

CPSC 471: Database Management Systems  
Winter 2021  
Professor Reda Alhajj

# **University Management System**

## **Final Report**

### ***GROUP 17***

Jiexin (Jessica) Liu, 30055965  
Deliar Mohammadi, 30072994  
Navroop Chahal, 30045291

## **Abstract**

In many areas of the world, large, bustling post-secondary institutions are a ubiquitous feature of the urban landscape. They provide an intersection for thousands of students, hundreds of professors and support staff, and a large menu of courses and degrees to sample from. As a result, the quantity of data that must be continually gathered, maintained, and even deleted, is immense. Our project offers a response to this challenge in the form of a simple university management system supported by a relational database for scalability. Built on fundamental database design principles, we created an application using HTML/CSS, PHP, and MySQL (managed by MySQL Workbench), to accommodate two different end-users -- administrative staff, and students. First, administrative users may add students and courses to the system, and then subsequently, these registered students may enroll in, or drop courses. Both types of users can access our system through a mobile-friendly webpage currently hosted on a local machine; after logging in with their assigned ID and password, they can interact with various tables in the SQL database by adding or removing courses (or students, if the user is an administrator). These selections affect the state of the database in real time, and status changes are reported back to the user immediately. The system also sanitizes all user provided inputs to prevent SQL injection attempts. In total, the database includes 11 different relations, while the code base is organized into 11 different php files, 2 different css files, and 2 javascript files.

## **Introduction**

Before the development of the internet, most universities and colleges only had small scale systems in place to manage their student body and course offerings. These systems required students to register for courses via telephone or hand written letters which would then be recorded on paper by academic advisors. But as the desire for higher education increased to match demands of industry and research, these institutions had to expand both their enrollment capacity, as well as program offerings. Eventually it was no longer feasible to collect, manage, and update this data manually. Indeed, at post-secondary students today, most interactions with the school are now managed online, for instance, obtaining professor and course information, as well as performing course enrollment.

Being able to manage all these moving pieces in an internally consistent and scalable way while also offering a usable interface was the task we wanted to address with our database. The goal was to apply the tools and knowledge acquired from our CPSC 471 course to first develop a conceptual representation of post-secondary

systems through an extended ER diagram, which was later translated into a more concrete relational model. After creating an initial design draft regarding the technical implementation of our system, we were able to develop a user-friendly and scalable application to help address the challenges of managing students and courses, based on these earlier design choices and project requirements.

## **System Created**

We developed a system that caters not only to students, but also provides an interface for administrators to directly manage the components of the university database by adding and removing student and course entities as required. In accommodating both types of users, we aimed to build a system that is a basic approximation to what a real institution might use in their regular operations. The final product offers both a student and admin interface, each with different functionalities and privileges according to the roles each plays in the institution. And given the integration of an SQL database in the back-end, our system is also configured to be scalable, which solves the problem of quick and consistent management of large quantities of data.

Specifically, we created a front-end schedule builder and student management system, built on a back-end relational database. There is a built in API to connect the front-end and back-end aspects, to allow the website and database to communicate efficiently. All of the SQL queries use prepared statements to perform any operations related to the database. After a statement is prepared it is then binded with the correct parameters for that statement, and finally it is executed. This prevents any possible SQL injections since we don't directly query the database. Moreover, there are several filters that the user inputs must go through to satisfy sanitation requirements -- for instance, the length of the field or providing the correct data type (int, string, etc.).

The website features a password-based authentication login system that keeps track of every student's and administrator's credentials to log in and the ability to create a new account with the security feature to avoid creating fake and multiple accounts of the same user. As for students they get the ability to; enroll in a course, drop a course and view all the courses that are available for every semester. For administrators they get the ability to; add and remove a new course, and create and remove a student in their courses.

Our system was built with the following technical specifications:

- **Front-end:** HTML5, CSS, and JavaScript.
- **Back-end:** PHP for database connection, and MySQL with MySQL workbench as the management system.
- **Others:** The website runs on a local machine via an Apache server on XAMPP.

