Faculty of Engineering and Applied Science

SOFE 3700U- Data Management Systems

Final Report

Project Group 13

Group Member 1

Name: <u>Sohaib Mohiuddin</u> 25%

Student ID: 100593657

Group Member 2

Name: <u>Umar Riaz</u> 25%

Student ID: 100600032

Group Member 3

Name: <u>Yusuf Shaik</u> 25%

Student ID: 100655451

Group Member 4

Name: <u>Curtis Whall</u> 25%

Student ID: 100655399

Date: November 29, 2019

CRN: 43511

Table of Contents

Abstract:

This project centralizes on the idea of having a single account for all different university applications. Currently, universities use separate platforms for personal information, and information pertaining to classes. Our database management system fixes that issue, and also integrates a compulsory review section for professors that students must complete. There are three types of accounts: Students, professors, and administrators. The main functionalities for students include viewing grades, updating personal information, viewing courses, and leaving reviews. The main functionalities for professors include viewing students grades, updating student grades, and viewing their reviews. The administrators can update information in both students and professors accounts, and also add/remove professors/students from/to the system/courses.

Introduction:

Students visiting universities for post secondary school and graduate studies are ever more abundant these days, and the more students who decide to attend schools, the greater the need for efficient, fully functioning, and robust systems. Knowing this, the problem that we decided to tackle was that of trying to make a system that meets the needs of all parties, while maintaining a simplistic approach to allow for easy access for users.

With the different age groups of students attending schools, as well as the age ranges of professors teaching, the importance of a system that requires little experience that awards maximum potential, is what this problem field requires.

This was a personal project for our team, being university students, the school database system is one that we interact with everyday, and the access of the information is vital to our learning. Trying to better the models that already exist to create something that we would be more inclined to use willingly, instead of out of necessity is our end goal.

The purpose of this report is to inform readers of the intricacies of our university database system, and accompany web application as well as the API interactions. The reader will be able to understand what can be expected while using the product, as well as the design process and decisions that were made during the creation.

This report will first explain the goals we had set in the beginning, and ultimately what we expect from our system. From there we will explain our work, including our design decisions and why the system is structured as it is. Next, we will list our diagrams and models that were used to design. Afterwards, we will list our ideas for future features and functionalities that we could not complete for the current iteration of the product. Next, we will list some related works, and any ideas we have taken from these. Lastly, we will acknowledge the relation back to our course and lectures as well as our results, finishing with a conclusion.

Goals:

Due to the nature of the demographic of the users we expect to serve, the simplicity and reliability of the web pages for our system were of the utmost importance. These web pages will give preset functions that the user can do, and will be expanded when needed. Our goal with our database is to allow for maximum security as well as query speed, intending for a seamless user experience that they can be enjoyed. We also hope to reduce the number of accounts that users have with the school down to one, allowing everything to be done through our single service. Lastly, with the API we hoped to integrate it to work as one with the database and web pages, also allowing for an easy and comfortable use experience. We also hoped to integrate most of the functionality into the project, allowing users to actually leave reviews, along with a functioning database management system.

Main body:

The whole product was made with ease of access and functionality in mind. Knowing this, we began our thought process with only ideas that came naturally, without many user requirements. With this in mind, we created some simple web pages that can interact with our database to allow for this ease of access. Along with our web pages we have our database, storing all the information regarding our students, classes, administrators, etc. This information can be seen in the screenshots of the tables in the database. We also have the python script that access information from the database, and creates an API.. Lastly we have our web application that uses openweathermap API that allows users to search current weather information based on location. The application also stores weather information and location in our weather table, and location and coordinates that have been searched into a coordinate table.

To begin, we have our database. This was the first component of the product was created, using a relational schema and an ER diagram, we modeled what tables we required, as well the relations between these tables. From here we created our database using SQL and MYSQL and populated with a few line of entry data. The purpose of this database is to store the information so that it can be easily accessed using certain SQL statements from a multitude of platforms. In our case, we can use PHP-myadmin as well as our web pages to return the results of our required queries. These results can be seen in the following images.

Next we implemented our web pages. To do so we used PHP, HTML as well as CSS and a little amount of Javascript. Originally we had considered Ruby, Node.JS and PHP but in the end, mostly due to prior experience we chose that PHP would be our best course of action. We already knew that the use of PHP would allow us to receive the results that we desired, while not having to learn a new language to program in. As HTML and CSS are the basic methods to achieve formatted web pages, these were also chosen.

One of the backbones of our web pages is the implementation of sessions. As we have various levels of users and permissions, as well as multiple users at once. The login page in the application consists of three input fields. One field is the input of the id of the member. The second field is the password. The third is the selection of what type of user the member is. All three of these fields are necessary in order for the member to login. The code is separated into three types of users which are the students, professors and admin. Depending on what type of user was selected, the id and the password are matched with the correct table. The corresponding query is selected if all the conditions are met. The three login tables are “slogin” for comparing the student ids entered and passwords, the “plogin” table which compares the professors data and finally an alogin table which compares the admin information. If successful login occurs the user is then redirected to the appropriate homepage. This structure ensures security of the members. A student cannot login to a professors homepage and a professor cannot login to a students homepage.. This also restricts the permissions of each member where they can only view/modify whatever is presented to them on that webpage. With the use of sessions, most queries that require input of certain SN’s or PN’s or AN’s instead can take the current user’s number from the session, instead of manual input.

