

# COURSE MANAGEET SYSTEM

Team 4

*Abdurrahman Mohammad  
Susmita Goswami  
Anudeep Gogineni*

# Project Overview

This is a prototype of a course management system that is similar to that used in colleges. It is a database application that displays course schedules, allows students to enroll in classes, and allows students to view their course history and transcripts.

## *Project Description*

This is a prototype of a course management system that is similar to that used in colleges. It is a database application that displays course schedules, allows students to enroll in classes, and allows students to view their course history and transcripts.

## *Goals*

The purpose of this application is to facilitate administration and student interactions for course enrollment and management. The goal is to make it easier for administrators to create, modify, and display course schedules just as how courses are displayed in a course schedule or catalog. Another goal is to simplify course viewing, course enrollment, and account history for students.

Administrators will be able to create courses offered by their institution and then with those courses, create a course schedule offered during the current term. Students will be able to view the current and past schedules and register for courses offered in the current term. Simply, the goal is to create a simple, user-friendly course management system that allows administrators to host classes and students to register.

## *Motivation*

Currently, course management systems are cluttered with unnecessary features and can sometimes become overwhelming. They are often hard to learn and adapt and look ugly. They often are buggy and are not user-friendly. An ideal course management system performs the basic task of displaying courses and allows students to register for these courses. Nowadays, course management systems have lost their focus and make the user experience extremely unpleasant when registering for courses. This system will solve these problems and will make it easier for administrators to host courses and students to add courses without much hassle. The system will make the user experience pleasant and will be robust and secure. Institutions need quality course management systems and there aren't many good ones. Another motivation for the development of this system is to create a simple course management system that can accommodate development for smaller institutions, businesses, non-profits, recreation centers, and community centers that offer courses but do not have a proper course management system that displays courses and allows students to register for their courses online. The motivation is to develop a product that is accessible and convenient for smaller public and private organizations that are in need of such a system like this.

## *Stakeholders*

The stakeholders of this system include the project drivers and designers, Abdurrahman Mohammad, Susmita Goswami, and Anudeep Gogineni. All future employees hired, such as web developers and Java software engineers, will also serve as stakeholders. The clients using this system are also stakeholders as well as their users, instructors, administrators - such as the school bursar -, employees, and students. Anyone responsible for the development, maintenance, and use of this system are stakeholders.

## *Application Domain*

The system will be used in the academic, the education industry, and private sector that offer courses and consequently have a need of such product. The system will be hosted on a web server. Interaction with this application will take place on a server which can be accessed via public or private URL. The system will contain an interface or GUI contacting the application and visualizing data and transactions. Data will be securely stored in a database on the server. The middleman between the database and the GUI will be the actual application responsible for conducting the tasks which lies between the interface and the database.

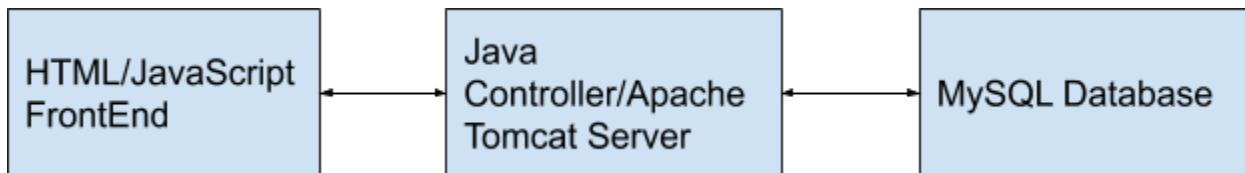
## *Benefits to Users*

This course management system will be fast, secure, reliable, clutter-free, easy to update, and simple to use which is not the case for most course management systems today. It will be simple to learn and will make it easier for administrators and students to conduct important course management activities. Additional benefit in the future may include students being able to view available classes and filter based on prerequisites and instructors being able to specify prerequisites and control enrollment with fewer manual overrides.

## **System Environment**

### *Structure of the system (graph based on client-server/3-tiered architecture)*

This is the 3-tiered architecture diagram of the system we plan to build:



## *Hardware/Software Used*

- For the initial release, the listed hardware will be used:
- Macbook Air 2013 and Macbook Air 2017
- IdeaPad Flex

For the initial release, the listed software will be used:

- Apache Tomcat - For hosting the server
- MySQL and MySQL Workbench - For database portion

For the initial release, the listed software tools will be used:

- MacOS Catalina (Version 10.15.5)
- Windows 10
- Eclipse

## *RDBMS Used*

- MySQL Community Server 8.0.20

## *Application languages*

- HTML
- CSS
- Java
- SQL

# Project Requirements

## **Functional Requirements (based on database manipulating activities)**

Admin can control the whole system. Instructors and students need to be able to access the system. Courses need to be added and students should be able to add/drop a course. The application should track when a course is full. Wait listing of students on a course should be allowed. The system should be able to calculate student dues to pass it on to the bursar fee payment system.

### *A list of detailed descriptions of users and how users interact with this application*

All users will enter the URL in their browser which will lead them to the server where the system is running. The users then log into their respective accounts using the sign-in page. New students can create an account by clicking on a link which will redirect them to create a new account page. After logging in, students, instructors, and administrators will have their own pages which allow different functions. Users can perform functions and make desired changes through the portal and the portal will contact the application which will contact the database and store the changes.

Administrators can control the courses, schedules, students, and instructors in the system, can modify system information and can call specific system functions such as suspend. Instructors may manage their course rosters by removing or adding students. Students can add, drop, and waitlist courses as well as view course history and financial transactions.

There are four categories of users for this application:

1. Visitors
  - People who are not registered with the system. They can register with the system by entering their information and an admin will email them their account credentials.
2. Administrative users
  - Administrators are responsible for the creation, deletion, and update of courses, course configurations, members, students, instructors, and other admins.
3. Instructor users
  - Instructors are responsible for managing registered and waitlist students for courses taught. They can override the system and manually add students from the waitlist or using the student's ID. They can drop students from the registration and waitlist for courses taught.
4. Student users
  - Students can register for courses or waitlist if class is filled, drop courses, view course history and transcript, view offered course and, view financial transactions.

*Description of each individual function/feature, functional process, and I/O.*

## Visitors

### *Register in the system*

Visitors go to the homepage. They click the “Register” button in the navigation bar.



They will then be directed to a form. The form will only submit if the fields are not null and if the account type is either ‘student’, ‘instructor’, or ‘administrator’.

## Register

First Name

Last Name

Phone

Email

Address

Type (student, instructor, administrator)

If wrong information is entered, they will get back to the form. If the requirements were met, they will get this confirmation message. An admin will soon email the account credentials.

**Registration successfully submitted!**

An administrator will be in touch with you shortly

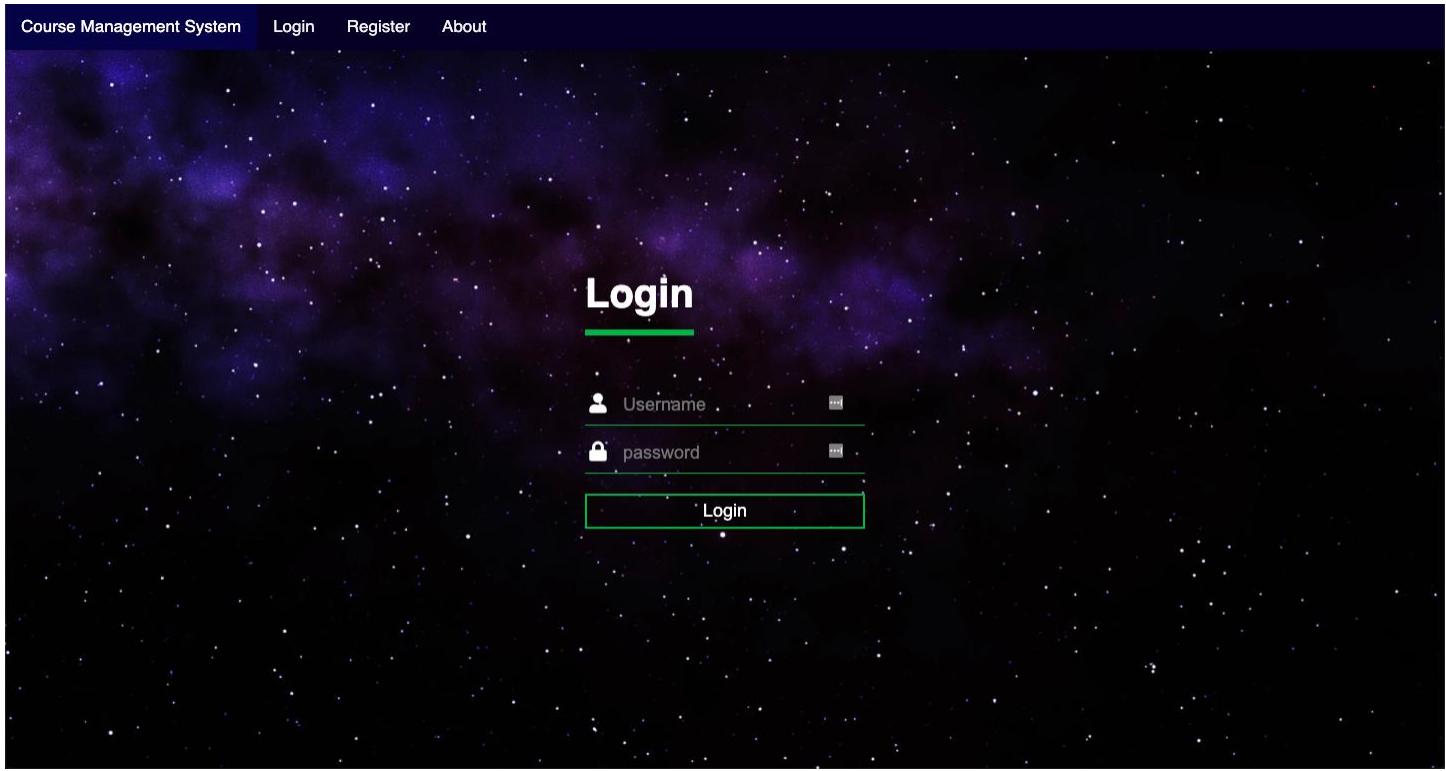
You will receive your username and password in your email

Please return to the homepage

The visitor can now get back to the homepage or leave.

## *Log in*

Visitors go to the homepage and then click the ‘Login’ button in the navigation bar. They will be redirected to this page.

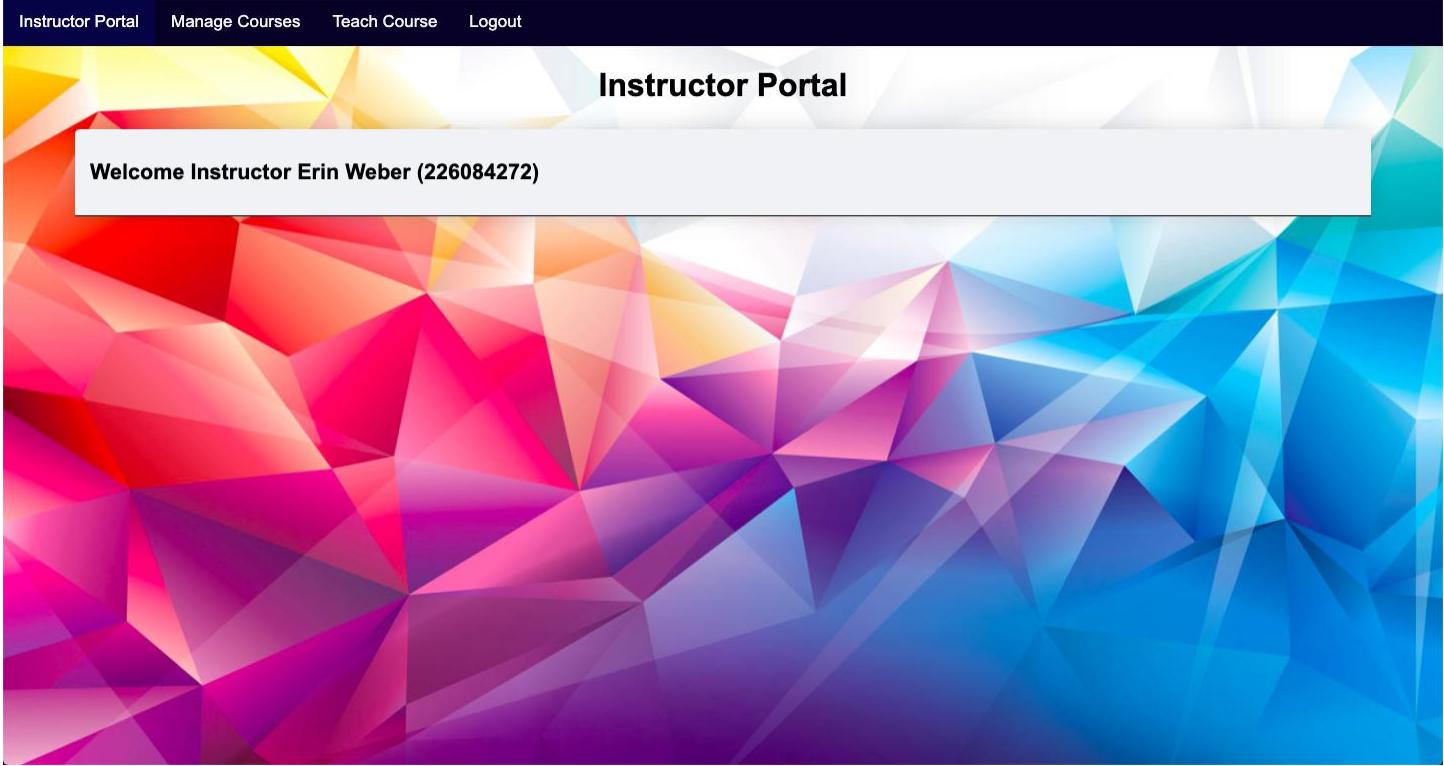


Then they enter credentials and be redirected to a custom portal depending on their user type.

## **Student Portal**

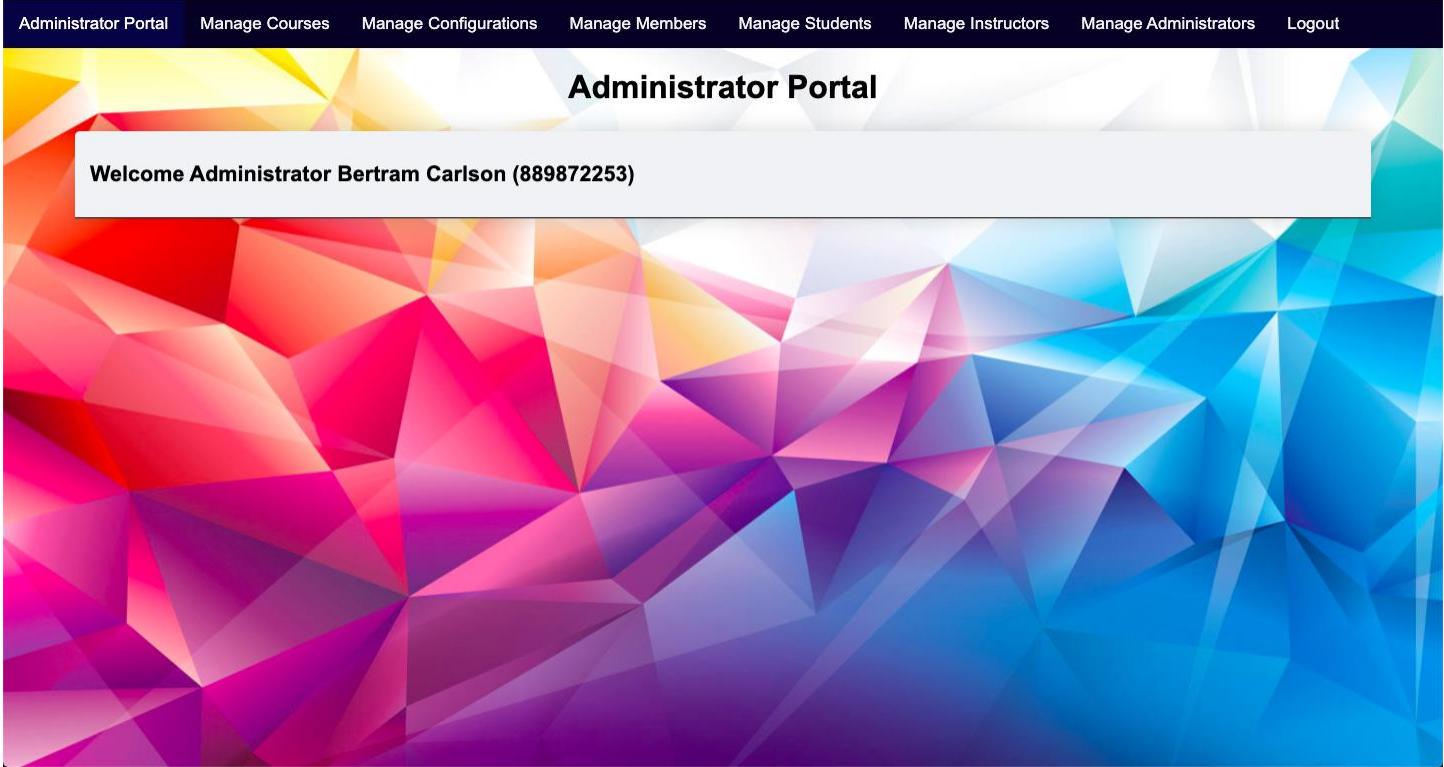
A screenshot of a student portal home page. At the top, there is a dark header bar with links for "Student Portal", "Add Courses", "Drop Courses", "View Courses", "Transcript", "Transactions", and "Logout". The main background features a vibrant, abstract geometric pattern of triangles in shades of red, orange, yellow, pink, purple, blue, and white. In the upper left area, there is a white rectangular box containing the text "Welcome Student Pasquale Nelson (199526438)". The overall design is modern and visually appealing.

## Instructor Portal:



The screenshot shows the Instructor Portal interface. At the top, there is a dark navigation bar with white text containing links: "Instructor Portal", "Manage Courses", "Teach Course", and "Logout". Below the navigation bar is a large, colorful abstract geometric background composed of triangles in shades of red, orange, yellow, purple, blue, and green. In the upper left area of the background, there is a white rectangular box with a black border. Inside this box, the text "Instructor Portal" is centered in bold black font. Below this, another white box contains the welcome message "Welcome Instructor Erin Weber (226084272)" in a smaller black font.

## Administrator Portal:



The screenshot shows the Administrator Portal interface. At the top, there is a dark navigation bar with white text containing links: "Administrator Portal", "Manage Courses", "Manage Configurations", "Manage Members", "Manage Students", "Manage Instructors", "Manage Administrators", and "Logout". Below the navigation bar is a large, colorful abstract geometric background composed of triangles in shades of red, orange, yellow, purple, blue, and green. In the upper left area of the background, there is a white rectangular box with a black border. Inside this box, the text "Administrator Portal" is centered in bold black font. Below this, another white box contains the welcome message "Welcome Administrator Bertram Carlson (889872253)" in a smaller black font.

## Administrators

After logging into the portal, an administrator can do certain tasks depending on the clearance.

## Manage courses – clearance 1 or 3

Click ‘Manage Courses’.

### View existing courses

A list of existing course is displayed

Department	Number	Title	Units	Cost	Update	Delete
BIO	1A	Biology I	5	230	<button>Update</button>	<button>Delete</button>
BIO	1B	Biology II	5	230	<button>Update</button>	<button>Delete</button>
CHEM	1A	Chemistry I	5	230	<button>Update</button>	<button>Delete</button>
CHEM	1B	Chemistry II	5	230	<button>Update</button>	<button>Delete</button>
COMM	2	Public Communication	3	138	<button>Update</button>	<button>Delete</button>
CS	146	Data Structures	4	184	<button>Update</button>	<button>Delete</button>
CS	147	Computer Architecture	4	184	<button>Update</button>	<button>Delete</button>
CS	149	Operating Systems	4	184	<button>Update</button>	<button>Delete</button>
CS	157A	Database	4	184	<button>Update</button>	<button>Delete</button>
CS	166	Information Security	4	184	<button>Update</button>	<button>Delete</button>
ENG	1A	Language & Composition	3	138	<button>Update</button>	<button>Delete</button>
ENG	1B	Literature	3	138	<button>Update</button>	<button>Delete</button>
GEO	2A	Introduction to Geology I	3	138	<button>Update</button>	<button>Delete</button>

### Create course

At the bottom of the page, you may enter new fields to create a course. If the course already exists, it will not be duplicated.

Create Course

Department	<input type="text"/>
Number	<input type="text"/>
Title	<input type="text"/>
Units	<input type="text"/>
Cost	<input type="text"/>
<input type="button" value="Submit"/>	

### Delete course

Click the delete button and the course will disappear from the list and database.

### Update course

Click the ‘Update’ button next to a course. You will be redirected to a page with the course’s info filled in. Modify the course. The course will update if the department and number do not conflict with another course.

## New Values

Department	<input type="text" value="BIO"/> <input type="button" value="..."/>
Number	<input type="text" value="1A"/>
Title	<input type="text" value="Biology I"/>
Units	<input type="text" value="5"/>
Cost	<input type="text" value="230"/>

*Manage course configurations – clearance 1 or 3*

### *View existing configurations*

Click ‘Manage Configurations’ You will see a page with all the existing configurations.

## Manage Configurations

ConfigID	Term	Year	Days	Time	Room	Seats	<a href="#">Update</a>	<a href="#">Delete</a>
1	Spring	2021	MW	4:00 - 6:00	41	31	<a href="#">Update</a>	<a href="#">Delete</a>
2	Summer	2021	TR	6:30-8:00	33	60	<a href="#">Update</a>	<a href="#">Delete</a>
3	Summer	2021	MWTR	9:00-10:30	34	56	<a href="#">Update</a>	<a href="#">Delete</a>
4	Summer	2021	MW	8:00-9:30	22	52	<a href="#">Update</a>	<a href="#">Delete</a>
5	Summer	2021	MWT	6:30-8:00	40	54	<a href="#">Update</a>	<a href="#">Delete</a>
6	Fall	2020	MW	1:30-3:00	50	53	<a href="#">Update</a>	<a href="#">Delete</a>
7	Fall	2020	TR	10:00-11:30	25	51	<a href="#">Update</a>	<a href="#">Delete</a>
8	Fall	2020	MWTR	8:00-9:30	18	35	<a href="#">Update</a>	<a href="#">Delete</a>
9	Fall	2020	MWF	9:30-11:00	18	33	<a href="#">Update</a>	<a href="#">Delete</a>
10	Winter	2020	MW	3:00-4:30	50	54	<a href="#">Update</a>	<a href="#">Delete</a>
11	Winter	2020	TR	6:30-8:00	24	46	<a href="#">Update</a>	<a href="#">Delete</a>
12	Winter	2020	MWTR	9:00-10:30	23	53	<a href="#">Update</a>	<a href="#">Delete</a>
13	Winter	2020	MW	3:00-4:30	48	48	<a href="#">Update</a>	<a href="#">Delete</a>

### Create configurations

At the bottom of the page you will see a form to add a configuration. Type in the desired information and click submit to create and add the configuration.

#### Create Configuration

Term	<input type="text"/>
Year	<input type="text"/>
Days	<input type="text"/>
Time	<input type="text"/>
Room	<input type="text"/>
Seats	<input type="text"/>
<input type="button" value="Submit"/>	

### Delete configurations

Click the delete button next to the configuration to delete it.

### Update configurations

Click the ‘Update’ button next to a configuration. You will be redirected to a page with the configuration’s info filled in. Modify the configuration and click submit.

## New Values

Term	<input type="text" value="Spring"/> <input type="button" value=""/>
Year	<input type="text" value="2018"/>
Days	<input type="text" value="MWTR"/>
Time	<input type="text" value="9:00-10:30"/>
Room	<input type="text" value="36"/>
Seats	<input type="text" value="42"/>
<input type="button" value="Submit"/>	

## Manage members – clearance 2 or 3

### View existing members

Click ‘Manage Members’ and a page with all the members and their info pops up.

## Manage Members

ID	First Name	Last Name	Type	Phone	Email	Address	Username	Password		
318547937	Tom	Adrian	Administrator	(207)400-8304	tomadrian@cms.com	Port Orange, FL 32127	tomadrian	WKSmMKQs	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
189826755	July	Anne	Administrator	(202)500-7430	julyanne@cms.com	4 Sugar Street	julyanne	QuWEn3JY	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
227209241	Raymon	Bailey	Student	(750) 488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.	raymonbailey	VsYJATqZ	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
549281974	Agustin	Barrett	Instructor	(852) 761-0810	agustinbarrett@cms.com	Bartlett, IL 60103	agustinbarrett	BeLrn6q5	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
600242738	Ward	Benitez	Administrator	(848)219-2884	wardbenitez@cms.com	Windermere, FL 34786	wardbenitez	E3zuH3XC	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
620381807	Jackson	Blankenship	Administrator	(305)742-8761	jacksonblankenship@cms.com	Bakersfield, CA 93306	jacksonblankenship	9YkgdG2W	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
851438868	Clara	Bradford	Instructor	(304)507-6675	clarabradford@cms.com	Sioux Falls, SD 57103	clarabradford	PQMkvujJ	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
320798136	Bob	Bryant	Administrator	(828)276-7614	bobbryant@cms.com	Lewiston, ME 04240	bobbryant	YP9buH86	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
889872253	Bertram	Carlson	Administrator	(614)341-4812	bertramcarlson@cms.com	Lake Zurich, IL 60047	bertramcarlson	9XUBTyct	<input type="button" value="Update"/>	<input type="button" value="Delete"/>
611668180	Edison	Carter	Instructor	(614)653-6066	edisoncarter@cms.com	Paterson, NJ 07501	edisoncarter	mX4j828U	<input type="button" value="Update"/>	<input type="button" value="Delete"/>

### Delete members

Click the delete button to delete a member.

## *Update members*

Click the update button. Enter the new fields in the next page and click submit. In this, you can change member attributes as well as user attributes. Use this to create a username and password for a new member.

The screenshot shows a web-based management system with a header containing links for Administrator Portal, Manage Courses, Manage Configurations, Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. Below the header is a decorative background with a geometric pattern of overlapping triangles in yellow, red, blue, and purple. A title 'New Values' is centered above a form. The form consists of eight input fields with labels: ID, First Name, Last Name, Phone, Email, Address, Username, and Password. Each field has a text input box and a small icon to its right. A 'Submit' button is located at the bottom right of the form area.

Field	Value
ID	318547937
First Name	Tom
Last Name	Adrian
Phone	(207)400-8304
Email	tomadrian@cms.com
Address	Port Orange, FL 32127
Username	tomadrian
Password	WKSmMKQs

## *Manage students – clearance 2 or 3*

### *View existing students*

Click the 'Manage Students' button.

## Manage Students

ID	First Name	Last Name	Balance	Unit Cap	
888134841	323	3232	0	12	<a href="#">Options</a>
227209241	Raymon	Bailey	696	18	<a href="#">Options</a>
339820994	Reyna	Colon	69	20	<a href="#">Options</a>
689582238	Dana	Conway	446	9	<a href="#">Options</a>
506190495	Verna	Copeland	153	11	<a href="#">Options</a>
483131196	Beatriz	Downs	623	20	<a href="#">Options</a>
573062892	Thanh	Everett	451	11	<a href="#">Options</a>
314843835	Myrtle	Haney	118	16	<a href="#">Options</a>
978594399	Gina	Hernandez	626	21	<a href="#">Options</a>
951597733	Lupe	Juarez	475	18	<a href="#">Options</a>
452018548	Bud	Kirby	776	18	<a href="#">Options</a>
796058602	Miles	Larson	320	15	<a href="#">Options</a>
503432266	Carol	Lynn	437	18	<a href="#">Options</a>

### Update student

Click options. You will have the ability to update a student's balance and unit cap. You can also change the ID and this will affect all tables.

### New Values

Name: 323 3232	
Student ID	888134841
Balance	0
Unit Cap	12
<a href="#">Submit</a>	

### Registered Courses

Term	Year	Department	Number	ConfigID
------	------	------------	--------	----------

### Transactions

Credit Card	Amount	Timestamp
-------------	--------	-----------

### View student's registered courses

Refer to picture above.

## *View student's transactions*

Refer to picture above.

## *Manage instructors – clearance 2 or 3*

### *View existing instructors*

Click the ‘Manage Students’ button.

ID	First Name	Last Name	Balance	Unit Cap
549281974	Agustin	Barrett	Active	<button>Options</button>
851438868	Clara	Bradford	Inactive	<button>Options</button>
611668180	Edison	Carter	Inactive	<button>Options</button>
337767185	Cherie	French	Inactive	<button>Options</button>
290327837	Nichole	Frost	Inactive	<button>Options</button>
957713740	Sylvia	Fuentes	Active	<button>Options</button>
307562770	Alton	Gallagher	Inactive	<button>Options</button>
996595763	Tamara	Gibson	Active	<button>Options</button>
661054866	Wes	Gill	Active	<button>Options</button>
100859622	Mia	Hodge	Inactive	<button>Options</button>
292211500	Antoinette	Marks	Active	<button>Options</button>
764777061	Miguel	Mathews	Inactive	<button>Options</button>
305981136	Polly	Rios	Inactive	<button>Options</button>

### *Update instructors*

Click ‘Options’. You will have the ability to activate or deactivate an instructor’s account. Enter ‘Active’ to activate. Enter anything else to deactivate, preferably ‘Inactive’. You can also change the ID and this will affect all tables.

## New Values

Name: Agustin Barrett	
Instructor ID	549281974
Status	Active
<input type="button" value="Submit"/>	

## Courses Taught

InstructorID	Department	Number	Term	Year	Days	Room	Seats
549281974	ENG	1A	Summer	2021	MW	22	52
549281974	CS	149	Spring	2020	MWF	43	55
549281974	GEO	2B	Fall	2018	MWTR	49	45

### *View courses taught*

Refer to picture above.

### *Manage administrators – clearance 2 or 3*

#### *View existing administrators*

Click the ‘Manage Administrators’ button.

## Manage Administrators

ID	First Name	Last Name	Clearance	
318547937	Tom	Adrian	2	<a href="#">Options</a>
189826755	July	Anne	1	<a href="#">Options</a>
600242738	Ward	Benitez	3	<a href="#">Options</a>
620381807	Jackson	Blankenship	3	<a href="#">Options</a>
320798136	Bob	Bryant	1	<a href="#">Options</a>
889872253	Bertram	Carlson	3	<a href="#">Options</a>
877078818	Hank	Colley	3	<a href="#">Options</a>
237393883	Angelique	Davila	1	<a href="#">Options</a>
546664173	Antoine	Deleon	1	<a href="#">Options</a>
798100317	Honey	Dent	1	<a href="#">Options</a>
409326430	Shelia	Elliott	3	<a href="#">Options</a>
587779626	Deangelo	Franklin	3	<a href="#">Options</a>
670334459	Brendon	Frazier	2	<a href="#">Options</a>

### Update administrators

Click 'Options'. You will have the ability to update an administrator's clearance. You can also change the ID and this will affect all tables.

### New Values

Name: Tom Adrian	
Administrator ID	<input type="text" value="889872253"/>
Clearance	<input type="text" value="2"/>
<input type="button" value="Submit"/>	

# Instructors

## *View courses taught*

From the instructor portal, click 'Manage Courses'.

The screenshot shows a user interface for managing courses. At the top, there is a navigation bar with links: 'Instructor Portal', 'Manage Courses', 'Teach Course', and 'Logout'. Below the navigation bar is a section titled 'Taught Courses' which displays a table of course information. The table has columns for Term, Year, Department, Number, Days, Time, Room, Seats, and ConfigID. Each row in the table includes two buttons: 'DROP' and 'Manage'. The background of the page features a colorful, abstract geometric pattern of triangles in shades of yellow, red, blue, and purple.

Term	Year	Department	Number	Days	Time	Room	Seats	ConfigID	DROP	Manage
Spring	2021	Math	127	MW	4:00 - 6:00	41	31	1	<button>DROP</button>	<button>Manage</button>
Winter	2020	CHEM	1A	TR	6:30-8:00	24	46	11	<button>DROP</button>	<button>Manage</button>
Summer	2019	HIS	17A	TR	9:00-10:30	31	33	21	<button>DROP</button>	<button>Manage</button>

## *Drop courses taught*

Click 'DROP' to drop a course.

## *View registered and waitlisted students*

Click 'Manage'.

## Registered Students

Student ID	First Name	Last Name	
199526438	Pasquale	Nelson	<input type="button" value="Drop"/>
339820994	Reyna	Colon	<input type="button" value="Drop"/>
Student ID	<input type="text"/> <input type="button" value="Add"/>		

## Waitlisted Students

Student ID	First Name	Last Name	
951597733	Lupe	Juarez	<input type="button" value="Drop"/> <input type="button" value="Add"/>

### *Drop registered students*

Click the drop button next to the student in the registrants table.

### *Drop waitlisted students*

Click the drop button next to the student in the waitlist table.

### *Manually add students on waitlist to registered list for a taught course*

Click add next to the student in the waitlist table.

### *Manually add students using their ID to registered list for a taught course*

Enter student's ID in the textbox and click add to add a student manually.

### *View offered courses*

Click 'Teach Course'.