## **System Requirements**

Please refer to the user manual provided near the end of this report (page 15) for details on how to configure your system to run the provided code. It provides detailed instructions on requirements for XAMPP, PHP, and MySQL.

## **Summary of System Features**

### ***General features***

- Authenticated login screen to prevent unauthorized users from using the system -- login permissions are specific to students and admin
- Effectively organized SQL database which follows solid design principles
- Capacity to prevent SQL injection attacks
- Page footer which provides links to current University of Calgary academic information, general news, information about project creators, and a logout button.

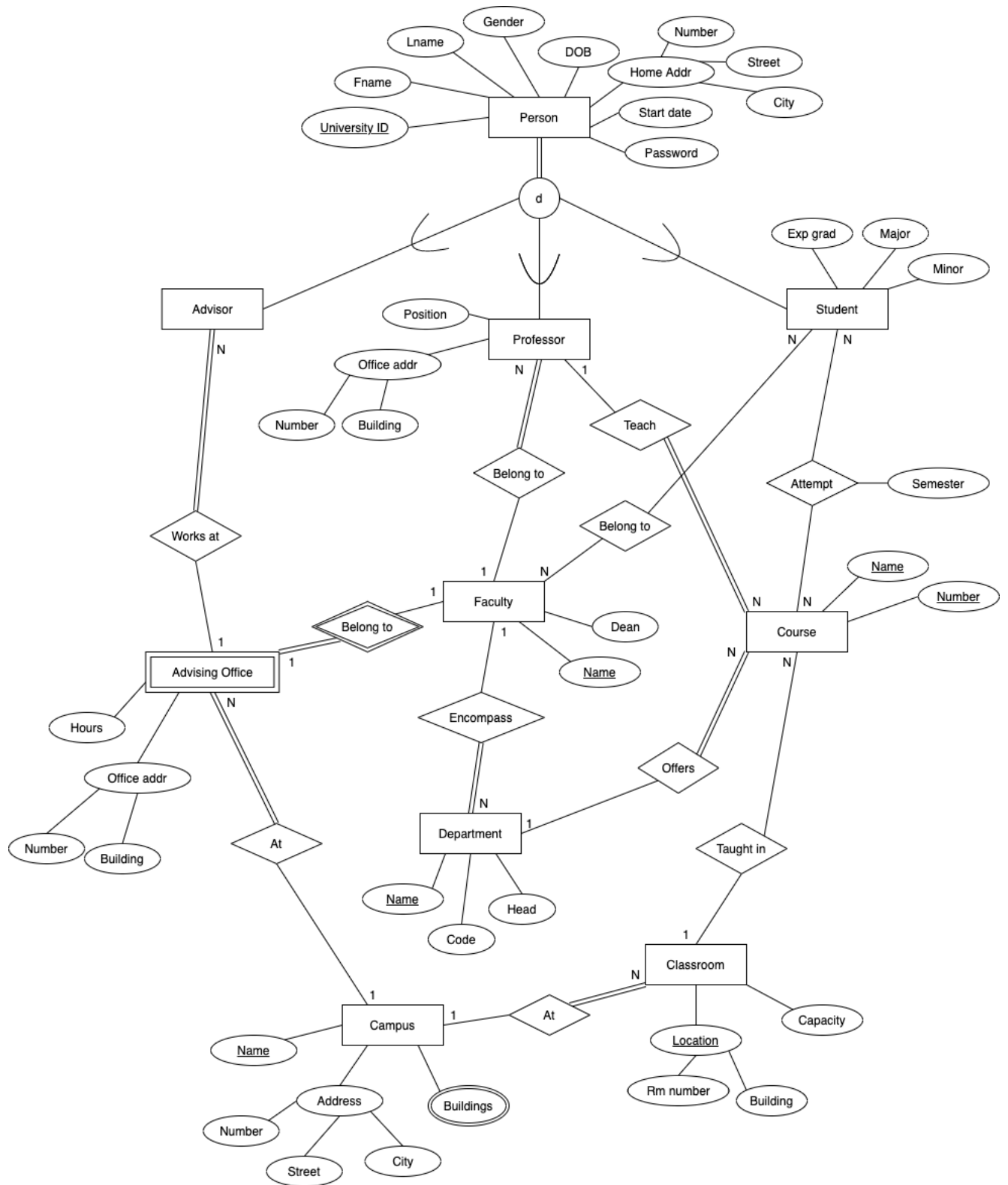
### ***Student user features***

- Option to enroll in, and drop courses
- Option to view list of all available courses each term