As PHP accepts query statements directly, we used query statements that are basic SQL and will search through our database for our required fields. The only other requirement for this was the actual physical connection to the database, which is tied to the session as well. Lastly, the HTML and CSS is basic, allowing for a simplistic and accessible design. We mostly use forms for the submission of data or requests for the queries.

API's (JSON):

External API's:

Accessing external API:

In order to access the information from an external API, we created a web application using flask to access openweathermap API. The format returned from openweathermap is formatted using JSON. We used jinja2 as a templating language to insert the information from the python program in the html page. We parsed through the json formatted information and acquired the weather in a certain location based on the users input. We also display a picture with custom activity recommendations based on the weather in the users area. In the future, we plan to implement this into the school webservice so that users can see the weather in their area.

Please view **Fig.1** in the appendix to see an example of the website displaying the weather in Whitby, Ontario. **Fig.2** is an example of the website displaying the weather in Los Angeles, California.

Images are displayed based on temperature and location. Users also have an ***Activity*** button that they can click to see activities in their area, based on the temperature. The button searches google for *{weatherType} weather activities in {area}* and takes the users to that page.

Queries Associated with the Functionality:

1. **Weather and Location**

   Within the flask application, once accessing the information from the API, we store the weather and location searched into the weather table in our database, all searches are tracked. This will eventually create a database with all weather information in the cities that our students stay in. We can use the Apriori's algorithm with weather in the area that students live in, and their attendance rate to determine which weather conditions students actually show up to school on.

2. **Location, coordinates**

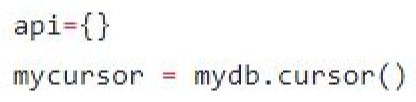
   Within the flask application, we also access the coordinates that users have searched on the web application. We parse through the JSON file to access the latitude and longitude of the searched area, and insert that information into our coordinates table. This information can be used to find all the students that live very far from school. Since the coordinates are the same for all cities that are close to each other (we implemented coordinates to 1 decimal place), we can find all students who live in cities with a difference of higher than 0.1, and conclude that those students live far from the university. This information can be useful to determine excused lates, or missed classes because of traffic in their commute.

JSON document generation (API Creation):

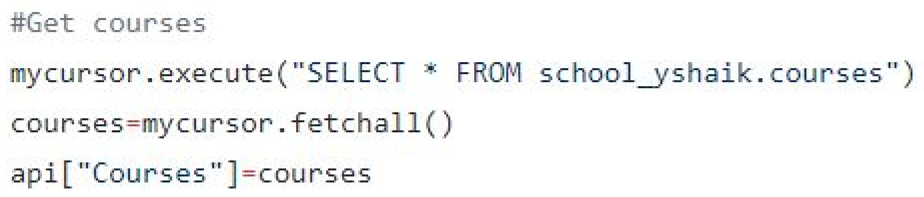
The JSON formatted API was created using python. We noticed that there were a few languages that, when accessing information from a database, returned the information in JSON format. This allows us to query the information, with minimal formatting Among those languages were Python, Ruby, Nodejs. After careful consideration, we decided on python because of simplicity, readability, and previous knowledge.

JSON formatting allows arrays, dictionaries, and tuples. When querying the database with python, the information is returned in those data structures only. In order to create the API, we utilised the hashmap data structure to associate various tables with the contents within each table. By initializing an empty hashmap at the beginning, as we queried each table, we were able to store the contents of each table with the key as the table name, and the value as the contents of each table.

For example:



```
api={}
mycursor = mydb.cursor()
```



```
#Get courses
mycursor.execute("SELECT * FROM school_yshaik.courses")
courses=mycursor.fetchall()
api["Courses"]=courses
```

The first line initializes the hashmap, and the two lines after *#Get Courses* queries the information from the database, and sets it equal to our courses variable. The last line adds the JSON formatted information from our database, into our hashmap with a key - *Courses.* Finally, the information is written to a JSON file. This file can be hosted anywhere online for the public to access the information in our database, while protecting it from injections. This is ultimately increasing our security, while also allowing others to access our source information.

Relation to Other work:

In order to create this idea, we did our research on common programs that completed similar tasks. Mainly we focused on the system that our university uses; Blackboard. During our experience as users of the Blackboard system, we found that the system was decent, but it could be better. As end users, we believe that we were able to design a system taking the best that Blackboard has to offer, along with our own ideas which would create the best system possible. While researching, we noticed that UOIT has 2 portals for their students: Blackboard, and MyCampus. Our first task was to rectify this issue, and allow both students, professors, and administrators to access all of their information through a single account. By doing this, we would have the functionality of both Blackboard and MyCampus in one system.

