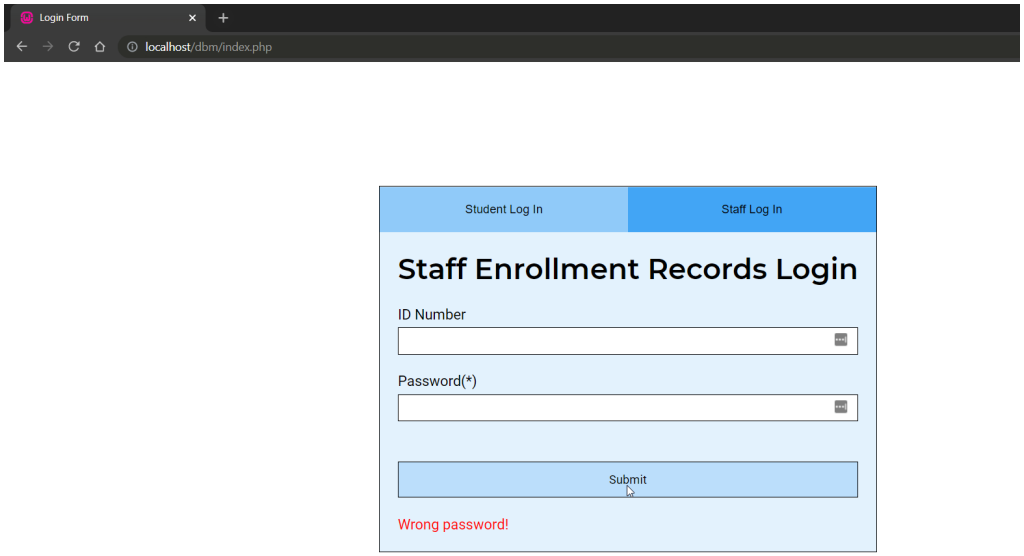


Figure 12: Staff Login Page

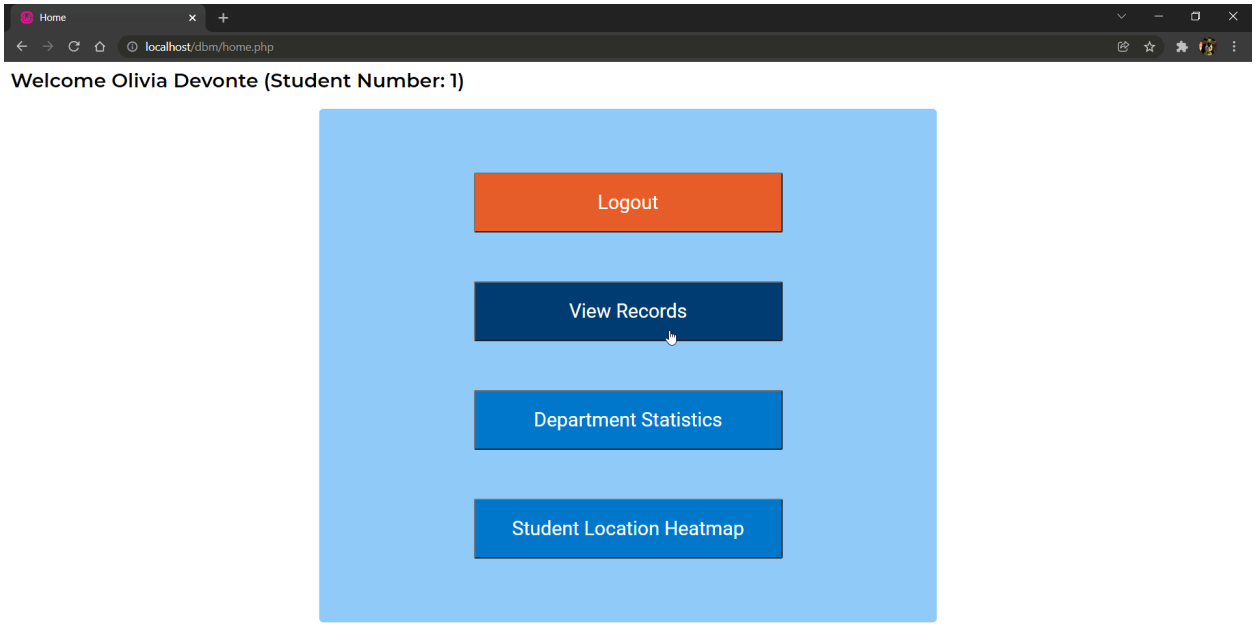
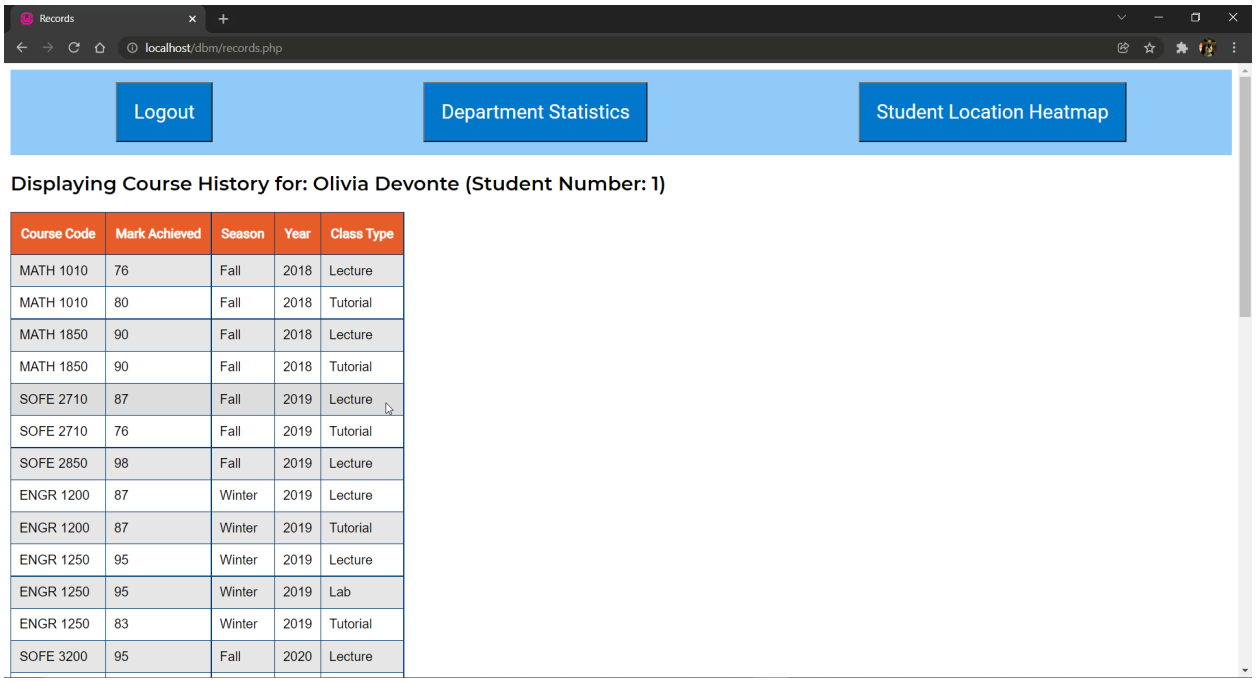
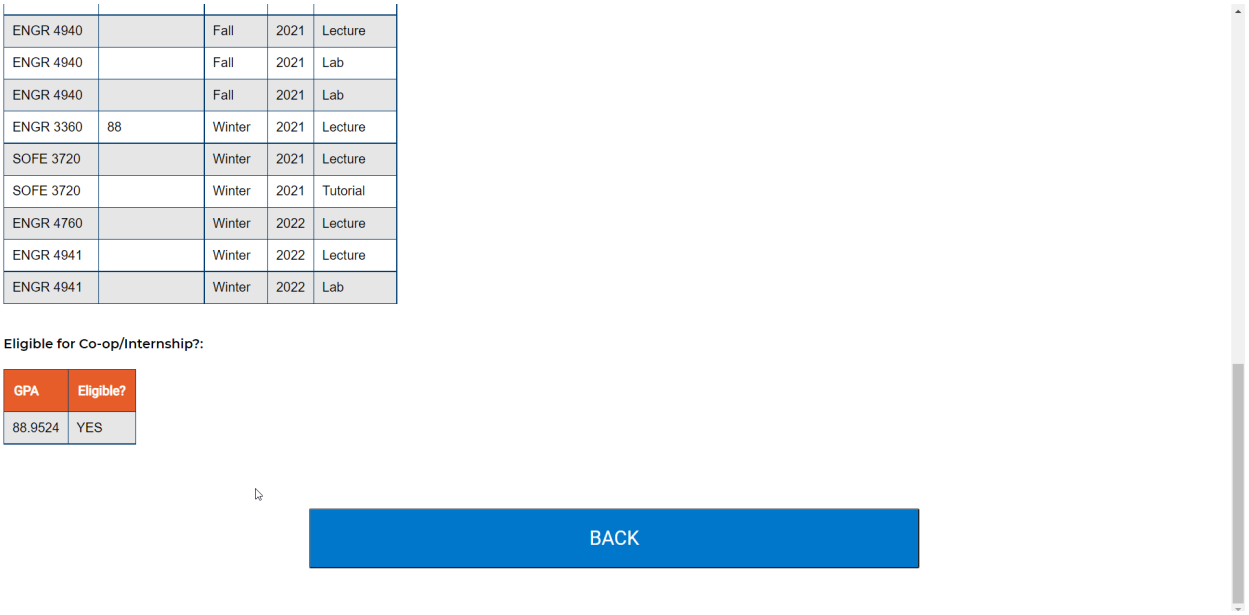


Figure 13: Student/Staff home page



Course Code	Mark Achieved	Season	Year	Class Type
MATH 1010	76	Fall	2018	Lecture
MATH 1010	80	Fall	2018	Tutorial
MATH 1850	90	Fall	2018	Lecture
MATH 1850	90	Fall	2018	Tutorial
SOFE 2710	87	Fall	2019	Lecture
SOFE 2710	76	Fall	2019	Tutorial
SOFE 2850	98	Fall	2019	Lecture
ENGR 1200	87	Winter	2019	Lecture
ENGR 1200	87	Winter	2019	Tutorial
ENGR 1250	95	Winter	2019	Lecture
ENGR 1250	95	Winter	2019	Lab
ENGR 1250	83	Winter	2019	Tutorial
SOFE 3200	95	Fall	2020	Lecture

Figure 14.1: Student Records Page with previous courses



ENGR 4940		Fall	2021	Lecture
ENGR 4940		Fall	2021	Lab
ENGR 4940		Fall	2021	Lab
ENGR 3360	88	Winter	2021	Lecture
SOFE 3720		Winter	2021	Lecture
SOFE 3720		Winter	2021	Tutorial
ENGR 4760		Winter	2022	Lecture
ENGR 4941		Winter	2022	Lecture
ENGR 4941		Winter	2022	Lab

Eligible for Co-op/Internship?:

GPA	Eligible?
88.9524	YES

BACK

Figure 14.2: Student Records, with previous courses (transcript) and average indicating co-op eligibility