### ***Administration user features***

- Option to add and remove courses as an administrator
- Option to add and remove students as an administrator

## Project Design



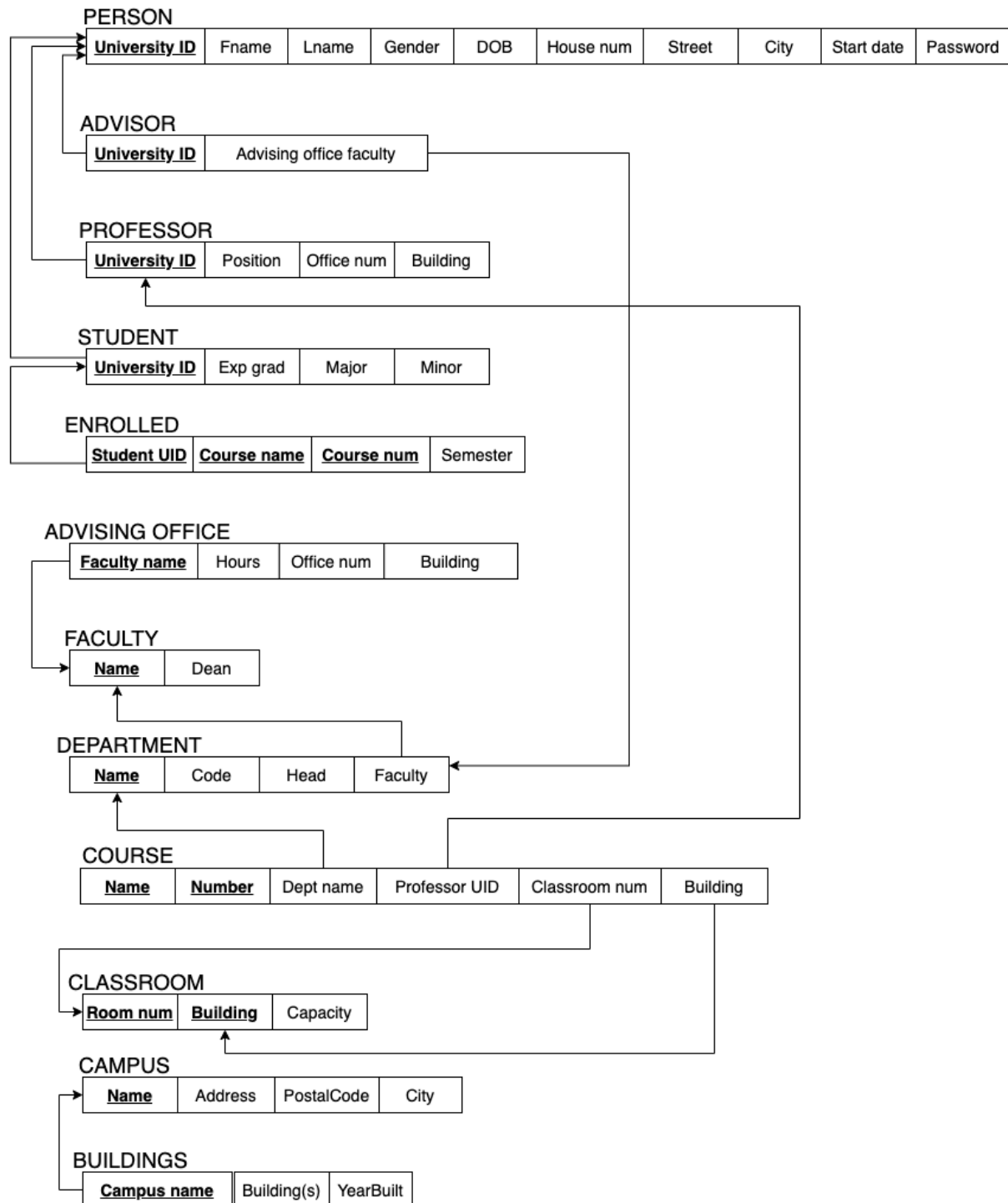
### ***Extended ER Diagram Design***

Our extended ER diagram was based on the main components we identified in a typical institution's academic operations. We focused on information that is essential to a student's course-taking experience, as well as the data that comes with enrolling each of these students at the institution. Because our conceptual design initially included students, advisors, and professors, we used the disjoint "Person" superclass to cover all three of these roles, with a few additional attributes for students and professors since they are the main actors in the day-to-day academic setting of a university. Additionally, we introduced "Advising Office" as a weak entity that depends on "Faculty", since each faculty often has their own set of specialized advisors.

### ***Implementation and End Users***

Although the ER diagram features three different types of people, our final implementation was streamlined to accommodate two users -- students and admin. The student end-user is a direct representation of the "Student" entity from the ER diagram, with the ability to view, add, or drop courses. However, the admin end-user was not modelled after any entity directly, rather, this user conceptually may represent an advisor or a professor. These users simply have the administrative powers to edit the student body and course list. And in the process of adding students and courses to the system, they then supply all other relevant data in our diagram, such as a student's name, ID, start date and expected graduation, assigned password, or a course's name, number, and professor, etc. This ensured that information and relationships from the above model were integrated in our implementation, so there was minimal deviation from the core structure and goal of our design. Note that, in our initial proposed ERD, there was an additional weak entity called "Grade Report" connected to "Student", as well as a self-referencing relationship called "Prerequisite" between the "Course" entity. However, we removed these aspects from our final implementation because they were non-essential to the main functionality of our management system. Additionally, we found that accounting for all prerequisites quickly became too complex for the scope of our project, and is something that can instead be considered in any future iterations.