## Teach Course

Department	Number	Config ID
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Teach"/>		

### Taught Courses

Term	Year	Department	Number	Days	Time	Room	Seats	ConfigID
Spring	2021	Math	127	MW	4:00 - 6:00	41	31	1
Winter	2020	CHEM	1A	TR	6:30-8:00	24	46	11
Summer	2019	HIS	17A	TR	9:00-10:30	31	33	21

### Offered Courses

Department	Number	Title	Units	Cost
BIO	1A	Biology I	5	230
BIO	1B	Biology II	5	230
CHEM	1A	Chemistry I	5	230
CHEM	1B	Chemistry II	5	230

## *View offered course configurations*

Click 'Teach Course'.

### Configurations

ConfigID	Term	Year	Days	Time	Room	Seats
1	Spring	2021	MW	4:00 - 6:00	41	31
2	Summer	2021	TR	6:30-8:00	33	60
3	Summer	2021	MWTR	9:00-10:30	34	56
4	Summer	2021	MW	8:00-9:30	22	52
5	Summer	2021	MWT	6:30-8:00	40	54
6	Fall	2020	MW	1:30-3:00	50	53
7	Fall	2020	TR	10:00-11:30	25	51
8	Fall	2020	MWTFD	9:00-9:30	10	25

## *Teach a course*

At the top of the page, enter the fields of the course and click submit. Select a course's department and number and config ID to add the course to teaching list. If the course is being taught by you or someone else – a certain combination of department, number, and configID are already in use by someone else – the course will not add.

## Teach Course

Department	Number	Config ID
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Teach"/>		

# Students

## *View registered courses*

In the student portal, click ‘Add Courses’.

The screenshot shows a student portal interface with a dark header bar containing links: Student Portal, Add Courses, Drop Courses, View Courses, Transcript, Transactions, and Logout. Below the header is a yellow section titled "Registered Courses" with a table:

Term	Year	Department	Number	ConfigID
Spring	2021	Math	127	1
Winter	2020	CHEM	1A	11
Summer	2019	HIS	17A	21

Below this is a section titled "Add Courses" with another table:

Term	Year	Department	Number	Firstname	Lastname	Days	Time	Room
Summer	2021	ENG	1A	Tamara	Gibson	MW	8:00-9:30	<input type="button" value="Add"/>
Spring	2021	Math	127	Catalina	Schaefer	MW	4:00 - 6:00	<input type="button" value="Add"/>
Summer	2021	MATH	3B	Mia	Hodge	MWTR	9:00-10:30	<input type="button" value="Add"/>
Summer	2021	ENG	1B	Clara	Bradford	MWT	6:30-8:00	<input type="button" value="Add"/>
Summer	2021	ENG	1B	Miguel	Mathews	MWT	6:30-8:00	<input type="button" value="Add"/>
Spring	2021	Math	127	Erin	Weber	MW	4:00 - 6:00	<input type="button" value="Add"/>
Summer	2021	ENG	1A	Agustin	Barrett	MW	8:00-9:30	<input type="button" value="Add"/>
Summer	2021	MATH	3A	Ivan	Tate	TR	6:30-8:00	<input type="button" value="Add"/>

## *Add courses*

In the picture above, click add next to the course to add. If the course is full, you will be added to the waitlist. If you are already registered or waitlisted, you cannot add the course twice.

## *Drop courses*

Click ‘Drop Courses’. Then click the drop button next to the course to drop.

## Registered Courses

Term	Year	Department	Number	ConfigID	
Spring	2021	Math	127	1	<button>Drop</button>
Winter	2020	CHEM	1A	11	<button>Drop</button>
Summer	2019	HIS	17A	21	<button>Drop</button>

*View offered courses  
Click ‘View Courses’.*

## Offered Courses

Term	Year	Department	Number	Firstname	Lastname	Days	Time	Room
Summer	2021	ENG	1A	Tamara	Gibson	MW	8:00-9:30	22
Spring	2021	Math	127	Catalina	Schaefer	MW	4:00 - 6:00	41
Summer	2021	MATH	3B	Mia	Hodge	MWTR	9:00-10:30	34
Summer	2021	ENG	1B	Clara	Bradford	MWT	6:30-8:00	40
Summer	2021	ENG	1B	Miguel	Mathews	MWT	6:30-8:00	40
Spring	2021	Math	127	Erin	Weber	MW	4:00 - 6:00	41
Summer	2021	ENG	1A	Agustin	Barrett	MW	8:00-9:30	22
Summer	2021	MATH	3A	Inez	Tate	TR	6:30-8:00	33
Summer	2021	MATH	3A	Jack	Vargas	TR	6:30-8:00	33
Summer	2021	MATH	3B	Alton	Gallagher	MWTR	9:00-10:30	34
Fall	2020	PHYS	4A	Antoinette	Marks	MWF	9:30-11:00	18
Winter	2020	CS	157A	Mia	Hodge	MW	3:00-4:30	48
Fall	2020	COMM	2	Polly	Rios	MW	1:30-3:00	50
Spring	2020	CS	149	Tamara	Gibson	MWF	4:00 - 6:00	43
Winter	2020	PHYS	4B	Sylvia	Eugenio	MW	3:00-4:30	50

*View transcript and registered courses  
Click ‘Transcript’.*

## Transcript

Term	Year	Department	Number
Spring	2021	Math	127
Winter	2020	CHEM	1A
Summer	2019	HIS	17A

*View transaction history*  
Click ‘Transactions’.

## Make a Transaction:

Credit Card	<input type="text"/>
Amount	<input type="text"/>
<input type="button" value="Submit"/>	

## Transaction History

Credit Card	Amount	Timestamp
4729921282617690	471	2020-08-04 14:18:09
7748348254287430	267	2020-08-03 19:38:15
8446756072343220	415	2020-08-02 07:09:12

## Pay for courses

Enter credit card information and amount and then click submit. The transaction will be in the portal after submitting.

# **Non-functional Issues**

## *Detailed descriptions of Graphical User Interface*

The online GUI will be made using HTML. It will initially contain the basic elements and may be beautified later. JavaScript may be added later to improve design. Initially, the pages will contain simple links, buttons, and tables. Color and visual appeal may be added later due to time constraints. Overall, the user will access the GUI by entering the server's URL. There will be a sign in page and a link to create a user. If the create user link is clicked, the user enters the required information and if it is valid, the account will be created followed by a success message. The user then clicks returns to the log in page and enters the username and password. After logging in, the user will be redirected to the respective page associated with the account type and will view the features and functions they are allowed to access by links. When the user clicks on a link, the respective page will be displayed allowing users to perform an action or view data.

Primarily there are four types of views. One is for the visitor who wants to register consisting of a form. Another is the student portal fulfilling the functional requirements. Instructors and administrators also have their respective portals. The portal will show limited options for an administrator with lower clearance. An inactive instructor account will not have any links except the logout button.

## *Detailed descriptions of Security*

- Input will be sanitized to prevent SQL injection attacks
- A sign-in page will ensure users log in to their correct accounts
- Users can log out by clicking a “logout” button. They will be immediately logged out when they exit the session.

## *Detailed descriptions of Access Control*

Separate accounts will be created for each type of user. Each type of user will have its own landing page. This ensures security as users don't access features they are not supposed to. This ensures access control and that users only do what they are allowed to and view data which is available for them to view. Users cannot view data they do not have access to nor perform actions they have no access to. The landing page will make sure of that. The status of the account can also be checked before performing any action to add an extra layer of security but that would be unnecessary.

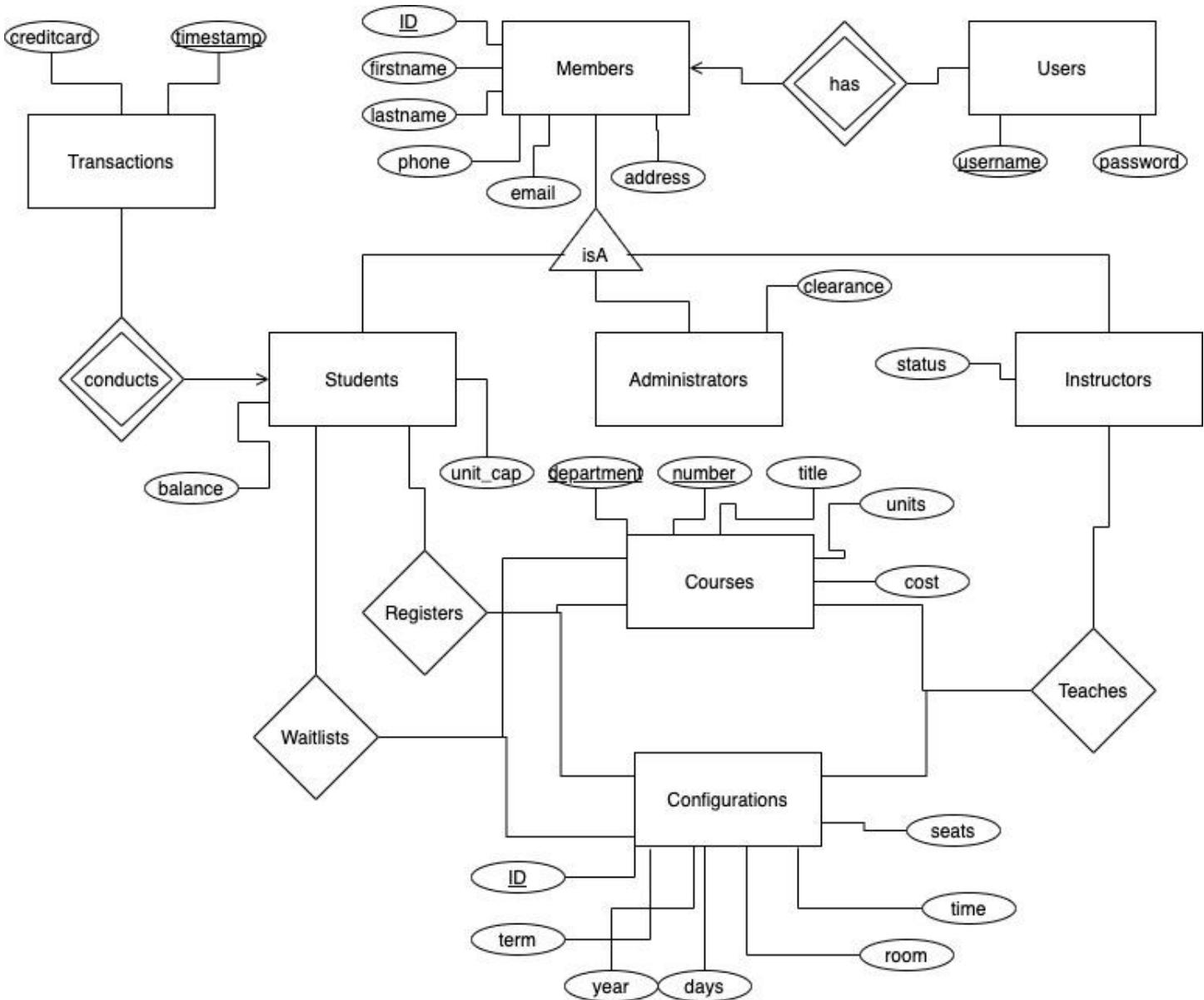
Student users are only able to access the courses available for the coming semester to register or waitlist for and are only able to alter their course schedule by dropping themselves from a selected course. However, they cannot enroll, waitlist, or drop other students into or out of a course or perform any actions reserved for admins and instructors. If they are waitlisted for a course, they are unable to perform any action that will alter their position.

Instructor users can manage the students who join the course. They can manually add and drop students. This is only possible for the courses an instructor teaches, not anyone else.

Administrator users are able to create new courses and configurations. They can manage accounts, create usernames, and reset passwords. They can control accounts and view information about other users but not delete their registered or taught courses.

# Project Design

## ERD



## Database Schemas

### Entities

**Members**(*ID, firstname, lastname, phone, email, address*)

An entity that represents a member in the system. This is created when you sign up on the homepage. A member is a student, instructor, or administrator. It also has a User account.

### ***Administrators(adminID, clearance)***

An administrator is a Member and holds a special attribute clearance for access control in the admin portal

### ***Configurations(configID, term, year, days, time, room, seats)***

A course is taught in a particular configuration. A single course can be taught in many configurations. Configurations describe when and where a course is taught.

### ***Courses(department, number, title, units, cost)***

An entity representing a course. It has the basic attributes that do not vary. A course combined with a configuration can be taught, registered, or waitlisted.

### ***Instructors(instructorID, status)***

A type of member with a special attribute status which maintains access control. If an instructor is inactive, it should not do anything.

### ***Students(studentID, balance, unit\_cap)***

A type of member with balance and unit cap. Balance is the amount of money a student has to pay for courses. The unit cap specifies the amount of course units a student can add.

### ***Transactions(studentID, creditcard, amount, timestamp)***

Is a weak entity set that store financial transaction information for buying a course. It needs studentID to help identify each unique tuple.

### ***Users(ID, username, password)***

Is a weak entity set that allows members to log into their portals. A User can have only one member. It borrows ID from Members.

## ***Relationships***

### ***Teaches(instructorID, department, number, configID)***

Holds information of course (department, number) and the configuration it is taught in.

### ***Registers(studentID, department, number, configID)***

Holds information of course (department, number) and the configuration it is registered for.

### ***Waitlists(studentID, department, number, configID)***

Holds information of course (department, number) and the configuration it is waitlisted for.

# Tables According to Schemas and Model of the Data Stored in the Database

Now we will show all the tables in the MySQL Workbench. We will first show the configuration of each table and the datatypes of each attribute. This will show the model of the data stored in the database. Then we will show 15 or more tuples for each table. Then we will show some sample data models in an excel spreadsheet.

## Administrators(adminID, clearance)

The screenshot shows the MySQL Workbench interface for the 'Course Management System' schema. The left sidebar lists various tables and objects. The main panel displays the 'Administrators' table configuration. The table has two columns: 'adminID' (CHAR(9), PK) and 'clearance' (INT). The 'adminID' column is set as Primary Key (PK) and Not NULL (Not NULL). The 'clearance' column is of type INT. Below the table definition, a 'Column details' panel shows the same information. At the bottom, a history pane shows three recent actions: a SELECT query on 'Waitlists' (10 rows returned), a SELECT query on 'Registers' (24 rows returned), and another SELECT query on 'Registers' (24 rows returned).

Table: Administrators

Columns:

adminID	char(9)	PK
clearance	int	

Action Output

Time	Action	Response	Duration / Fetch Time
18 19:40:19	SELECT * FROM `Course Management System`.`Waitlists` LIMIT 0, 1000	10 row(s) returned	0.0028 sec / 0.00023...
19 19:42:16	SELECT * FROM `Course Management System`.`Registers` LIMIT 0, 1000	24 row(s) returned	0.00053 sec / 0.000...
20 19:42:17	SELECT * FROM `Course Management System`.`Registers` LIMIT 0, 1000	24 row(s) returned	0.00055 sec / 0.000...

Screenshot of MySQL Workbench showing the execution of a SELECT query on the Administrators table.

**Query:**

```
1 •  SELECT * FROM `Course Management System`.Administrators;
```

**Result Grid:**

adminID	clearance
189826755	1
237393883	1
318547937	2
320798136	1
409326430	3
435313478	2
546664173	1
587779626	3
600242738	3
620381807	3
622426523	2
638993187	2
670334459	2
722640650	1
798100317	1
861394637	3
877078818	3
889872253	3
943667551	1
944806400	2

**Object Info:**

Table: Administrators

Columns:

- adminID: char(9) PK
- clearance: int

**Action Output:**

Action	Time	Action	Response	Duration / Fetch Time
29	20:32:06	SELECT * FROM `Course Management System`.Administrators LIMIT 0, 1000	20 row(s) returned	0.00053 sec / 0.000...
30	20:32:06	SELECT * FROM `Course Management System`.Administrators LIMIT 0, 1000	20 row(s) returned	0.00031 sec / 0.000...

Query Completed

A	B	C	D
1 Administrators			
2 Tuple Number	adminID	clearance	SQL Statement
3 1	546664173		2 INSERT INTO Administrators VALUES ('546664173', 2);
4 2	620381807		1 INSERT INTO Administrators VALUES ('620381807', 1);
5 3	409326430		2 INSERT INTO Administrators VALUES ('409326430', 2);
6 4	943667551		1 INSERT INTO Administrators VALUES ('943667551', 1);
7 5	237393883		2 INSERT INTO Administrators VALUES ('237393883', 2);
8 6	435313478		1 INSERT INTO Administrators VALUES ('435313478', 1);
9 7	670334459		3 INSERT INTO Administrators VALUES ('670334459', 3);
10 8	600242738		2 INSERT INTO Administrators VALUES ('600242738', 2);
11 9	622426523		1 INSERT INTO Administrators VALUES ('622426523', 1);
12 10	587779626		3 INSERT INTO Administrators VALUES ('587779626', 3);
13 11	189826755		3 INSERT INTO Administrators VALUES ('189826755', 3);
14 12	638993187		1 INSERT INTO Administrators VALUES ('638993187', 1);
15 13	798100317		2 INSERT INTO Administrators VALUES ('798100317', 2);
16 14	944806400		3 INSERT INTO Administrators VALUES ('944806400', 3);
17 15	722640650		3 INSERT INTO Administrators VALUES ('722640650', 3);
18 16	889872253		1 INSERT INTO Administrators VALUES ('889872253', 1);
19 17	861394637		2 INSERT INTO Administrators VALUES ('861394637', 2);
20 18	320798136		2 INSERT INTO Administrators VALUES ('320798136', 2);
21 19	877078818		1 INSERT INTO Administrators VALUES ('877078818', 1);
22 20	318547937		1 INSERT INTO Administrators VALUES ('318547937', 1);

## Configurations(configID, term, year, days, time, room, seats)

The screenshot shows the MySQL Workbench interface for creating a new table named 'Configurations'. The table structure is defined as follows:

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
configID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
term	VARCHAR(7)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
year	CHAR(4)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
days	VARCHAR(7)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
time	VARCHAR(15)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
room	VARCHAR(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
seats	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

Column details for 'configID':

- Column Name: configID
- Datatype: INT
- Charset/Collation: Default Charset
- Comments:
- Storage: VIRTUAL (selected)
- Primary Key:
- Not NULL:
- Unique:
- Binary:
- Unsigned:
- ZeroFill:
- Auto Increment:
- Generated:

Action Output:

Action	Response	Duration / Fetch Time
18 19:40:19 SELECT * FROM 'Course Management System'.Waitlists LIMIT 0, 1000	10 row(s) returned	0.0028 sec / 0.00023...
19 19:42:16 SELECT * FROM 'Course Management System'.Registers LIMIT 0, 1000	24 row(s) returned	0.00053 sec / 0.000...
20 19:42:17 SELECT * FROM 'Course Management System'.Registers LIMIT 0, 1000	24 row(s) returned	0.00055 sec / 0.000...

Query Completed

MySQL Workbench

Administration Schemas Query 1 Users Registers Configurations Members Registers Context Help Snippets

Course Management System

Tables Administrators Configurations Courses Instructors Members Registers Students Teaches Transactions Users Waitlists Views Stored Procedures Functions demo Project\_data sys

Object Info Session

Table: Configurations

Columns:

configID	term	year	days	time	room	seats
1	Spring	2021	MW	4:00 - 6:00	41	31
2	Summer	2021	TR	6:30-8:00	33	60
3	Summer	2021	MWTR	9:00-10:30	34	56
4	Summer	2021	MW	8:00-9:30	22	52
5	Summer	2021	MWT	6:30-8:00	40	54
6	Fall	2020	MW	1:30-3:00	50	53
7	Fall	2020	TR	10:00-11:30	25	51
8	Fall	2020	MWTR	8:00-9:30	18	35
9	Fall	2020	MWF	9:30-11:00	18	33
10	Winter	2020	MW	3:00-4:30	50	54
11	Winter	2020	TR	6:30-8:00	24	46
12	Winter	2020	MWTR	9:00-10:30	23	53
13	Winter	2020	MW	3:00-4:30	48	48
14	Spring	2020	MWF	4:00 - 6:00	43	55
15	Spring	2019	MW	8:00 - 10:00	33	48
16	Spring	2019	TR	1:00-2:00	43	52
17	Spring	2019	MWTR	8:00-9:30	34	39
18	Summer	2019	MW	9:30-11:00	48	38
19	Summer	2019	MWT	3:00-4:30	12	51
20	Summer	2019	MWTR	6:30-8:00	18	37
21	Summer	2019	TR	9:00-10:30	31	33
22	Fall	2019	MW	6:30-8:00	41	38
23	Fall	2018	TR	3:00-6:00	37	51
24	Fall	2018	MWTR	9:00-10:30	49	45
25	Fall	2018	MW	8:00-9:30	12	53
26	Winter	2018	TR	9:30-11:00	15	34
27	Winter	2018	MWTR	3:00-4:30	31	60
28	Winter	2018	MW	6:30-8:00	30	46
29	Winter	2018	MWF	9:00-10:30	21	43
30	Spring	2018	MWTR	9:00-10:30	36	42

Action Output

Time	Action	Response	Duration / Fetch Time
20:32:06	SELECT * FROM 'Course Management System'.Administrators LIMIT 0, 1000	20 row(s) returned	0.00031 sec / 0.0000...
20:34:47	SELECT * FROM 'Course Management System'.Configurations LIMIT 0, 1000	30 row(s) returned	0.00066 sec / 0.000...

Query Completed

A	B	C	D	E	F	G	H
<b>Configurations</b>							
1	ID	Term	Year	Days	Time	Room	Seats
2	1	Spring	2021	MW	4:00 - 6:00	10	57
3	2	Summer	2021	TR	6:30-8:00	48	42
4	3	Summer	2021	MWTR	9:00-10:30	39	35
5	4	Summer	2021	MW	8:00-9:30	22	32
6	5	Summer	2021	MWT	6:30-8:00	49	37
7	6	Fall	2020	MW	1:30-3:00	15	53
8	7	Fall	2020	TR	10:00-11:30	34	60
9	8	Fall	2020	MWTR	8:00-9:30	44	49
10	9	Fall	2020	MWF	9:30-11:00	21	40
11	10	Winter	2020	MW	3:00-4:30	41	55
12	11	Winter	2020	TR	6:30-8:00	45	36
13	12	Winter	2020	MWTR	9:00-10:30	39	44
14	13	Winter	2020	MW	3:00-4:30	48	31
15	14	Spring	2020	MWF	4:00 - 6:00	45	52
16	15	Spring	2019	MW	8:00 - 10:00	50	60
17	16	Spring	2019	TR	1:00-2:00	48	60
18	17	Spring	2019	MWTR	8:00-9:30	16	33
19	18	Summer	2019	MW	9:30-11:00	16	58
20	19	Summer	2019	MWT	3:00-4:30	14	39
21	20	Summer	2019	MWTR	6:30-8:00	27	60
22	21	Summer	2019	TR	9:00-10:30	15	32
23	22	Fall	2019	MW	6:30-8:00	47	41
24	23	Fall	2018	TR	3:00-6:00	18	30
25	24	Fall	2018	MWTR	9:00-10:30	23	38
26	25	Fall	2018	MW	8:00-9:30	48	35
27	26	Winter	2018	TR	9:30-11:00	29	41
28	27	Winter	2018	MWTR	3:00-4:30	36	53
29	28	Winter	2018	MW	6:30-8:00	18	59
30	29	Winter	2018	MWF	9:00-10:30	50	44
31	30	Spring	2018	MWTR	9:00-10:30	41	54

*Courses(department, number, title, units, cost)*

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

Course Management System

Tables

Administrators Configurations Courses Instructors Members Registers Students Teaches Transactions Users Waitlists Views Stored Procedures Functions

demo Project\_data sys

Object Info Session

Table: Courses

Columns:

department	varchar(20) PK			
number	varchar(10) PK			
title	varchar(45)			
units	int			
cost	int			
BIO	1A	Biology I	5	230
BIO	1B	Biology II	5	230
CHEM	1A	Chemistry I	5	230
CHEM	1B	Chemistry II	5	230
COMM	2	Public Communication	3	138
CS	146	Data Structures	4	184
CS	147	Computer Architecture	4	184
CS	149	Operating Systems	4	184
CS	157A	Database	4	184
CS	166	Information Security	4	184
ENG	1A	Language & Compos...	3	138
ENG	1B	Literature	3	138
GEO	2A	Introduction to Geolo...	3	138
GEO	2B	Introduction to Geolo...	3	138
HIS	17A	U.S. History I	3	138
HIS	17B	U.S. History II	3	138
HUM	5A	Humanities Intro I	3	138
HUM	5B	Humanities Intro II	3	138
KIN	1	Soccer	1	46
KIN	2	Basketball	1	46
KIN	3	Swimming	1	46
KIN	4	Badminton	1	46
Math	127	Linear Algebra	4	184
MATH	3A	Calculus I	4	184
PHYS	4A	Physics I	5	230
PHYS	4B	Physics II	5	230
SPA	1	Spanish I	5	230
SPA	2	Spanish II	5	230
SPA	3	Spanish III	5	230

Action Output

Time	Action	Response	Duration / Fetch Time
20:34:48	SELECT * FROM 'Course Management System'.Configurations LIMIT 0, 1000	30 row(s) returned	0.00037 sec / 0.0000...
20:35:49	SELECT * FROM 'Course Management System'.Courses LIMIT 0, 1000	29 row(s) returned	0.0018 sec / 0.00004...

Query Completed

	A	B	C	D	E	F	G
2		Department	Number	Title	Units	Cost	Insert Statement
3	1	Math	127	Linear Algebra	4	184	INSERT INTO Courses VALUES ('Math', '127', 'Linear Algebra', 4, 184);
4	2	MATH	3A	Calculus I	4	184	INSERT INTO Courses VALUES ('MATH', '3A', 'Calculus I', 4, 184);
5	3	MATH	3B	Calculus II	4	184	INSERT INTO Courses VALUES ('MATH', '3B', 'Calculus II', 4, 184);
6	4	ENG	1A	Language & Composition	3	138	INSERT INTO Courses VALUES ('ENG', '1A', 'Language & Composition', 3, 138);
7	5	ENG	1B	Literature	3	138	INSERT INTO Courses VALUES ('ENG', '1B', 'Literature', 3, 138);
8	6	COMM	2	Public Communication	3	138	INSERT INTO Courses VALUES ('COMM', '2', 'Public Communication', 3, 138);
9	7	BIO	1A	Biology I	5	230	INSERT INTO Courses VALUES ('BIO', '1A', 'Biology I', 5, 230);
10	8	BIO	1B	Biology II	5	230	INSERT INTO Courses VALUES ('BIO', '1B', 'Biology II', 5, 230);
11	9	PHYS	4A	Physics I	5	230	INSERT INTO Courses VALUES ('PHYS', '4A', 'Physics I', 5, 230);
12	10	PHYS	4B	Physics II	5	230	INSERT INTO Courses VALUES ('PHYS', '4B', 'Physics II', 5, 230);
13	11	CHEM	1A	Chemistry I	5	230	INSERT INTO Courses VALUES ('CHEM', '1A', 'Chemistry I', 5, 230);
14	12	CHEM	1B	Chemistry II	5	230	INSERT INTO Courses VALUES ('CHEM', '1B', 'Chemistry II', 5, 230);
15	13	CS	157A	Database	4	184	INSERT INTO Courses VALUES ('CS', '157A', 'Database', 4, 184);
16	14	CS	149	Operating Systems	4	184	INSERT INTO Courses VALUES ('CS', '149', 'Operating Systems', 4, 184);
17	15	CS	146	Data Structures	4	184	INSERT INTO Courses VALUES ('CS', '146', 'Data Structures', 4, 184);
18	16	CS	147	Computer Architecture	4	184	INSERT INTO Courses VALUES ('CS', '147', 'Computer Architecture', 4, 184);
19	17	CS	166	Information Security	4	184	INSERT INTO Courses VALUES ('CS', '166', 'Information Security', 4, 184);
20	18	SPA	1	Spanish I	5	230	INSERT INTO Courses VALUES ('SPA', '1', 'Spanish I', 5, 230);
21	19	SPA	2	Spanish II	5	230	INSERT INTO Courses VALUES ('SPA', '2', 'Spanish II', 5, 230);
22	20	SPA	3	Spanish III	5	230	INSERT INTO Courses VALUES ('SPA', '3', 'Spanish III', 5, 230);
23	21	HIS	17A	U.S. History I	3	138	INSERT INTO Courses VALUES ('HIS', '17A', 'U.S. History I', 3, 138);
24	22	HIS	17B	U.S. History II	3	138	INSERT INTO Courses VALUES ('HIS', '17B', 'U.S. History II', 3, 138);
25	23	GEO	2A	Introduction to Geology I	3	138	INSERT INTO Courses VALUES ('GEO', '2A', 'Introduction to Geology I', 3, 138);
26	24	GEO	2B	Introduction to Geology II	3	138	INSERT INTO Courses VALUES ('GEO', '2B', 'Introduction to Geology II', 3, 138);
27	25	HUM	5A	Humanities Intro I	3	138	INSERT INTO Courses VALUES ('HUM', '5A', 'Humanities Intro I', 3, 138);
28	26	HUM	5B	Humanities Intro II	3	138	INSERT INTO Courses VALUES ('HUM', '5B', 'Humanities Intro II', 3, 138);
29	27	KIN	1	Soccer	1	46	INSERT INTO Courses VALUES ('KIN', '1', 'Soccer', 1, 46);
30	28	KIN	2	Basketball	1	46	INSERT INTO Courses VALUES ('KIN', '2', 'Basketball', 1, 46);
31	29	KIN	3	Swimming	1	46	INSERT INTO Courses VALUES ('KIN', '3', 'Swimming', 1, 46);
32	30	KIN	4	Badminton	1	46	INSERT INTO Courses VALUES ('KIN', '4', 'Badminton', 1, 46);

## Instructors(instructorID, status)

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** Course Management System
- Tables:** Instructors
- Table Definition:**

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
instructorID	CHAR(9)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<click to edit>				
status	VARCHAR(8)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>
- Column Details for 'instructorID':**
  - Column Name: instructorID
  - Datatype: CHAR(9)
  - Storage: VIRTUAL (Primary Key, Not NULL)
  - Comments:
- Query Results:**

Action Output	Time	Action	Response	Duration / Fetch Time
27	20:43:06	SELECT * FROM 'Course Management System'.Members LIMIT 0, 1000	33 row(s) returned	0.00040 sec / 0.000...
28	20:43:51	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	10 row(s) returned	0.00034 sec / 0.000...
29	20:43:51	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	10 row(s) returned	0.00089 sec / 0.000...

MySQL Workbench - MySQL Workbench

File Edit View Query Database Server Tools Scripting Help Tue 8:36 PM Anshuman Goswami 65% 100% 1:1

Administration Schemas

Schemas Course Management System

Tables Administrators Configurations Courses Instructors Members Registers Students Teaches Transactions Users Waitlists Views Stored Procedures Functions demo Project\_data sys

1 • |SELECT \* FROM `Course Management System`.Instructors;

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

instructorID	status
100859622	Inactive
226084272	Active
290327837	Inactive
292211500	Active
305981136	Inactive
307562770	Inactive
337767185	Inactive
378503098	Inactive
418130858	Active
549281974	Active
603160432	Active
611668180	Inactive
661054866	Active
764777061	Inactive
851438868	Inactive
875783848	Inactive
891149217	Active
916994127	Active
957713740	Active
996595763	Active
HULL	HULL

Object Info Session

Table: Instructors

Columns:

- instructorID** char(9) PK
- status varchar(8)

Action Output

Time	Action	Response	Duration / Fetch Time
34 20:35:49	SELECT * FROM 'Course Management System'.Courses LIMIT 0, 1000	29 row(s) returned	0.00048 sec / 0.000...
35 20:36:45	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	20 row(s) returned	0.00066 sec / 0.000...

Query Completed

A	B	C	D
1	Instructors		
2	Tuple Number	instructorID	status SQL Statement
3	1	226084272	Active INSERT INTO Instructors VALUES ('226084272', 'Active');
4	2	378503098	Inactive INSERT INTO Instructors VALUES ('378503098', 'Inactive');
5	3	307562770	Inactive INSERT INTO Instructors VALUES ('307562770', 'Inactive');
6	4	549281974	Active INSERT INTO Instructors VALUES ('549281974', 'Active');
7	5	764777061	Inactive INSERT INTO Instructors VALUES ('764777061', 'Inactive');
8	6	305981136	Inactive INSERT INTO Instructors VALUES ('305981136', 'Inactive');
9	7	916994127	Active INSERT INTO Instructors VALUES ('916994127', 'Active');
10	8	290327837	Inactive INSERT INTO Instructors VALUES ('290327837', 'Inactive');
11	9	292211500	Active INSERT INTO Instructors VALUES ('292211500', 'Active');
12	10	337767185	Inactive INSERT INTO Instructors VALUES ('337767185', 'Inactive');
13	11	891149217	Active INSERT INTO Instructors VALUES ('891149217', 'Active');
14	12	418130858	Active INSERT INTO Instructors VALUES ('418130858', 'Active');
15	13	100859622	Inactive INSERT INTO Instructors VALUES ('100859622', 'Inactive');
16	14	996595763	Active INSERT INTO Instructors VALUES ('996595763', 'Active');
17	15	851438868	Inactive INSERT INTO Instructors VALUES ('851438868', 'Inactive');
18	16	603160432	Active INSERT INTO Instructors VALUES ('603160432', 'Active');
19	17	875783848	Inactive INSERT INTO Instructors VALUES ('875783848', 'Inactive');
20	18	661054866	Active INSERT INTO Instructors VALUES ('661054866', 'Active');
21	19	611668180	Inactive INSERT INTO Instructors VALUES ('611668180', 'Inactive');
22	20	957713740	Active INSERT INTO Instructors VALUES ('957713740', 'Active');

## *Members*(*ID, firstname, lastname, phone, email, address*)

The screenshot shows the MySQL Workbench interface for creating a new table named 'Members' in the 'Course Management System' schema.