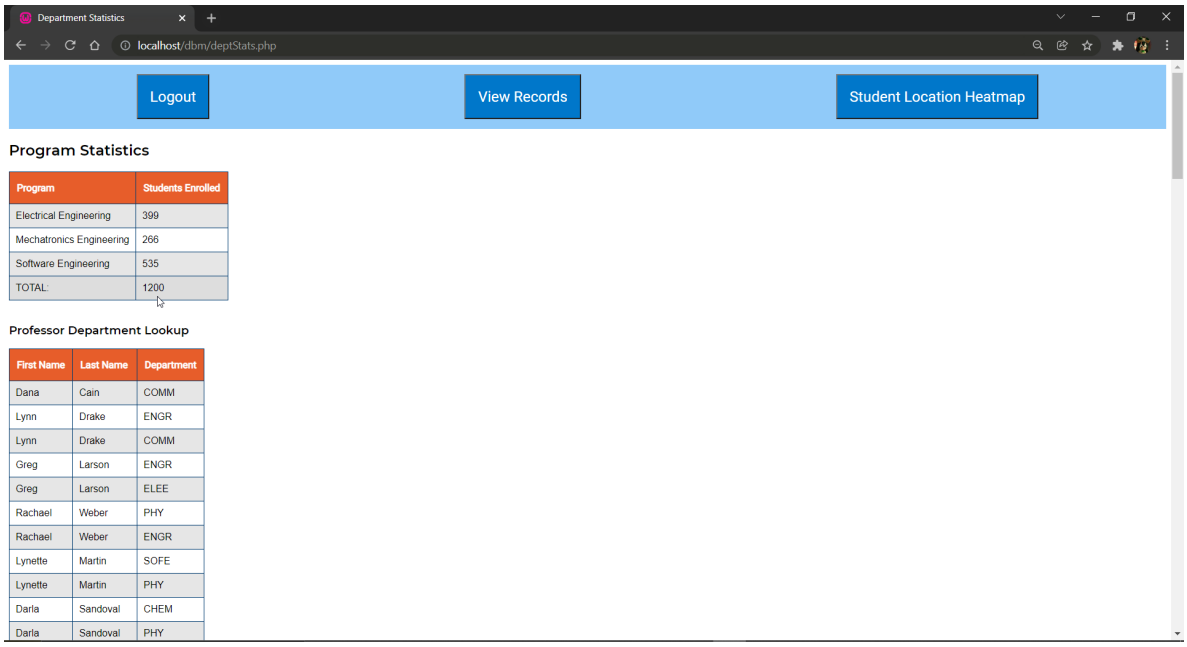
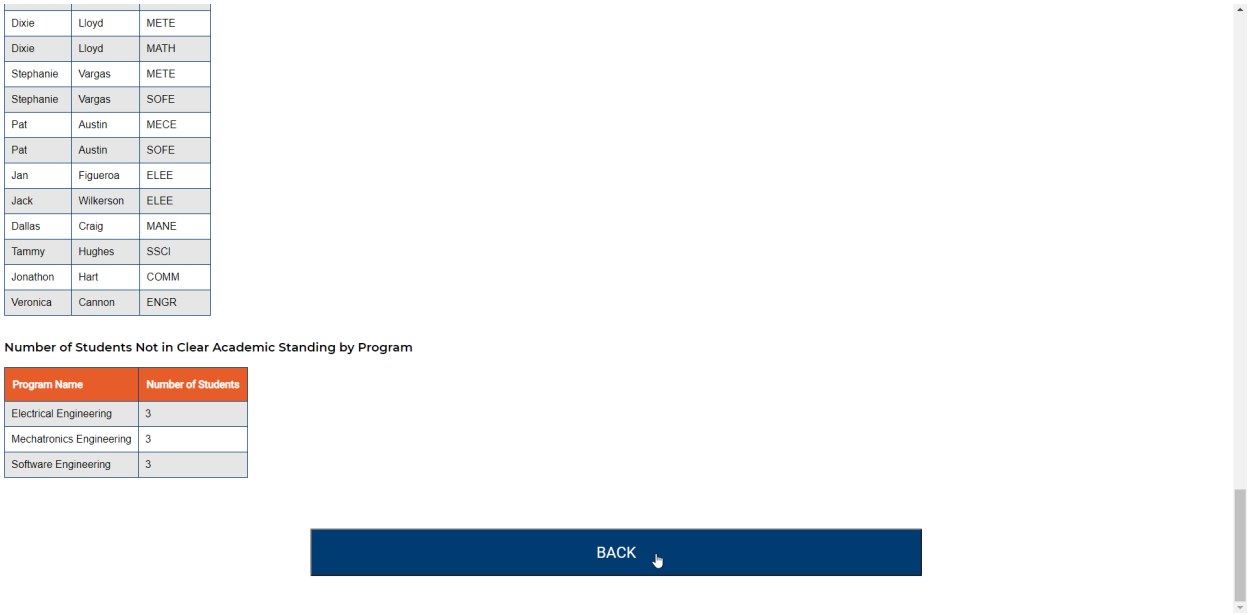


Figure 15.1:Department Statistics



Dixie	Lloyd	METE
Dixie	Lloyd	MATH
Stephanie	Vargas	METE
Stephanie	Vargas	SOFE
Pat	Austin	MECE
Pat	Austin	SOFE
Jan	Figueroa	ELEE
Jack	Wilkerson	ELEE
Dallas	Craig	MANE
Tammy	Hughes	SSCI
Jonathon	Hart	COMM
Veronica	Cannon	ENGR

Figure 15.2: Department Statistics.