## Implementation



**Notes on Implementation:**

The relational model was created directly from the ER diagram, using conversion techniques taught in the CPSC 471 class. The process was quite standard, with no unusual decisions made during the process.

**SQL Queries**

MySQL was used as the relational database management system for the entirety of this project.

The following queries are used to log users in and assert that they are authorized to deploy all other queries.

**Authentication**

Student Login:

```
SELECT * FROM student
WHERE UniversityID=$UniID AND
password = $pass;
```

Professor/Admin Login:

```
SELECT * FROM professor
WHERE UniversityID=$UniID AND
password = $pass;
```

***After logging in, there are several different queries, we will start with the student queries.***

**Enroll In a Course**

```
SELECT * FROM course
WHERE CourseNum=$CourseID
AND StudentUID=$studentID;
```



```
SELECT * FROM course
WHERE Number=$CourseID;
```

```
INSERT INTO enrolled (StudentUid, CourseName,
CourseNum, Semester)
VALUES
($studentID, $courseName, $courseID, $semester);
```

### **Drop a Course**

```
DELETE FROM enrolled
WHERE StudentUID=$studentNum
AND CourseNum=$courseNum;
```

### **View Courses**

```
SELECT * FROM course
ORDER BY DeptName;
```

```
SELECT *FROM person WHERE
UniversityID=$professorNum;
```

```
SELECT *FROM classroom WHERE
RoomNum=$room;
```

***We will now take a look at all the SQL statements for administrators/professors.***

### **Add a Course**

```
SELECT * FROM course
ORDER BY DeptName;
```

```
SELECT *FROM person WHERE
UniversityID=$professorNum;
```

```
SELECT *FROM classroom WHERE  
RoomNum=$room;
```

**Add a Student**

```
SELECT * FROM person  
WHERE UniversityID=$studentID;
```

```
INSERT INTO person  
(UniversityID, Fname, Name, Gender, DOB, StreetAddress,  
City, StartDate, Password)  
VALUES  
($IDnum, $Fname, $Lname, $gender, $dob, $address, $city,  
$startDate, $pass);
```

```
INSERT INTO student (UniversityID, ExpGrad, Major, Minor)  
VALUES  
($IDnum, $gradDate, $major, $minor);
```

**Remove a Student**

```
DELETE FROM person  
WHERE UniversityID=$studentID;
```

**Remove a Course**

```
DELETE FROM course  
WHERE Number=$Cnum;
```

## API Documentation and Sample Request & Response

Below, you will find the sample requests and their corresponding responses for all the endpoints. The project used plain queries for all requests to the database, I'm showing what the API requests and responses would look like had they been implemented with stored procedures.