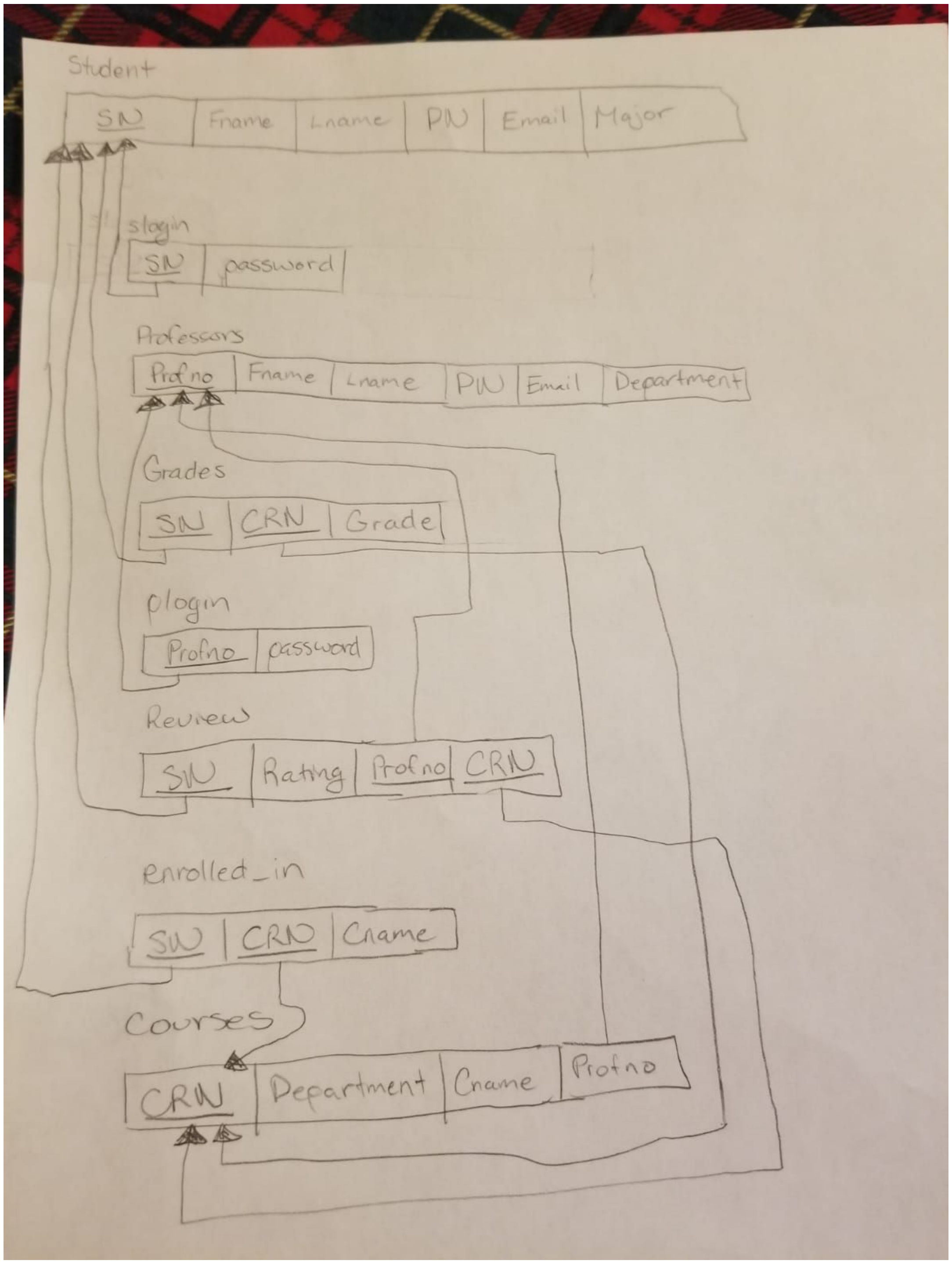
Furthermore, we found that Blackboard also requires an external system for professor reviews. In our database, we decided to add a review table that keeps track of all professors reviews, and allows students to leave a review for a professor at the end of the year. This could be integrated into their grades immediately (1 bonus mark for leaving a review). Currently, if professors wanted to give a mark to students who left a review, they would have to view the external webpage used for reviews, see the students who left a review, and upload marks manually. With our database, if someone were to leave a review, their identity would be protected, and their grade would be updated automatically, and hidden from the professor. In all cases, the professor would not be able to see who got the extra bonus mark in their class, because that specific information would be hidden from them. By integrating the review system into our program, we are able to control the amount of information encapsulated, thus, providing a higher level of security to each user's information.