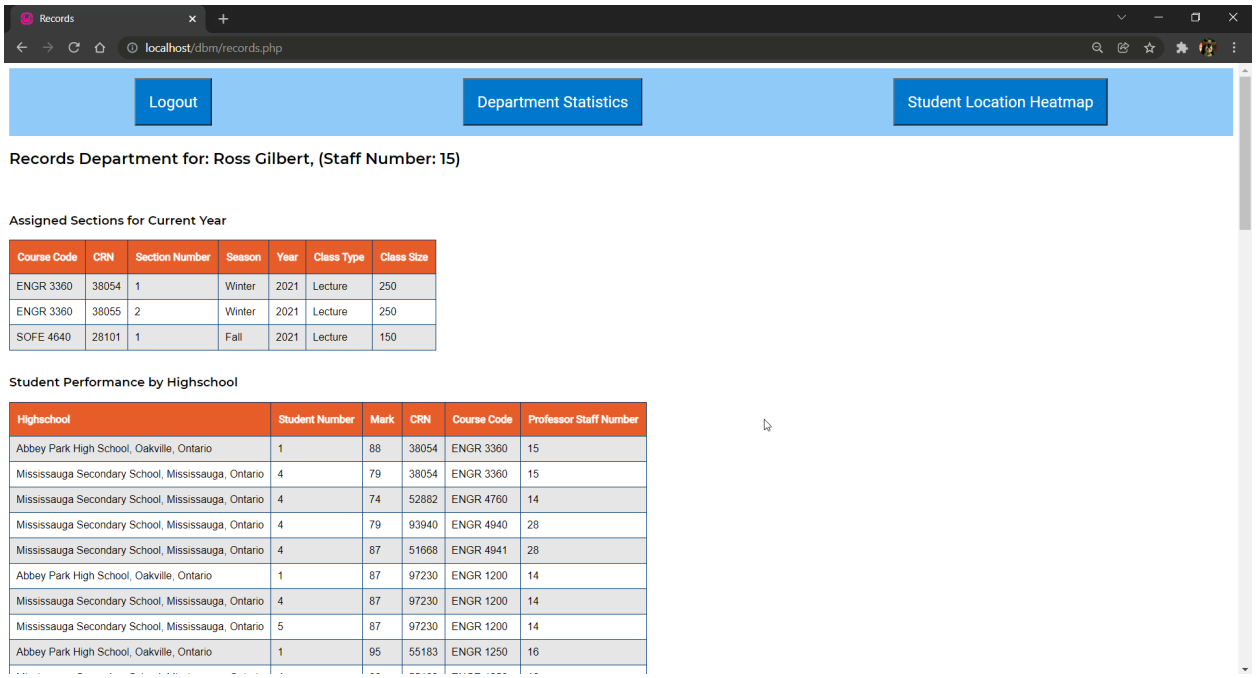
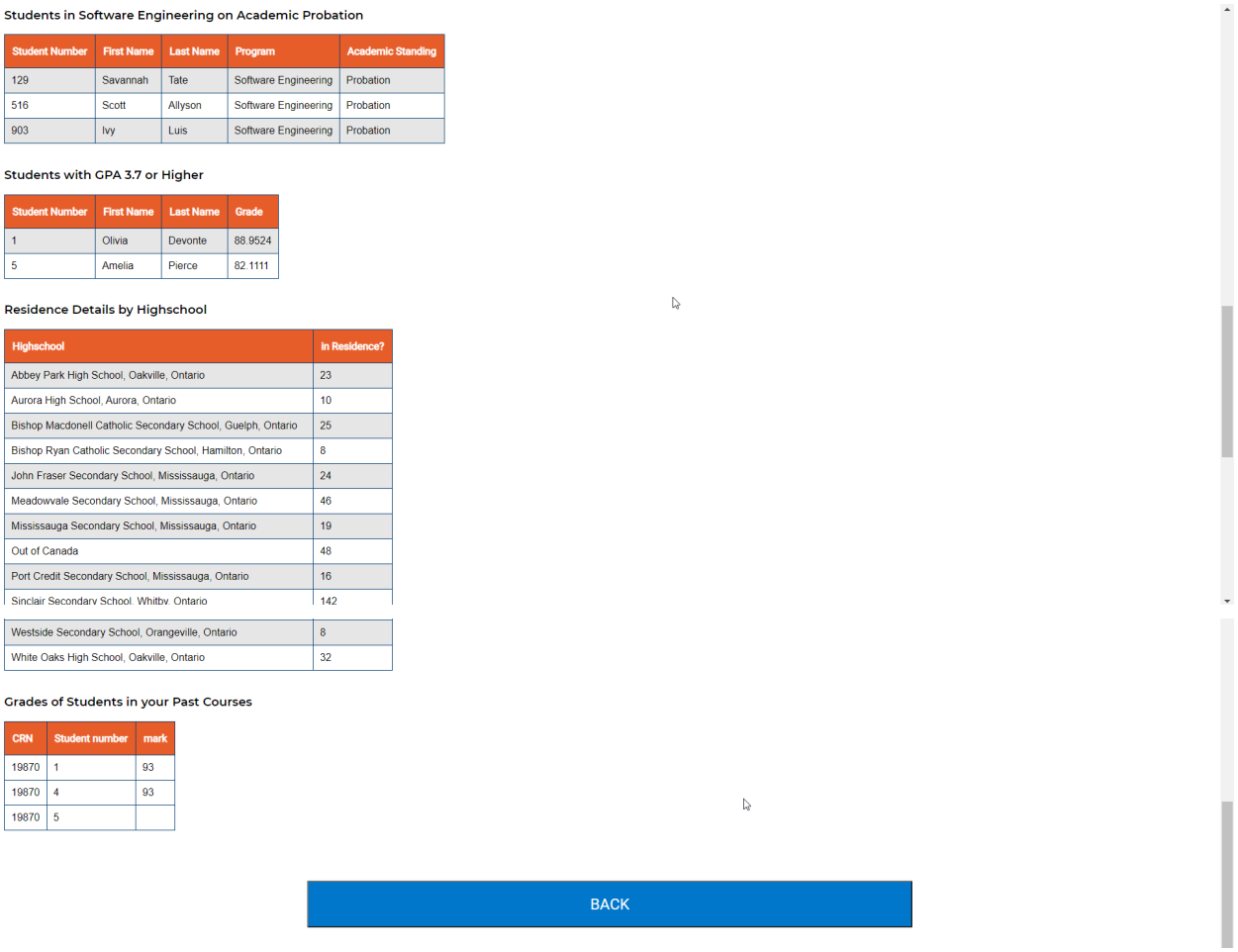


Figure 16.1: Records page for staff users. Shows the staff assigned sections to teach, and student performance.