**Table Structure:**

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
ID	CHAR(9)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<click to edit>				
firstname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>
lastname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>
phone	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>
email	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>
address	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<click to edit>

**Column Details for 'ID':**

- Column Name: ID
- Datatype: CHAR(9)
- Default:
- Storage: Primary Key (checked), Not NULL (checked), Unique (unchecked), Binary (unchecked), Unsigned (unchecked), ZeroFill (unchecked), Auto Increment (unchecked), Generated (unchecked).

**Action Output:**

Action	Time	Response	Duration / Fetch Time
19	19:42:16	SELECT * FROM 'Course Management System'.Registers LIMIT 0, 1000	24 row(s) returned 0.00053 sec / 0.000...
20	19:42:17	SELECT * FROM 'Course Management System'.Registers LIMIT 0, 1000	24 row(s) returned 0.00055 sec / 0.000...
21	20:38:52	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	10 row(s) returned 0.0034 sec / 0.0002...

MySQL Workbench

Administration Schemas Query 1 Users Registers Configurations Members Registers Context Help Snippets

MySQL Workbench

Schemas Course Management System

Tables Administrators Configurations Courses Instructors Members Registers Students Teaches Transactions Users Waitlists Views Stored Procedures Functions

Members

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

ID firstname lastname phone email address

100% 1:1

1 • SELECT \* FROM `Course Management System`.Members;

100859622 Mia Hodge (310)704-6500 miahodge@cms.com Worcester, MA 01604  
 189826755 July Anne (202)500-7430 julianne@cms.com 4 Sugar Street  
 199526438 Pasquale Nelson (745) 243-1323 pasqualnelson@cms.com Battle Creek, MI 49015  
 204722767 Alice Sims (589) 871-3681 alicesims@cms.com San Jose, CA 95127  
 226084272 Erin Weber (346) 266-2473 erinweber@cms.com 8267 Rock Maple Drive  
 227209241 Raymon Bailey (750) 488-7011 raymonbailey@cms.com 229 SW. Ocean Ave.  
 237393883 Angelique Davila (270)493-4957 angelique davila@cms.com 40 Jockey Hollow Dr.  
 290327837 Nichole Frost (891) 297-0455 nicholefrost@cms.com Clearwater, FL 33756  
 292211500 Antoinette Marks (939) 935-8994 antoinettemarks@cms.com 75 Longbranch Dr.  
 305981196 Polly Rios (809) 914-2643 pollyrios@cms.com Peabody, MA 01960  
 307562770 Alton Gallagher (359) 351-7168 alton gallagher@cms.com 90 Eagle Street  
 314843835 Myrtle Haney (953) 493-3646 myrtlehaney@cms.com Auburndale, FL 33823  
 318547937 Tom Adrian (207)400-8303 tomadian@cms.com Port Orange, FL 32127  
 320798136 Bob Bryant (828)276-7614 bobbryant@cms.com Lewiston, ME 04240  
 337767185 Cherie French (241) 768-9215 cheriefrench@cms.com 9117 Arnold Street  
 339820994 Reyna Colon (383) 592-3560 reynacolon@cms.com Saint Cloud, MN 56301  
 378503098 Jack Vargas (908) 442-4062 jackvargas@cms.com Pasadena, MD 21122  
 409326430 Shelia Elliott (386)438-4907 shelialelliott@cms.com 45 Circle Avenue  
 418130858 Inez Tate (302)422-6181 ineztate@cms.com 556 Trusey Ave.  
 435313478 Wilber Melton (802)398-4468 wilbermelton@cms.com Niceville, FL 32578  
 452018548 Bud Kirby (919) 601-5112 budkirby@cms.com 85 Selby Street  
 483131196 Beatriz Downs (780) 422-7078 beatrizdowns@cms.com Aiken, SC 29803  
 503432266 Carol Lynn (446) 747-1918 carolynn@cms.com 78 Summer St.  
 506190495 Vern Copeland (395) 913-5631 vernacopeland@cms.com 7 East Trusel Drive  
 538782975 Dustin Warren (501) 342-1537 dustinwarren@cms.com Simpsonville, SC 29680  
 Columns:

ID char(9) PK  
 firstname varchar(45)  
 lastname varchar(45)  
 phone varchar(45)  
 email varchar(45)  
 address varchar(45)

Members 1

Action Output

	Time	Action	Response	Duration / Fetch Time
37	20:37:32	SELECT * FROM 'Course Management System'.Members LIMIT 0, 1000	60 row(s) returned	0.0019 sec / 0.00003...
38	20:37:32	SELECT * FROM 'Course Management System'.Members LIMIT 0, 1000	60 row(s) returned	0.00040 sec / 0.000...

Query Completed

A	B	C	D	E	F	G	H	I	J
1	Members								
2	Tuple Number	ID	Firstname	Lastname	Phone	Email	Address		
3	1	199526438	Pasquale	Nelson	(745) 243-1323	pasqualnelson@cms.com	Battle Creek, MI 49015		
4	2	951597735	Lupe	Juarez	(816) 842-3578	lupejuarez@cms.com	647 Vale Street		Student
5	3	483131196	Beatriz	Downs	(780) 422-7078	beatrizdowns@cms.com	Aiken, SC 29803		Student
6	4	503432266	Carol	Lynn	(446) 747-1918	carolynn@cms.com	78 Summer St.		Student
7	5	314843835	Myrtle	Haney	(953) 493-3646	myrtlehaney@cms.com	Auburndale, FL 33823		Student
8	6	760511911	Luciano	Smith	(592) 205-4019	lucianosmith@cms.com	251 San Carlos Ave.		Student
9	7	760508602	Miles	Larson	(898) 732-7801	mileslarson@cms.com	South Portland, ME 04101		Student
10	8	57906941	Steven	Suarez	(254) 567-1538	stevensuarez@cms.com	49 Lees Creek Ave.		Student
11	9	538782975	Dustin	Warren	(501) 342-1537	dustinwarren@cms.com	Simpsonville, SC 29680		Student
12	10	670202202	Mose	McDonald	(627) 579-8894	mosemcdonald@cms.com	254 Littleton Drive		Student
13	11	339820994	Reyna	Colon	(383) 592-3560	reynacolon@cms.com	Saint Cloud, MN 56301		Student
14	12	978594399	Gina	Hernandez	(454) 582-4008	ginahernandez@cms.com	6 High Point St.		Student
15	13	689582238	Dana	Conway	(582) 630-1598	danaconway@cms.com	Shirley, NY 11967		Student
16	14	506190495	Verna	Copeland	(395) 913-5631	vernacopeland@cms.com	7 East Trusel Drive		Student
17	15	932823238	Jonas	Padilla	(605) 452-4957	jonaspadilla@cms.com	Klamath Falls, OR 97603		Student
18	16	452018548	Bud	Kirby	(919) 601-5112	budkirby@cms.com	85 Seby Street		Student
19	17	176801899	Roscoe	Richard	(303) 819-2145	roscoerichard@cms.com	Bardstown, KY 40004		Student
20	18	573062892	Thanh	Everett	(339) 936-4988	thanheverett@cms.com	8227 William St.		Student
21	19	204722757	Alice	Sims	(589) 871-3681	alicesims@cms.com	San Jose, CA 95127		Student
22	20	227209241	Raymon	Bailey	(750) 488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.		Student
23	21	226084272	Erin	Weber	(346) 266-2473	erinweber@cms.com	8267 Rock Maple Drive		Instructor
24	22	378503098	Jack	Vargas	(908) 442-4062	jackvargas@cms.com	Pasadena, MD 21122		Instructor
25	23	307562770	Alton	Gallagher	(359) 351-7168	alton gallagher@cms.com	90 Eagle Street		Instructor
26	24	549281974	Agustin	Barrett	(852) 761-0810	agustinbarrett@cms.com	Bartlett, IL 60103		Instructor
27	25	674777061	Miguel	Mathews	(841) 394-2348	miguelmathews@cms.com	595 Finch Drive		Instructor
28	26	305981136	Polly	Rios	(809) 914-2643	pollyrios@cms.com	Peabody, MA 01960		Instructor
29	27	916941127	Quinn	Warner	(383) 269-9498	quinnwarner@cms.com	12 N. Oak Circle		Instructor
30	28	290327837	Nichole	Frost	(891) 297-0455	nicholefrost@cms.com	Clearwater, FL 33756		Instructor
31	29	292211500	Antoinette	Marks	(939) 935-8994	antoinettemarks@cms.com	75 Longbranch Dr.		Instructor
32	30	337767185	Cherie	French	(241) 768-9215	cheriefrench@cms.com	9117 Arnold Street		Instructor
33	31	891149217	Catalina	Schaefer	(575) 208-7495	catalinaschaefer@cms.com	Lilburn, GA 30047		Instructor
34	32	418130858	Inez	Tate	(302) 422-6181	ineztate@cms.com	556 Trusey Ave.		Instructor
35	33	100859622	Mia	Hodge	(310) 704-6500	miahodge@cms.com	Worcester, MA 01604		Instructor
36	34	96595763	Tamara	Gibson	(289)276-8056	tamaragibson@cms.com	182 Sherman Ave.		Instructor
37	35	85148868	Clara	Bradford	(304)507-6675	clarabradford@cms.com	Sioux Falls, SD 57103		Instructor
38	36	603160432	Normand	Sheppard	(346)932-8189	normandsheppard@cms.com	94 Glen Ridge Drive		Instructor
39	37	87783848	Allen	Zuniga	(671)635-4007	allen.zuniga@cms.com	Vernon Hills, IL 60061		Instructor
40	38	661054866	Wes	Gill	(473)820-5552	wesgill@cms.com	9089 Edgewater Court		Instructor
41	39	611668180	Edison	Carter	(614)653-6066	edisoncarter@cms.com	Paterson, NJ 07501		Instructor
42	40	557713740	Sylvia	Fuentes	(331)254-9063	sylviafuentes@cms.com	7890 Cross Road		Instructor

45	41	546664173	Antoine	Deleon	(224)441-8546	antoinedeleon@cms.com	7731 S. Wagon Street	INSERT INTO Members VALUES ('546664173', 'Antoine', 'Deleon', '(224)441-8546', 'antoinedeleon@cms.com', '7731 S. Wagon Street');	Administrator
46	42	620381807	Jackson	Blankenship	(305)742-8761	jacksonblankenship@cms.co	Bakersfield, CA 93306	INSERT INTO Members VALUES ('620381807', 'Jackson', 'Blankenship', '(305)742-8761', 'jacksonblankenship@cms.co', 'Bakersfield, CA 93306');	Administrator
47	43	409326430	Shelia	Elliott	(386)438-4907	sheliaelliott@cms.com	45 Circle Avenue	INSERT INTO Members VALUES ('409326430', 'Shelia', 'Elliott', '(386)438-4907', 'sheliaelliott@cms.com', '45 Circle Avenue');	Administrator
48	44	943667551	Justine	Mcbride	(607)898-3165	justinemcbride@cms.com	Derby, KS 67037	INSERT INTO Members VALUES ('943667551', 'Justine', 'Mcbride', '(607)898-3165', 'justinemcbride@cms.com', 'Derby, KS 67037');	Administrator
49	45	237393883	Angelique	Davila	(270)493-4957	angelique davila@cms.com	40 Jockey Hollow Dr.	INSERT INTO Members VALUES ('237393883', 'Angelique', 'Davila', '(270)493-4957', 'angelique davila@cms.com', '40 Jockey Hollow Dr.');	Administrator
50	46	435134748	Wilber	Melton	(802)398-4468	wilbermelton@cms.com	Niceville, FL 32578	INSERT INTO Members VALUES ('435134748', 'Wilber', 'Melton', '(802)398-4468', 'wilbermelton@cms.com', 'Niceville, FL 32578');	Administrator
51	47	670334459	Brendon	Frazier	(205)879-2848	brendonfrazier@cms.com	8527 Madison Lane	INSERT INTO Members VALUES ('670334459', 'Brendon', 'Frazier', '(205)879-2848', 'brendonfrazier@cms.com', '8527 Madison Lane');	Administrator
52	48	600242738	Ward	Benitez	(848)219-2884	wardbenitez@cms.com	Windermere, FL 34786	INSERT INTO Members VALUES ('600242738', 'Ward', 'Benitez', '(848)219-2884', 'wardbenitez@cms.com', 'Windermere, FL 34786');	Administrator
53	49	622426523	Colette	Stark	(832)665-2899	colettestark@cms.com	95 Gulf Drive	INSERT INTO Members VALUES ('622426523', 'Colette', 'Stark', '(832)665-2899', 'colettestark@cms.com', '95 Gulf Drive');	Administrator
54	50	567779526	Deangelo	Franklin	(650)559-4049	deangelofranklin@cms.com	Bradenton, FL 34203	INSERT INTO Members VALUES ('567779526', 'Deangelo', 'Franklin', '(650)559-4049', 'deangelofranklin@cms.com', 'Bradenton, FL 34203');	Administrator
55	51	189526755	July	Anne	(202)590-7430	julyanne@cms.com	4 Sugar Street	INSERT INTO Members VALUES ('189526755', 'July', 'Anne', '(202)590-7430', 'julyanne@cms.com', '4 Sugar Street');	Administrator
56	52	638993187	Cary	Lank	(567)200-5875	carylank @cms.com	Warren, MI 48089	INSERT INTO Members VALUES ('638993187', 'Cary', 'Lank', '(567)200-5875', 'carylank @cms.com', 'Warren, MI 48089');	Administrator
57	53	798100317	Honey	Dent	(305)448-4944	honeydent @cms.com	72 N. Pumpkin Hill St.	INSERT INTO Members VALUES ('798100317', 'Honey', 'Dent', '(305)448-4944', 'honeydent @cms.com', '72 N. Pumpkin Hill St.');	Administrator
58	54	944806400	Artie	Smoak	(229)496-5839	artiesmoak@cms.com	Victoria, TX 77904	INSERT INTO Members VALUES ('944806400', 'Artie', 'Smoak', '(229)496-5839', 'artiesmoak@cms.com', 'Victoria, TX 77904');	Administrator
59	55	722640550	Jeff	Hanson	(339)224-3622	jeffhanson@cms.com	892 Walt Whitman Dr.	INSERT INTO Members VALUES ('722640550', 'Jeff', 'Hanson', '(339)224-3622', 'jeffhanson@cms.com', '892 Walt Whitman Dr.');	Administrator
60	56	889872253	Bertram	Carlson	(614)341-4812	bertramcarlson@cms.com	Lake Zurich, IL 60047	INSERT INTO Members VALUES ('889872253', 'Bertram', 'Carlson', '(614)341-4812', 'bertramcarlson@cms.com', 'Lake Zurich, IL 60047');	Administrator
61	57	861394637	Sam	Heister	(270)447-3045	samheister@cms.com	258 Kent Drive	INSERT INTO Members VALUES ('861394637', 'Sam', 'Heister', '(270)447-3045', 'samheister@cms.com', '258 Kent Drive');	Administrator
62	58	320798136	Bob	Bryant	(828)276-7614	bobbryant@cms.com	Lewiston, ME 04240	INSERT INTO Members VALUES ('320798136', 'Bob', 'Bryant', '(828)276-7614', 'bobbryant@cms.com', 'Lewiston, ME 04240');	Administrator
63	59	877078318	Hank	Colley	(530)655-4881	hankcolley@cms.com	444 Hartford Street	INSERT INTO Members VALUES ('877078318', 'Hank', 'Colley', '(530)655-4881', 'hankcolley@cms.com', '444 Hartford Street');	Administrator
64	60	318547937	Tom	Adrian	(207)400-8304	tomadrian@cms.com	Port Orange, FL 32127	INSERT INTO Members VALUES ('318547937', 'Tom', 'Adrian', '(207)400-8304', 'tomadrian@cms.com', 'Port Orange, FL 32127');	Administrator

## Students(*studentID*, *balance*, *unit\_cap*)

Project (Course Management ...)

Administration   Schemas   Query 1   Waitlists   Waitlists   Students   Students   Teaches   Teaches   Transactions   Transactions   Context Help   Snippets

**SCHEMAS**

Name: **Students**   Schema: Course Management System

**Course Management System...**

**Tables**

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students**
  - Teaches
  - Transactions
  - Users
  - Waitlists
- Views
- Stored Procedures
- Functions

**demo**

**Project\_data**

**sys**

**Object Info**   **Session**

**Table: Students**

**Columns:**

studentID	varchar(45) PK
balance	int
unit_cap	int
studentID	varchar(45)
balance	int
unit_cap	int

**Column details 'studentID'**

Column Name: studentID   Datatype: VARCHAR(45)

CharSet/Collation: Default Charset   Default Collation

Comments:

Storage:  VIRTUAL  STORED

Primary Key  Not NULL  Unique  
 Binary  Unsigned  ZeroFill  
 Auto Increment  Generated

**Columns**   **Indexes**   **Foreign Keys**   **Triggers**   **Partitioning**   **Options**   **Apply**   **Revert**

Action Output

Time	Action	Response	Duration / Fetch Time
21 20:38:52	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	10 row(s) returned	0.0034 sec / 0.000...
22 20:38:53	SELECT * FROM 'Course Management System'.Instructors LIMIT 0, 1000	10 row(s) returned	0.00050 sec / 0.000...
23 20:41:17	SELECT * FROM 'Course Management System'.Students LIMIT 0, 1000	10 row(s) returned	0.00069 sec / 0.000...

Query Completed

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

Schemas

Course Management Syst...

Tables

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students**
- Teaches
- Transactions
- Users
- Waitlists
- Views
- Stored Procedures
- Functions
- demo
- Project\_data
- sys

Result Grid Filter Rows: Search Edit: Export/Import:

studentID	balance	unit_cap
199526438	602	15
204722757	72	18
227209241	696	18
314843835	118	16
339820994	69	20
452018548	776	18
483131196	623	20
503432266	437	18
506190495	153	11
538782975	507	20
573062892	451	11
57906941	640	19
670202202	214	11
689582238	446	9
716801899	940	16
760511911	376	10
796058602	320	15
932823238	491	13
951597733	475	18
978594399	626	21
NULL	NULL	NULL

Object Info Session

Table: Students

Columns:

studentID	varchar(45) PK
balance	int
unit_cap	int

Action Output

Time	Action	Response	Duration / Fetch Time
20:38:20	SELECT * FROM `Course Management System`.Registers LIMIT 0, 1000	60 row(s) returned	0.00036 sec / 0.000...
20:40:25	SELECT * FROM `Course Management System`.Students LIMIT 0, 1000	20 row(s) returned	0.00058 sec / 0.000...

Query Completed

	A	B	C	D	E
1	Students				
2	Tuple Number	studentID	balance	unit_cap	
3	1	199526438	107	18	INSERT INTO Students VALUES ('199526438', 107, 18);
4	2	951597733	474	18	INSERT INTO Students VALUES ('951597733', 474, 18);
5	3	483131196	875	17	INSERT INTO Students VALUES ('483131196', 875, 17);
6	4	503432266	927	11	INSERT INTO Students VALUES ('503432266', 927, 11);
7	5	314843835	282	10	INSERT INTO Students VALUES ('314843835', 282, 10);
8	6	760511911	884	11	INSERT INTO Students VALUES ('760511911', 884, 11);
9	7	796058602	256	21	INSERT INTO Students VALUES ('796058602', 256, 21);
10	8	57906941	575	19	INSERT INTO Students VALUES ('57906941', 575, 19);
11	9	538782975	814	11	INSERT INTO Students VALUES ('538782975', 814, 11);
12	10	670202202	833	16	INSERT INTO Students VALUES ('670202202', 833, 16);
13	11	339820994	449	21	INSERT INTO Students VALUES ('339820994', 449, 21);
14	12	978594399	677	8	INSERT INTO Students VALUES ('978594399', 677, 8);
15	13	689582238	802	18	INSERT INTO Students VALUES ('689582238', 802, 18);
16	14	506190495	940	21	INSERT INTO Students VALUES ('506190495', 940, 21);
17	15	932823238	37	18	INSERT INTO Students VALUES ('932823238', 37, 18);
18	16	452018548	548	19	INSERT INTO Students VALUES ('452018548', 548, 19);
19	17	716801899	143	8	INSERT INTO Students VALUES ('716801899', 143, 8);
20	18	573062892	853	9	INSERT INTO Students VALUES ('573062892', 853, 9);
21	19	204722757	762	16	INSERT INTO Students VALUES ('204722757', 762, 16);
22	20	227209241	534	10	INSERT INTO Students VALUES ('227209241', 534, 10);

## *Transactions(studentID, creditcard, amount, timestamp)*

The screenshot shows the MySQL Workbench interface with the 'Transactions' table selected in the 'Tables' section of the left sidebar. The main pane displays the table structure:

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
studentID	CHAR(9)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
creditcard	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
amount	INT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
timestamp	TIMESTAMP	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP

A tooltip on the right says: "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

The 'Object Info' tab at the bottom shows the table definition:

```
Table: Transactions
Columns:
studentID    char(9) PK
creditcard   varchar(45)
amount       int
timestamp    timestamp PK
```

The 'Session' tab shows the execution history:

Action Output	Time	Action	Response	Duration / Fetch Time
9	19:32:17	SELECT * FROM `Course Management System`.Users LIMIT 0, 1000	30 row(s) returned	0.00067 sec / 0.0000...
10	19:32:17	SELECT * FROM `Course Management System`.Users LIMIT 0, 1000	30 row(s) returned	0.00066 sec / 0.000...
11	19:32:46	SELECT * FROM `Course Management System`.Waitlists LIMIT 0, 1000	10 row(s) returned	0.00049 sec / 0.000...

Query Completed

Schemas

Course Management System

Tables

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students
- Teaches
- Transactions**
- Users
- Waitlists

Views

Stored Procedures

Functions

demo

Project\_data

sys

Object Info

Session

Table: Transactions

Columns:

studentID	char(9) PK
creditcard	varchar(45)
amount	int
timestamp	timestamp PK

Transactions 1

Action Output

	Time	Action	Response	Duration / Fetch Time
45	20:42:01	SELECT * FROM 'Course Management System'.Transactions LIMIT 0, 1000	60 row(s) returned	0.00089 sec / 0.000...
46	20:42:02	SELECT * FROM 'Course Management System'.Transactions LIMIT 0, 1000	60 row(s) returned	0.00065 sec / 0.000...

Query Completed

A	B	C	D	E	F
<b>1 Transactions</b>					
2 Tuple Numb:	studentID	creditcard	amount	timestamp	SQL Statement
3 1	199526438	4.72992E+15	23	2020-08-04 14:18:09	INSERT INTO Transactions VALUES ('199526438', 4729921282617690, 23, '2020-08-04 14:18:09');
4 2	951597733	5.57464E+15	23	2020-08-04 14:18:10	INSERT INTO Transactions VALUES ('951597733', 5574644723349960, 23, '2020-08-04 14:18:10');
5 3	483131196	2.95806E+14	401	2020-08-04 14:18:11	INSERT INTO Transactions VALUES ('483131196', 295806044123801, 401, '2020-08-04 14:18:11');
6 4	503432266	8.68541E+15	243	2020-08-04 14:18:12	INSERT INTO Transactions VALUES ('503432266', 8685406565773660, 243, '2020-08-04 14:18:12');
7 5	314843835	7.27848E+15	481	2020-08-04 14:18:13	INSERT INTO Transactions VALUES ('314843835', 727848301932900, 481, '2020-08-04 14:18:13');
8 6	760511911	5.25354E+15	357	2020-08-04 14:18:14	INSERT INTO Transactions VALUES ('760511911', 5253539480864210, 357, '2020-08-04 14:18:14');
9 7	796058602	2.45476E+15	448	2020-08-04 14:18:15	INSERT INTO Transactions VALUES ('796058602', 2454760920399580, 448, '2020-08-04 14:18:15');
10 8	57906941	2.66599E+15	200	2020-08-04 14:18:16	INSERT INTO Transactions VALUES ('57906941', 2665986493051550, 200, '2020-08-04 14:18:16');
11 9	538782975	2.69632E+15	384	2020-08-04 14:18:17	INSERT INTO Transactions VALUES ('538782975', 2696320301756680, 384, '2020-08-04 14:18:17');
12 10	670202202	6.39899E+15	303	2020-08-04 14:18:18	INSERT INTO Transactions VALUES ('670202202', 6398990889602200, 303, '2020-08-04 14:18:18');
13 11	339820994	9.42305E+15	19	2020-08-04 14:18:19	INSERT INTO Transactions VALUES ('339820994', 9423048167971790, 19, '2020-08-04 14:18:19');
14 12	978594399	4.09564E+15	435	2020-08-04 14:18:20	INSERT INTO Transactions VALUES ('978594399', 4095637251471560, 435, '2020-08-04 14:18:20');
15 13	689582238	3.45492E+15	410	2020-08-04 14:18:21	INSERT INTO Transactions VALUES ('689582238', 3454922700343390, 410, '2020-08-04 14:18:21');
16 14	506190495	4.27219E+15	85	2020-08-04 14:18:22	INSERT INTO Transactions VALUES ('506190495', 4272194249026740, 85, '2020-08-04 14:18:22');
17 15	932823238	8.45386E+15	265	2020-08-04 14:18:23	INSERT INTO Transactions VALUES ('932823238', 8453863461888190, 265, '2020-08-04 14:18:23');
18 16	452018548	9.03971E+15	390	2020-08-04 14:18:24	INSERT INTO Transactions VALUES ('452018548', 9039713967280070, 390, '2020-08-04 14:18:24');
19 17	716801899	6.00465E+15	78	2020-08-04 14:18:25	INSERT INTO Transactions VALUES ('716801899', 6004645319995770, 78, '2020-08-04 14:18:25');
20 18	573062892	5.25561E+15	376	2020-08-04 14:18:26	INSERT INTO Transactions VALUES ('573062892', 5255610566974170, 376, '2020-08-04 14:18:26');
21 19	204722757	6.03917E+15	210	2020-08-04 14:18:27	INSERT INTO Transactions VALUES ('204722757', 6039171052900360, 210, '2020-08-04 14:18:27');
22 20	227209241	3.67919E+15	339	2020-08-04 14:18:28	INSERT INTO Transactions VALUES ('227209241', 3679186864519600, 339, '2020-08-04 14:18:28');

24	21	199526438	8.44676E+15	468	2020-08-02 07:09:12	INSERT INTO Transactions VALUES ('199526438', 8446756072343220, 468, '2020-08-02 07:09:12');
25	22	951597733	7.31115E+15	407	2020-08-02 07:09:13	INSERT INTO Transactions VALUES ('951597733', 7311145967256280, 407, '2020-08-02 07:09:13');
26	23	483131196	3.52566E+15	408	2020-08-02 07:09:14	INSERT INTO Transactions VALUES ('483131196', 3525662627988230, 408, '2020-08-02 07:09:14');
27	24	503432266	2.92718E+15	343	2020-08-02 07:09:15	INSERT INTO Transactions VALUES ('503432266', 2927180332102520, 343, '2020-08-02 07:09:15');
28	25	314843835	6.77505E+14	97	2020-08-02 07:09:16	INSERT INTO Transactions VALUES ('314843835', 677504755406390, 97, '2020-08-02 07:09:16');
29	26	760511911	2.02323E+15	402	2020-08-02 07:09:17	INSERT INTO Transactions VALUES ('760511911', 2023228702884830, 402, '2020-08-02 07:09:17');
30	27	796058602	9.74346E+15	479	2020-08-02 07:09:18	INSERT INTO Transactions VALUES ('796058602', 9743461695855310, 479, '2020-08-02 07:09:18');
31	28	57906941	6.37424E+15	379	2020-08-02 07:09:19	INSERT INTO Transactions VALUES ('57906941', 6374235457377450, 379, '2020-08-02 07:09:19');
32	29	538782975	3.58166E+15	281	2020-08-02 07:09:20	INSERT INTO Transactions VALUES ('538782975', 3581656041727260, 281, '2020-08-02 07:09:20');
33	30	670202202	2.1284E+15	226	2020-08-02 07:09:21	INSERT INTO Transactions VALUES ('670202202', 2128397314854200, 226, '2020-08-02 07:09:21');
34	31	339820994	5.38883E+15	438	2020-08-02 07:09:22	INSERT INTO Transactions VALUES ('339820994', 5388827244482690, 438, '2020-08-02 07:09:22');
35	32	978594399	8.83104E+15	250	2020-08-02 07:09:23	INSERT INTO Transactions VALUES ('978594399', 8831037965403030, 250, '2020-08-02 07:09:23');
36	33	689582238	3.23971E+15	175	2020-08-02 07:09:24	INSERT INTO Transactions VALUES ('689582238', 3239714305043840, 175, '2020-08-02 07:09:24');
37	34	506190495	8.23771E+15	440	2020-08-02 07:09:25	INSERT INTO Transactions VALUES ('506190495', 8237707225753060, 440, '2020-08-02 07:09:25');
38	35	932823238	6.00574E+15	487	2020-08-02 07:09:26	INSERT INTO Transactions VALUES ('932823238', 6005742490778090, 487, '2020-08-02 07:09:26');
39	36	452018548	7.53906E+15	42	2020-08-02 07:09:27	INSERT INTO Transactions VALUES ('452018548', 7539058489488120, 42, '2020-08-02 07:09:27');
40	37	716801899	7.35141E+15	112	2020-08-02 07:09:28	INSERT INTO Transactions VALUES ('716801899', 7351406002102060, 112, '2020-08-02 07:09:28');
41	38	573062892	4.29262E+15	18	2020-08-02 07:09:29	INSERT INTO Transactions VALUES ('573062892', 4292623879684210, 18, '2020-08-02 07:09:29');
42	39	204722757	5.72689E+15	118	2020-08-02 07:09:30	INSERT INTO Transactions VALUES ('204722757', 5726891121146300, 118, '2020-08-02 07:09:30');
43	40	227209241	5.71815E+15	128	2020-08-02 07:09:31	INSERT INTO Transactions VALUES ('227209241', 5718149113272290, 128, '2020-08-02 07:09:31');
45	41	199526438	7.74835E+15	490	2020-08-03 19:38:15	INSERT INTO Transactions VALUES ('199526438', 7748348254287430, 490, '2020-08-03 19:38:15');
46	42	951597733	7.92296E+15	319	2020-08-03 19:38:16	INSERT INTO Transactions VALUES ('951597733', 7922956099596220, 319, '2020-08-03 19:38:16');
47	43	483131196	2.47787E+15	28	2020-08-03 19:38:17	INSERT INTO Transactions VALUES ('483131196', 2477869810505720, 28, '2020-08-03 19:38:17');
48	44	503432266	8.22657E+15	276	2020-08-03 19:38:18	INSERT INTO Transactions VALUES ('503432266', 8226574521531990, 276, '2020-08-03 19:38:18');
49	45	314843835	2.59517E+15	123	2020-08-03 19:38:19	INSERT INTO Transactions VALUES ('314843835', 2595167065235700, 123, '2020-08-03 19:38:19');
50	46	760511911	4.95344E+15	416	2020-08-03 19:38:20	INSERT INTO Transactions VALUES ('760511911', 4953438450105470, 416, '2020-08-03 19:38:20');
51	47	796058602	1.64423E+14	240	2020-08-03 19:38:21	INSERT INTO Transactions VALUES ('796058602', 164423418596531, 240, '2020-08-03 19:38:21');
52	48	57906941	2.32964E+15	388	2020-08-03 19:38:22	INSERT INTO Transactions VALUES ('57906941', 2329642515421580, 388, '2020-08-03 19:38:22');
53	49	538782975	7.58959E+15	112	2020-08-03 19:38:23	INSERT INTO Transactions VALUES ('538782975', 7589588117374810, 112, '2020-08-03 19:38:23');
54	50	670202202	1.95301E+15	248	2020-08-03 19:38:24	INSERT INTO Transactions VALUES ('670202202', 1953010143480940, 248, '2020-08-03 19:38:24');
55	51	339820994	3.53861E+15	126	2020-08-03 19:38:25	INSERT INTO Transactions VALUES ('339820994', 3538607479323430, 126, '2020-08-03 19:38:25');
56	52	978594399	5.72018E+15	383	2020-08-03 19:38:26	INSERT INTO Transactions VALUES ('978594399', 5720183787576280, 383, '2020-08-03 19:38:26');
57	53	689582238	4.90607E+15	385	2020-08-03 19:38:27	INSERT INTO Transactions VALUES ('689582238', 4906070290874800, 385, '2020-08-03 19:38:27');
58	54	506190495	4.34154E+15	239	2020-08-03 19:38:28	INSERT INTO Transactions VALUES ('506190495', 4341541452464450, 239, '2020-08-03 19:38:28');
59	55	932823238	1.67709E+14	464	2020-08-03 19:38:29	INSERT INTO Transactions VALUES ('932823238', 167708681282153, 464, '2020-08-03 19:38:29');
60	56	452018548	1.56154E+14	226	2020-08-03 19:38:30	INSERT INTO Transactions VALUES ('452018548', 156154426351262, 226, '2020-08-03 19:38:30');
61	57	716801899	5.5894E+15	288	2020-08-03 19:38:31	INSERT INTO Transactions VALUES ('716801899', 5589400512089820, 288, '2020-08-03 19:38:31');
62	58	573062892	7.47879E+15	192	2020-08-03 19:38:32	INSERT INTO Transactions VALUES ('573062892', 7478788495416250, 192, '2020-08-03 19:38:32');
63	59	204722757	4.78541E+15	24	2020-08-03 19:38:33	INSERT INTO Transactions VALUES ('204722757', 4785408902176240, 24, '2020-08-03 19:38:33');
64	60	227209241	6.62772E+15	25	2020-08-03 19:38:34	INSERT INTO Transactions VALUES ('227209241', 6627718472249090, 25, '2020-08-03 19:38:34');

## Users(ID, username, password)

Project (Course Management ...)