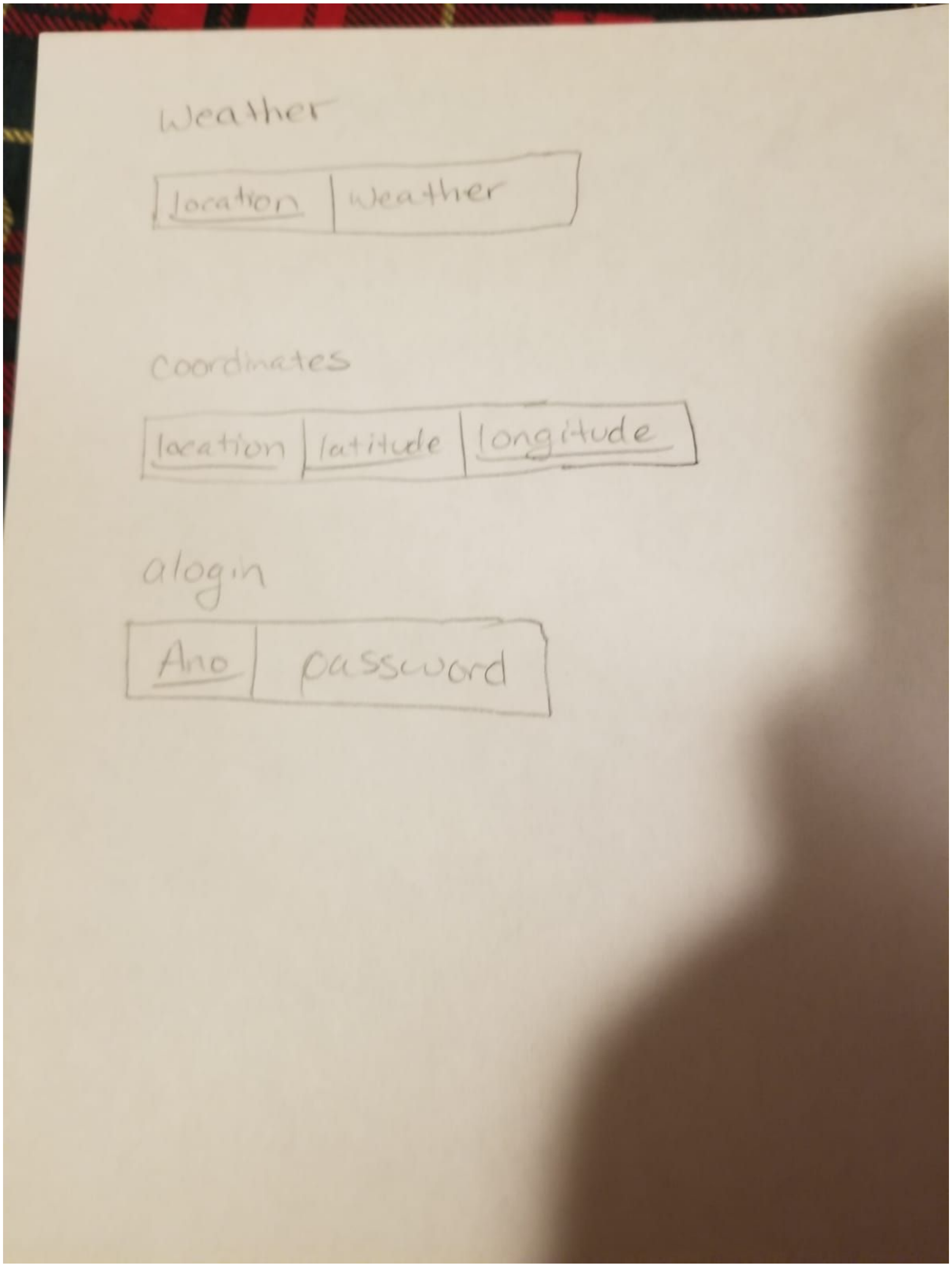
Contributions:

All members participated equally on all aspects of the project. From the initial concept to the final product, all group members actively participated in the meetings. Tasks were done in a timely manner as the work was distributed equally amongst the group members. At the beginning of the project, before our first meeting each group member was required to research the best way to implement this project. After a few days of research, each group member presented their ideas with pros and cons of each language, taking into consideration prior knowledge, ease of implementation, scalability, and maintainability. For this project, we considered, Python, Ruby on Rails, Node.js, and PHP. After careful consideration, we ultimately decided on nodejs, then ended up switching to PHP due to unforeseen issues. Each group member understood PHP well due to prior knowledge, so this was an advantage. For the coding of the application we divided the API creation, the web page creation, database management and the implementation of the queries into the application.

Since these were the four major tasks, we divided them amongst the four members. Curtis did queries, Umar did login the implementation of sessions and some views, Sohaib did views, and Yusuf did weather website and API creation. All members participated in creating a skeleton project then understanding of what our task were and we combined all the work once completed. After the original implementation, we all worked together to understand each others thought process and code. We also helped each other with improving our code, and addressed any issues we had. After creating a skeleton of our project, the teamwork we demonstrated played a vital part in producing our final result.

Schemas:


Student
SN
Fname
Lname
PIN
Email
Major
slogin
SN
password
Professors
Profno
Fname
Lname
PIN
Email
Department
Grades
SN
CRN
Grade
plogin
Profno
password
Review
SN
Rating
Profno
CRN
enrolled_in
SN
CRN
Cname
Courses
CRN
Department
Cname
Profno