Students in Software Engineering on Academic Probation

Student Number	First Name	Last Name	Program	Academic Standing
129	Savannah	Tate	Software Engineering	Probation
516	Scott	Allyson	Software Engineering	Probation
903	Ivy	Luis	Software Engineering	Probation

Students with GPA 3.7 or Higher

Student Number	First Name	Last Name	Grade
1	Olivia	Devonte	88.9524
5	Amelia	Pierce	82.1111

Residence Details by Highschool

Highschool	In Residence?
Abbey Park High School, Oakville, Ontario	23
Aurora High School, Aurora, Ontario	10
Bishop Macdonell Catholic Secondary School, Guelph, Ontario	25
Bishop Ryan Catholic Secondary School, Hamilton, Ontario	8
John Fraser Secondary School, Mississauga, Ontario	24
Meadowvale Secondary School, Mississauga, Ontario	46
Mississauga Secondary School, Mississauga, Ontario	19
Out of Canada	48
Port Credit Secondary School, Mississauga, Ontario	16
Sinclair Secondary School, Whitby, Ontario	142
Westside Secondary School, Orangeville, Ontario	8
White Oaks High School, Oakville, Ontario	32

Grades of Students in your Past Courses

CRN	Student number	mark
19870	1	93
19870	4	93
19870	5	

BACK

Figure 16.2: Staff Records page showing sections for this semester, highschool overall performance in each class, students on academic probation, students with a GPA of 3.7 or higher, number of students in residence ordered by highschool, and the grades in the viewing professor's previously taught courses.