Administration Schemas Query 1 Waitlists Students Students Teaches Transactions Transactions > Context Help Snippets

**Schemas**

Name: Users Schema: Course Management System

**Course Management System**

- Tables
  - Administrators
  - Configurations
  - Courses
  - Instructors
  - Members
  - Registers
  - Students
  - Teaches
  - Transactions
  - Users**
  - Waitlists
- Views
- Stored Procedures
  - Functions
- demo
- Project\_data
- sys

**Object Info** **Session**

**Table: Users**

**Columns:**

ID	char(9) PK
username	varchar(45)
password	varchar(45)

**Column details 'ID'**

Column Name: ID Datatype: CHAR(9)  
 Charset/Collation: Default Charset Default Collation  
 Comments:  
 Storage: VIRTUAL STORED  
 Primary Key  Not NULL  Unique  
 Binary  Unsigned  ZeroFill  
 Auto Increment  Generated

**Columns** **Indexes** **Foreign Keys** **Triggers** **Partitioning** **Options** [Apply] [Revert]

**Action Output**

Action	Time	Response	Duration / Fetch Time
23 20:41:17	SELECT * FROM `Course Management System`.Students LIMIT 0, 1000	10 row(s) returned	0.00069 sec / 0.000...
24 20:41:17	SELECT * FROM `Course Management System`.Students LIMIT 0, 1000	10 row(s) returned	0.00084 sec / 0.000...
25 20:41:57	SELECT * FROM `Course Management System`.Users LIMIT 0, 1000	30 row(s) returned	0.00050 sec / 0.000...

Query Completed

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas Query 1 Users Registers Registers Configurations Members Members Registers Registers > Context Help Snippets

**Schemas**

**Course Management System**

- Tables
  - Administrators
  - Configurations
  - Courses
  - Instructors
  - Members
  - Registers
  - Students
  - Teaches
  - Transactions
  - Users**
  - Waitlists
- Views
- Stored Procedures
  - Functions
- demo
- Project\_data
- sys

**Object Info** **Session**

**Table: Users**

**Columns:**

ID	char(9) PK
username	varchar(45)
password	varchar(45)

**Result Grid** Filter Rows: Search Edit: Export/Import: [grid] [form]

ID	username	password
199526438	pasqualenelson	dtnXMeXQ
204722757	alicesims	PRccsbw
226084272	erinweber	me86fRTE
227209241	raymonbailey	VsYJA7qZ
237393883	angeliqueudavila	EVeemcd8
290327837	nicholefrost	MfcSwbK2
29221500	antoninelemarks	CHWpcLrFy
305981136	pollyrios	aNmfmjZz
307562770	allongalagher	4jBjBy2V
314843835	myrtlehaney	MNN2AiPk
318547937	tomadian	WKSmMK...
320798136	bobbyrant	VP9buH86
337767185	cheriefranch	VlAxDNc
339820994	reynacolon	7V6MXU3a
378503098	jackvargas	75dE3A9S
409326430	sheliaeliot	yMSHdrl5
418130858	inezata	q7Djx3SQ
435313478	wilbermelton	KkLgXbZy
452018548	budkirby	nKGRC6C5
483131196	beatrizdowns	Sv9HCyJ3
503432266	carolynlynn	2M6ubrYH
506190495	vernacopeland	Y2Ue3gBp
538782975	dustinwarren	V6ubrxJ
546664173	antoinedeleon	9BB99DA
549281974	agustinabarret	Bel.lm6q5
573062892	thanheverett	Uusw4YzL
57906941	stevensuarez	P4mvhLsM
587779626	deangelofranklin	vA2u75f3
600242738	wardbenitez	E3zuH3xC
600242738	wardbenitez	vOuMh1D9

Users 1

**Action Output**

Action	Time	Response	Duration / Fetch Time
47 20:42:40	SELECT * FROM `Course Management System`.Users LIMIT 0, 1000	60 row(s) returned	0.00054 sec / 0.000...
48 20:42:40	SELECT * FROM `Course Management System`.Users LIMIT 0, 1000	60 row(s) returned	0.00034 sec / 0.000...

Query Completed

A	B	C	D	E	F	G
1	Users					
2	Tuple Number	ID	Username	Password	SQL Statement	
3	1	199526438	pasqualnelson	dtmXMeXQ	INSERT INTO Users VALUES('199526438', 'pasqualnelson', 'dtmXMeXQ');	Student
4	2	951597733	lupejuarez	sczEflZF	INSERT INTO Users VALUES('951597733', 'lupejuarez', 'sczEflZF');	Student
5	3	483131196	beatrizdowns	Sv9HCyJ3	INSERT INTO Users VALUES('483131196', 'beatrizdowns', 'Sv9HCyJ3');	Student
6	4	503432266	carollynn	2M6u6rYH	INSERT INTO Users VALUES('503432266', 'carollynn', '2M6u6rYH');	Student
7	5	314843835	myrtlehaney	MNM2AtPk	INSERT INTO Users VALUES('314843835', 'myrtlehaney', 'MNM2AtPk');	Student
8	6	760511911	lucianosmith	DZvh6tbG	INSERT INTO Users VALUES('760511911', 'lucianosmith', 'DZvh6tbG');	Student
9	7	796058602	mileslarson	5X4MuBZq	INSERT INTO Users VALUES('796058602', 'mileslarson', '5X4MuBZq');	Student
10	8	57906941	stevensuarez	P4mvNLsM	INSERT INTO Users VALUES('57906941', 'stevensuarez', 'P4mvNLsM');	Student
11	9	538782975	dustinwarren	V6ubrbxJ	INSERT INTO Users VALUES('538782975', 'dustinwarren', 'V6ubrbxJ');	Student
12	10	670202202	mosemcDonald	NaCkSyed	INSERT INTO Users VALUES('670202202', 'mosemcDonald', 'NaCkSyed');	Student
13	11	339820994	reynacolon	7V6MXU3a	INSERT INTO Users VALUES('339820994', 'reynacolon', '7V6MXU3a');	Student
14	12	978594399	ginahernandez	MWwxgHrK	INSERT INTO Users VALUES('978594399', 'ginahernandez', 'MWwxgHrK');	Student
15	13	689582238	danaconway	SG7rYYvr	INSERT INTO Users VALUES('689582238', 'danaconway', 'SG7rYYvr');	Student
16	14	506190495	vernacopeland	Y2Ue3gBp	INSERT INTO Users VALUES('506190495', 'vernacopeland', 'Y2Ue3gBp');	Student
17	15	932823238	jonaspadilla	vYWxrnrY9	INSERT INTO Users VALUES('932823238', 'jonaspadilla', 'vYWxrnrY9');	Student
18	16	452018548	budkirby	nKGR6CR5	INSERT INTO Users VALUES('452018548', 'budkirby', 'nKGR6CR5');	Student
19	17	716801899	roscoerichard	jqtkkEqv	INSERT INTO Users VALUES('716801899', 'roscoerichard', 'jqtkkEqv');	Student
20	18	573062892	thanheverett	Uusw4YzL	INSERT INTO Users VALUES('573062892', 'thanheverett', 'Uusw4YzL');	Student
21	19	204722757	alicesims	PRccSb8w	INSERT INTO Users VALUES('204722757', 'alicesims', 'PRccSb8w');	Student
22	20	227209241	raymonbailey	VsYJATqZ	INSERT INTO Users VALUES('227209241', 'raymonbailey', 'VsYJATqZ');	Student
24	21	226084272	erinweber	me86RtTE	INSERT INTO Users VALUES('226084272', 'erinweber', 'me86RtTE');	Instructor
25	22	378503098	jackvargas	75dE3A9S	INSERT INTO Users VALUES('378503098', 'jackvargas', '75dE3A9S');	Instructor
26	23	307562770	altongallagher	4jBjBy2V	INSERT INTO Users VALUES('307562770', 'altongallagher', '4jBjBy2V');	Instructor
27	24	549281974	agustínbarrett	BeLn6q5	INSERT INTO Users VALUES('549281974', 'agustínbarrett', 'BeLn6q5');	Instructor
28	25	764777061	miguelmathews	EebfEAPh	INSERT INTO Users VALUES('764777061', 'miguelmathews', 'EebfEAPh');	Instructor
29	26	305981136	pollyrios	aNmfmjZz	INSERT INTO Users VALUES('305981136', 'pollyrios', 'aNmfmjZz');	Instructor
30	27	916994127	quinnwarner	eqbUql4c	INSERT INTO Users VALUES('916994127', 'quinnwarner', 'eqbUql4c');	Instructor
31	28	290327837	nicholefrost	MfcswbK2	INSERT INTO Users VALUES('290327837', 'nicholefrost', 'MfcswbK2');	Instructor
32	29	292211500	antoinettemarks	CHWpcLRy	INSERT INTO Users VALUES('292211500', 'antoinettemarks', 'CHWpcLRy');	Instructor
33	30	337767185	cheriefrench	VfLaPDNC	INSERT INTO Users VALUES('337767185', 'cheriefrench', 'VfLaPDNC');	Instructor
34	31	891149217	catalinaschaefer	XYnPd2yF	INSERT INTO Users VALUES('891149217', 'catalinaschaefer', 'XYnPd2yF');	Instructor
35	32	418130858	ineztate	q7Dxj3SQ	INSERT INTO Users VALUES('418130858', 'ineztate', 'q7Dxj3SQ');	Instructor
36	33	100859622	miahodge	d5VjTxRv	INSERT INTO Users VALUES('100859622', 'miahodge', 'd5VjTxRv');	Instructor
37	34	996595763	tamaragibson	K4QpExve	INSERT INTO Users VALUES('996595763', 'tamaragibson', 'K4QpExve');	Instructor
38	35	851438868	clarabradford	PQMkvujJ	INSERT INTO Users VALUES('851438868', 'clarabradford', 'PQMkvujJ');	Instructor
39	36	603160432	normandsheppard	yQVv2gMB	INSERT INTO Users VALUES('603160432', 'normandsheppard', 'yQVv2gMB');	Instructor
40	37	875783848	allenzuniga	YCmHmynj	INSERT INTO Users VALUES('875783848', 'allenzuniga', 'YCmHmynj');	Instructor
41	38	661054866	wesgill	AJCvTtW3	INSERT INTO Users VALUES('661054866', 'wesgill', 'AJCvTtW3');	Instructor
42	39	611668180	edisoncarter	mX4j828U	INSERT INTO Users VALUES('611668180', 'edisoncarter', 'mX4j828U');	Instructor
43	40	957713740	sylviafuente	P4FnnKvG	INSERT INTO Users VALUES('957713740', 'sylviafuente', 'P4FnnKvG');	Instructor
45	41	546664173	antoinedeleon	9BB99DAD	INSERT INTO Users VALUES('546664173', 'antoinedeleon', '9BB99DAD');	Administrator
46	42	620381807	jacksonblankenship	9YkgdG2W	INSERT INTO Users VALUES('620381807', 'jacksonblankenship', '9YkgdG2W');	Administrator
47	43	409326430	sheliaelliott	yMSRdrA5	INSERT INTO Users VALUES('409326430', 'sheliaelliott', 'yMSRdrA5');	Administrator
48	44	943667551	justinemcbride	7xfP2bZJ	INSERT INTO Users VALUES('943667551', 'justinemcbride', '7xfP2bZJ');	Administrator
49	45	237393883	angeliquedavila	EVeecmD8	INSERT INTO Users VALUES('237393883', 'angeliquedavila', 'EVeecmD8');	Administrator
50	46	435313478	wilbermelton	KkLgXbZy	INSERT INTO Users VALUES('435313478', 'wilbermelton', 'KkLgXbZy');	Administrator
51	47	670334459	brendonfrazier	yNQCzf32	INSERT INTO Users VALUES('670334459', 'brendonfrazier', 'yNQCzf32');	Administrator
52	48	600242738	wardbenitez	E3zuH3XC	INSERT INTO Users VALUES('600242738', 'wardbenitez', 'E3zuH3XC');	Administrator
53	49	622426523	colettestark	fBV6czBzR	INSERT INTO Users VALUES('622426523', 'colettestark', 'fBV6czBzR');	Administrator
54	50	587797962	deangeloFranklin	vA2u75f3	INSERT INTO Users VALUES('587797962', 'deangeloFranklin', 'vA2u75f3');	Administrator
55	51	189826755	julyanne	QuWEn3JY	INSERT INTO Users VALUES('189826755', 'julyanne', 'QuWEn3JY');	Administrator
56	52	638993187	carylank	7T4gK8rD	INSERT INTO Users VALUES('638993187', 'carlank', '7T4gK8rD');	Administrator
57	53	798100317	honeydent	HunPcxcs	INSERT INTO Users VALUES('798100317', 'honeydent', 'HunPcxcs');	Administrator
58	54	944806400	artiesmoak	RAw3dPpX	INSERT INTO Users VALUES('944806400', 'artiesmoak', 'RAw3dPpX');	Administrator
59	55	722640650	jeffhanson	35nv3LG3	INSERT INTO Users VALUES('722640650', 'jeffhanson', '35nv3LG3');	Administrator
60	56	889872253	bertramcarlson	9XUBTyct	INSERT INTO Users VALUES('889872253', 'bertramcarlson', '9XUBTyct');	Administrator
61	57	861394637	samheister	cvJuMuLk	INSERT INTO Users VALUES('861394637', 'samheister', 'cvJuMuLk');	Administrator
62	58	320798136	bobbryant	YP9buH86	INSERT INTO Users VALUES('320798136', 'bobbryant', 'YP9buH86');	Administrator
63	59	877078818	hankcolley	uFg2BP3n	INSERT INTO Users VALUES('877078818', 'hankcolley', 'uFg2BP3n');	Administrator
64	60	318547937	tomadrian	WKSmMKQs	INSERT INTO Users VALUES('318547937', 'tomadrian', 'WKSmMKQs');	Administrator

Teaches(instructorID, department, number, configID)

Project (Course Management ...)

Administration Schemas Query 1 Waitlists Students Students Teaches Transactions Transactions > Context Help Snippets

**Schemas**

Name: **Teaches** Schema: Course Management System

**Course Management Syst...**

- Tables
  - Administrators
  - Configurations
  - Courses
  - Instructors
  - Members
  - Registers
  - Students
  - Teaches**
  - Transactions
  - Users
  - Waitlists
- Views
- Stored Procedures
  - demo
- Functions
- sys

**Object Info Session**

**Table: Teaches**

**Columns:**

<b>instructorID</b>	char(9) PK
<b>department</b>	varchar(20) PK
<b>number</b>	varchar(10) PK
<b>configID</b>	int PK

Column details 'instructorID'

Column Name: **instructorID** Datatype: CHAR(9)  
 Charset/Collation: Default Charset Default Collation  
 Comments:  
 Storage: VIRTUAL STORED  
 Primary Key  Not NULL  Unique  
 Binary  Unsigned  Zerofill  
 Auto Increment  Generated

**Columns Indexes Foreign Keys Triggers Partitioning Options** [Apply] [Revert]

Action Output

Action	Response	Duration / Fetch Time
14 19:38:25 SELECT * FROM 'Course Management System'.Transactions LIMIT 0, 1000	12 row(s) returned	0.00052 sec / 0.0000...
15 19:38:34 SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	10 row(s) returned	0.00088 sec / 0.000...
16 19:38:34 SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	10 row(s) returned	0.00049 sec / 0.000...

Query Completed

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas Query 1 Users Registers Registers Configurations Members Members Registers Registers > Context Help Snippets

**Schemas**

1 • **SELECT \* FROM 'Course Management System'.Teaches;**

100% 1:1

**Result Grid** Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

instructorID	departem...	number	configID
100859622	CS	157A	13
100859622	GEO	2A	23
100859622	MATH	3B	3
226084272	CHEM	1A	11
226084272	HIS	17A	21
226084272	Math	127	1
290327837	BIO	1B	8
290327837	KIN	2	28
290327837	SPA	1	18
292211500	KIN	3	29
292211500	PHYS	4A	9
292211500	SPA	2	19
305981136	COMM	2	6
305981136	CS	147	16
305981136	HUM	5B	26
307562770	CS	157A	13
307562770	GEO	2A	23
307562770	MATH	3B	3
337767185	KIN	4	30
337767185	PHYS	4B	10
337767185	SPA	3	20
378503098	CHEM	1B	12
378503098	HIS	17B	22
378503098	MATH	3A	2
419130858	CHEM	1B	12
419130858	HIS	17B	22
419130858	MATH	3A	2
549281974	CS	149	14
549281974	ENG	1A	4
549281974	GEO	2B	24

Teaches 1

Action Output

Action	Response	Duration / Fetch Time
43 20:41:19 SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	60 row(s) returned	0.00050 sec / 0.000...
44 20:41:19 SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	60 row(s) returned	0.00035 sec / 0.000...

Query Completed

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

A	B	C	D	E	F	
1	Teaches					
2	Tuple Number	instructorID	Department	Number	ConfigID	SQL Statement
3	1	226084272	Math	127	1	INSERT INTO Teaches VALUES('226084272', 'Math', '127', 1);
4	2	378503098	MATH	3A	2	INSERT INTO Teaches VALUES('378503098', 'MATH', '3A', 2);
5	3	307562770	MATH	3B	3	INSERT INTO Teaches VALUES('307562770', 'MATH', '3B', 3);
6	4	549281974	ENG	1A	4	INSERT INTO Teaches VALUES('549281974', 'ENG', '1A', 4);
7	5	764777061	ENG	1B	5	INSERT INTO Teaches VALUES('764777061', 'ENG', '1B', 5);
8	6	305981136	COMM	2	6	INSERT INTO Teaches VALUES('305981136', 'COMM', '2', 6);
9	7	916994127	BIO	1A	7	INSERT INTO Teaches VALUES('916994127', 'BIO', '1A', 7);
10	8	290327837	BIO	1B	8	INSERT INTO Teaches VALUES('290327837', 'BIO', '1B', 8);
11	9	292211500	PHYS	4A	9	INSERT INTO Teaches VALUES('292211500', 'PHYS', '4A', 9);
12	10	337767185	PHYS	4B	10	INSERT INTO Teaches VALUES('337767185', 'PHYS', '4B', 10);
13	11	891149217	CHEM	1A	11	INSERT INTO Teaches VALUES('891149217', 'CHEM', '1A', 11);
14	12	418130858	CHEM	1B	12	INSERT INTO Teaches VALUES('418130858', 'CHEM', '1B', 12);
15	13	100859622	CS	157A	13	INSERT INTO Teaches VALUES('100859622', 'CS', '157A', 13);
16	14	996595763	CS	149	14	INSERT INTO Teaches VALUES('996595763', 'CS', '149', 14);
17	15	851438868	CS	146	15	INSERT INTO Teaches VALUES('851438868', 'CS', '146', 15);
18	16	603160432	CS	147	16	INSERT INTO Teaches VALUES('603160432', 'CS', '147', 16);
19	17	875783848	CS	166	17	INSERT INTO Teaches VALUES('875783848', 'CS', '166', 17);
20	18	661054866	SPA	1	18	INSERT INTO Teaches VALUES('661054866', 'SPA', '1', 18);
21	19	611668180	SPA	2	19	INSERT INTO Teaches VALUES('611668180', 'SPA', '2', 19);
22	20	957713740	SPA	3	20	INSERT INTO Teaches VALUES('957713740', 'SPA', '3', 20);
23	21	226084272	HIS	17A	21	INSERT INTO Teaches VALUES('226084272', 'HIS', '17A', 21);
24	22	378503098	HIS	17B	22	INSERT INTO Teaches VALUES('378503098', 'HIS', '17B', 22);
25	23	307562770	GEO	2A	23	INSERT INTO Teaches VALUES('307562770', 'GEO', '2A', 23);
26	24	549281974	GEO	2B	24	INSERT INTO Teaches VALUES('549281974', 'GEO', '2B', 24);
27	25	764777061	HUM	5A	25	INSERT INTO Teaches VALUES('764777061', 'HUM', '5A', 25);
28	26	305981136	HUM	5B	26	INSERT INTO Teaches VALUES('305981136', 'HUM', '5B', 26);
29	27	916994127	KIN	1	27	INSERT INTO Teaches VALUES('916994127', 'KIN', '1', 27);
30	28	290327837	KIN	2	28	INSERT INTO Teaches VALUES('290327837', 'KIN', '2', 28);

31		29	292211500	KIN	3	29	INSERT INTO Teaches VALUES('292211500', 'KIN', '3', 29);
32		30	337767185	KIN	4	30	INSERT INTO Teaches VALUES('337767185', 'KIN', '4', 30);
33		31	891149217	Math	127	1	INSERT INTO Teaches VALUES('891149217', 'Math', '127', 1);
34		32	418130858	MATH	3A	2	INSERT INTO Teaches VALUES('418130858', 'MATH', '3A', 2);
35		33	100859622	MATH	3B	3	INSERT INTO Teaches VALUES('100859622', 'MATH', '3B', 3);
36		34	996595763	ENG	1A	4	INSERT INTO Teaches VALUES('996595763', 'ENG', '1A', 4);
37		35	851438868	ENG	1B	5	INSERT INTO Teaches VALUES('851438868', 'ENG', '1B', 5);
38		36	603160432	COMM	2	6	INSERT INTO Teaches VALUES('603160432', 'COMM', '2', 6);
39		37	875783848	BIO	1A	7	INSERT INTO Teaches VALUES('875783848', 'BIO', '1A', 7);
40		38	661054866	BIO	1B	8	INSERT INTO Teaches VALUES('661054866', 'BIO', '1B', 8);
41		39	611668180	PHYS	4A	9	INSERT INTO Teaches VALUES('611668180', 'PHYS', '4A', 9);
42		40	957713740	PHYS	4B	10	INSERT INTO Teaches VALUES('957713740', 'PHYS', '4B', 10);
43		41	226084272	CHEM	1A	11	INSERT INTO Teaches VALUES('226084272', 'CHEM', '1A', 11);
44		42	378503098	CHEM	1B	12	INSERT INTO Teaches VALUES('378503098', 'CHEM', '1B', 12);
45		43	307562770	CS	157A	13	INSERT INTO Teaches VALUES('307562770', 'CS', '157A', 13);
46		44	549281974	CS	149	14	INSERT INTO Teaches VALUES('549281974', 'CS', '149', 14);
47		45	764777061	CS	146	15	INSERT INTO Teaches VALUES('764777061', 'CS', '146', 15);
48		46	305981136	CS	147	16	INSERT INTO Teaches VALUES('305981136', 'CS', '147', 16);
49		47	916994127	CS	166	17	INSERT INTO Teaches VALUES('916994127', 'CS', '166', 17);
50		48	290327837	SPA	1	18	INSERT INTO Teaches VALUES('290327837', 'SPA', '1', 18);
51		49	292211500	SPA	2	19	INSERT INTO Teaches VALUES('292211500', 'SPA', '2', 19);
52		50	337767185	SPA	3	20	INSERT INTO Teaches VALUES('337767185', 'SPA', '3', 20);
53		51	891149217	HIS	17A	21	INSERT INTO Teaches VALUES('891149217', 'HIS', '17A', 21);
54		52	418130858	HIS	17B	22	INSERT INTO Teaches VALUES('418130858', 'HIS', '17B', 22);
55		53	100859622	GEO	2A	23	INSERT INTO Teaches VALUES('100859622', 'GEO', '2A', 23);
56		54	996595763	GEO	2B	24	INSERT INTO Teaches VALUES('996595763', 'GEO', '2B', 24);
57		55	851438868	HUM	5A	25	INSERT INTO Teaches VALUES('851438868', 'HUM', '5A', 25);
58		56	603160432	HUM	5B	26	INSERT INTO Teaches VALUES('603160432', 'HUM', '5B', 26);
59		57	875783848	KIN	1	27	INSERT INTO Teaches VALUES('875783848', 'KIN', '1', 27);
60		58	661054866	KIN	2	28	INSERT INTO Teaches VALUES('661054866', 'KIN', '2', 28);
61		59	611668180	KIN	3	29	INSERT INTO Teaches VALUES('611668180', 'KIN', '3', 29);
62		60	957713740	KIN	4	30	INSERT INTO Teaches VALUES('957713740', 'KIN', '4', 30);

Registers(studentID, department, number, configID)

Project (Course Management ...)

Administration Schemas

**SCHEMAS**

Name: Registers Schema: Course Management System

Tables

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers**
- Students
- Teaches
- Transactions
- Users
- Waitlists
- Views
- Stored Procedures
- Functions
- demo
- Project\_data
- sys

Object Info Session

Table: Registers

Columns:

studentID	char(9) PK
department	varchar(20) PK
number	varchar(10) PK
configID	int PK

Column details 'studentID'

Column Name: studentID Datatype: CHAR(9)  
Charset/Collation: Default Charset Default  
Comments:  
Storage: VIRTUAL STORED Primary Key Not NULL Unique  
Binary Unsigned ZeroFill Auto Increment Generated

Columns Indexes Foreign Keys Triggers Partitioning Options Apply Revert

Action Output

Time	Action	Response	Duration / Fetch Time
18 19:40:19	SELECT * FROM `Course Management System`.Waitlists LIMIT 0, 1000	10 row(s) returned	0.0028 sec / 0.00023...
19 19:42:16	SELECT * FROM `Course Management System`.Registers LIMIT 0, 1000	24 row(s) returned	0.00053 sec / 0.000...
20 19:42:17	SELECT * FROM `Course Management System`.Registers LIMIT 0, 1000	24 row(s) returned	0.00055 sec / 0.000...

Query Completed

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas

**SCHEMAS**

Filter objects

Course Management Syst...

Tables

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers**
- Students
- Teaches
- Transactions
- Users
- Waitlists
- Views
- Stored Procedures
- Functions
- demo
- Project\_data
- sys

Object Info Session

Table: Registers

Columns:

studentID	char(9) PK
department	varchar(20) PK
number	varchar(10) PK
configID	int PK

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

1 • `SELECT * FROM `Course Management System`.Registers;`

100% 1:1

studentID department number configID

199526438	CHEM	1A	11
199526438	HIS	17A	21
199526438	Math	127	1
204722757	KIN	3	29
204722757	PHYS	4A	9
204722757	SPA	2	19
227209241	KIN	4	30
227209241	PHYS	4B	10
227209241	SPA	3	20
314843835	CS	146	15
314843835	ENG	1B	5
314843835	HUM	5A	25
339820994	CHEM	1A	11
339820994	HIS	17A	21
339820994	Math	127	1
452018548	COMM	2	6
452018548	CS	147	16
452018548	HUM	5B	26
483131196	CS	157A	13
483131196	GEO	2A	23
483131196	MATH	3B	3
503432266	CS	149	14
503432266	ENG	1A	4
503432266	GEO	2B	24
506190495	CS	149	14
506190495	ENG	1A	4
506190495	GEO	2B	24
538782975	KIN	3	29
538782975	PHYS	4A	9
538782975	SPA	2	19

Registers 1

Action Output

Time	Action	Response	Duration / Fetch Time
38 20:37:32	SELECT * FROM `Course Management System`.Members LIMIT 0, 1000	60 row(s) returned	0.00040 sec / 0.000...
39 20:38:20	SELECT * FROM `Course Management System`.Registers LIMIT 0, 1000	60 row(s) returned	0.00071 sec / 0.000...

Query Completed

	A	B	C	D	E	F
1		Registers				Each person 2 courses
2		SID	Department	Number	ConfigID	SQL Statement
3	1	199526438	Math	127	1	INSERT INTO Registers VALUES('199526438', 'Math', '127', 1);
4	2	951597733	MATH	3A	2	INSERT INTO Registers VALUES('951597733', 'MATH', '3A', 2);
5	3	483131196	MATH	3B	3	INSERT INTO Registers VALUES('483131196', 'MATH', '3B', 3);
6	4	503432266	ENG	1A	4	INSERT INTO Registers VALUES('503432266', 'ENG', '1A', 4);
7	5	314843835	ENG	1B	5	INSERT INTO Registers VALUES('314843835', 'ENG', '1B', 5);
8	6	760511911	COMM	2	6	INSERT INTO Registers VALUES('760511911', 'COMM', '2', 6);
9	7	796058602	BIO	1A	7	INSERT INTO Registers VALUES('796058602', 'BIO', '1A', 7);
10	8	57906941	BIO	1B	8	INSERT INTO Registers VALUES('57906941', 'BIO', '1B', 8);
11	9	538782975	PHYS	4A	9	INSERT INTO Registers VALUES('538782975', 'PHYS', '4A', 9);
12	10	670202202	PHYS	4B	10	INSERT INTO Registers VALUES('670202202', 'PHYS', '4B', 10);
13	11	339820994	CHEM	1A	11	INSERT INTO Registers VALUES('339820994', 'CHEM', '1A', 11);
14	12	978594399	CHEM	1B	12	INSERT INTO Registers VALUES('978594399', 'CHEM', '1B', 12);
15	13	689582238	CS	157A	13	INSERT INTO Registers VALUES('689582238', 'CS', '157A', 13);
16	14	506190495	CS	149	14	INSERT INTO Registers VALUES('506190495', 'CS', '149', 14);
17	15	932823238	CS	146	15	INSERT INTO Registers VALUES('932823238', 'CS', '146', 15);
18	16	452018548	CS	147	16	INSERT INTO Registers VALUES('452018548', 'CS', '147', 16);
19	17	716801899	CS	166	17	INSERT INTO Registers VALUES('716801899', 'CS', '166', 17);
20	18	573062892	SPA	1	18	INSERT INTO Registers VALUES('573062892', 'SPA', '1', 18);
21	19	204722757	SPA	2	19	INSERT INTO Registers VALUES('204722757', 'SPA', '2', 19);
22	20	227209241	SPA	3	20	INSERT INTO Registers VALUES('227209241', 'SPA', '3', 20);
23	21	199526438	HIS	17A	21	INSERT INTO Registers VALUES('199526438', 'HIS', '17A', 21);
24	22	951597733	HIS	17B	22	INSERT INTO Registers VALUES('951597733', 'HIS', '17B', 22);
25	23	483131196	GEO	2A	23	INSERT INTO Registers VALUES('483131196', 'GEO', '2A', 23);
26	24	503432266	GEO	2B	24	INSERT INTO Registers VALUES('503432266', 'GEO', '2B', 24);
27	25	314843835	HUM	5A	25	INSERT INTO Registers VALUES('314843835', 'HUM', '5A', 25);
28	26	760511911	HUM	5B	26	INSERT INTO Registers VALUES('760511911', 'HUM', '5B', 26);
29	27	796058602	KIN	1	27	INSERT INTO Registers VALUES('796058602', 'KIN', '1', 27);
30	28	57906941	KIN	2	28	INSERT INTO Registers VALUES('57906941', 'KIN', '2', 28);

33	31	339820994	Math	127	1	INSERT INTO Registers VALUES('339820994', 'Math', '127', 1);
34	32	978594399	MATH	3A	2	INSERT INTO Registers VALUES('978594399', 'MATH', '3A', 2);
35	33	689582238	MATH	3B	3	INSERT INTO Registers VALUES('689582238', 'MATH', '3B', 3);
36	34	506190495	ENG	1A	4	INSERT INTO Registers VALUES('506190495', 'ENG', '1A', 4);
37	35	932823238	ENG	1B	5	INSERT INTO Registers VALUES('932823238', 'ENG', '1B', 5);
38	36	452018548	COMM	2	6	INSERT INTO Registers VALUES('452018548', 'COMM', '2', 6);
39	37	716801899	BIO	1A	7	INSERT INTO Registers VALUES('716801899', 'BIO', '1A', 7);
40	38	573062892	BIO	1B	8	INSERT INTO Registers VALUES('573062892', 'BIO', '1B', 8);
41	39	204722757	PHYS	4A	9	INSERT INTO Registers VALUES('204722757', 'PHYS', '4A', 9);
42	40	227209241	PHYS	4B	10	INSERT INTO Registers VALUES('227209241', 'PHYS', '4B', 10);
43	41	199526438	CHEM	1A	11	INSERT INTO Registers VALUES('199526438', 'CHEM', '1A', 11);
44	42	951597733	CHEM	1B	12	INSERT INTO Registers VALUES('951597733', 'CHEM', '1B', 12);
45	43	483131196	CS	157A	13	INSERT INTO Registers VALUES('483131196', 'CS', '157A', 13);
46	44	503432266	CS	149	14	INSERT INTO Registers VALUES('503432266', 'CS', '149', 14);
47	45	314843835	CS	146	15	INSERT INTO Registers VALUES('314843835', 'CS', '146', 15);
48	46	760511911	CS	147	16	INSERT INTO Registers VALUES('760511911', 'CS', '147', 16);
49	47	796058602	CS	166	17	INSERT INTO Registers VALUES('796058602', 'CS', '166', 17);
50	48	57906941	SPA	1	18	INSERT INTO Registers VALUES('57906941', 'SPA', '1', 18);
51	49	538782975	SPA	2	19	INSERT INTO Registers VALUES('538782975', 'SPA', '2', 19);
52	50	670202202	SPA	3	20	INSERT INTO Registers VALUES('670202202', 'SPA', '3', 20);
53	51	339820994	HIS	17A	21	INSERT INTO Registers VALUES('339820994', 'HIS', '17A', 21);
54	52	978594399	HIS	17B	22	INSERT INTO Registers VALUES('978594399', 'HIS', '17B', 22);
55	53	689582238	GEO	2A	23	INSERT INTO Registers VALUES('689582238', 'GEO', '2A', 23);
56	54	506190495	GEO	2B	24	INSERT INTO Registers VALUES('506190495', 'GEO', '2B', 24);
57	55	932823238	HUM	5A	25	INSERT INTO Registers VALUES('932823238', 'HUM', '5A', 25);
58	56	452018548	HUM	5B	26	INSERT INTO Registers VALUES('452018548', 'HUM', '5B', 26);
59	57	716801899	KIN	1	27	INSERT INTO Registers VALUES('716801899', 'KIN', '1', 27);
60	58	573062892	KIN	2	28	INSERT INTO Registers VALUES('573062892', 'KIN', '2', 28);
61	59	204722757	KIN	3	29	INSERT INTO Registers VALUES('204722757', 'KIN', '3', 29);
62	60	227209241	KIN	4	30	INSERT INTO Registers VALUES('227209241', 'KIN', '4', 30);

Waitlists(studentID, department, number, configID)

Project (Course Management ...)