### Student/Administrator Login:

#### Example Request:

```
curl --location --request GET 'http://localhost/471-final/frontend/login.php' \
--header 'Content-Type: application/json' \
--data-raw '{
    "UniversityID": "12078",
    "Password": "MINTY1"
}
```

#### Example Response:

```
{
    "Message": "Login Successful"
}
```

### Inserting A Course:

#### Example Request:

```
curl --location --request POST 'http://localhost/471-final/frontend/addCourse.php' \
--header 'Content-Type: application/json' \
--data-raw '{
    "Number": "471",
    "Name": "Database Management Systems",
    "ProfessorID": "42132",
    "Department": "Computer Science"
}
```

#### Example Response:

```
{
    "Message": "Successfully Added New Course"
}
```

**Deleting a Course:**Example Request:

```
curl --location --request DELETE 'http://localhost/471-final/frontend/removeCourse.php'\
--header 'Content-Type: application/json' \
--data-raw '{
    "Number": "471",
}'
```

Example Response:

```
{
    "Message" : "Successfully Deleted Course"
}
```

**Adding A Student:**Example Request:

```
curl --location --request POST 'http://localhost/471-final/frontend/AddStudent.php'\
--header 'Content-Type: application/json' \
--data-raw '{
    "UniversityID" : "12345",
    "Fname": "Susan",
    "Lname": "Johnson",
    "Gender": "Female",
    "DOB": "2001-02-03",
    "StreetAddress": "321 Rover St"
    "City" : "Calgary",
    "StartDate": "2020-01-01",
    "Password": "password"
}'
```

Example Response:

```
{
    "Message" : "Successfully Added New Student"
}
```

**Removing A Student:**Example Request:

```
curl --location --request DELETE
'http://localhost/471-final/frontend/RemoveStudent.php'\
--header 'Content-Type: application/json' \
--data-raw '{
    "UniversityID" : "12345",
}
```

Example Response:

```
{
    "Message" : "Successfully Removed Student"
}
```

**Enroll In A Course:**Example Request:

```
curl --location --request POST 'http://localhost/471-final/frontend/RemoveStudent.php'\
--header 'Content-Type: application/json' \
--data-raw '{
    "UniversityID" : "12345",
    "CourseNum": "471",
    "Semester": "Fall 2021"
}
```

Example Response:

```
{
    "Message" : "Successfully Enrolled In Course"
}
```

**Drop A Course:**Example Request:

```
curl --location --request DELETE
'http://localhost/471-final/frontend/RemoveStudent.php'\
--header 'Content-Type: application/json' \
--data-raw '{
    "UniversityID" : "12345",
    "CourseNum": "471",
}
```

```
}
```

Example Response:

```
{
  "Message" : "Successfully Dropped Course"
}
```

**View All Courses:**

Example Request:

```
curl --location --request GET 'http://localhost/471-final/frontend/Student.php'\
--header 'Content-Type: application/json' \
```

Example Response:

```
{
  {
    "CourseNum": "271",
    "CourseName": "Discrete Mathematics",
    "ProfessorUID": "12078",
    "DeptName": "Mathematics",
    "ClassroomNum": "120"
  }
  {
    "CourseNum": "471",
    "CourseName": "Database Management",
    "ProfessorUID": "12138",
    "DeptName": "Computer Science",
    "ClassroomNum": "80"
  }
}
```

## User Guide/Manual

### ***Requirements/Installations:***

1. XAMPP
2. PHP
3. MySQL

### ***Step 1) Configure XAMPP***

- Make sure that XAMPP is located in the C: drive after downloading
- Download the source code from Github and move it to the “htdocs” folder
- Start Apache and MySQL from the terminal