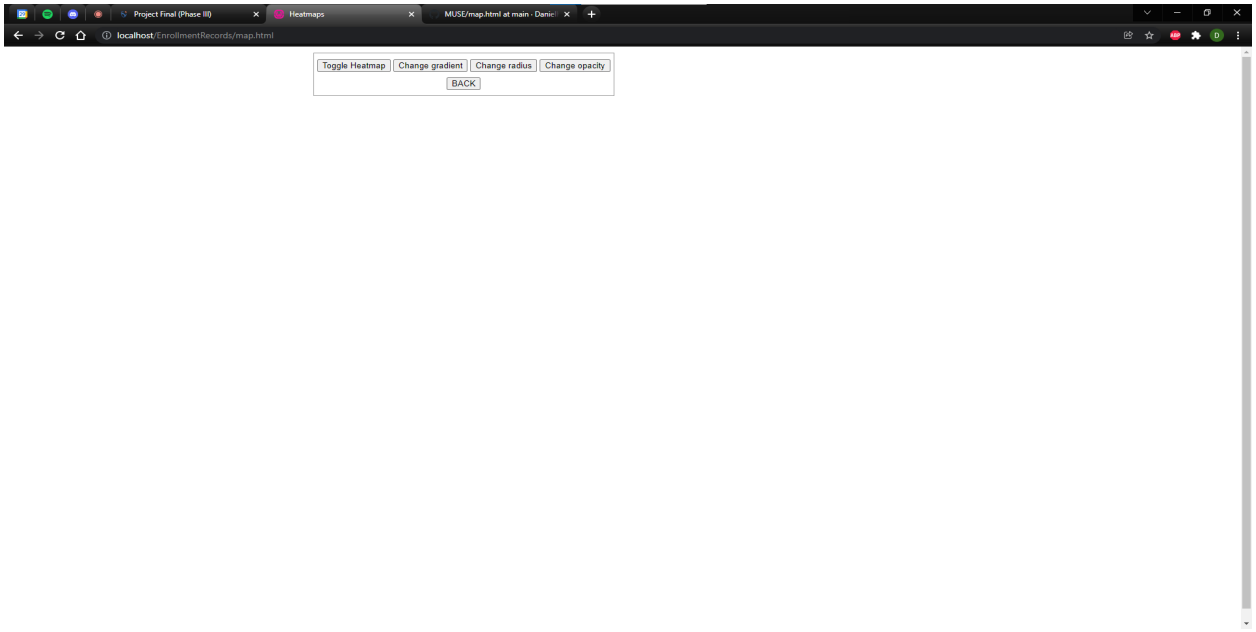


Figure 17 : before the map loads

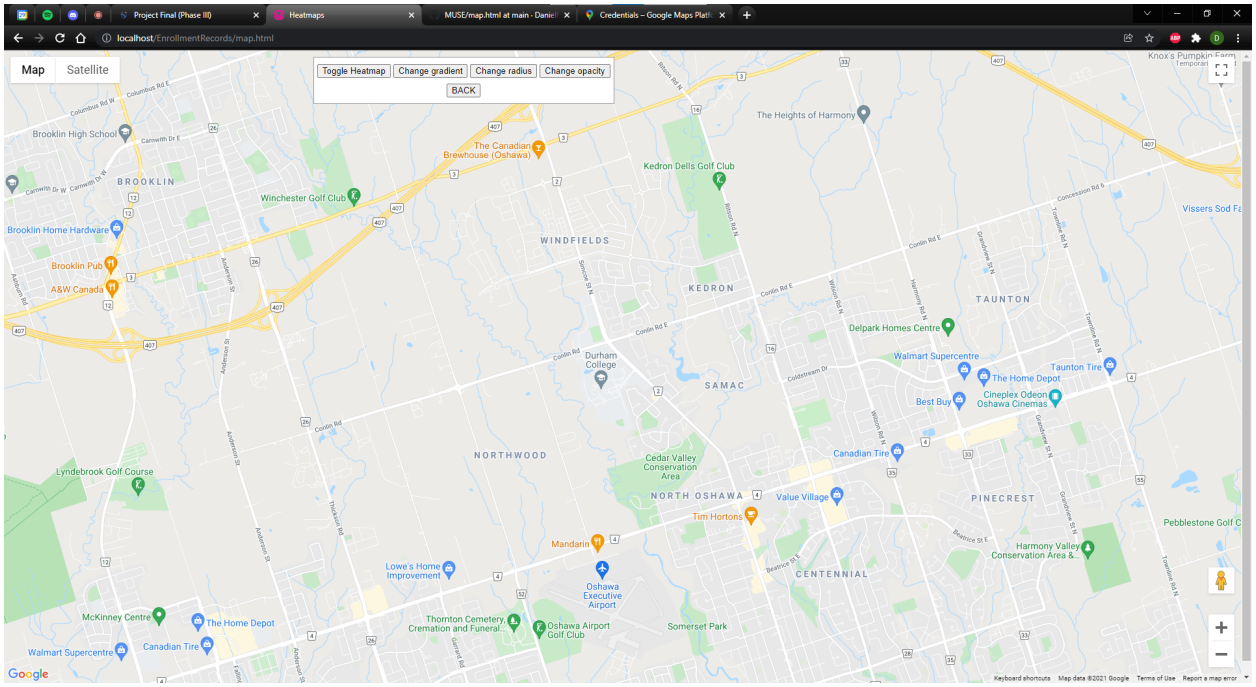


Figure 18 First load of the map

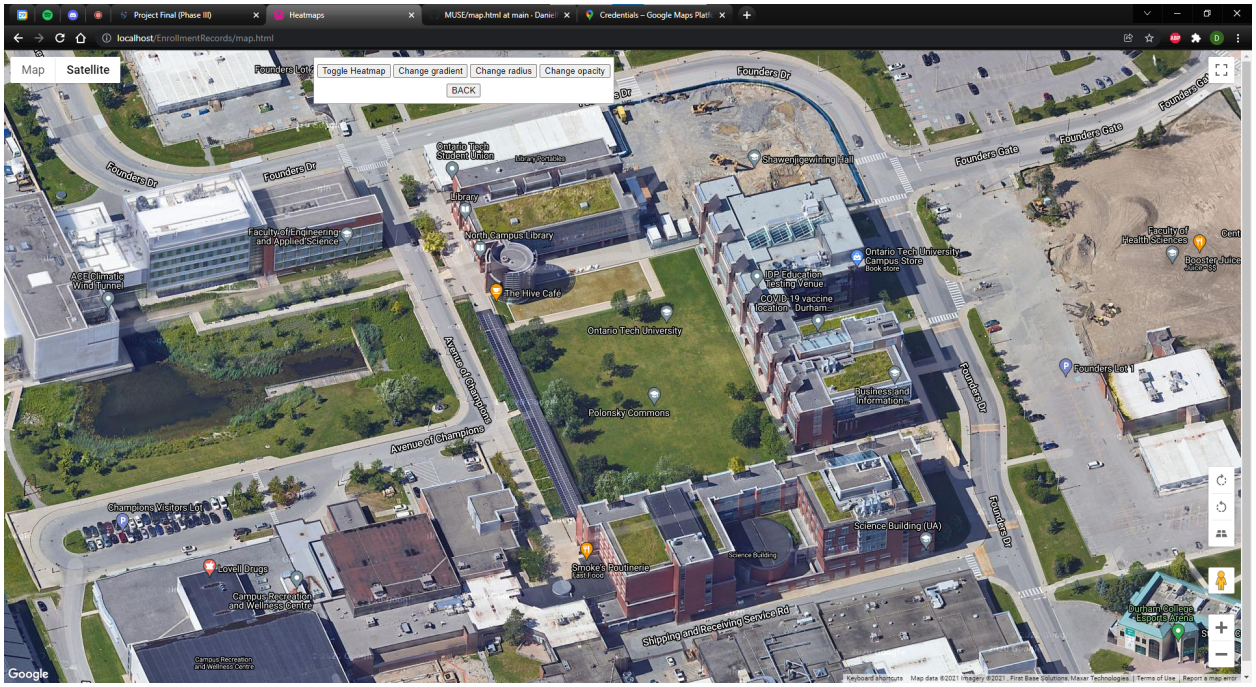


Figure 19: Street view

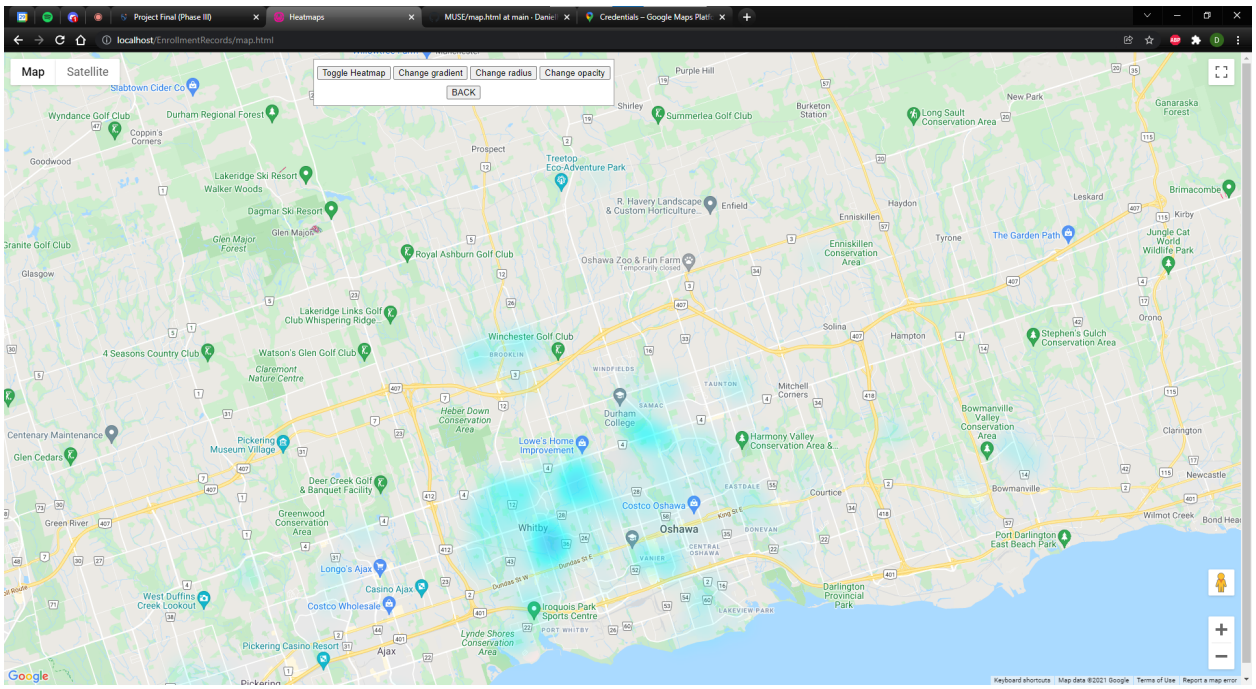


Figure 20.1: Heat map view

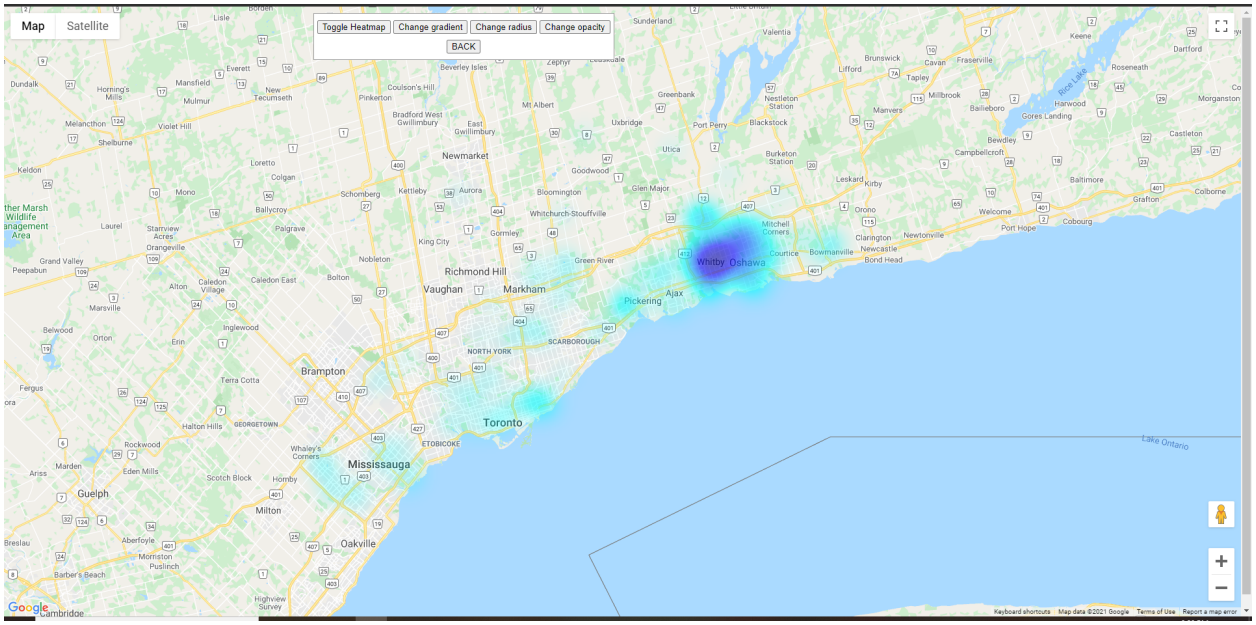


Figure 20.2: Heat map view #2

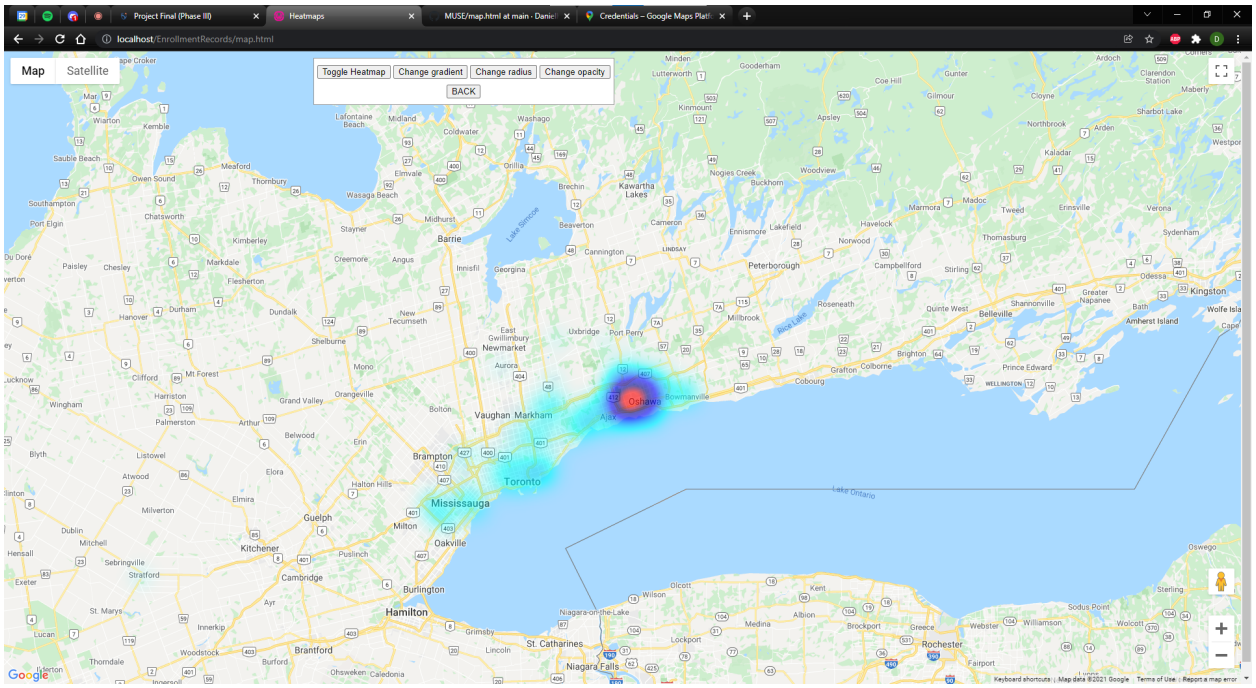


Figure 20.3: Heatmap view #3

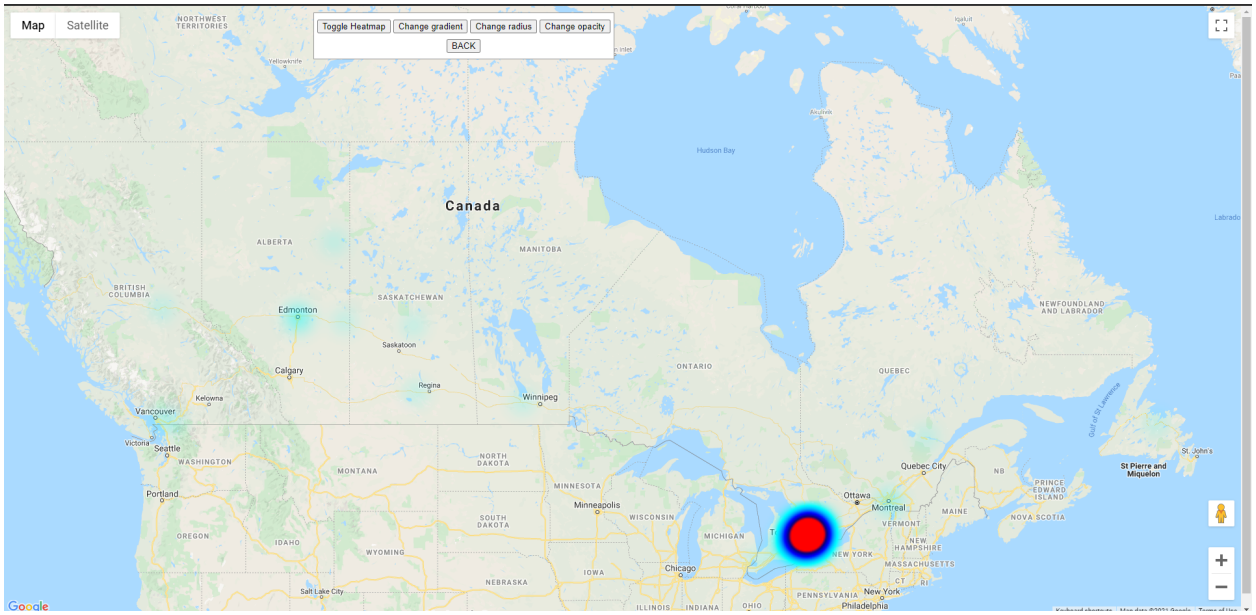


Figure 20.4 Heat map view #4

Contribution Matrix

Group 9 Task Distribution for Management of Enrollment Records

	Project Phase and Tasks Completed				