Weather

location	weather

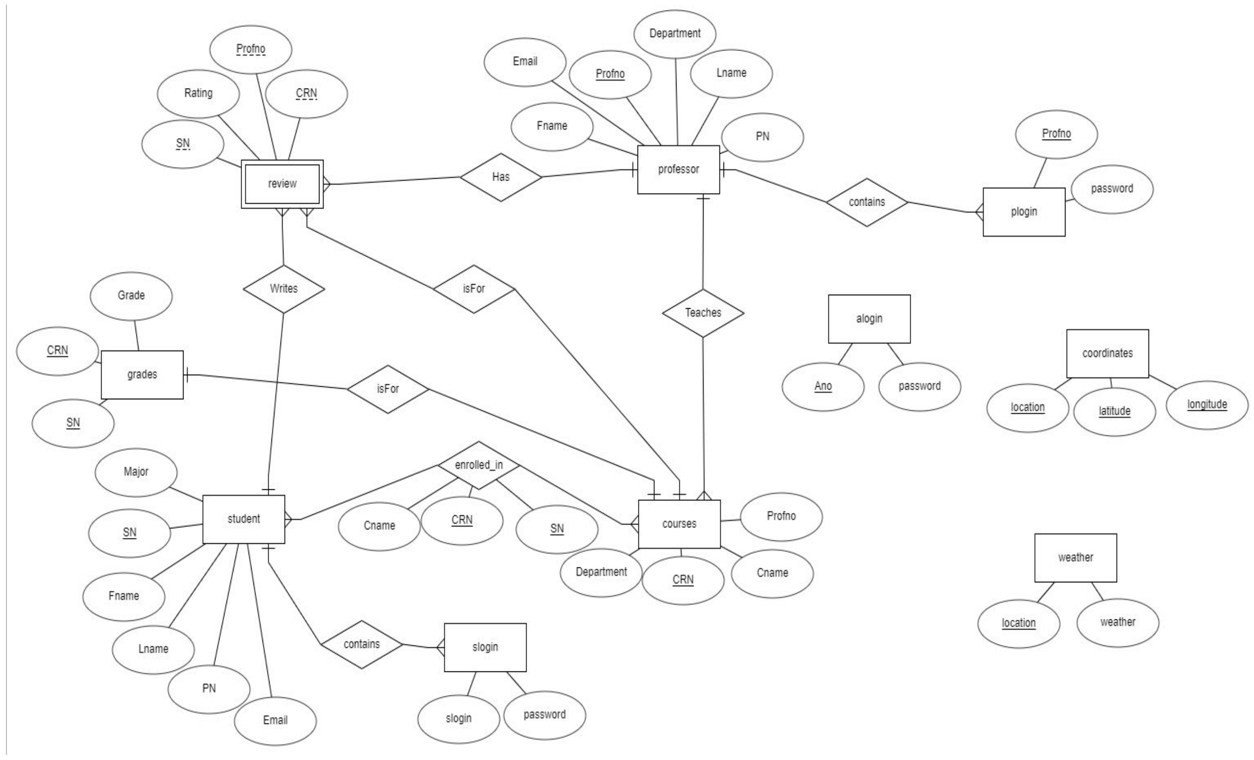
coordinates

location	latitude	longitude

alogin

Ano	password

Design diagrams:



Future work:

While we are quite happy with the progress we have made, there still are some areas that we would like to further develop and test. The product is not completely finished, and could do with some functionality and cosmetic changes. Firstly, overall the project needs a final sense of completion and polish. For example, instead of opening new windows on form submissions, with the use of AJAX and JQuery we could instead alter the contents of the current page without having to load a new web page with the query results. This could help improve the overall feel of operation of the system. Moving on, we would like to allow for the uploading and downloading of files so the students and professors have another way to interact. Professors would be able to upload lecture slides and assignments, and the students could submit work that is due. Another desired feature would be the ability for users to update their personal information while logged in. Due to this only really requiring a form that calls an update statement, this would be one of the easiest features to implement. In the future, we could add a schedule maker, as well as time slots for classes, allowing students to register for classes and have a created schedule to be able to view. With the use of the weather API, we would also like to collect data on when students are skipping class, and test if it is based on certain weather conditions or if they are unrelated. Lastly, more management options for the administrators would be a good addition, allowing for more control over the database and web pages.

Relationship to course:

This project has many relations back to our original data management systems course. Firstly, we followed the design process outlined in class, with the creation of a relational schema and ER diagram to originally map out our plan and structure of the database. From here, the creation of the database was similar to the method used in the lab, with some minor changes, as we mostly used PHPmyAdmin instead of MYSQL Workbench. Next, the query statements were based off of, and involved commands learned in class that were instead applied to our problem. The relation back can also be seen in our choice of web programming language that we used. This language can be seen in lecture, but was also still not a major portion of the lecture content. Overall, we managed to leverage the design process of creating a database, as well as commands from SQL and little amounts of the operation of PHP.

Results and analysis:

We successfully implemented the functionality of the system that we originally planned, even if we did switch languages along the way. After our careful planning, we successfully implemented the project in PHP, with all of our planned views, and more. We also created an additional weather application that stores information into our database, in a specific table that can be used for statistical analysis in the future. After careful analysis of our projects, we found that we exceeded our results, producing a much smarter database management system. We believe this was made possible due to our careful planning and consideration.

Conclusion:

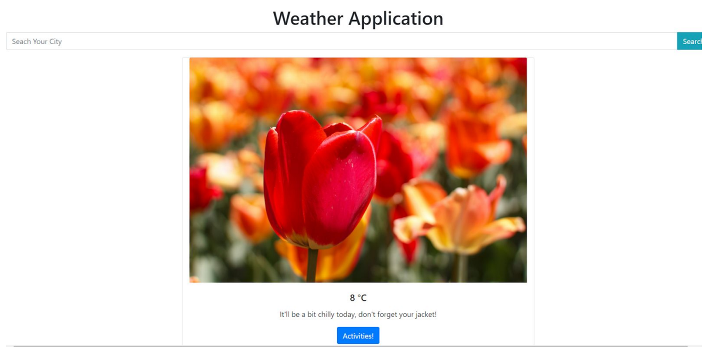
In conclusion, we have learned immensely from this project. Since we started from scratch, thinking of our own operation and functionality, we went through almost the entire process of creating and leveraging a database system. We honed our skills of creating ER diagrams and relational schemas, and learned even more of SQL creation and management commands. We were able to once again practice our web programming skills as well.