Project (Course Management ...)

Administration Schemas Query 1 Waitlists Students Students Teaches Transactions Transactions > Context Help Snippets

**Schemas**

Name: Waitlists Schema: Course Management System

**Tables**

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students
- Teaches
- Transactions
- Users
- Waitlists

**Views**

Stored Procedures Functions

demo Project\_data sys

**Object Info Session**

**Table: Waitlists**

**Columns:**

studentID	char(9) PK
department	varchar(20) PK
number	varchar(10) PK
configID	int PK

**Column details 'studentID'**

Column Name: studentID Datatype: CHAR(9) Default: Charset/Collation: Default Charset / Default Collation Comments: Storage: VIRTUAL / STORED Primary Key: ✓ Not NULL: ✓ Unique: Binary: Unsigned: ZeroFill: Auto Increment: Generated

**Action Output**

Time	Action	Response	Duration / Fetch Time
15 19:38:34	SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	10 row(s) returned	0.00088 sec / 0.000...
16 19:38:34	SELECT * FROM 'Course Management System'.Teaches LIMIT 0, 1000	10 row(s) returned	0.00049 sec / 0.000...
17 19:40:19	SELECT * FROM 'Course Management System'.Waitlists LIMIT 0, 1000	10 row(s) returned	0.00068 sec / 0.000...

Query Completed

MySQLWorkbench File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas Query 1 Users Registers Configurations Members Registers Registers > Context Help Snippets

**Schemas**

Filter objects

**Tables**

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students
- Teaches
- Transactions
- Users
- Waitlists

**Views**

Stored Procedures Functions

demo Project\_data sys

**Object Info Session**

**Table: Waitlists**

**Result Grid**

1 • | SELECT \* FROM `Course Management System`.Waitlists;

100% 1:1

studentID	departme...	number	configID
199526438	SPA	3	20
204722757	SPA	1	18
227209241	SPA	2	19
314843835	ENG	1A	4
339820994	PHYS	4B	10
452018548	CS	146	15
483131196	MATH	3A	2
603432266	MATH	3B	3
506190495	CS	157A	13
538782975	BIO	1B	8
573062892	CS	166	17
57906941	BIO	1A	7
670202202	PHYS	4A	9
689582238	CHEM	1B	12
716801899	CS	147	16
760519111	ENG	1B	5
796058602	COMM	2	6
932823238	CS	149	14
951597733	Math	127	1
978594399	CHEM	1A	11
NULL	NULL	NULL	NULL

**Execution Plan**

**Action Output**

Time	Action	Response	Duration / Fetch Time
48 20:42:40	SELECT * FROM 'Course Management System'.Users LIMIT 0, 1000	60 row(s) returned	0.00034 sec / 0.000...
49 20:43:23	SELECT * FROM 'Course Management System'.Waitlists LIMIT 0, 1000	20 row(s) returned	0.00058 sec / 0.000...

Query Completed

A	B	C	D	E	F
1	<b>Waitlists</b>				Each person 2 courses
2	SID	Department	Number	ConfigID	SQL Statement
3 1	199526438	SPA	3	20	INSERT INTO Waitlists VALUES('199526438', 'SPA', '3', 20);
4 2	951597733	Math	127	1	INSERT INTO Waitlists VALUES('951597733', 'Math', '127', 1);
5 3	483131196	MATH	3A	2	INSERT INTO Waitlists VALUES('483131196', 'MATH', '3A', 2);
6 4	503432266	MATH	3B	3	INSERT INTO Waitlists VALUES('503432266', 'MATH', '3B', 3);
7 5	314843835	ENG	1A	4	INSERT INTO Waitlists VALUES('314843835', 'ENG', '1A', 4);
8 6	760511911	ENG	1B	5	INSERT INTO Waitlists VALUES('760511911', 'ENG', '1B', 5);
9 7	796058602	COMM	2	6	INSERT INTO Waitlists VALUES('796058602', 'COMM', '2', 6);
10 8	57906941	BIO	1A	7	INSERT INTO Waitlists VALUES('57906941', 'BIO', '1A', 7);
11 9	538782975	BIO	1B	8	INSERT INTO Waitlists VALUES('538782975', 'BIO', '1B', 8);
12 10	670202202	PHYS	4A	9	INSERT INTO Waitlists VALUES('670202202', 'PHYS', '4A', 9);
13 11	339820994	PHYS	4B	10	INSERT INTO Waitlists VALUES('339820994', 'PHYS', '4B', 10);
14 12	978594399	CHEM	1A	11	INSERT INTO Waitlists VALUES('978594399', 'CHEM', '1A', 11);
15 13	689582238	CHEM	1B	12	INSERT INTO Waitlists VALUES('689582238', 'CHEM', '1B', 12);
16 14	506190495	CS	157A	13	INSERT INTO Waitlists VALUES('506190495', 'CS', '157A', 13);
17 15	932823238	CS	149	14	INSERT INTO Waitlists VALUES('932823238', 'CS', '149', 14);
18 16	452018548	CS	146	15	INSERT INTO Waitlists VALUES('452018548', 'CS', '146', 15);
19 17	716801899	CS	147	16	INSERT INTO Waitlists VALUES('716801899', 'CS', '147', 16);
20 18	573062892	CS	166	17	INSERT INTO Waitlists VALUES('573062892', 'CS', '166', 17);
21 19	204722757	SPA	1	18	INSERT INTO Waitlists VALUES('204722757', 'SPA', '1', 18);
22 20	227209241	SPA	2	19	INSERT INTO Waitlists VALUES('227209241', 'SPA', '2', 19);

# Project Implementation

We started with a simple index.jsp and LoginAction.java servlet. We ran these two files to check how to run a simple web application using Tomcat server. Then eventually we added the main java files: Instructors, Administrators, Students, Courses and their corresponding .jsp files.

The java classes use SQL statements to interact with the database. During development, we first developed the queries and tested on MySQLWorkbench. Once they were functional, we incorporated the SQL statements inside the Java class and tested the output on the console. Finally, we connected the Java class to the jsp file, ran the same statement and displayed the output in the browser. The figure below shows the workflow in the MySQLWorkbench.

The screenshot shows the MySQL Workbench interface. On the left is a tree view of the 'Course Management Syst...' schema, with 'Registers' selected. The central pane displays a query results grid for a JOIN operation between 'Registers' and 'Members'. The right pane shows the query text and a status message about context help. Below the main area, there's a 'Session' tab showing table definitions and a history of actions at the bottom.

studentID	department	number	configID	ID	firstname	lastname	phone	email	address
227209241	KIN	4	30	227209241	Raymon	Bailey	(750) 488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.
227209241	PHYS	4B	10	227209241	Raymon	Bailey	(750) 488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.
227209241	SPA	3	20	227209241	Raymon	Bailey	(750) 488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.

The figure below shows the workflow in the Java class.

```

try {
    // Check if course is already being taught
    if (isTeaching(department, number, configID)) return false;
    pstate = SQLMethods.con.prepareStatement(sql: "INSERT INTO Teaches VALUES (?, ?, ?, ?)");
    pstate.setString( parameterIndex: 1, instructorID);
    pstate.setString( parameterIndex: 2, department);
    pstate.setInt( parameterIndex: 3, number);
    pstate.setInt( parameterIndex: 4, configID);
    pstate.executeUpdate(); // Execute query
    SQLMethods.closeConnection(); // Close connection
    return true; // Successful insert
} catch (SQLException e) { // Print error and terminate program
    SQLMethods.mysql_fatal_error("Query error: " + e.toString());
}
return false; // Default value: false
}

private static boolean isTeaching(String department, String number, int configID) throws SQLException {
    /* Check for invalid inputs. If any input is null, return false */
    if (department == null || number == null || configID < 0) return false;
    pstate = SQLMethods.con.prepareStatement(sql: "SELECT COUNT(*) FROM Teaches WHERE department = ? AND number = ? AND configID = ?");
    pstate.setString( parameterIndex: 1, department);
    pstate.setString( parameterIndex: 2, number);
    pstate.setInt( parameterIndex: 3, configID);
    result = pstate.executeQuery();
    result.next();
    int rowcount = result.getInt( columnIndex: 1); // Get row count
    result.close(); // Close result
    return rowcount > 0; // If greater than 0, course is taught
}

```

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Tree:** Shows the project structure under "Team4FinalProject". The "teaches" package contains a class named "Teaches".
- Code Editor:** Displays the Java code for "Teaches.java". The code includes methods for inserting data into the "Teaches" table and checking if a specific course (department, number, configID) is already taught.
- Run Tab:** Set to "Tomcat 9.0.36".
- Server Tab:** Shows logs for "Tomcat Catalina Log" and "Tomcat Localhost Log".
- Output Tab:** Shows deployment logs for "Team4FinalProject:war exploded".
- Bottom Status Bar:** Shows the build status: "Build completed successfully in 4 s 514 ms (today 8:02 PM)".

## *Register in the system*

When we run the project, we start from the homepage index.jsp. Visitors go to the homepage.



Code for index.jsp.

```

8@<style>
9  body {
10    background: url(pictures/home.jpg) no-repeat;
11    background-size: cover;
12    margin: 0;
13    padding: 0;
14    font-family: Arial, Helvetica, sans-serif;
15  }
16
17 .topnav {
18   overflow: hidden;
19   background-color: #030024;
20 }
21
22 .topnav a {
23   float: left;
24   color: #f2f2f2;
25   text-align: center;
26   padding: 14px 16px;
27   text-decoration: none;
28   font-size: 17px;
29 }
30
31 .topnav a:hover {
32   background-color: #ddd;
33   color: black;
34 }
35
36 .topnav a.active {
37   background-color: #010d42;
38   color: white;
39 }
40 </style>
41 </head>
42@<body>
43@  <div class="topnav">
44@    <a class="active" href="index.jsp">Course Management System</a> <a
45@      href="login.jsp">Login</a> <a href="register.jsp">Register</a> <a href="#about">About</a>
46@  </div>
47@  <h1 style="color: white; text-align: center; font-size: 200%;">Welcome
48@    to the Course Management System</h1>
49
50@  <h1 style="color: white; text-align: center; font-size: 150%;">Please
51@    log in or register!</h1>
52 </body>
53 </html>
54

```

If visitors want to register, they click the “Register” button in the navigation bar. They will then be directed to a form. The form will only submit if the fields are not null and if the account type is either ‘student’, ‘instructor’, or ‘administrator’.

## Register

First Name

Last Name

Phone

Email

Address

Type (student, instructor, administrator)

[submit](#)

Code for register.jsp

```

76 .topnav {
77   overflow: hidden;
78   background-color: #030024;
79 }
80
81 .topnav a {
82   float: left;
83   color: #f2f2f2;
84   text-align: center;
85   padding: 14px 16px;
86   text-decoration: none;
87   font-size: 17px;
88 }
89
90 .topnav a:hover {
91   background-color: #ddd;
92   color: black;
93 }
94
95 .topnav a.active {
96   background-color: #010d42;
97   color: white;
98 }
99 </style>
100 </head>
101<body>
102<div class="topnav">
103  <a class="active" href="index.jsp">Course Management System</a> <a
104    href="login.jsp">Login</a> <a href="register.jsp">Register</a> <a href="#about">About</a>
105 </div>
106<div class="contact-section">
107   <h1>Register</h1>
108   <div class="border"></div>
109<form class="contact-form" action="registerAction" method="post">
110   <input type="text" class="contact-form-text" placeholder="First Name" name="firstname">
111   <input type="text" class="contact-form-text" placeholder="Last Name" name="lastname">
112   <input type="text" class="contact-form-text" placeholder="Phone" name="phone">
113   <input type="text" class="contact-form-text" placeholder="Email" name="email">
114   <input type="text" class="contact-form-text" placeholder="Address" name="address">
115   <input type="text" class="contact-form-text" placeholder="Type (student, instructor, administrat
116   <input type="submit" class="contact-form-btn" value="submit">
117 </form>
118 </div>
119 </body>
120 </html>
---
```

When we submit this form, we then go to the serverlet registerAction which is registerAction.java.

```

▲36④ protected void doPost(HttpServletRequest request, HttpServletResponse response)
37     throws ServletException, IOException {
38     // Get fields from form
39     String firstname = request.getParameter("firstname");
40     String lastname = request.getParameter("lastname");
41     String phone = request.getParameter("phone");
42     String email = request.getParameter("email");
43     String address = request.getParameter("address");
44     String type = request.getParameter("type");
45
46     /** If any fields are empty, return back to the form */
47     if (firstname.isEmpty() || lastname.isEmpty() || phone.isEmpty() || email.isEmpty() || address.isEmpty()
48         || type.isEmpty()) {
49         RequestDispatcher req = request.getRequestDispatcher("register.jsp"); // Return to the form
50         req.include(request, response);
51     } else {
52         /** Generate ID */
53         Random rand = new Random();
54         String ID = Integer.toString(rand.nextInt(1000000000)); // Generate ID (9 digits)
55         /** Check if ID already exists. Keep on generating until find an ID that DNE */
56         while (!Members.checkID(ID))
57             ID = Integer.toString(rand.nextInt(1000000000)); // Generate ID (9 digits)
58         /** Get type of user (student, instructor, administrator). */
59         if (type.equalsIgnoreCase("administrator")) {
60             Administrators.insert(ID, 0); // Clearance = 0 (no access until an existing admin updates)
61             Members.insert(ID, firstname, lastname, phone, email, address); // Add member to DB
62             /** Go to confirmation page */
63             RequestDispatcher req = request.getRequestDispatcher("confirmation.jsp"); // Go to confirmation
64             req.forward(request, response);
65         } else if (type.equalsIgnoreCase("instructor")) {
66             Instructors.insert(ID, "Inactive");
67             Members.insert(ID, firstname, lastname, phone, email, address); // Add member to DB
68             /** Go to confirmation page */
69             RequestDispatcher req = request.getRequestDispatcher("confirmation.jsp"); // Go to confirmation
70             req.forward(request, response);
71         } else if (type.equalsIgnoreCase("student")) { // Student
72             Students.insert(ID, 0, 12); // Initial unit cap is 12
73             Members.insert(ID, firstname, lastname, phone, email, address); // Add member to DB
74             /** Go to confirmation page */
75             RequestDispatcher req = request.getRequestDispatcher("confirmation.jsp"); // Go to confirmation
76             req.forward(request, response);
77         } else { // Invalid type. Return to form.
78             /** If invalid type, return back to form */
79             RequestDispatcher req = request.getRequestDispatcher("register.jsp"); // Return to the form
80             req.include(request, response);
81         }
82     }
83 }

```

If any fields are null, return to the user. If the type is not valid, return. Otherwise, add the member to the database using this method. Also, insert into the table corresponding to the type of user.

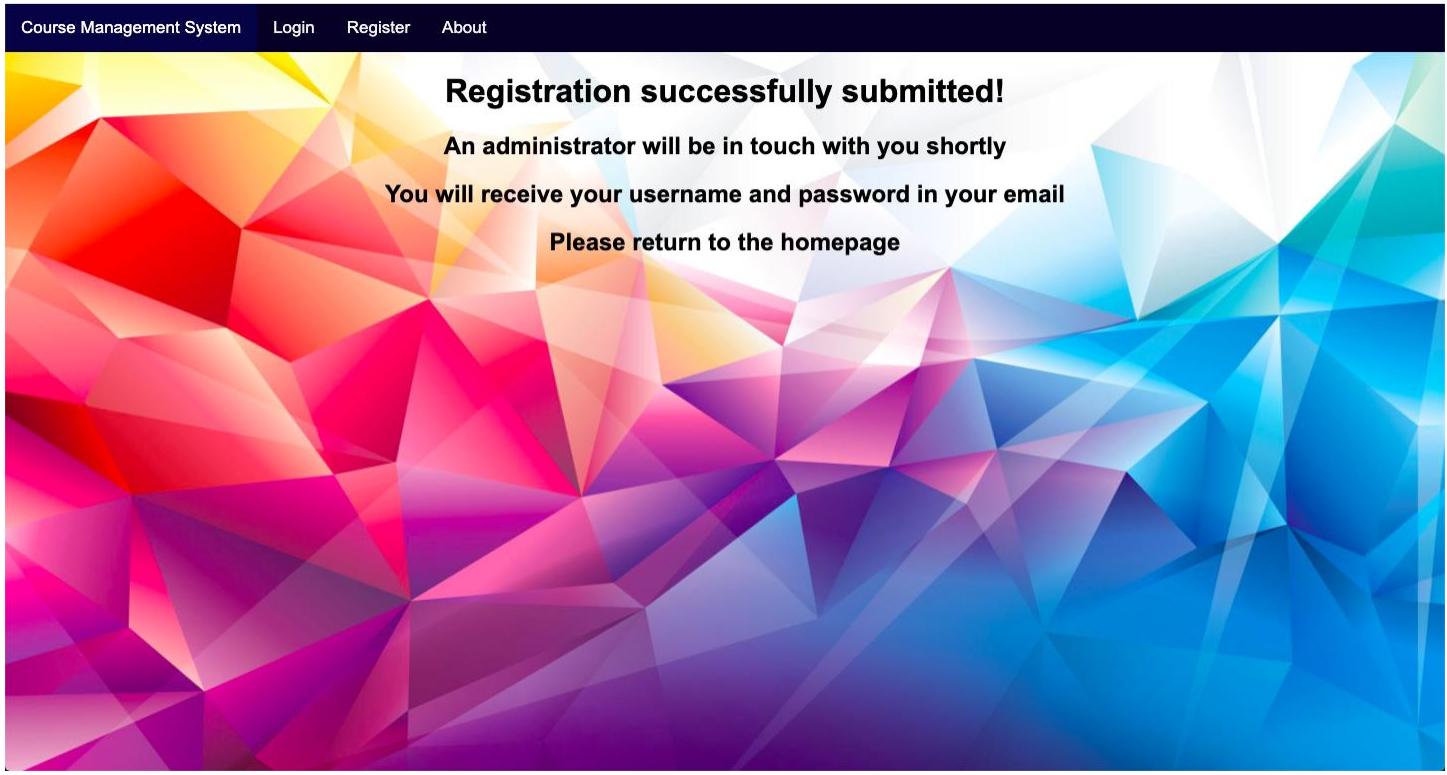
Insert method in members to add user to database:

```

*/
public static boolean insert(String ID, String firstname, String lastname, String phone, String email,
    String address) {
    /** Check for invalid inputs. If any input is null, return false */
    if (ID == null || firstname == null || lastname == null || phone == null || email == null || address == null)
        return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to insert
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Members VALUES(?, ?, ?, ?, ?, ?);");
        pstate.setString(1, ID);
        pstate.setString(2, firstname);
        pstate.setString(3, lastname);
        pstate.setString(4, phone);
        pstate.setString(5, email);
        pstate.setString(6, address);
        int rowcount = pstate.executeUpdate();
        SQLMethods.closeConnection(); // Close connection
        return (rowcount == 1); // If rowcount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}

```

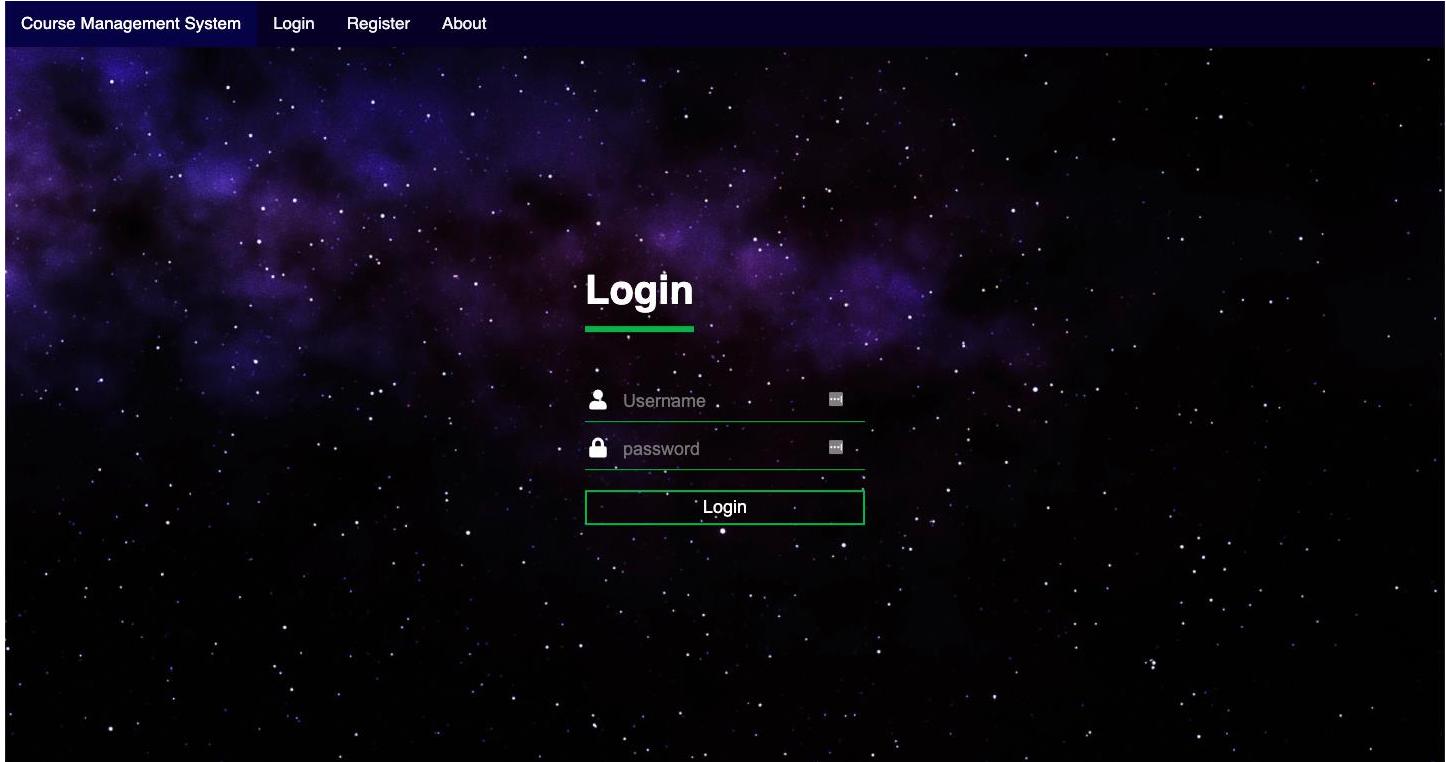
If wrong information is entered, they will get back to the form. If the requirements were met, they will get this confirmation message. An admin will soon email the account credentials.



The visitor can now get back to the homepage or leave.

### *Log in*

Visitors go to the homepage and then click the 'Login' button in the navigation bar. They will be redirected to this page.



Code for login.jsp:

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4<html>
5<head>
6 <meta charset="UTF-8">
7 <title>User Login</title>
8
9 <link href="style.css" rel="stylesheet" type="text/css"/>
10 </head>
11<body>
12<div class="topnav">
13    <a class="active" href="index.jsp">Course Management System</a> <a
14        href="login.jsp">Login</a> <a href="register.jsp">Register</a> <a
15        href="#about">About</a>
16</div>
17<div class="login-box">
18    <h1>Login</h1>
19<form action="LoginAction" method="post">
20    <div class="textbox">
21        <i class="fas fa-user"></i> <input type="text"
22            placeholder="Username" name="username">
23    </div>
24    <div class="textbox">
25        <i class="fas fa-lock"></i> <input type="password"
26            placeholder="password" name="password">
27    </div>
28    <input type="submit" class="btn" value="Login">
29</form>
30</div>
31</body>
32</html>
33
```

Then they enter credentials and be redirected to a custom portal depending on their user type.

## Student Portal

Student Portal Add Courses Drop Courses View Courses Transcript Transactions Logout

### Student Portal

Welcome Student Pasquale Nelson (199526438)

Code for studentPortal.jsp:

```
1  %@ page import="users.Users,members.Members,java.util.HashMap" language="java"
2  contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  @<head>
6  <meta charset="UTF-8">
7  <title>Student Portal</title>
8  <link href="portal.css" rel="stylesheet" type="text/css"/>
9  </head>
10 @<body>
11 @  <div class="topnav">
12 @    String username = request.getParameter("username");
13 @    String studentID = request.getParameter("studentID");
14 @    if(username == null) username = Users.search(studentID).get("username");
15 @    if(studentID == null) studentID = Users.getID(username);
16 @    HashMap<String, String> student = Members.search(studentID);%
17 @    <a class="active" href=<%="studentPortal.jsp?studentID="+studentID%>>Student Portal</a>
18 @    <a href=<%="add.jsp?studentID="+studentID%>>Add Courses</a>
19 @    <a href=<%="drop.jsp?studentID="+studentID%>>Drop Courses</a>
20 @    <a href=<%="viewCourses.jsp?studentID="+studentID%>>View Courses</a>
21 @    <a href=<%="transcript.jsp?studentID="+studentID%>>Transcript</a>
22 @    <a href=<%="transactions.jsp?studentID="+studentID%>>Transactions</a>
23 @    <a href="index.jsp">Logout</a>
24 @  </div>
25 @  <h1 style="text-align:center;">Student Portal</h1>
26 @  <table class="content-table">
27 @    <tr>
28 @      <td><h2>Welcome Student <%out.println(student.get("firstname") + " " + student.get("lastname"));out.p
29 @      </td>
30 @    </tr>
31 @  </table>
32 @</body>
```

## Instructor Portal:

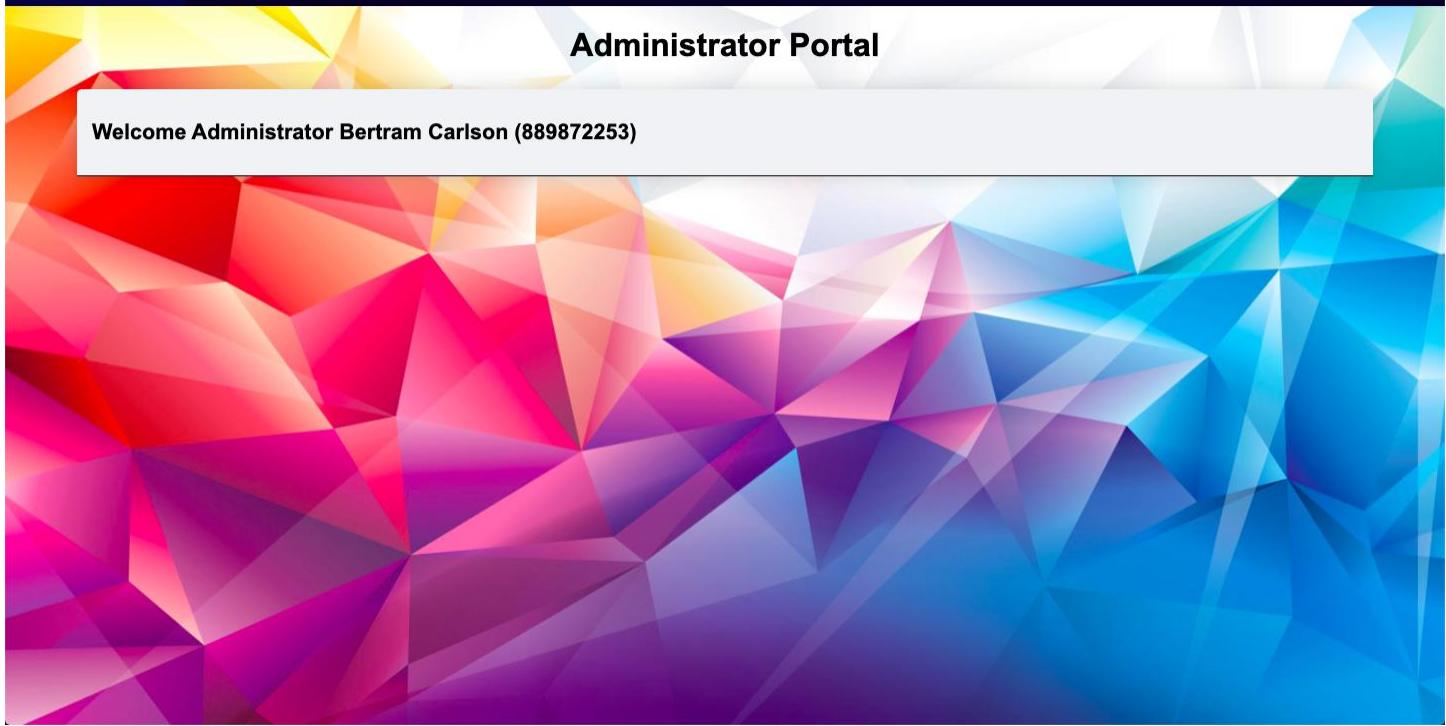
The screenshot shows the Instructor Portal interface. At the top, there is a dark blue header bar with white text containing links: "Instructor Portal", "Manage Courses", "Teach Course", and "Logout". Below the header is a large, vibrant background image composed of a low-poly geometric pattern in shades of red, orange, yellow, purple, blue, and green. In the upper left area of the main content area, there is a white rectangular box containing the text "Welcome Instructor Erin Weber (226084272)" in a black sans-serif font.

## Code for InstructorPortal.jsp

```
1 <%@ page
2     import="users.Users,instructors.Instructors,members.Members,java.util.HashMap"
3     language="java" contentType="text/html; charset=UTF-8"
4     pageEncoding="UTF-8"%>
5 <!DOCTYPE html>
6<html>
7<head>
8 <meta charset="UTF-8">
9 <title>Instructor Portal</title>
10 <link href="portal.css" rel="stylesheet" type="text/css" />
11 </head>
12<body>
13<%
14     String username = request.getParameter("username");
15     String instructorID = request.getParameter("instructorID");
16     if (username == null) username = Users.search(instructorID).get("username");
17     if (instructorID == null) instructorID = Users.getID(username);
18     HashMap<String, String> instructor = Members.search(instructorID);
19     String status = Instructors.getStatus(instructorID);
20 %>
21<div class="topnav">
22<a class="active"
23     href=<%="InstructorPortal.jsp?instructorID=" + instructorID%>>Instructor
24     Portal</a>
25     <% if (status.equalsIgnoreCase("Active")) { %>
26     <a href=<%="roster.jsp?instructorID=" + instructorID%>>Manage Courses</a>
27     <a href=<%="teachCourse.jsp?instructorID=" + instructorID%>>Teach Course</a>
28     <%
29     }
30     %>
31     <a href="index.jsp">Logout</a>
32 </div>
33 <h1 style="text-align: center;">Instructor Portal</h1>
34<table class="content-table">
35<tr>
36    <td>
37        <h2>Welcome Instructor <%out.println(instructor
38 .get("firstname") + " " + instructor
39 .get("lastname"));out.println("(" + instructorID + ")");%>
40        </h2>
41    </td>
42</tr>
43</table>
44</body>
45</html>
```

## Administrator Portal:

Administrator Portal   Manage Courses   Manage Configurations   Manage Members   Manage Students   Manage Instructors   Manage Administrators   Logout



## Code for administratorPortal.jsp

```
1  <%@ page import="users.Users, administrators.Administrators, members.Members, java.util.HashMap" language="java"
2  pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Administrator Portal</title>
8  <link href="portal.css" rel="stylesheet" type="text/css"/>
9  </head>
10 <body>
11     <% String username = request.getParameter("username");
12     String adminID = request.getParameter("adminID");
13     if(username == null) username = Users.search(adminID).get("username");
14     if(adminID == null) adminID = Users.getID(username);
15     HashMap<String, String> admin = Members.search(adminID);
16     int clearance = Administrators.getClearance(adminID);%>
17     <div class="topnav">
18         <a class="active" href="<%= "administratorPortal.jsp?adminID=" + adminID %>">Administrator Portal</a>
19         <% if(clearance == 1 || clearance == 3) { // Clearance 1 = manage courses and configs %>
20             <a href="<%= "manageCourses.jsp?adminID=" + adminID %>">Manage Courses</a>
21             <a href="<%= "manageConfigurations.jsp?adminID=" + adminID %>">Manage Configurations</a>
22             <% } %>
23             <% if(clearance == 2 || clearance == 3) { // Clearance 2 = manage accounts %>
24                 <a href="<%= "manageMembers.jsp?adminID=" + adminID %>">Manage Members</a>
25                 <a href="<%= "manageStudents.jsp?adminID=" + adminID %>">Manage Students</a>
26                 <a href="<%= "manageInstructors.jsp?adminID=" + adminID %>">Manage Instructors</a>
27                 <a href="<%= "manageAdministrators.jsp?adminID=" + adminID %>">Manage Administrators</a>
28                 <% } %>
29                 <a href="index.jsp">Logout</a>
30     </div>
31     <h1 style="text-align:center;">Administrator Portal</h1>
32     <table class="content-table">
33         <tr>
34             <td><h2>Welcome Administrator <%out.println(admin.get("firstname") + " " + admin.get("lastname"));out
35             </tr>
36     </table>
37
38 </body>
39 </html>
40
```

# Administrators

After logging into the portal, an administrator can do certain tasks depending on the clearance.

## *Manage courses – clearance 1 or 3*

Click ‘Manage Courses’.

### *View existing courses*

A list of existing course is displayed

Administrator Portal	Manage Courses	Manage Configurations	Manage Members	Manage Students	Manage Instructors	Manage Administrators	Logout
<b>Manage Courses</b>							
Department	Number	Title	Units	Cost			
BIO	1A	Biology I	5	230	<a href="#">Update</a>	<a href="#">Delete</a>	
BIO	1B	Biology II	5	230	<a href="#">Update</a>	<a href="#">Delete</a>	
CHEM	1A	Chemistry I	5	230	<a href="#">Update</a>	<a href="#">Delete</a>	
CHEM	1B	Chemistry II	5	230	<a href="#">Update</a>	<a href="#">Delete</a>	
COMM	2	Public Communication	3	138	<a href="#">Update</a>	<a href="#">Delete</a>	
CS	146	Data Structures	4	184	<a href="#">Update</a>	<a href="#">Delete</a>	
CS	147	Computer Architecture	4	184	<a href="#">Update</a>	<a href="#">Delete</a>	
CS	149	Operating Systems	4	184	<a href="#">Update</a>	<a href="#">Delete</a>	
CS	157A	Database	4	184	<a href="#">Update</a>	<a href="#">Delete</a>	
CS	166	Information Security	4	184	<a href="#">Update</a>	<a href="#">Delete</a>	
ENG	1A	Language & Composition	3	138	<a href="#">Update</a>	<a href="#">Delete</a>	
ENG	1B	Literature	3	138	<a href="#">Update</a>	<a href="#">Delete</a>	
GEO	2A	Introduction to Geology I	3	138	<a href="#">Update</a>	<a href="#">Delete</a>	

In this portal, we are executing this method and printing it on the webpage. This will execute the SQL statement and will return an array list of HashMap containing the attributes for each registered course which we print to the user.