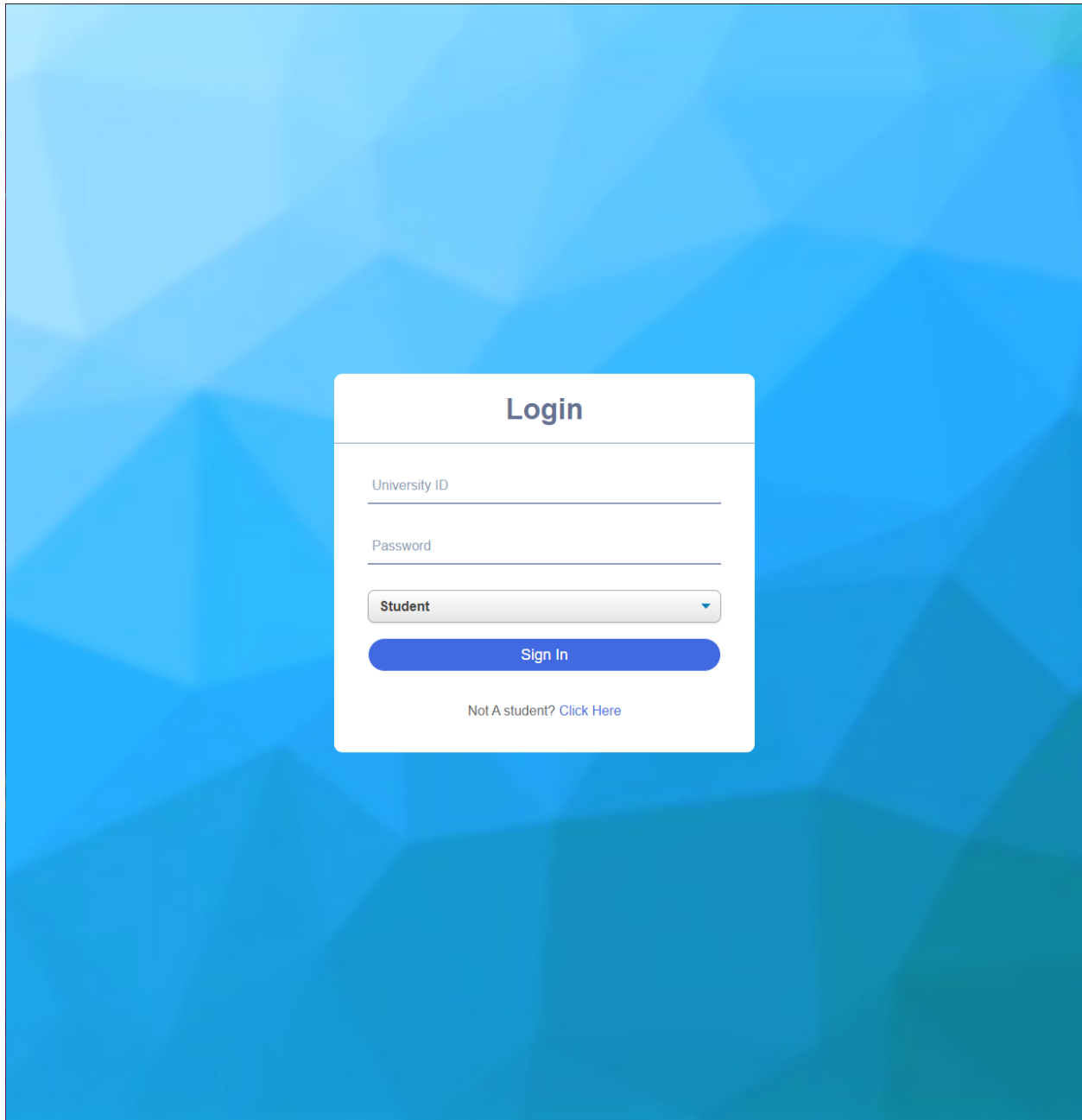
### ***Step 2) Create the database in your local machine***