Not only learning, we met most of our goals and created a functioning product. Although not completely finished, it is a solid working prototype and will only get better once some of our proposed future additions are added.

In the end, we all had a ton of fun working on this project, and we learned a lot. We cannot wait to see where this experience can lead us to in the future.

Appendix:

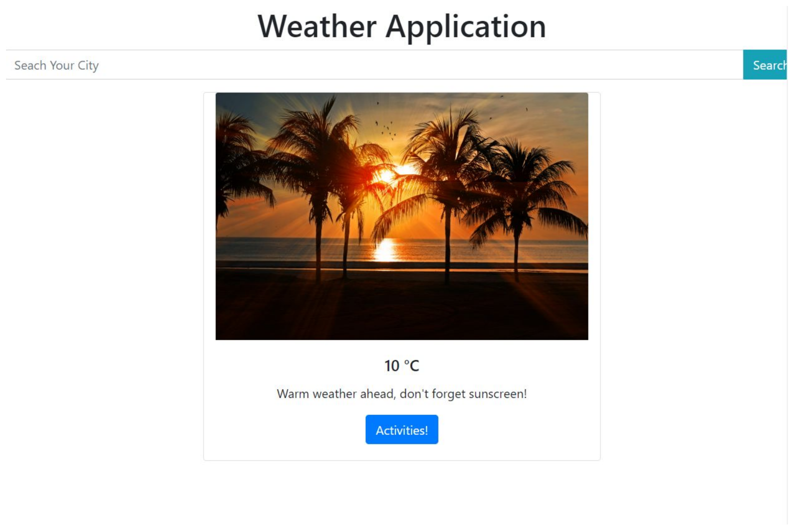
Fig.1: Weather in Whitby



8 °C

It'll be a bit chilly today, don't forget your jacket!

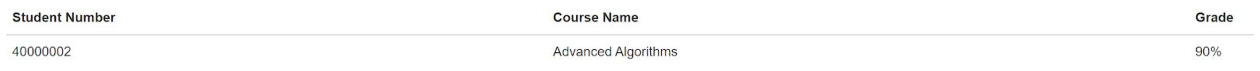
Fig.2: Weather in Los Angeles



10 °C

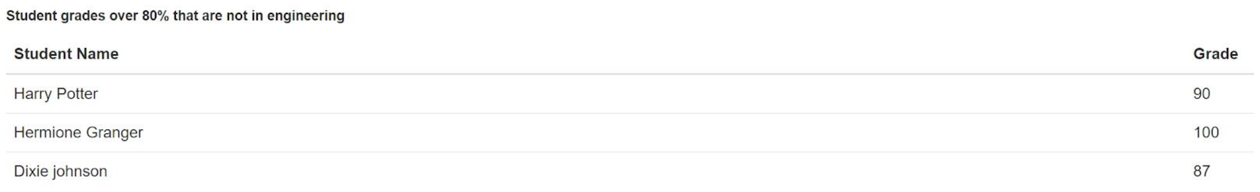
Warm weather ahead, don't forget sunscreen!

View 1: Grades of the student currently logged in



Student Number	Course Name	Grade
40000002	Advanced Algorithms	90%

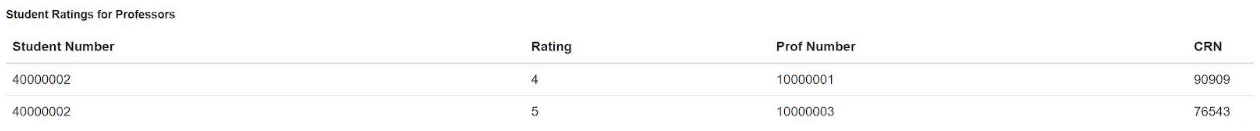
View 2: Grades of students that have over 80% but are not enrolled in Engineering



Student grades over 80% that are not in engineering

Student Name	Grade
Harry Potter	90
Hermione Granger	100
Dixie johnson	87

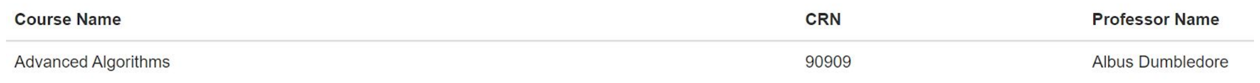
View 3: Reviews of any professor that the logged in student has submitted



Student Ratings for Professors

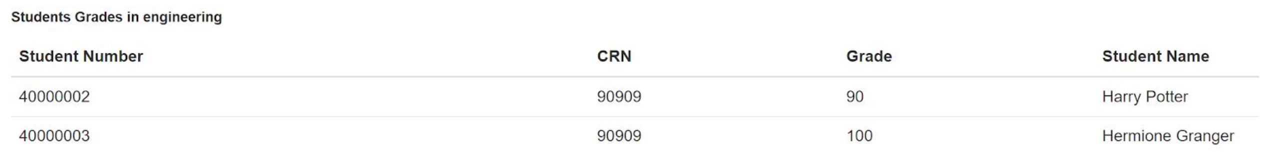
Student Number	Rating	Prof Number	CRN
40000002	4	10000001	90909
40000002	5	10000003	76543