```
    */
    public static ArrayList<HashMap<String, String>> viewRegisteredCourses(String studentID) {
        ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
        /** Check for invalid inputs. If any input is null, return false */
        if (studentID == null) return output; // Check if studentID is null, return empty list if so
        SQLMethods.mysqlConnect(); // Connect to DB
        try { // Attempt to search
            /** Search and retrieve tuple */
            pstate = SQLMethods.con.prepareStatement(
                "SELECT * FROM Registers JOIN Configurations Using (configID) WHERE studentID = ? ORDER BY year DESC;");
            pstate.setString(1, studentID);
            result = pstate.executeQuery(); // Execute query
            /** Extract tuple data */
            while (result.next()) {
                HashMap<String, String> tuple = new HashMap<String, String>();
                tuple.put("term", result.getString("term"));
                tuple.put("year", result.getString("year"));
                tuple.put("department", result.getString("department"));
                tuple.put("number", result.getString("number"));
                tuple.put("days", result.getString("days"));
                tuple.put("time", result.getString("time"));
                tuple.put("room", result.getString("room"));
                tuple.put("configID", Integer.toString(result.getInt("configID")));
                output.add(tuple);
            }
            result.close(); // Close result
            SQLMethods.closeConnection(); // Close connection
            return output; // Return output
        } catch (SQLException e) { // Print error and terminate program
            SQLMethods.mysql_fatal_error("Query error: " + e.toString());
        }
        return output; // Return false as a default value
    }
```

## Create course

At the bottom of the page, you may enter new fields to create a course. If the course already exists, it will not be duplicated.

The screenshot shows a web form titled "Create Course". It contains five input fields: "Department", "Number", "Title", "Units", and "Cost", each with a text input box. Below the input fields is a "Submit" button.

Servlet to process form:

```
/*
 * protected void doPost(HttpServletRequest request, HttpServletResponse response)
 *      throws ServletException, IOException {
    String adminID = request.getParameter("adminID");
    String department = request.getParameter("department");
    String number = request.getParameter("number");
    String title = request.getParameter("title");
    int units = Integer.parseInt(request.getParameter("units"));
    int cost = Integer.parseInt(request.getParameter("cost"));
    // Add only if course does not already exist
    if (Courses.search(department, number).size() == 0) Courses.insert(department, number, title, units, cost);
    RequestDispatcher req = request.getRequestDispatcher("manageCourses.jsp?adminID=" + adminID);
    req.forward(request, response);
}
```

Method containing the SQL statement which is executed:

```
/*
 * public static boolean insert(String department, String number, String title, int units, int cost) {
    /** Check for invalid inputs. If any input is null, return false */
    if (department == null || number == null || title == null || units < 0 || cost < 0) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to insert
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Courses Values (?, ?, ?, ?, ?, ?);");
        pstate.setString(1, department);
        pstate.setString(2, number);
        pstate.setString(3, title);
        pstate.setInt(4, units);
        pstate.setInt(5, cost);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

## Delete course

Click the delete button and the course will disappear from the list and database.

Servlet to process delete:

```
/*
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    /** Get inputs */
    String adminID = request.getParameter("adminID");
    String department = request.getParameter("department");
    String number = request.getParameter("number");
    /** Delete Course */
    Courses.delete(department, number); // Remove course
    /** Return to manageCourses.jsp */
    RequestDispatcher req = request.getRequestDispatcher("manageCourses.jsp?adminID=" + adminID);
    req.forward(request, response); // Return to manageCourses.jsp
}
```

Delete method:

```
public static boolean delete(String department, String number) {
    /** Check for invalid inputs. If any input is null, return false */
    if (department == null || number == null) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete using PreparedStatement
        pstate = SQLMethods.con.prepareStatement("DELETE FROM Courses WHERE department = ? AND number = ?;");
        pstate.setString(1, department);
        pstate.setString(2, number);
        int rowcount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowcount == 1); // If rowcount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

## Update course

Click the ‘Update’ button next to a course. You will be redirected to a page with the course’s info filled in. Modify the course. The course will update if the department and number do not conflict with another course.

The screenshot shows a web-based administrative interface. At the top, there is a navigation bar with links: Administrator Portal, Manage Courses, Manage Configurations, Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. Below the navigation bar is a title 'New Values'. The main content area contains a form with five input fields: 'Department' (value: BIO), 'Number' (value: 1A), 'Title' (value: Biology I), 'Units' (value: 5), and 'Cost' (value: 230). A 'Submit' button is located at the bottom right of the form. The background of the page features a colorful, abstract geometric pattern.

Serverlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    /* Get inputs */
    String adminID = request.getParameter("adminID");
    String oldDept = request.getParameter("oldDepartment");
    String oldNum = request.getParameter("oldNumber");
    String newDept = request.getParameter("department");
    String newNum = request.getParameter("number");
    String title = request.getParameter("title");
    int units = Integer.parseInt(request.getParameter("units"));
    int cost = Integer.parseInt(request.getParameter("cost"));
    // If new attributes conflict with an existing course, return
    if (Courses.search(newDept, newNum).size() == 0) {
        // Update references to the course
        Registers.updateCourse(oldDept, oldNum, newDept, newNum);
        Teaches.updateCourse(oldDept, oldNum, newDept, newNum);
        // Update course
        Courses.update(oldDept, oldNum, newDept, newNum, title, units, cost);
    }
    RequestDispatcher req = request.getRequestDispatcher("manageCourses.jsp?adminID=" + adminID);
    req.forward(request, response);
}
```

Update method:

```

public static boolean update(String oldDepartment, String oldNumber, String newDepartment, String newNumber,
    String newTitle, int newUnits, int newCost) {
    /** Check for invalid inputs. If any input is null, return false */
    if (oldDepartment == null || oldNumber == null || newDepartment == null || newNumber == null || newTitle == null
        || newUnits < 0 || newCost < 0)
        return false; // PK cannot be null
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        /*
         * HashMap<String, String> course = search(oldDepartment, oldNumber); // Search for course to update
         * if (newDepartment == null) newDepartment = oldDepartment; // Determine if we
         * want to change old value if (newNumber == null) newNumber = oldNumber; //
         * Determine if we want to change old value if (newTitle == null) newTitle =
         * course.get("title"); // Determine if we want to change old value if (newUnits
         * == -1) newUnits = Integer.parseInt(course.get("units")); if (newCost == -1)
         * newCost = Integer.parseInt(course.get("cost"));
         */
        pstate = SQLMethods.con.prepareStatement(
            "UPDATE Courses SET department = ?, number = ?, title = ?, units = ?, cost = ? WHERE department = ? AND number = ?;");
        pstate.setString(1, oldDepartment);
        pstate.setString(2, oldNumber);
        pstate.setInt(3, newTitle);
        pstate.setInt(4, Courses(department, number, title, units, cost));
        pstate.setInt(5, newUnits);
        pstate.setDouble(6, newCost);
        pstate.setString(7, oldNumber);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}

```

## Manage course configurations – clearance 1 or 3

### View existing configurations

Click ‘Manage Configurations’ You will see a page with all the existing configurations.

The screenshot shows a web application interface titled "Manage Configurations". At the top, there is a navigation bar with links: Administrator Portal, Manage Courses, Manage Configurations (which is the active link), Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. The main content area has a yellow and orange abstract background. A table titled "Manage Configurations" lists 13 rows of course configuration data. Each row includes columns for ConfigID, Term, Year, Days, Time, Room, Seats, and two buttons: "Update" and "Delete".

ConfigID	Term	Year	Days	Time	Room	Seats	Update	Delete
1	Spring	2021	MW	4:00 - 6:00	41	31	<button>Update</button>	<button>Delete</button>
2	Summer	2021	TR	6:30-8:00	33	60	<button>Update</button>	<button>Delete</button>
3	Summer	2021	MWTR	9:00-10:30	34	56	<button>Update</button>	<button>Delete</button>
4	Summer	2021	MW	8:00-9:30	22	52	<button>Update</button>	<button>Delete</button>
5	Summer	2021	MWT	6:30-8:00	40	54	<button>Update</button>	<button>Delete</button>
6	Fall	2020	MW	1:30-3:00	50	53	<button>Update</button>	<button>Delete</button>
7	Fall	2020	TR	10:00-11:30	25	51	<button>Update</button>	<button>Delete</button>
8	Fall	2020	MWTR	8:00-9:30	18	35	<button>Update</button>	<button>Delete</button>
9	Fall	2020	MWF	9:30-11:00	18	33	<button>Update</button>	<button>Delete</button>
10	Winter	2020	MW	3:00-4:30	50	54	<button>Update</button>	<button>Delete</button>
11	Winter	2020	TR	6:30-8:00	24	46	<button>Update</button>	<button>Delete</button>
12	Winter	2020	MWTR	9:00-10:30	23	53	<button>Update</button>	<button>Delete</button>
13	Winter	2020	MW	3:00-4:30	48	48	<button>Update</button>	<button>Delete</button>

Java method:

```
public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement("SELECT * FROM Configurations ORDER BY year DESC;");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("configID", Integer.toString(result.getInt("configID")));
            tuple.put("term", result.getString("term"));
            tuple.put("year", Integer.toString(result.getInt("year")));
            tuple.put("days", result.getString("days"));
            tuple.put("time", result.getString("time"));
            tuple.put("room", result.getString("room"));
            tuple.put("seats", Integer.toString(result.getInt("seats")));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}
```

JSP file:

```

3   <!DOCTYPE html>
4   <html>
5     <head>
6       <meta charset="UTF-8">
7       <title>Manage Configurations</title>
8       <link href="portal.css" rel="stylesheet" type="text/css"/>
9     </head>
10    <body>
11      <% String adminID = request.getParameter("adminID");
12      int clearance = Administrators.getClearance(adminID);
13      %>
14      <div class="topnav">
15        <a class="active" href=<%="administratorPortal.jsp?adminID=" + adminID%>>Administrator Portal</a>
16        <% if(clearance == 1 || clearance == 3) { // Clearance 1 = manage courses and configs %>
17        <a href=<%="manageCourses.jsp?adminID=" + adminID%>>Manage Courses</a>
18        <a href=<%="manageConfigurations.jsp?adminID=" + adminID%>>Manage Configurations</a>
19        <% } %>
20        <% if(clearance == 2 || clearance == 3) { // Clearance 2 = manage accounts %>
21        <a href=<%="manageMembers.jsp?adminID=" + adminID%>>Manage Members</a>
22        <a href=<%="manageStudents.jsp?adminID=" + adminID%>>Manage Students</a>
23        <a href=<%="manageInstructors.jsp?adminID=" + adminID%>>Manage Instructors</a>
24        <a href=<%="manageAdministrators.jsp?adminID=" + adminID%>>Manage Administrators</a>
25        <% } %>
26        <a href="index.jsp">Logout</a>
27      </div>
28      <h1 style="text-align:center;">Manage Configurations</h1>
29      <table class="content-table">
30        <tr>
31          <td>ConfigID</td>
32          <td>Term</td>
33          <td>Year</td>
34          <td>Days</td>
35          <td>Time</td>
36          <td>Room</td>
37          <td>Seats</td>
38          <td></td>
39          <td></td>
40          <td></td>
41        </tr>
42        <%
43        ArrayList<HashMap<String, String>> configurations = Configurations.getAll();
44        for (int i = 0; i < configurations.size(); i++) {
45          HashMap<String, String> configuration = configurations.get(i);
46        %>
47        <tr>
48          <td><%=configuration.get("configID")%></td>
49          <td><%=configuration.get("term")%></td>
50          <td><%=configuration.get("year")%></td>
51          <td><%=configuration.get("days")%></td>
52          <td><%=configuration.get("time")%></td>
53          <td><%=configuration.get("room")%></td>
54          <td><%=configuration.get("seats")%></td>
55          <td>
56            <form action=<%="updateConfigurations.jsp?adminID=" + adminID%> method="post">
57              <input type="hidden" id="configID" name="configID" value=<%=configuration.get("configID")%>>
58              <input type="submit" value="Update">
59            </form>
60          </td>
61          <td>
62            <form action="deleteConfiguration" method="post">
63              <input type="hidden" id="adminID" name="adminID" value=<%=adminID%>>
64              <input type="hidden" id="configID" name="configID" value=<%=configuration.get("configID")%>>
65              <input type="submit" value="Delete">
66            </form>
67          </td>
68        </tr>
69        <%
70        %>
71      </table>
72      <%
73      <h1 style="text-align:center;">Create Configuration</h1>
74      <form action="insertConfiguration" method="post">

```

## Create configurations

At the bottom of the page you will see a form to add a configuration. Type in the desired information and click submit to create and add the configuration.

### Create Configuration

Term	<input type="text"/>
Year	<input type="text"/>
Days	<input type="text"/>
Time	<input type="text"/>
Room	<input type="text"/>
Seats	<input type="text"/>

Serverlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
// String configID = request.getParameter("configID"); // configID is
// autogenerated
String adminID = request.getParameter("adminID");
String term = request.getParameter("term");
int year = Integer.parseInt(request.getParameter("year"));
String days = request.getParameter("days");
String time = request.getParameter("time");
String room = request.getParameter("room");
int seats = Integer.parseInt(request.getParameter("seats"));
Configurations.insert(term, year, days, time, room, seats);
RequestDispatcher req = request.getRequestDispatcher("manageConfigurations.jsp?adminID=" + adminID);
req.forward(request, response);
}
```

Insert method:

```
public static boolean insert(String term, int year, String days, String time, String room, int seats) {
    /** Check for invalid inputs. If any input is null, return false */
    SQLMethods.mysqlConnect();
    try {
        // configID is auto incremented in SQL table
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Configurations VALUES (NULL, ?, ?, ?, ?, ?, ?, ?);");
        pstate.setString(1, term);
        pstate.setInt(2, year);
        pstate.setString(3, days);
        pstate.setString(4, time);
        pstate.setString(5, room);
        pstate.setInt(6, seats);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

## Delete configurations

Click the delete button next to the configuration to delete it.

Servlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    /** Get inputs */
    String adminID = request.getParameter("adminID");
    String configID = request.getParameter("configID");
    /** Delete configuration */
    Configurations.delete(configID); // Remove configuration
    /** Return to manageConfigurations.jsp */
    RequestDispatcher req = request.getRequestDispatcher("manageConfigurations.jsp?adminID=" + adminID);
    req.forward(request, response); // Return to manageConfigurations.jsp
}
```

Delete method:

```
public static boolean delete(String configID) {
    /** Check for invalid inputs. If any input is null, return false */
    if (configID == null) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete using PreparedStatement
        pstate = SQLMethods.con.prepareStatement("DELETE FROM Configurations WHERE configID = ?;");
        pstate.setString(1, configID);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, 1 row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Return false as a default value
}
```

## Update configurations

Click the ‘Update’ button next to a configuration. You will be redirected to a page with the configuration’s info filled in. Modify the configuration and click submit.

New Values

Term	Spring
Year	2018
Days	MWTR
Time	9:00-10:30
Room	36
Seats	42

Submit

Serverlet:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // int oldConfigID = Integer.parseInt(request.getParameter("oldConfigID"));
    // int newConfigID = Integer.parseInt(request.getParameter("newConfigID"));
    // It's better to not change configID - autogenerated
    String adminID = request.getParameter("adminID");
    int configID = Integer.parseInt(request.getParameter("configID"));
    String term = request.getParameter("term");
    int year = Integer.parseInt(request.getParameter("year"));
    String days = request.getParameter("days");
    String time = request.getParameter("time");
    String room = request.getParameter("room");
    int seats = Integer.parseInt(request.getParameter("seats"));

    // Registers.updateConfigID(oldConfigID, newConfigID);
    // Teaches.updateConfigID(oldConfigID, newConfigID);
    // Configurations.update(oldConfigID, newConfigID, term, year, days, time, room,
    // seats);
    Configurations.update(configID, term, year, days, time, room, seats);

    RequestDispatcher req = request.getRequestDispatcher("manageConfigurations.jsp?adminID=" + adminID);
    req.forward(request, response);
}
```

Update method:

```

public static boolean update(int configID, String term, int year, String days, String time, String room,
    int seats) {
    /** Check for invalid inputs. If any input is null, return false */
    if (configID < 0 || term == null || year < 1900 || days == null || time == null || room == null || seats < 0)
        return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        // HashMap<String, String> course = search(oldConfigID); // Search for config to
        // update
        SQLMethods.mysqlConnect(); // Connect to DB
        pstate = SQLMethods.con.prepareStatement(
            "UPDATE Configurations SET term = ?, year = ?, days = ?, time = ?, room = ?, seats = ? WHERE configID = ?;");
        pstate.setString(1, term);
        pstate.setInt(2, year);
        pstate.setString(3, days);
        pstate.setString(4, time);
        pstate.setString(5, room);
        SQLMethods.executeUpdate(pstate);
        return true;
    } catch (SQLException e) {
        SQLMethods.printError("Error updating configuration: " + e.getMessage());
    }
    return false;
}

```

**void java.sql.PreparedStatement.setString(int parameterIndex, String x) throws SQLException**

Sets the designated parameter to the given Java `String` value. The driver converts this to an SQL `VARCHAR` or `LONGVARCHAR` value (depending on the argument's size relative to the driver's limits on `VARCHAR` values) when it sends it to the database.

**Parameters:**

- `parameterIndex` the first parameter is 1, the second is 2, ...
- `x` the parameter value

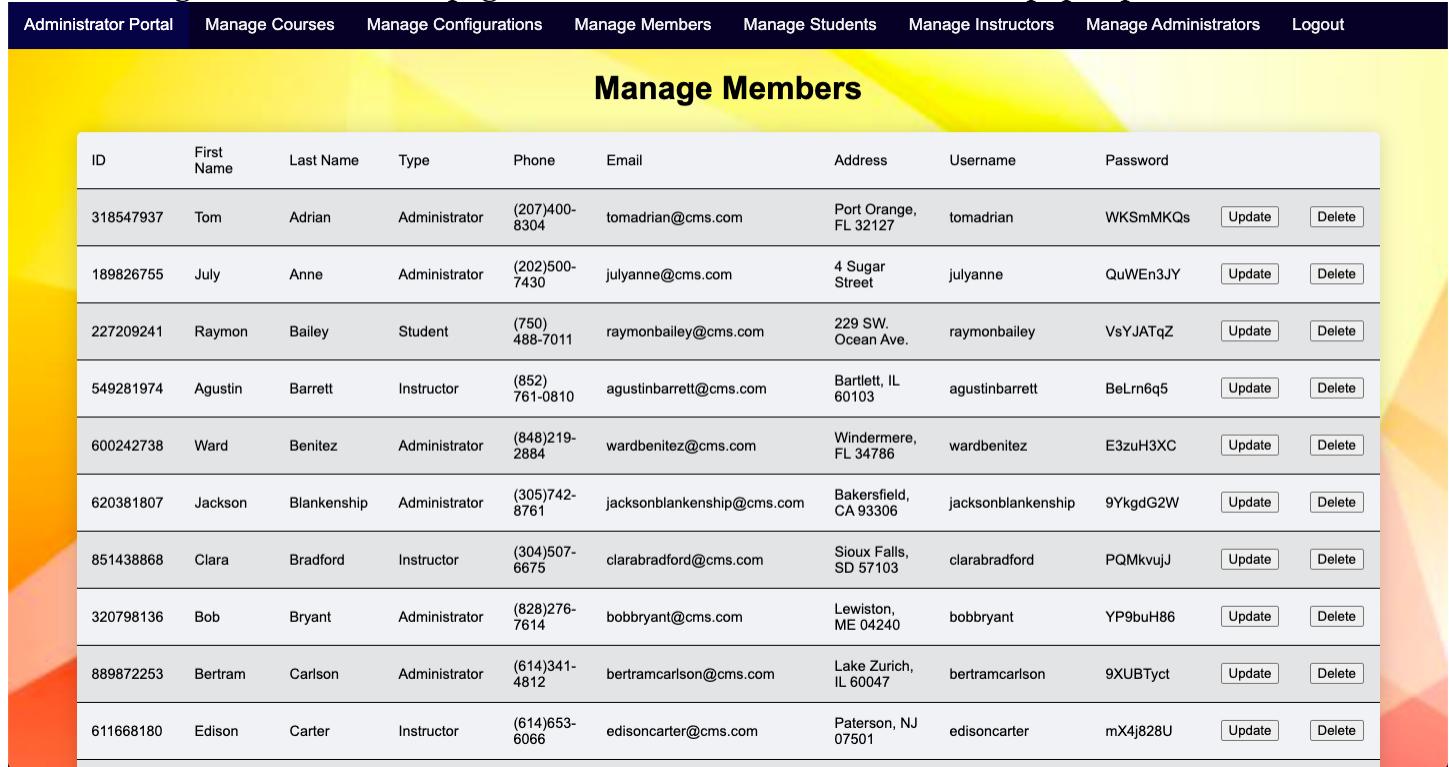
**Throws:**

- `SQLException` - if `parameterIndex` does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed `PreparedStatement`.

## Manage members – clearance 2 or 3

### View existing members

Click ‘Manage Members’ and a page with all the members and their info pops up.



The screenshot shows a web-based application interface titled 'Manage Members'. At the top, there is a navigation bar with links: 'Administrator Portal', 'Manage Courses', 'Manage Configurations', 'Manage Members' (which is the active link), 'Manage Students', 'Manage Instructors', 'Manage Administrators', and 'Logout'. The main content area is titled 'Manage Members' and contains a table with 10 rows of member data. Each row includes columns for ID, First Name, Last Name, Type, Phone, Email, Address, Username, and Password, along with 'Update' and 'Delete' buttons. The data in the table is as follows:

ID	First Name	Last Name	Type	Phone	Email	Address	Username	Password
318547937	Tom	Adrian	Administrator	(207)400-8304	tomadrian@cms.com	Port Orange, FL 32127	tomadrian	WKSrnMKQs
189826755	July	Anne	Administrator	(202)500-7430	julyanne@cms.com	4 Sugar Street	julyanne	QuWEEn3JY
227209241	Raymon	Bailey	Student	(750)488-7011	raymonbailey@cms.com	229 SW. Ocean Ave.	raymonbailey	VsYJATqZ
549281974	Agustin	Barrett	Instructor	(852)761-0810	agustinbarrett@cms.com	Bartlett, IL 60103	agustinbarrett	BeLrn6q5
600242738	Ward	Benitez	Administrator	(848)219-2884	wardbenitez@cms.com	Windermere, FL 34786	wardbenitez	E3zuH3XC
620381807	Jackson	Blankenship	Administrator	(305)742-8761	jacksonblankenship@cms.com	Bakersfield, CA 93306	jacksonblankenship	9YkgdG2W
851438868	Clara	Bradford	Instructor	(304)507-6675	clarabradford@cms.com	Sioux Falls, SD 57103	clarabradford	PQMkvujJ
320798136	Bob	Bryant	Administrator	(828)276-7614	bobbryant@cms.com	Lewiston, ME 04240	bobbryant	YP9buH86
889872253	Bertram	Carlson	Administrator	(614)341-4812	bertramcarlson@cms.com	Lake Zurich, IL 60047	bertramcarlson	9XUBTyct
611668180	Edison	Carter	Instructor	(614)653-6066	edisoncarter@cms.com	Paterson, NJ 07501	edisoncarter	mX4j828U

Method to retrieve all tuples:

```

public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement("SELECT * FROM Members NATURAL JOIN Users ORDER BY lastname ASC");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("ID", result.getString("ID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            tuple.put("phone", result.getString("phone"));
            tuple.put("email", result.getString("email"));
            tuple.put("address", result.getString("address"));
            tuple.put("username", result.getString("username"));
            tuple.put("password", result.getString("password"));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}

```

## Delete members

Click the delete button to delete a member.

Code for deleteMember:

```
<td>
    <form action="deleteMember" method="post">
        <input type="hidden" id="adminID" name="adminID" value=<%=adminID%>>
        <input type="hidden" id="ID" name="ID" value=<%=member.get("ID")%>>
        <input type="submit" value="Delete">
    </form>
</td>
```

Serverlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    /* Get inputs */
    String adminID = request.getParameter("adminID");
    String ID = request.getParameter("ID");
    String number = request.getParameter("number");
    /* Delete Member and User */
    Members.delete(ID);
    Users.delete(ID);
    /* Delete all dependencies to this ID */
    Administrators.delete(ID);
    Students.delete(ID);
    Registers.dropAll(ID);
    Transactions.deleteAll(ID);
    Instructors.delete(ID);
    Teaches.dropAll(ID);

    /* Return back */
    RequestDispatcher req = request.getRequestDispatcher("manageMembers.jsp?adminID=" + adminID);
    req.forward(request, response);
}
```

Delete method:

```
public static boolean delete(String ID) {
    if (ID == null) return false; // Check if ID is null
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete
        pstate = SQLMethods.con.prepareStatement("DELETE FROM Members WHERE ID = ?;");
        pstate.setString(1, ID);
        int rowcount = pstate.executeUpdate();
        SQLMethods.closeConnection(); // Close connection
        return (rowcount == 1); // If rowcount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; / S void SQL.SQLMethods.mysql_fatal_error(String
error)
```

## Update members

Click the update button. Enter the new fields in the next page and click submit. In this, you can change member attributes as well as user attributes. Use this to create a username and password for a new member.

New Values

ID	318547937
First Name	Tom
Last Name	Adrian
Phone	(207)400-8304
Email	tomadrian@cms.com
Address	Port Orange, FL 32127
Username	tomadrian
Password	WKSmMKQs

Submit

Serverlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String adminID = request.getParameter("adminID");
    String oldID = request.getParameter("oldID");
    String newID = request.getParameter("newID");
    String firstname = request.getParameter("firstname");
    String lastname = request.getParameter("lastname");
    String phone = request.getParameter("phone");
    String email = request.getParameter("email");
    String address = request.getParameter("address");
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    // If new ID conflicts with an existing member, return
    if (newID.equals(oldID) || Members.search(newID).size() == 0) {
        // Update references in all tables
        Students.updateID(oldID, newID);
        Instructors.updateID(oldID, newID);
        Administrators.updateID(oldID, newID);
        Registers.updateID(oldID, newID);
        Transactions.updateID(oldID, newID);
        Teaches.updateID(oldID, newID);
        // Update Members
        Members.update(oldID, newID, firstname, lastname, phone, email, address);
        // Update User
        Users.update(oldID, newID, username, password);
    }
    RequestDispatcher req = request.getRequestDispatcher("manageMembers.jsp?adminID=" + adminID);
    req.forward(request, response);
}
```

## Update method:

```
public static boolean update(String oldID, String newID, String firstname, String lastname, String phone,
    String email, String address) {
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        /** Update tuple in Members */
        pstate = SQLMethods.con.prepareStatement(
            "UPDATE Members SET ID = ?, firstname = ?, lastname = ?, phone = ?, email = ?, address = ? WHERE ID = ?;");
        pstate.setString(1, newID);
        pstate.setString(2, firstname);
        pstate.setString(3, lastname);
        pstate.setString(4, phone);
        pstate.setString(5, email);
        pstate.setString(6, address);
        pstate.setString(7, oldID);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

# Manage students – clearance 2 or 3

## View existing students

Click the ‘Manage Students’ button.

The screenshot shows a web-based application interface titled "Manage Students". At the top, there is a navigation bar with links: "Administrator Portal", "Manage Courses", "Manage Configurations", "Manage Members", "Manage Students" (which is the active link), "Manage Instructors", "Manage Administrators", and "Logout". Below the navigation bar is a decorative background with abstract geometric shapes in yellow, red, blue, and purple. The main content area is titled "Manage Students" and contains a table with 12 rows of student data. The columns are labeled "ID", "First Name", "Last Name", "Balance", "Unit Cap", and "Options". Each row includes a "Options" button. The data in the table is as follows:

ID	First Name	Last Name	Balance	Unit Cap	Options
888134841	323	3232	0	12	<button>Options</button>
227209241	Raymon	Bailey	696	18	<button>Options</button>
339820994	Reyna	Colon	69	20	<button>Options</button>
689582238	Dana	Conway	446	9	<button>Options</button>
506190495	Verna	Copeland	153	11	<button>Options</button>
483131196	Beatriz	Downs	623	20	<button>Options</button>
573062892	Thanh	Everett	451	11	<button>Options</button>
314843835	Myrtle	Haney	118	16	<button>Options</button>
978594399	Gina	Hernandez	626	21	<button>Options</button>
951597733	Lupe	Juarez	475	18	<button>Options</button>
452018548	Bud	Kirby	776	18	<button>Options</button>
796058602	Miles	Larson	320	15	<button>Options</button>
503432266	Carol	Lynn	437	18	<button>Options</button>

Method to get all tuples:

```
public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con
            .prepareStatement("SELECT * FROM Students JOIN Members ON studentID = ID ORDER BY lastname ASC");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("studentID", result.getString("studentID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            tuple.put("balance", Integer.toString(result.getInt("balance")));
            tuple.put("unit_cap", Integer.toString(result.getInt("unit_cap")));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}
```

## Update student

Click options. You will have the ability to update a student's balance and unit cap. You can also change the ID and this will affect all tables.

The screenshot shows a web-based course management system with a modern, colorful geometric background. At the top, a navigation bar includes links for Administrator Portal, Manage Courses, Manage Configurations, Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. Below the navigation, there are three main sections: 1) A form titled "New Values" for updating student information, with fields for Name (323 3232), Student ID (888134841), Balance (0), and Unit Cap (12), followed by a "Submit" button. 2) A section titled "Registered Courses" with columns for Term, Year, Department, Number, and ConfigID. 3) A section titled "Transactions" with columns for Credit Card, Amount, and Timestamp.

Update method:

```
public static boolean update(String oldStudentID, String newStudentID, int balance, int unit_cap) {  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try {  
        pstate = SQLMethods.con.prepareStatement(  
            "UPDATE Students SET studentID = ?, balance = ?, unit_cap = ? WHERE studentID = ?");  
        pstate.setString(1, newStudentID);  
        pstate.setInt(2, balance);  
        pstate.setInt(3, unit_cap);  
        pstate.setString(4, oldStudentID);  
        int rowCount = pstate.executeUpdate(); // Number of rows affected  
        SQLMethods.closeConnection(); // Close connection  
        return (rowCount == 1); // If rowCount == 1, row successfully updated  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Default value: false  
}
```

## *View student's registered courses*

Refer to picture above.

Method:

```
public static ArrayList<HashMap<String, String>> viewRegisteredCourses(String studentID) {  
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();  
    /** Check for invalid inputs. If any input is null, return false */  
    if (studentID == null) return output; // Check if studentID is null, return empty list if so  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to search  
        /** Search and retrieve tuple */  
        pstate = SQLMethods.con.prepareStatement(  
            "SELECT * FROM Registers JOIN Configurations Using (configID) WHERE studentID = ? ORDER BY year DESC;");  
        pstate.setString(1, studentID);  
        result = pstate.executeQuery(); // Execute query  
        /** Extract tuple data */  
        while (result.next()) {  
            HashMap<String, String> tuple = new HashMap<String, String>();  
            tuple.put("term", result.getString("term"));  
            tuple.put("year", result.getString("year"));  
            tuple.put("department", result.getString("department"));  
            tuple.put("number", result.getString("number"));  
            tuple.put("days", result.getString("days"));  
            tuple.put("time", result.getString("time"));  
            tuple.put("room", result.getString("room"));  
            tuple.put("configID", Integer.toString(result.getInt("configID")));  
            output.add(tuple);  
        }  
        result.close(); // Close result  
        SQLMethods.closeConnection(); // Close connection  
        return output; // Return output  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return output; // Return false as a default value  
}
```

## *View student's transactions*

Refer to picture above.

Method:

```
public static ArrayList<HashMap<String, String>> viewTransactions(String studentID) {  
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();  
    if (studentID == null) return output; // Return empty list  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to search  
        pstate = SQLMethods.con  
            .prepareStatement("SELECT * FROM Transactions WHERE studentID = ? ORDER BY timestamp DESC;");  
        pstate.setString(1, studentID);  
        result = pstate.executeQuery(); // Execute query  
        while (result.next()) {  
            HashMap<String, String> tuple = new HashMap<String, String>();  
            tuple.put("studentID", result.getString("studentID"));  
            tuple.put("creditcard", result.getString("creditcard"));  
            tuple.put("amount", result.getString("amount"));  
            tuple.put("timestamp", result.getString("timestamp"));  
            output.add(tuple);  
        }  
        result.close(); // Close result  
        SQLMethods.closeConnection(); // Close connection  
        return output; // Success  
    } catch (SQLException e) {  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return output; // default value  
}
```

## Manage instructors – clearance 2 or 3

### View existing instructors

Click the ‘Manage Instructors’ button.

The screenshot shows a web application interface titled "Manage Instructors". At the top, there is a navigation bar with links: "Administrator Portal", "Manage Courses", "Manage Configurations", "Manage Members", "Manage Students", "Manage Instructors" (which is the active link), "Manage Administrators", and "Logout". Below the navigation bar is a decorative header section with geometric shapes in yellow, red, and blue. The main content area is titled "Manage Instructors" and contains a table with 13 rows of data. The table has columns for "ID", "First Name", "Last Name", "Balance", and "Unit Cap". Each row includes an "Options" button in the last column. The data in the table is as follows:

ID	First Name	Last Name	Balance	Unit Cap
549281974	Agustin	Barrett	Active	<button>Options</button>
851438868	Clara	Bradford	Inactive	<button>Options</button>
611668180	Edison	Carter	Inactive	<button>Options</button>
337767185	Cherie	French	Inactive	<button>Options</button>
290327837	Nichole	Frost	Inactive	<button>Options</button>
957713740	Sylvia	Fuentes	Active	<button>Options</button>
307562770	Alton	Gallagher	Inactive	<button>Options</button>
996595763	Tamara	Gibson	Active	<button>Options</button>
661054866	Wes	Gill	Active	<button>Options</button>
100859622	Mia	Hodge	Inactive	<button>Options</button>
292211500	Antoinette	Marks	Active	<button>Options</button>
764777061	Miguel	Mathews	Inactive	<button>Options</button>
305981136	Polly	Rios	Inactive	<button>Options</button>

HTML:

```
<h1 style="text-align: center;">Manage Instructors</h1>
<table class="content-table">
  <tr>
    <td>ID</td>
    <td>First Name</td>
    <td>Last Name</td>
    <td>Balance</td>
    <td>Unit Cap</td>
  </tr>
<%
  ArrayList<HashMap<String, String>> instructors = Instructors.getAll();
  for (int i = 0; i < instructors.size(); i++) {
    HashMap<String, String> instructor = instructors.get(i);
%>
```

Java Method:

```

public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement(
            "SELECT * FROM Instructors JOIN Members ON instructorID = ID ORDER BY lastname ASC");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("instructorID", result.getString("instructorID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            tuple.put("status", result.getString("status"));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}

```

## Update instructors

Click 'Options'. You will have the ability to activate or deactivate an instructor's account. Enter 'Active' to activate. Enter anything else to deactivate, preferably 'Inactive'. You can also change the ID and this will affect all tables.

The screenshot shows a web application interface. At the top, there is a navigation bar with links: Administrator Portal, Manage Courses, Manage Configurations, Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. Below the navigation bar, the main content area has a title "New Values". Under this title, there is a form with three fields: Name (Agustin Barrett), Instructor ID (549281974), and Status (Active). A "Submit" button is located at the bottom right of the form. Below the form, there is a section titled "Courses Taught" which contains a table with the following data:

InstructorID	Department	Number	Term	Year	Days	Room	Seats
549281974	ENG	1A	Summer	2021	MW	22	52
549281974	CS	149	Spring	2020	MWF	43	55
549281974	GEO	2B	Fall	2018	MWTR	49	45

HTML:

```
<td>
<form action=<%="updateInstructors.jsp?adminID="+adminID%> method="post">
    <input type="hidden" id="adminID" name="adminID" value=<%=adminID%>>
    <input type="hidden" id="instructorID" name="instructorID" value=<%=instructor.get("instructorID")%>>
    <input type="hidden" id="firstname" name="firstname" value=<%=instructor.get("firstname")%>>
    <input type="hidden" id="lastname" name="lastname" value=<%=instructor.get("lastname")%>>
    <input type="hidden" id="status" name="status" value=<%=instructor.get("status")%>>
    <input type="submit" value="Options">
</form>
</td>
```

Serverlet method:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
String adminID = request.getParameter("adminID");
String oldInstructorID = request.getParameter("oldInstructorID");
String newInstructorID = request.getParameter("newInstructorID");
String status = request.getParameter("status");

// If new ID conflicts with an existing member, return
if (oldInstructorID.equals(newInstructorID) || Members.search(newInstructorID).size() == 0) {
    // Update references of ID in all tables (if we want to change ID)
    Members.updateID(oldInstructorID, newInstructorID); // Update reference in Members
    Users.updateID(oldInstructorID, newInstructorID); // Update reference in Users
    Teaches.updateID(oldInstructorID, newInstructorID); // Update reference in Teaches
    // Update instructor
    Instructors.update(oldInstructorID, newInstructorID, status);
}
RequestDispatcher req = request.getRequestDispatcher("manageInstructors.jsp?adminID=" + adminID);
req.forward(request, response);
}
```

Update method:

```
public static boolean update(String oldInstructorID, String newInstructorID, String status) {
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        /** Update instructor tuple */
        pstate = SQLMethods.con
            .prepareStatement("UPDATE Instructors SET instructorID = ?, status = ? WHERE instructorID = ?;");
        pstate.setString(1, newInstructorID);
        pstate.setString(2, status);
        pstate.setString(3, oldInstructorID);
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

## *View courses taught*

Refer to picture above.

Method:

```
public static ArrayList<HashMap<String, String>> viewTaughtCourses(String instructorID) {  
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();  
    /* Check for invalid inputs. If any input is null, return false */  
    if (instructorID == null) return output; // Check if studentID is null, return empty list if so  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to search  
        /** Search and retrieve tuple */  
        pstate = SQLMethods.con.prepareStatement(  
            "SELECT * FROM Teaches JOIN Configurations Using (configID) WHERE instructorID = ? ORDER BY year DESC;");  
        pstate.setString(1, instructorID);  
        result = pstate.executeQuery(); // Execute query  
        /** Extract tuple data */  
        while (result.next()) {  
            HashMap<String, String> tuple = new HashMap<String, String>();  
            tuple.put("instructorID", result.getString("instructorID"));  
            tuple.put("department", result.getString("department"));  
            tuple.put("number", result.getString("number"));  
            tuple.put("term", result.getString("term"));  
            tuple.put("year", result.getString("year"));  
            tuple.put("days", result.getString("days"));  
            tuple.put("time", result.getString("time"));  
            tuple.put("room", result.getString("room"));  
            tuple.put("seats", result.getString("seats"));  
            tuple.put("configID", Integer.toString(result.getInt("configID")));  
            output.add(tuple);  
        }  
        result.close(); // Close result  
        SQLMethods.closeConnection(); // Close connection  
        return output; // Return output  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return output; // Return false as a default value  
}
```

## Manage administrators – clearance 2 or 3

### View existing administrators

Click the ‘Manage Administrators’ button.

The screenshot shows a web-based application interface titled "Manage Administrators". At the top, there is a navigation bar with links: Administrator Portal, Manage Courses, Manage Configurations, Manage Members, Manage Students, Manage Instructors, Manage Administrators, and Logout. Below the navigation bar is a decorative header section with geometric patterns in yellow, red, and blue. The main content area is titled "Manage Administrators" and contains a table with 12 rows of data. The columns are labeled "ID", "First Name", "Last Name", "Clearance", and "Options". Each row represents an administrator with their ID, first name, last name, clearance level (1, 2, or 3), and a "Options" button. The data is as follows:

ID	First Name	Last Name	Clearance	Options
318547937	Tom	Adrian	2	Options
189826755	July	Anne	1	Options
600242738	Ward	Benitez	3	Options
620381807	Jackson	Blankenship	3	Options
320798136	Bob	Bryant	1	Options
889872253	Bertram	Carlson	3	Options
877078818	Hank	Colley	3	Options
237393883	Angelique	Davila	1	Options
546664173	Antoine	Deleon	1	Options
798100317	Honey	Dent	1	Options
409326430	Shelia	Elliott	3	Options
587779626	Deangelo	Franklin	3	Options
670334459	Brendon	Frazier	2	Options

Method:

```
public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement(
            "SELECT * FROM Administrators JOIN Members ON adminID = ID ORDER BY lastname ASC;");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("adminID", result.getString("adminID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            tuple.put("clearance", Integer.toString(result.getInt("clearance")));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}
```

## Update administrators

Click ‘Options’. You will have the ability to update an administrator’s clearance. You can also change the ID and this will affect all tables.

The screenshot shows a web-based administrative interface. At the top, there is a navigation bar with links: 'Administrator Portal', 'Manage Courses', 'Manage Configurations', 'Manage Members', 'Manage Students', 'Manage Instructors', 'Manage Administrators', and 'Logout'. Below the navigation bar is a section titled 'New Values' which contains three input fields. The first field is labeled 'Name' with the value 'Tom Adrian'. The second field is labeled 'Administrator ID' with the value '889872253'. The third field is labeled 'Clearance' with the value '2'. At the bottom right of this section is a 'Submit' button. The background of the page features a vibrant, abstract geometric pattern of triangles in shades of red, orange, yellow, green, blue, and purple.

Method:

```
public static boolean update(String oldAdminID, String newAdminID, int clearance) {
    Members.updateID(oldAdminID, newAdminID); // Update reference in Members
    Users.updateID(oldAdminID, newAdminID); // Update reference in Users
    SQLMethods.mysqlConnect(); // Connect to DB
    try { /* Update admin tuple */
        pstate = SQLMethods.con
            .prepareStatement("UPDATE Administrators SET adminID = ?, clearance = ? WHERE adminID = ?;");
        pstate.setString(1, newAdminID);
        pstate.setInt(2, clearance);
        pstate.setString(3, oldAdminID);
        pstate.executeUpdate();
        int rowCount = pstate.executeUpdate(); // Number of rows affected
        SQLMethods.closeConnection(); // Close connection
        return (rowCount == 1); // If rowCount == 1, row successfully inserted
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

# Instructors

## View courses taught

From the instructor portal, click 'Manage Courses'.

The screenshot shows a web-based instructor portal with a dark header bar containing links for 'Instructor Portal', 'Manage Courses', 'Teach Course', and 'Logout'. Below the header is a colorful geometric background. In the center, there is a table titled 'Taught Courses' with the following data:

Term	Year	Department	Number	Days	Time	Room	Seats	ConfigID	Drop	Manage
Spring	2021	Math	127	MW	4:00 - 6:00	41	31	1	<button>Drop</button>	<button>Manage</button>
Winter	2020	CHEM	1A	TR	6:30-8:00	24	46	11	<button>Drop</button>	<button>Manage</button>
Summer	2019	HIS	17A	TR	9:00-10:30	31	33	21	<button>Drop</button>	<button>Manage</button>

```
public static ArrayList<HashMap<String, String>> viewTaughtCourses(String instructorID) {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    /* Check for invalid inputs. If any input is null, return false */
    if (instructorID == null) return output; // Check if studentID is null, return empty list if so
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        // Attempt to search
        G SQL.SQLMethods
            .retrieve tuple */
        SQLMethods.con.prepareStatement(
            "/* FROM Teaches JOIN Configurations Using (configID) WHERE instructorID = ? ORDER BY year DESC;");
        pstate.setString(1, instructorID);
        result = pstate.executeQuery(); // Execute query
        /* Extract tuple data */
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("instructorID", result.getString("instructorID"));
            tuple.put("department", result.getString("department"));
            tuple.put("number", result.getString("number"));
            tuple.put("term", result.getString("term"));
            tuple.put("year", result.getString("year"));
            tuple.put("days", result.getString("days"));
            tuple.put("time", result.getString("time"));
            tuple.put("room", result.getString("room"));
            tuple.put("seats", result.getString("seats"));
            tuple.put("configID", Integer.toString(result.getInt("configID")));
            output.add(tuple);
        }
        result.close(); // Close result
        SQLMethods.closeConnection(); // Close connection
        return output; // Return output
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output; // Return false as a default value
}
```

## Drop courses taught

Click 'DROP' to drop a course.

```
/*
public static boolean delete(String instructorID, String department, String number, int configID) {
    /** Check for invalid inputs. If any input is null, return false */
    if (instructorID == null || department == null || number == null || configID < 0) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete
        pstate = SQLMethods.con.prepareStatement(
            "DELETE FROM Teaches WHERE instructorID = ? AND department = ? AND number = ? AND configID = ?");
        pstate.setString(1, instructorID);
        pstate.setString(2, department);
        pstate.setString(3, number);
        pstate.setInt(4, configID);
        int rowCount = pstate.executeUpdate(); // Execute query

        // When course is dropped, remove registered and waitlisted students
        pstate = SQLMethods.con
            .prepareStatement("DELETE FROM Registers WHERE department = ? AND number = ? AND configID = ?");
        pstate.setString(1, department);
        pstate.setString(2, number);
        pstate.setInt(3, configID);
        rowCount += pstate.executeUpdate(); // Execute query

        pstate = SQLMethods.con
            .prepareStatement("DELETE FROM Waitlists WHERE department = ? AND number = ? AND configID = ?");
        pstate.setString(1, department);
        pstate.setString(2, number);
        pstate.setInt(3, configID);
        rowCount += pstate.executeUpdate(); // Execute query

        SQLMethods.closeConnection(); // Close connection
        return true; // Successful insert
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Default value: false
}
```

# *View registered and waitlisted students*

Click 'Manage'.

The screenshot shows a web-based course management system. At the top, there is a navigation bar with links: Instructor Portal, Manage Courses, Teach Course, and Logout. Below the navigation bar, the main content area has a title "Registered Students". A table displays two registered students: Pasquale Nelson and Reyna Colon. Each student entry includes a "Drop" button. Below this section is another title "Waitlisted Students". A table displays one waitlisted student: Lupe Juarez. This student also has a "Drop" button. At the bottom of the page, there is a search bar with the placeholder "Student ID" and an "Add" button.

Student ID	First Name	Last Name	
199526438	Pasquale	Nelson	<input type="button" value="Drop"/>
339820994	Reyna	Colon	<input type="button" value="Drop"/>

Student ID	First Name	Last Name		
951597733	Lupe	Juarez	<input type="button" value="Drop"/>	<input type="button" value="Add"/>

```
public static ArrayList<HashMap<String, String>> getRegisteredStudents(String department, String number,
    int configID) {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        pstate = SQLMethods.con.prepareStatement(
            "SELECT * FROM Registers JOIN Members ON studentID = ID WHERE department = ? AND number = ? AND configID = ?;");
        pstate.setString(1, department);
        pstate.setString(2, number);
        pstate.setInt(3, configID);
        result = pstate.executeQuery(); // Execute query
        /** Extract tuple data */
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("studentID", result.getString("studentID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            output.add(tuple);
        }
        result.close(); // Close result
        SQLMethods.closeConnection(); // Close connection
        return output; // Return output
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}
```

```

public static ArrayList<HashMap<String, String>> getWaitlistedStudents(String department, String number,
    int configID) {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try {
        pstate = SQLMethods.con.prepareStatement(
            "SELECT * FROM Waitlists JOIN Members ON studentID = ID WHERE department = ? AND number = ? AND configID = ?;");
        pstate.setString(1, department);
        pstate.setString(2, number);
        pstate.setInt(3, configID);
        result = pstate.executeQuery(); // Execute query
        /** Extract tuple data */
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("studentID", result.getString("studentID"));
            tuple.put("firstname", result.getString("firstname"));
            tuple.put("lastname", result.getString("lastname"));
            output.add(tuple);
        }
        result.close(); // Close result
        SQLMethods.closeConnection(); // Close connection
        return output; // Return output
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}

```

## Drop registered students

Click the drop button next to the student in the registrants table.

```
/*
public static boolean drop(String studentID, String department, String number, int configID) {
    if (studentID == null || department == null || number == null || configID < 0) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete
        /**
         * Check if student is in Registers. If so, we check else statement. If not, we
         * have another student waiting on waitlist, so we move waiting student to
         * Registers and delete old student.
        */
        int value = 0;
        if (isWaitlisted(studentID, department, number, configID)) { // If student is on waitlist
            pstate = SQLMethods.con.prepareStatement(
                "DELETE FROM Waitlists WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");
            pstate.setString(1, studentID);
            pstate.setString(2, department);
            pstate.setString(3, number);
            pstate.setInt(4, configID);
            value = pstate.executeUpdate();
        } else {
            pstate = SQLMethods.con.prepareStatement(
                "DELETE FROM Registers WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");
            pstate.setString(1, studentID);
            pstate.setString(2, department);
            pstate.setString(3, number);
            pstate.setInt(4, configID);
            value = pstate.executeUpdate();

            /** Adjust the waitlist if needed */
            adjustWaitlist(studentID, studentID, number, configID);
        }
        SQLMethods.closeConnection(); // Close connection
        return true; // Success
    } catch (SQLException e) {
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Return false as a default value
}
```

## Drop waitlisted students

Click the drop button next to the student in the waitlist table.

```
/*
public static boolean drop(String studentID, String department, String number, int configID) {
    if (studentID == null || department == null || number == null || configID < 0) return false;
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to delete
        /**
         * Check if student is in Registers. If so, we check else statement. If not, we
         * have another student waiting on waitlist, so we move waiting student to
         * Registers and delete old student.
        */
        int value = 0;
        if (isWaitlisted(studentID, department, number, configID)) { // If student is on waitlist
            pstate = SQLMethods.con.prepareStatement(
                "DELETE FROM Waitlists WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");
            pstate.setString(1, studentID);
            pstate.setString(2, department);
            pstate.setString(3, number);
            pstate.setInt(4, configID);
            value = pstate.executeUpdate();
        } else {
            pstate = SQLMethods.con.prepareStatement(
                "DELETE FROM Registers WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");
            pstate.setString(1, studentID);
            pstate.setString(2, department);
            pstate.setString(3, number);
            pstate.setInt(4, configID);
            value = pstate.executeUpdate();

            /** Adjust the waitlist if needed */
            adjustWaitlist(studentID, studentID, number, configID);
        }
        SQLMethods.closeConnection(); // Close connection
        return true; // Success
    } catch (SQLException e) {
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return false; // Return false as a default value
}
```

## *Manually add students on waitlist to registered list for a taught course*

Click add next to the student in the waitlist table.

```
public static boolean overrideRegister(String studentID, String department, String number, int configID) {  
    /** Check for invalid inputs. If any input is null, return false */  
    if (studentID == null || department == null || number == null || configID < 0) return false;  
    /** Check if student already registered for this course */  
    if (isRegistered(studentID, department, number, configID)) return false;  
    drop(studentID, department, number, configID); // Drop student from waitlist if possible  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to insert  
        // Ignore if seats are filled  
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Registers VALUES (?, ?, ?, ?);");  
        pstate.setString(1, studentID);  
        pstate.setString(2, department);  
        pstate.setString(3, number);  
        pstate.setInt(4, configID);  
        int rowCount = pstate.executeUpdate(); // Number of rows affected  
        SQLMethods.closeConnection(); // Close connection  
        return (rowCount == 1); // If rowCount == 1, row successfully inserted  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Default value: false  
}
```

## *Manually add students using their ID to registered list for a taught course*

Enter student's ID in the textbox and click add to add a student manually.

```
public static boolean overrideRegister(String studentID, String department, String number, int configID) {  
    /** Check for invalid inputs. If any input is null, return false */  
    if (studentID == null || department == null || number == null || configID < 0) return false;  
    /** Check if student already registered for this course */  
    if (isRegistered(studentID, department, number, configID)) return false;  
    drop(studentID, department, number, configID); // Drop student from waitlist if possible  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to insert  
        // Ignore if seats are filled  
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Registers VALUES (?, ?, ?, ?);");  
        pstate.setString(1, studentID);  
        pstate.setString(2, department);  
        pstate.setString(3, number);  
        pstate.setInt(4, configID);  
        int rowcount = pstate.executeUpdate(); // Number of rows affected  
        SQLMethods.closeConnection(); // Close connection  
        return (rowcount == 1); // If rowcount == 1, row successfully inserted  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Default value: false  
}
```

# *View offered courses*

Click 'Teach Course'.

Instructor Portal   Manage Courses   Teach Course   Logout

## Teach Course

Department	Number	Config ID
<input type="text"/>	<input type="text"/>	<input type="text"/>

## Taught Courses

Term	Year	Department	Number	Days	Time	Room	Seats	ConfigID
Spring	2021	Math	127	MW	4:00 - 6:00	41	31	1
Winter	2020	CHEM	1A	TR	6:30-8:00	24	46	11
Summer	2019	HIS	17A	TR	9:00-10:30	31	33	21

## Offered Courses

Department	Number	Title	Units	Cost
BIO	1A	Biology I	5	230
BIO	1B	Biology II	5	230
CHEM	1A	Chemistry I	5	230
CHEM	1B	Chemistry II	5	230

```

/*
public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement("SELECT * FROM Courses ORDER BY department ASC;");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("department", result.getString("department"));
            tuple.put("number", result.getString("number"));
            tuple.put("title", result.getString("title"));
            tuple.put("units", Integer.toString(result.getInt("units")));
            tuple.put("cost", Integer.toString(result.getInt("cost")));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}

```

## *View offered course configurations*

Click 'Teach Course'.

### Configurations

ConfigID	Term	Year	Days	Time	Room	Seats
1	Spring	2021	MW	4:00 - 6:00	41	31
2	Summer	2021	TR	6:30-8:00	33	60
3	Summer	2021	MWTR	9:00-10:30	34	56
4	Summer	2021	MW	8:00-9:30	22	52
5	Summer	2021	MWT	6:30-8:00	40	54
6	Fall	2020	MW	1:30-3:00	50	53
7	Fall	2020	TR	10:00-11:30	25	51
8	Fall	2020	MWTD	8:00-9:30	10	25

```

public static ArrayList<HashMap<String, String>> getAll() {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con.prepareStatement("SELECT * FROM Configurations ORDER BY year DESC;");
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("configID", Integer.toString(result.getInt("configID")));
            tuple.put("term", result.getString("term"));
            tuple.put("year", Integer.toString(result.getInt("year")));
            tuple.put("days", result.getString("days"));
            tuple.put("time", result.getString("time"));
            tuple.put("room", result.getString("room"));
            tuple.put("seats", Integer.toString(result.getInt("seats")));
            output.add(tuple);
        }
        result.close(); // Close result
        return output; // Successful search
    } catch (SQLException e) { // Print error and terminate program
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output;
}

```

## Teach a course

At the top of the page, enter the fields of the course and click submit. Select a course's department and number and config ID to add the course to teaching list. If the course is being taught by you or someone else – a certain combination of department, number, and configID are already in use by someone else – the course will not add.

The screenshot shows a web page titled "Teach Course". At the top, there is a navigation bar with links: "Instructor Portal", "Manage Courses", "Teach Course", and "Logout". Below the navigation bar is a form with three input fields: "Department", "Number", and "Config ID". Each field has a small icon next to it. To the right of the "Config ID" field is a "Teach" button. Below the form, there is a large block of Java code:

```
public static boolean insert(String instructorID, String department, String number, int configID) {  
    /** Check for invalid inputs. If any input is null, return false */  
    if (instructorID == null || department == null || number == null || configID < 0) return false;  
    /** Insert tuple */  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try {  
        // Check if course is already being taught  
        if (isTeaching(department, number, configID)) return false;  
        pstate = SQLMethods.con.prepareStatement("INSERT INTO Teaches VALUES (?, ?, ?, ?);");  
        pstate.setString(1, instructorID);  
        pstate.setString(2, department);  
        pstate.setString(3, number);  
        pstate.setInt(4, configID);  
        pstate.executeUpdate(); // Execute query  
        SQLMethods.closeConnection(); // Close connection  
        return true; // Successful insert  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Default value: false  
}
```

# Students

## View registered courses

In the student portal, click 'Add Courses'.

The screenshot shows a student portal interface. At the top, there is a navigation bar with links: Student Portal, Add Courses, Drop Courses, View Courses, Transcript, Transactions, and Logout. Below the navigation bar, the title "Registered Courses" is centered above a table. The table has columns: Term, Year, Department, Number, and ConfigID. The data in the table is as follows:

Term	Year	Department	Number	ConfigID
Spring	2021	Math	127	1
Winter	2020	CHEM	1A	11
Summer	2019	HIS	17A	21

Below the registered courses section is another section titled "Add Courses". It also has a table with columns: Term, Year, Department, Number, Firstname, Lastname, Days, Time, and Room. The data in the table is as follows:

Term	Year	Department	Number	Firstname	Lastname	Days	Time	Room
Summer	2021	ENG	1A	Tamara	Gibson	MW	8:00-9:30	<input type="button" value="Add"/>
Spring	2021	Math	127	Catalina	Schaefer	MW	4:00 - 6:00	<input type="button" value="Add"/>
Summer	2021	MATH	3B	Mia	Hodge	MWTR	9:00-10:30	<input type="button" value="Add"/>
Summer	2021	ENG	1B	Clara	Bradford	MWT	6:30-8:00	<input type="button" value="Add"/>
Summer	2021	ENG	1B	Miguel	Mathews	MWT	6:30-8:00	<input type="button" value="Add"/>
Spring	2021	Math	127	Erin	Weber	MW	4:00 - 6:00	<input type="button" value="Add"/>
Summer	2021	ENG	1A	Agustin	Barrett	MW	8:00-9:30	<input type="button" value="Add"/>
Summer	2021	MATH	3A	Ivan	Tate	TR	6:30-8:00	<input type="button" value="Add"/>

```
public static ArrayList<HashMap<String, String>> viewRegisteredCourses(String studentID) {  
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();  
    /* Check for invalid inputs. If any input is null, return false */  
    if (studentID == null) return output; // Check if studentID is null, return empty list if so  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to search  
        /* Search and retrieve tuple */  
        pstate = SQLMethods.con.prepareStatement(  
            "SELECT * FROM Registers JOIN Configurations Using (configID) WHERE studentID = ? ORDER BY year DESC");  
        pstate.setString(1, studentID);  
        result = pstate.executeQuery(); // Execute query  
        /* Extract tuple data */  
        while (result.next()) {  
            HashMap<String, String> tuple = new HashMap<String, String>();  
            tuple.put("term", result.getString("term"));  
            tuple.put("year", result.getString("year"));  
            tuple.put("department", result.getString("department"));  
            tuple.put("number", result.getString("number"));  
            tuple.put("days", result.getString("days"));  
            tuple.put("time", result.getString("time"));  
            tuple.put("room", result.getString("room"));  
            tuple.put("configID", Integer.toString(result.getInt("configID")));  
            output.add(tuple);  
        }  
        result.close(); // Close result  
        SQLMethods.closeConnection(); // Close connection  
        return output; // Return output  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return output; // Return false as a default value  
}
```

## Add courses

In the picture above, click add next to the course to add. If the course is full, you will be added to the waitlist. If you are already registered or waitlisted, you cannot add the course twice.

```
public static boolean register(String studentID, String department, String number, int configID) {  
    /** Check for invalid inputs. If any input is null, return false */  
    if (studentID == null || department == null || number == null || configID < 0) return false;  
    /** Check if student already registered for this course */  
    if (isRegistered(studentID, department, number, configID)) return false;  
    /** Check if student already waitlisted this course */  
    if (isWaitlisted(studentID, department, number, configID)) return false;  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to insert  
        /** Check if seats are open */  
        // If course is open, register for course  
        // If course is not open, waitlist course  
        if (isOpen(department, number, configID))  
            pstate = SQLMethods.con.prepareStatement("INSERT INTO Registers VALUES (?, ?, ?, ?, ?);");  
        else  
            pstate = SQLMethods.con.prepareStatement("INSERT INTO Waitlists VALUES (?, ?, ?, ?, ?);");  
        pstate.setString(1, studentID);  
        pstate.setString(2, department);  
        pstate.setString(3, number);  
        pstate.setInt(4, configID);  
        int rowCount = pstate.executeUpdate(); // Number of rows affected  
        SQLMethods.closeConnection(); // Close connection  
        return (rowCount == 1); // If rowCount == 1, row successfully inserted  
    } catch (SQLException e) { // Print error and terminate program  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Default value: false  
}
```

## Drop courses

Click 'Drop Courses'. Then click the drop button next to the course to drop.

Term	Year	Department	Number	ConfigID	
Spring	2021	Math	127	1	<input type="button" value="Drop"/>
Winter	2020	CHEM	1A	11	<input type="button" value="Drop"/>
Summer	2019	HIS	17A	21	<input type="button" value="Drop"/>

```
public static boolean drop(String studentID, String department, String number, int configID) {  
    if (studentID == null || department == null || number == null || configID < 0) return false;  
    SQLMethods.mysqlConnect(); // Connect to DB  
    try { // Attempt to delete  
        /**  
         * Check if student is in Registers. If so, we check else statement. If not, we  
         * have another student waiting on waitlist, so we move waiting student to  
         * Registers and delete old student.  
         */  
        int value = 0;  
        if (isWaitlisted(studentID, department, number, configID)) { // If student is on waitlist  
            pstate = SQLMethods.con.prepareStatement(  
                "DELETE FROM Waitlists WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");  
            pstate.setString(1, studentID);  
            pstate.setString(2, department);  
            pstate.setString(3, number);  
            pstate.setInt(4, configID);  
            value = pstate.executeUpdate();  
        } else {  
            pstate = SQLMethods.con.prepareStatement(  
                "DELETE FROM Registers WHERE studentID = ? AND department = ? AND number = ? AND configID = ?");  
            pstate.setString(1, studentID);  
            pstate.setString(2, department);  
            pstate.setString(3, number);  
            pstate.setInt(4, configID);  
            value = pstate.executeUpdate();  
  
            /** Adjust the waitlist if needed */  
            adjustWaitlist(studentID, studentID, number, configID);  
        }  
        SQLMethods.closeConnection(); // Close connection  
        return true; // Success  
    } catch (SQLException e) {  
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());  
    }  
    return false; // Return false as a default value  
}
```

# View offered courses

Click 'View Courses'.

Student Portal	Add Courses	Drop Courses	View Courses	Transcript	Transactions	Logout		
Offered Courses								
Term	Year	Department	Number	Firstname	Lastname	Days	Time	Room
Summer	2021	ENG	1A	Tamara	Gibson	MW	8:00-9:30	22
Spring	2021	Math	127	Catalina	Schaefer	MW	4:00 - 6:00	41
Summer	2021	MATH	3B	Mia	Hodge	MWTR	9:00-10:30	34
Summer	2021	ENG	1B	Clara	Bradford	MWT	6:30-8:00	40
Summer	2021	ENG	1B	Miguel	Mathews	MWT	6:30-8:00	40
Spring	2021	Math	127	Erin	Weber	MW	4:00 - 6:00	41
Summer	2021	ENG	1A	Agustin	Barrett	MW	8:00-9:30	22
Summer	2021	MATH	3A	Inez	Tate	TR	6:30-8:00	33
Summer	2021	MATH	3A	Jack	Vargas	TR	6:30-8:00	33
Summer	2021	MATH	3B	Alton	Gallagher	MWTR	9:00-10:30	34
Fall	2020	PHYS	4A	Antoinette	Marks	MWF	9:30-11:00	18
Winter	2020	CS	157A	Mia	Hodge	MW	3:00-4:30	48
Fall	2020	COMM	2	Polly	Rios	MW	1:30-3:00	50
Spring	2020	CS	149	Tamara	Gibson	MWF	4:00 - 6:00	43
Winter	2020	PHYS	4B	Sylvia	Fuentes	MW	2:00-4:30	50

```

package courses;

import ...

/** Courses(department, number, title, units, cost) */
public class Courses {
    private static ResultSet result;
    private static PreparedStatement pstate;

    /**
     * Insert a course
     *
     * @param department
     * @param number
     * @param title
     * @param units
     * @param cost
     * @return true if successful insert, else false
     */
    public static boolean insert(String department, String number, String title, int units, int cost) {
        /* Check for invalid inputs. If any input is null, return false */
        if (department == null || number == null || title == null || units < 0 || cost < 0) return false;
        SQLMethods.mysqlConnect(); // Connect to DB
        try { // Attempt to insert
            pstate = SQLMethods.con.prepareStatement("INSERT INTO Courses Values (?, ?, ?, ?, ?)");
            pstate.setString(parameterIndex: 1, department);
            pstate.setString(parameterIndex: 2, number);
            pstate.setString(parameterIndex: 3, title);
            pstate.setInt(parameterIndex: 4, units);
            pstate.executeUpdate();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }
}

```

The screenshot shows the IntelliJ IDEA interface with the 'Courses.java' file open. The code implements a `Courses` class with a static method `insert` that inserts a new course record into a MySQL database. The code includes validation for null inputs and connects to the database using `SQLMethods.mysqlConnect()`. The `insert` method uses a `PreparedStatement` to execute the `INSERT INTO Courses` SQL query with five parameters: department, number, title, units, and cost.