- Create a database named “finalproject”
- Run the SQL code from github in a query
- Open C:\xampp\mysql\bin and locate my.ini, configure the settings to your local database credentials

### ***Step3) Run the webpage from your local machine***

- Open a browser (preferably google chrome)
- Paste the following into your search bar:  
<http://localhost/471-final/frontEnd/index.php>

After executing steps 1-3, you will be greeted with this login page:



The image shows a login page with a blue geometric background. In the center is a white login form with the title "Login". The form contains two input fields: "University ID" and "Password". Below these is a dropdown menu currently set to "Student". At the bottom of the form is a blue "Sign In" button. Below the button is a link that says "Not A student? Click Here".

**Login**

University ID

Password

Student

Sign In

Not A student? [Click Here](#)



After logging in with correct credentials you will be greeted by the “Add Course” page (For administrators/professors).

Hello Peter Parker	Add Course	Remove Course	Add Student	Remove Student
--------------------	------------	---------------	-------------	----------------

## Add a New Course

New Course Number

New Course Name

Select Department

Select Professor

Select Room and Building

Add Course



### About Us

[Navroop Chahal](#)  
[Deliar Mohammadi](#)  
[Jiexin \(Jessica\) Liu](#)

### Academics

[Ucalgary](#)  
[Department & Programs](#)  
[Undergraduate Studies](#)  
[University Library](#)

### UofC News

[News](#)  
[UThisWeek](#)  
[Student Union](#)

### External

[Logout](#)


## Remove Course Page:

Hello Peter Parker	Add Course	Remove Course	Add Student	Remove Student
--------------------	------------	---------------	-------------	----------------

### Remove A Course

Select Course To Be Removed

Remove Course

	<b>About Us</b> <a href="#">Navroop Chahal</a> <a href="#">Deliar Mohammadi</a> <a href="#">Jiexin (Jessica) Liu</a>	<b>Academics</b> <a href="#">Ucalgary</a> <a href="#">Department &amp; Programs</a> <a href="#">Undergraduate Studies</a> <a href="#">University Library</a>	<b>UofC News</b> <a href="#">News</a> <a href="#">UThisWeek</a> <a href="#">Student Union</a>	<b>External</b> <a href="#">Logout</a>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------	-------------------------------------------

## Add Student Page:

Hello Peter Parker	Add Course	Remove Course	Add Student	Remove Student
--------------------	------------	---------------	-------------	----------------

### Add A New Student

New ID Number

Student's First Name

Student's Last Name

Date Of Birth  
yyyy-mm-dd

Student's Street Address

City Of Residence

Starting Date  
yyyy-mm-dd

Expected Graduation  
yyyy-mm-dd


Student's Major

Student's Minor ("N/A" if unspecified)

Student's Password

Male

Add Student

	<b>About Us</b> <a href="#">Navroop Chahal</a> <a href="#">Delia Mohammadi</a> <a href="#">Jixin (Jessica) Liu</a>	<b>Academics</b> <a href="#">Ucalgary</a> <a href="#">Department &amp; Programs</a> <a href="#">Undergraduate Studies</a> <a href="#">University Library</a>	<b>UofC News</b> <a href="#">News</a> <a href="#">UThisWeek</a> <a href="#">Student Union</a>	<b>External</b> <a href="#">Logout</a>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------	-------------------------------------------


## Remove Student Page:

Hello Peter Parker	Add Course	Remove Course	Add Student	Remove Student
--------------------	------------	---------------	-------------	----------------

### Remove A Student

Select Student To Be Removed

Remove Student

	<b>About Us</b> <a href="#">Navroop Chahal</a> <a href="#">Deliar Mohammadi</a> <a href="#">Jiexin (Jessica) Liu</a>	<b>Academics</b> <a href="#">Ucalgary</a> <a href="#">Department &amp; Programs</a> <a href="#">Undergraduate Studies</a> <a href="#">University Library</a>	<b>UofC News</b> <a href="#">News</a> <a href="#">UThisWeek</a> <a href="#">Student Union</a>	<b>External</b> <a href="#">Logout</a>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------	-------------------------------------------