Team Member	**1** **Proposal** *October 17*	**2** **Schema, Tuples, Views, ER** *November 1 - 7*	**3** **(S.D. refactor, Mass Data Creation)** *November 15 - 22*	**4** **Implementation / Presentation** *November 23 - 25*	**5** **Final Report / Repository** *November 26 - 29*
Daniel Gohara Kamel	Overview: 10% Foreseen Prob.: 40% Related works: 2%	Schema diagram design: 95% S.D.implementation: 5% ER Diagram: 100%	Diagram Planning: 100% Diagram remaster: 15% New create statements: 15% Data: all students, edit all belongs_to, Redid views: 1, 2, 5, 7, 9	Index.php map.html map.php map2.js	Background: 15% System Design: 42% Observations: 3% References: 100% Appendix A: 50% Appendix B: 100% Appendix D: pg. 22-25
Jessica Leishman	Overview: 90% Objective: 100% Goals + Motiv.: 48% Foreseen Prob.: 22% Related works: 3%	S.D. design: 5% S.D.implementation: 95% Create statements: 100% Data tuples: 85%	Diagram remaster: 85% New create statements: 85% Redid views: 3, 4 using L/R join and union, 6, 10	home.php records.php deptStats.php all CSS Presentation slides System Navigation Flowchart, System Layer Diagram	Introduction: 100% Background: 28% System Design: 13% Use Case Diagram, Use Case Descriptions Conclusion: 80% Appendix A: 50% Appendix C: 100% Appendix D: pg. 17-21 Github README
Abida Choudhury	Related works: 95%	Views: 55%	Data: all staff, all belongs_to, all sections Redid view: 9		Background: 39% Observations: 97% Conclusion: 20%
Adris Azimi	Goals + Motiv.: 52% Foreseen Prob.: 38%	Views: 45% Data tuples: 15%	Data: all enrolled, all linked_sections		Background: 18% System Design: 50%