View 4: Courses of the student that is currently logged in



Course Name	CRN	Professor Name
Advanced Algorithms	90909	Albus Dumbledore

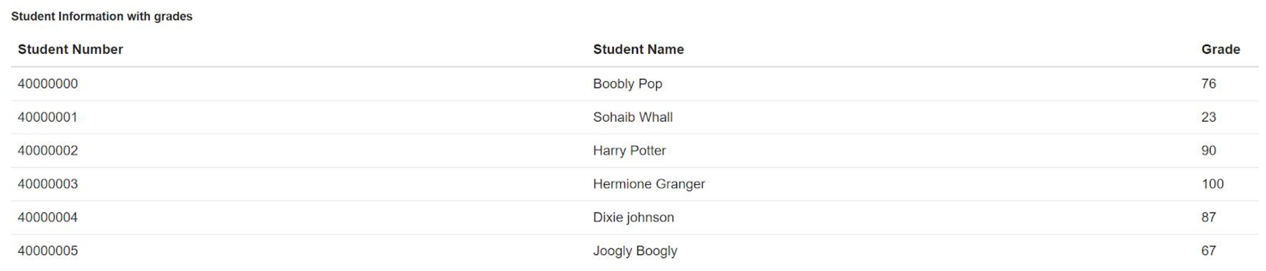
View 5: Grades of the students that are enrolled in Engineering



Students Grades in engineering

Student Number	CRN	Grade	Student Name
40000002	90909	90	Harry Potter
40000003	90909	100	Hermione Granger

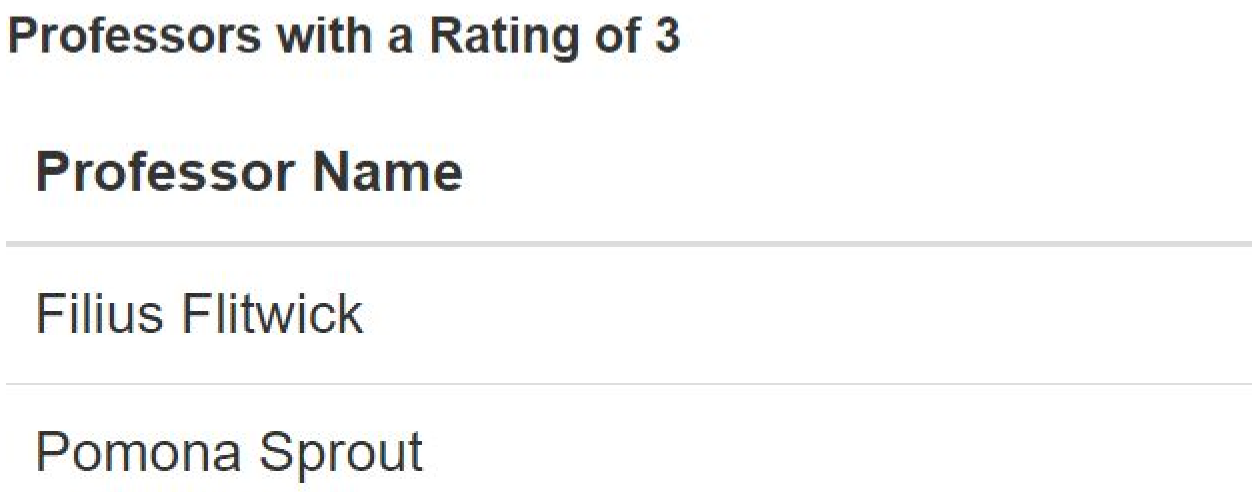
View 6: Full Join - Student information along with their cumulative grade



Student Information with grades

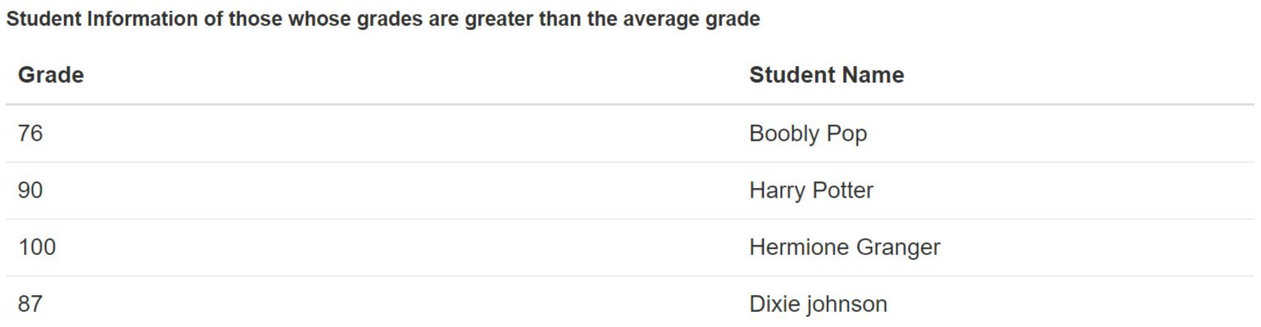
Student Number	Student Name	Grade
40000000	Boobly Pop	76
40000001	Sohaib Whall	23
40000002	Harry Potter	90
40000003	Hermione Granger	100
40000004	Dixie johnson	87
40000005	Joogly Boogly	67