## Enroll In A Course Page:

Hello Lindsey johnson

Enroll in a Course

Drop a Course

View Courses

Enroll In A Course

Select A Course

Fall 2021

Enroll In Course

## Drop A Course Page:

## Drop A Course

Select A Course

Drop Course



## About Us

[Navroop Chahal](#)  
[Delia Mohammad](#)  
[Jiexin \(Jessica\) Liu](#)

## Academics

[Ucalgary](#)  
[Department & Programs](#)  
[Undergraduate Studies](#)  
[University Library](#)

## UofC News

[News](#)  
[UThisWeek](#)  
[Student Union](#)

## External


[Logout](#)

View All Courses Page:

Hello Lindsey Johnson
Enroll in a Course
Drop a Course
View Courses

### View All Courses

Department	Number	Name	Professor	Location
Computer Science	233	Introduction to Computer Science II	Martha Gonzalez	146, Engineering Building
Computer Science	329	Explorations in Information Security and Priv	Randy Dell	142, Science A
Computer Science	331	Data Structures, Algorithms, and Their Analys	John Locke	148, Science B
Computer Science	355	Computing Machinery I	Martha Gonzalez	80, ICT Building
Computer Science	409	History of Computation	Arnold Stewart	142, Science A
Computer Science	411	Compiler Construction	Arnold Stewart	123, Murray Fraser Hall
Computer Science	418	Introduction to Cryptography	Arnold Stewart	146, Engineering Building
Computer Science	231	Introduction to Computer I	Randy Dell	146, Engineering Building
Computer Science	433	Artificial Intelligence	Arnold Stewart	80, ICT Building
Computer Science	441	Computer Networks	Randy Dell	162, Mac Hall
Computer Science	525	Principles of Computer Security	Martin King	142, Science A
Computer Science	519	Introduction to Quantum Computation	Randy Dell	19, Engineering Building
Economics	431	Labour Economics	Dmitri Migrow	60, Engineering Building
Economics	328	Petroleum Economics	Alexis Finch	123, Murray Fraser Hall
Economics	453	Cost Benefit Analysis	Lucia Vojtassak	123, Murray Fraser Hall
Economics	349	Economics of Social Problems	Gordon James	19, Engineering Building
Economics	353	Chinese Economy	Alexis Finch	123, Murray Fraser Hall
Economics	481	Behavioural Economics	Dmitri Migrow	19, Engineering Building



About Us
[Navroop Chahal](#)  
[Deliar Mohammadi](#)  
[Jiexin \(Jessica\) Liu](#)

Academics
[Ucalgary](#)  
[Department & Programs](#)  
[Undergraduate Studies](#)  
[University Library](#)

UofC News
[News](#)  
[UThisWeek](#)  
[Student Union](#)

External
[Logout](#)

## References

- [1] - <https://d2l.ucalgary.ca/d2l/le/content/358051/viewContent/4452596/View>
- [2] - <https://www.w3schools.com/sql/>
- [3] - <https://www.youtube.com/watch?v=HXV3zeQKqGY>
- [4] - <https://www.khanacademy.org/computing/computer-programming/sql>
- [5] - *University Of Calgary Scheduler:*  
<https://www.ucalgary.ca/registrar/registration/schedule-builder>
- [6] - <https://www.youtube.com/watch?v=KUHrMzN7ey8&t=297s>
- [7] - *Elmasri R. , & Navathe, S.B. , Fundamentals Of Database Systems 7th edition.*
- [8] - <https://developer.mozilla.org/en-US/docs/Web/HTML>
- [9] - <https://www.w3schools.com/css/>
- [10] - <https://www.php.net/>
- [11] - [https://www.w3schools.com/howto/howto\\_js\\_redirect\\_webpage.asp](https://www.w3schools.com/howto/howto_js_redirect_webpage.asp)
- [12] - [https://www.w3schools.com/js/js\\_htmlDOM\\_html.asp](https://www.w3schools.com/js/js_htmlDOM_html.asp)