View 7: Any professor that has a rating of 3


Professors with a Rating of 3

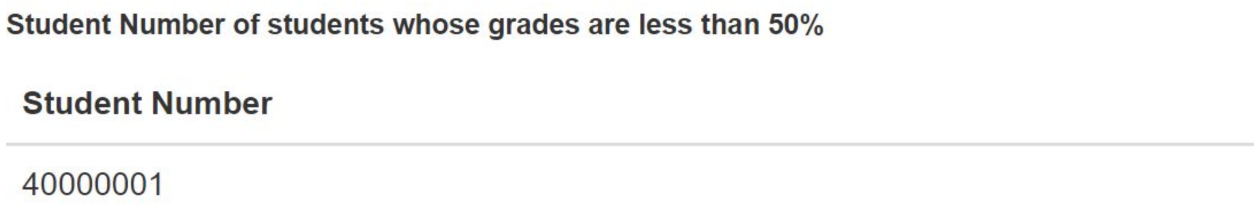
Professor Name
Filius Flitwick
Pomona Sprout

View 8:Student information and grade of those students that have a higher grade than the average grade


Student Information of those whose grades are greater than the average grade

Grade	Student Name
76	Boobly Pop
90	Harry Potter
100	Hermione Granger
87	Dixie johnson

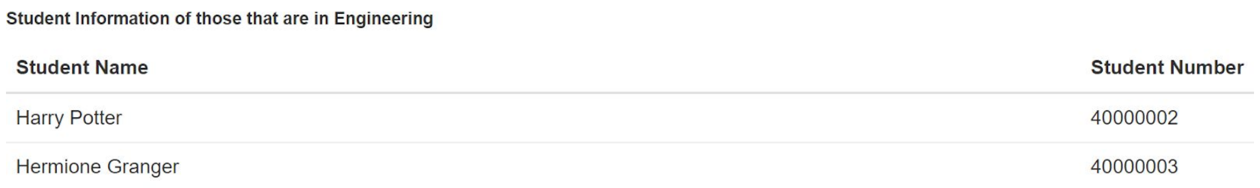
View 9: Student number of those students whose grades are less than 50%



Student Number of students whose grades are less than 50%

Student Number
40000001

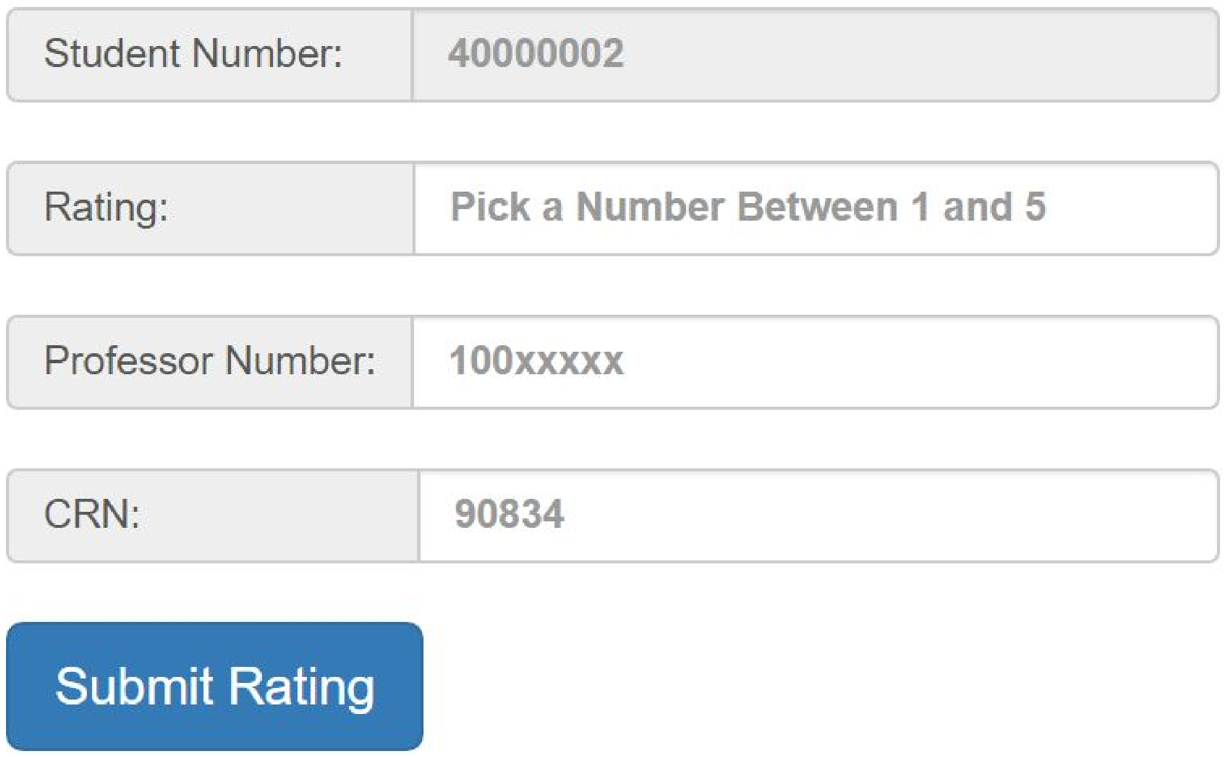
View 10: Student information of those students that are enrolled in Engineering



Student Information of those that are in Engineering

Student Name	Student Number
Harry Potter	40000002
Hermione Granger	40000003

Review Submission Form:



Student Number:	40000002
Rating:	Pick a Number Between 1 and 5
Professor Number:	100xxxxx
CRN:	90834