Project

Administration Schemas Query 1 Users Registers Configurations Members Registers Context Help Snippets

Schemas Course Management Syst...

Tables Administrators Configurations Courses Instructors Members Registers Students Teaches Transactions Users Waitlists Views Stored Procedures Functions

Registers

Object Info Session

**Table: Registers**

**Columns:**

- studentID char(9) PK
- department varchar(20) PK
- number varchar(10) PK
- configID int PK

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

1 • SELECT \* FROM Courses;

department	number	title	units	cost
BIO	1A	Biology I	5	230
BIO	1B	Biology II	5	230
CHEM	1A	Chemistry I	5	230
CHEM	1B	Chemistry II	5	230
COMM	2	Public Communication	3	138
CS	146	Data Structures	4	184
CS	147	Computer Architecture	4	184
CS	157A	Database	4	184
CS	166	Information Security	4	184
ENG	1A	Language & Compos...	3	138
ENG	1B	Literature	3	138
GEO	2A	Introduction to Geolo...	3	138
GEO	2B	Introduction to Geolo...	3	138
HIS	17A	U.S. History I	3	138
HIS	17B	U.S. History II	3	138
HUM	5A	Humanities Intro I	3	138
HUM	5B	Humanities Intro II	3	138
KIN	1	Soccer	1	46
KIN	2	Basketball	1	46
KIN	3	Swimming	1	46
KIN	4	Badminton	1	46
MATH	127	Linear Algebra	4	184
MATH	3A	Calculus I	4	184
PHYS	4A	Physics I	5	230
PHYS	4B	Physics II	5	230
SPA	1	Spanish I	5	230
SPA	2	Spanish II	5	230
SPA	3	Spanish III	5	230

Courses 2 Apply Revert

Action Output

Time	Action	Response	Duration / Fetch Time
69 23:10:32	SELECT * FROM 'Course Management System'.Courses LIMIT 0, 1000	29 row(s) returned	0.00040 sec / 0.000...
70 23:10:42	SELECT * FROM Courses LIMIT 0, 1000	29 row(s) returned	0.00037 sec / 0.000...

Query Completed

# *View transcript and registered courses*

Click ‘Transcript’.

The screenshot shows a web application interface with a header containing links: Student Portal, Add Courses, Drop Courses, View Courses, Transcript, Transactions, and Logout. The main content area is titled "Transcript" and displays a table with four rows of student data:

Term	Year	Department	Number
Spring	2021	Math	127
Winter	2020	CHEM	1A
Summer	2019	HIS	17A

The screenshot shows an IDE interface with the project structure on the left and the code editor on the right. The code editor displays the JSP file "transcript.jsp". The code includes imports for "Registers", "Registers.java", "ArrayList", "HashMap", and "Transactions.java". It defines a string "studentID" from the request parameters and uses it to generate a navigation bar with links to various pages like "Student Portal", "Add Courses", etc. Below the navigation bar is a heading "Transcript" and a table with columns "Term", "Year", "Department", and "Number".

```

1 <%@page import="registers.Registers,java.util.ArrayList,java.util.HashMap"
2   languages="java" contentType="text/html; charset=UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta charset="UTF-8">
7     <title>Transcript</title>
8     <link href="portal.css" rel="stylesheet" type="text/css"/>
9   </head>
10  <body>
11    <String studentID = request.getParameter("studentID"); // get ID from previous page %>
12    <div class="topnav">
13      <a class="active" href="<%=studentPortal.jsp?studentID%>+studentID%>>Student Portal</a>
14      <a href="<%=add.jsp?studentID%>+studentID%>>Add Courses</a>
15      <a href="<%=drop.jsp?studentID%>+studentID%>>Drop Courses</a>
16      <a href="<%=viewCourses.jsp?studentID%>+studentID%>>View Courses</a>
17      <a href="<%=transcript.jsp?studentID%>+studentID%>>Transcript</a>
18      <a href="<%=transactions.jsp?studentID%>+studentID%>>Transactions</a>
19      <a href="index.jsp">Logout</a>
20    </div>
21    <h1 style="...>Transcript</h1>
22    <table class="content-table">
23      <tr>
24        <td>Term</td>
25        <td>Year</td>
26        <td>Department</td>
27        <td>Number</td>
28      </tr>
29    </table>
30  </root> <page>

```

The IDE also shows the Tomcat 9.0.36 server running and the Catalina Log output window displaying deployment logs for the "Team4FinalProject.war" artifact.

# *View transaction history*

## *Click 'Transactions'.*

The screenshot shows a web application interface. At the top, there is a navigation bar with links: Student Portal, Add Courses, Drop Courses, View Courses, Transcript, Transactions, and Logout. Below the navigation bar, the main content area has a title "Make a Transaction:" followed by a form with fields for "Credit Card" and "Amount", and a "Submit" button. Below this, there is a section titled "Transaction History" containing a table with three columns: Credit Card, Amount, and Timestamp. The table displays three transactions:

Credit Card	Amount	Timestamp
4729921282617690	471	2020-08-04 14:18:09
7748348254287430	267	2020-08-03 19:38:15
8446756072343220	415	2020-08-02 07:09:12

```
/*
public static ArrayList<HashMap<String, String>> viewTransactions(String studentID) {
    ArrayList<HashMap<String, String>> output = new ArrayList<HashMap<String, String>>();
    if (studentID == null) return output; // Return empty list
    SQLMethods.mysqlConnect(); // Connect to DB
    try { // Attempt to search
        pstate = SQLMethods.con
            .prepareStatement("SELECT * FROM Transactions WHERE studentID = ? ORDER BY timestamp DESC;");
        pstate.setString(1, studentID);
        result = pstate.executeQuery(); // Execute query
        while (result.next()) {
            HashMap<String, String> tuple = new HashMap<String, String>();
            tuple.put("studentID", result.getString("studentID"));
            tuple.put("creditcard", result.getString("creditcard"));
            tuple.put("amount", result.getString("amount"));
            tuple.put("timestamp", result.getString("timestamp"));
            output.add(tuple);
        }
        result.close(); // Close result
        SQLMethods.closeConnection(); // Close connection
        return output; // Success
    } catch (SQLException e) {
        SQLMethods.mysql_fatal_error("Query error: " + e.toString());
    }
    return output; // default value
}
```

## Pay for courses

Enter credit card information and amount and then click submit. The transaction will be in the portal after submitting.

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Team4FinalProject". It contains several packages: dropFromKoster, manualAdd, teachCourse, unteachCourse, instructors, members, registers, SQL, studentPortal, students, teaches, transactions, userPortal, web, and users. The "Transactions" class is selected in the code editor.
- Code Editor:** The code for the "Transactions" class is displayed. It includes a static method "pay" that inserts a record into the "Transactions" table. The code handles null inputs and generates a timestamp.
- Run Tab:** The "Tomcat 9.0.36" run configuration is selected.
- Deployment Log:** The log shows the deployment of "Team4FinalProject.war" to Tomcat. It includes logs from the Catalina log and the Tomcat localhost log, indicating successful deployment and artifact scanning.
- Bottom Status:** The status bar shows "Build completed successfully in 4 s 514 ms (today 8:02 PM)".

```

1 <%@ page
2 import="transactions.Transactions,java.util.ArrayList,java.util.HashMap"
3 language="java" contentType="text/html; charset=UTF-8"
4 pageEncoding="UTF-8%>
5 <!DOCTYPE html>
6 <html>
7 <head>
8 <meta charset="UTF-8">
9 <title>Transactions</title>
10 <link href="portal.css" rel="stylesheet" type="text/css"/>
11 </head>
12 <body>
13 <% String studentID = request.getParameter("studentID");%>
14 <div class="topnav">
15 <a class="active" href="<%=studentPortal.jsp?studentID%>+studentID>>Student Portal</a>
16 <a href="<%=add.jsp?studentID%>+studentID>>Add Courses</a>
17 <a href="<%=drop.jsp?studentID%>+studentID>>Drop Courses</a>
18 <a href="<%=viewCourses.jsp?studentID%>+studentID>>View Courses</a>
19 <a href="<%=transcript.jsp?studentID%>+studentID>>Transcript</a>
20 <a href="<%=transactions.jsp?studentID%>+studentID>>Transactions</a>
21 <a href="index.jsp">Logout</a>
22 </div>
23 <%
24 ArrayList<HashMap<String, String>> transactions = Transactions.viewTransactions(studentID);
25 %>
26 <h1 style="...">Make a Transaction:</h1>
27 <form action="transactionAction" method="post">
28 <table class="content-table">
29 <tr>

```

Run: Tomcat 9.0.36

Server Tomcat Catalina Log Tomcat Localhost Log

Deployment Output

Connected to server

[2020-08-04 08:02:44,075] Artifact Team4FinalProject:war exploded: Artifact is being deployed, please wait...

[04-Aug-2020 20:02:45.556 INFO [RMI TCP Connection(2)-127.0.0.1] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained

[2020-08-04 08:02:45,631] Artifact Team4FinalProject:war exploded: Artifact is deployed successfully

[2020-08-04 08:02:45,631] Artifact Team4FinalProject:war exploded: Deploy took 1,556 milliseconds

[04-Aug-2020 20:02:53.829 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory /usr/local/apache-tomcat-9.0.36/webapps/Team4FinalProject

[04-Aug-2020 20:02:54.564 INFO [Catalina-utility-1] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs. Enabled TLD scanning.

[04-Aug-2020 20:02:54.575 INFO [Catalina-utility-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory /usr/local/apache-tomcat-9.0.36/webapps/Team4FinalProject has finished in 1,556 ms

Event Log

Build completed successfully in 4 s 514 ms (today 8:02 PM)

Add a new transaction:

INSERT INTO Transactions VALUES (204722757, 5726891121146300, 319, Now());

SCHEMAS

Course Management Syst...

Tables

- Administrators
- Configurations
- Courses
- Instructors
- Members
- Registers
- Students
- Teaches
- Transactions
- Users
- Waitlists

Views

Stored Procedures

Functions

demo

Project\_data

sys

Result Grid

Filter Rows:  Search

Edit:   Export/Import:

1 INSERT INTO Transactions VALUES (204722757, 5726891121146300, 319, Now());

studentID	creditcard	amount	timestamp
199526438	8446756072343220	415	2020-08-02 07:09:12
199526438	77483482542874730	267	2020-08-03 19:38:15
199526438	4729921286176930	471	2020-08-04 14:18:09
204722757	726891121146300	312	2020-08-02 07:09:30
204722757	4785406902176240	217	2020-08-03 19:38:33
204722757	6039171052903600	187	2020-08-04 14:18:27
204722757	5726891121146300	318	2020-08-03 19:38:13
227209241	58711489272990	498	2020-08-02 07:09:31
227209241	682771147224990	498	2020-08-03 19:38:34
227209241	367918864519800	286	2020-08-04 14:18:28
314843835	677504755406390	323	2020-08-02 07:09:16
314843835	2595167085235700	307	2020-08-03 19:38:19
314843835	727848303193290	411	2020-08-04 14:18:13
339820994	53988272448269	360	2020-08-02 07:09:22
339820994	353860747933430	202	2020-08-03 19:38:25
339820994	9423048167971790	122	2020-08-04 14:18:19
452018548	753905848948120	373	2020-08-02 07:09:27
452018548	15615442635126	147	2020-08-03 19:38:30
452018548	9039713967280070	351	2020-08-04 14:18:24
483131196	352586282798930	493	2020-08-02 07:09:14
483131196	24988692798759	486	2020-08-03 19:38:17
483131196	259600944125801	486	2020-08-04 14:18:11
503423266	2297190332105200	466	2020-08-02 07:09:15
503423266	8226574521511990	477	2020-08-03 19:38:18
503423266	8685406565773860	362	2020-08-04 14:18:12
506190495	8237707225753060	111	2020-08-02 07:09:25
506190495	4341541452444450	470	2020-08-03 19:38:28
506190495	4272194249026740	419	2020-08-04 14:18:22
538782975	3581656041727260	267	2020-08-02 07:09:20
538782975	7589588117374810	260	2020-08-03 19:38:23

Action Output

Time	Action	Response	Duration / Fetch Time
66 22:58:13	INSERT INTO Transactions VALUES (204722757, 5726891121146300, 319, Now())	1 row(s) affected	0.026 sec
67 22:58:22	SELECT * FROM 'Course Management System'.Transactions LIMIT 0,1000	61 row(s) returned	0.00059 sec / 0.000...

Query Completed

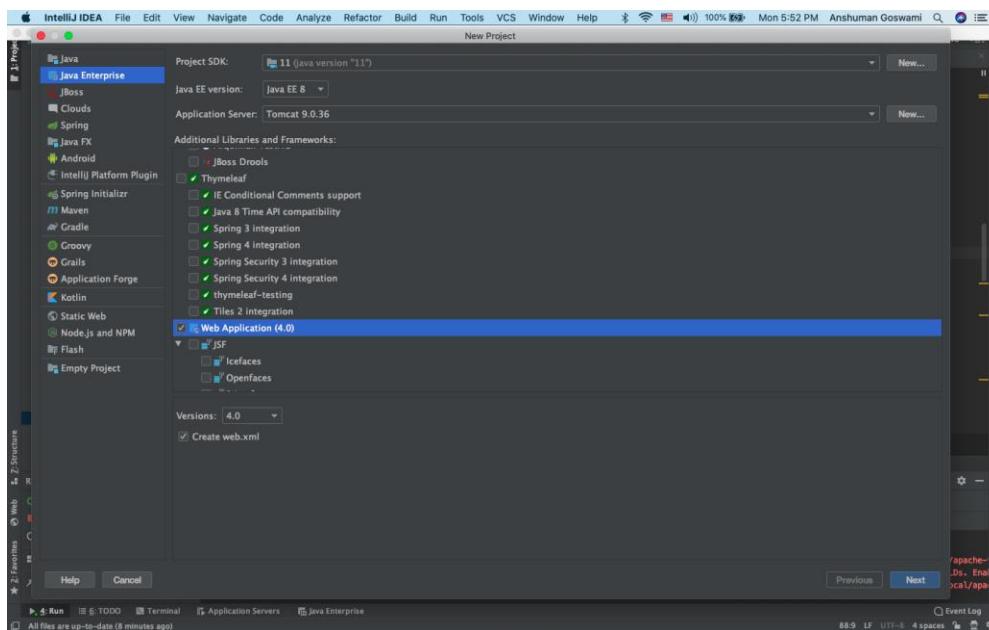
## *Procedures how to set up and run our system:*

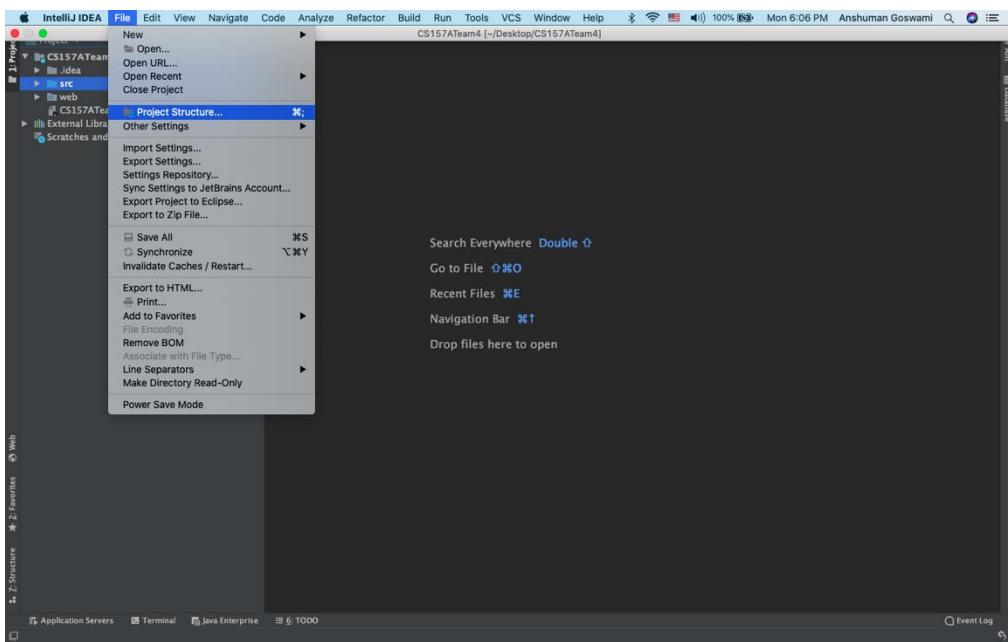
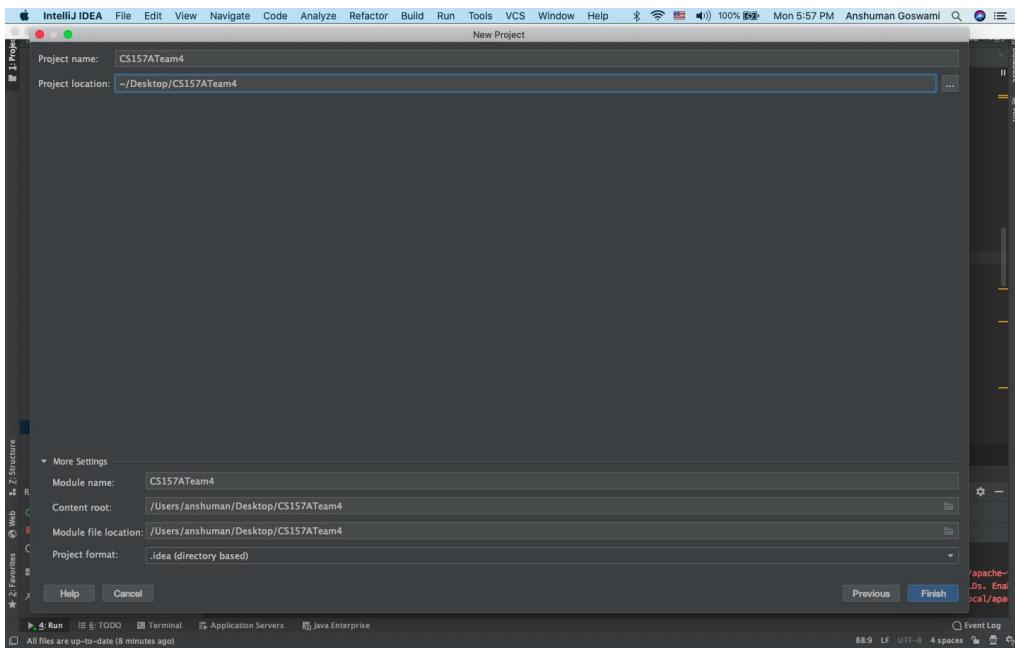
- Installed Apache Tomcat server to run the program.
- Downloaded my-sql-connector-java-8.0.21.jar to connect java files with the database.
- In our team we used a mix of Eclipse and IntelliJ.

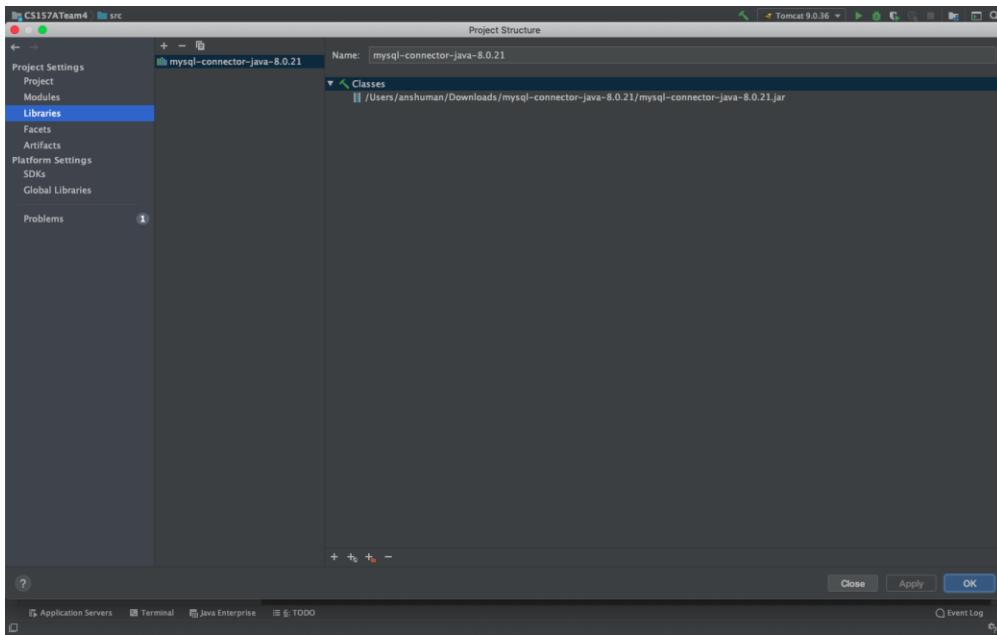
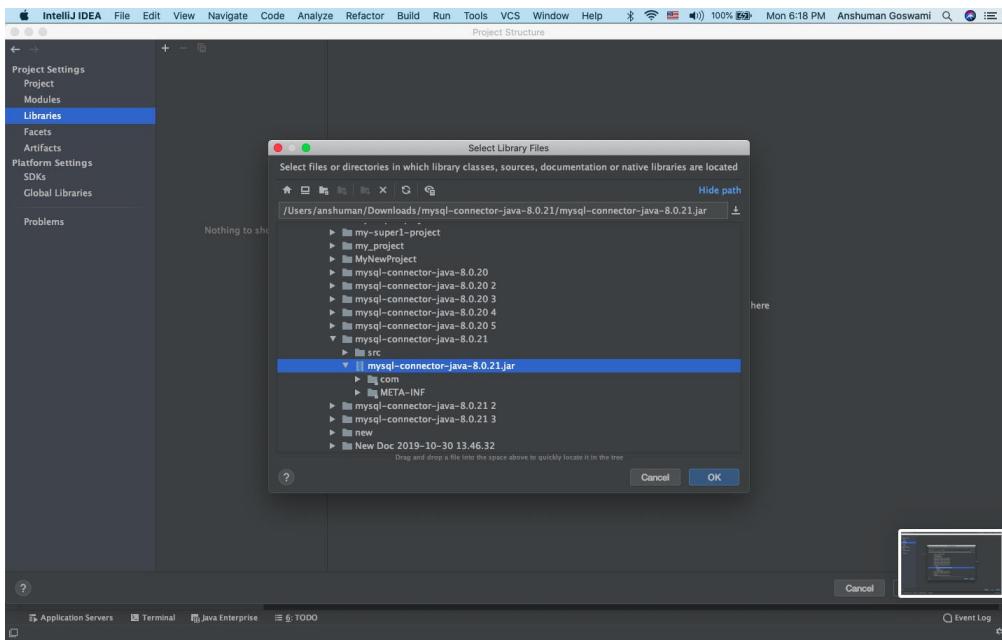
## *Notes when testing*

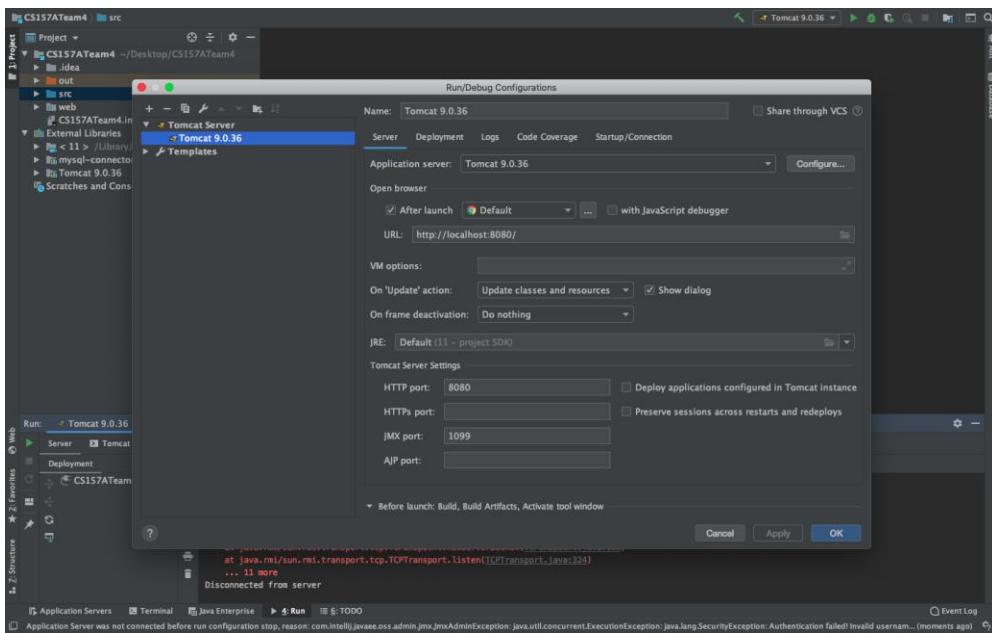
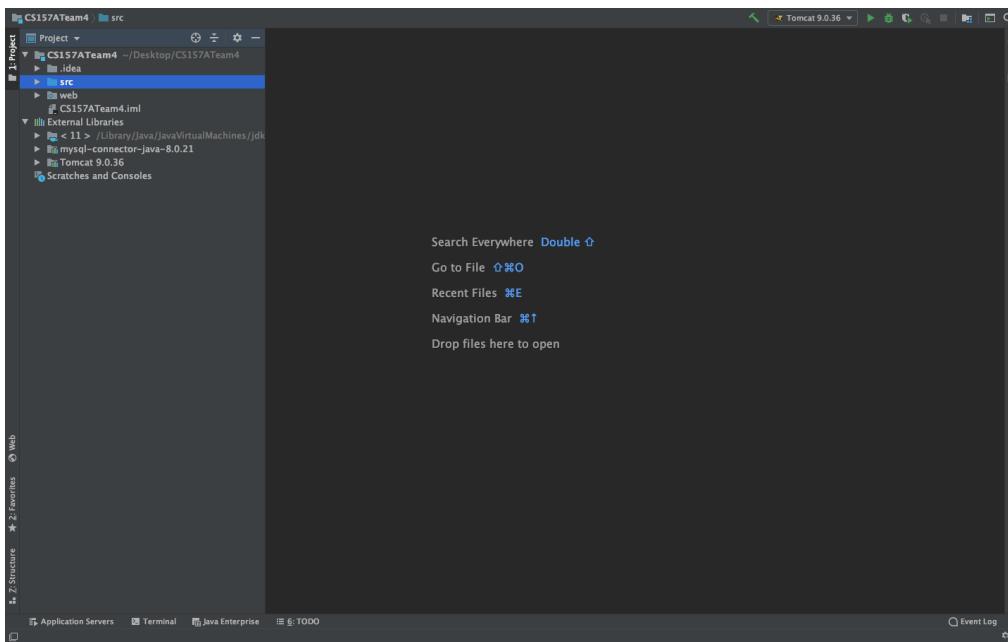
- Please use admin account with clearance 3
  - You can find this in the excel sheet or use the pictures
- You may have difficulty adding existing courses. That is because other instructors are teaching this section. Try logging in as admin, creating a course, and then teach the course with an instructor account.

*Here are some screenshots of the step by step procedure in IntelliJ how we set and run our system.*









```

1 package administrators;
2
3 import ...
4
5 /**
6  * Clearance 1 = manage courses and configs
7  * Clearance 2 = manage accounts
8  * Clearance 3 = manage all
9 */
10 public class Administrators {
11     private static ResultSet result;
12     private static PreparedStatement pstate;
13
14     /**
15      * Insert an admin in Admin table
16      *
17      * @param adminID
18      * @param clearance
19      * @return Returns true if successful, otherwise false
20     */
21     public static boolean insert(String adminID, int clearance) {
22         if (adminID == null || clearance < 0 || clearance > 3) return false;
23         if (adminID == null || clearance < 0 || clearance > 3) return false;
24         SQLMethods.mysqlConnect(); // Connect to DB
25         try { // Attempt to insert
26             pstate = SQLMethods.con.prepareStatement("INSERT INTO Administrators VALUES (?, ?)");
27             pstate.setString(1, adminID);
28             pstate.setInt(2, clearance);
29             int rowcount = pstate.executeUpdate();
30             SQLMethods.mysqlDisconnect(); // Close connection
31             return (rowcount == 1); // If rowcount = 1, row successfully inserted
32         } catch (SQLException e) { // Print error and terminate program
33             SQLMethods.mysql_fatal_error("Query error: " + e.toString());
34         }
35         return false; // Default value: false
36     }
37
38     /**
39      * Delete an admin in Admin table
40      *
41      * @param adminID
42      * @return
43     */
44     public static boolean delete(String adminID) {
45         if (adminID == null) return false;
46         SQLMethods.mysqlConnect(); // Connect to DB
47         try {
48             pstate = SQLMethods.con.prepareStatement("DELETE FROM Administrators WHERE adminID = ?");
49             pstate.setString(1, adminID);
50             int rowcount = pstate.executeUpdate();
51             SQLMethods.mysqlDisconnect(); // Close connection
52             return (rowcount == 1); // If rowcount = 1, row successfully deleted
53         } catch (SQLException e) { // Print error and terminate program
54             SQLMethods.mysql_fatal_error("Query error: " + e.toString());
55         }
56         return false; // Default value: false
57     }
58 }

```

*Run the program using Tomcat server.*

```

1 package teaches;
2
3 import ...
4
5 /**
6  * Teaches(instructorID, department, number, configID)
7  */
8 public class Teaches {
9     private static ResultSet result;
10    private static PreparedStatement pstate;
11
12    /**
13     * Insert a taught course into DB
14     *
15     * @param instructorID
16     * @param department
17     * @param number
18     * @param configID
19     * @return true if successful insert, else false
20     */
21    public static boolean insert(String instructorID, String department, String number, String configID) {
22        if (instructorID == null || department == null || number == null || configID == null) return false;
23        SQLMethods.mysqlConnect(); // Connect to DB
24        try {
25            // Check if course is already being taught
26            if (!isTeaching(department, number, configID)) return false;
27            pstate = SQLMethods.con.prepareStatement("INSERT INTO Teaches VALUES (?, ?, ?, ?)");
28            pstate.setString(1, instructorID);
29            pstate.setString(2, department);
30            pstate.setString(3, number);
31            pstate.setString(4, configID);
32        } catch (SQLException e) {
33            // Check if course is already being taught
34            if (!isTeaching(department, number, configID)) return false;
35            pstate = SQLMethods.con.prepareStatement("UPDATE Teaches SET instructorID = ?, department = ?, number = ?, configID = ? WHERE instructorID = ? AND department = ? AND number = ? AND configID = ?");
36            pstate.setString(1, instructorID);
37            pstate.setString(2, department);
38            pstate.setString(3, number);
39            pstate.setString(4, configID);
40            pstate.setString(5, instructorID);
41            pstate.setString(6, department);
42            pstate.setString(7, number);
43            pstate.setString(8, configID);
44        }
45    }
46
47    /**
48     * Check if course is already being taught
49     * @param department
50     * @param number
51     * @param configID
52     * @return true if course is already being taught, false otherwise
53     */
54    private static boolean isTeaching(String department, String number, String configID) {
55        SQLMethods.mysqlConnect(); // Connect to DB
56        try {
57            pstate = SQLMethods.con.prepareStatement("SELECT * FROM Teaches WHERE department = ? AND number = ? AND configID = ?");
58            pstate.setString(1, department);
59            pstate.setString(2, number);
60            pstate.setString(3, configID);
61            result = pstate.executeQuery();
62            if (result.next()) return true;
63        } catch (SQLException e) {
64            e.printStackTrace();
65        }
66        return false;
67    }
68 }

```

*This is our landing page.*



*Success!*

# **Lessons Learned**

## **Abdurrahman Mohammad**

This was the first time I used GitHub. It was a real learning experience. I learned how to install it on Eclipse and set it up. I learned how to make a web project in Java using Java, HTML, and CSS. I learned how to use JSP files. I learned how to make tables and input forms and then retrieve input from these forms. I learned how to style with CSS and link CSS sheets to pages. I learned and practiced SQL and learned how to use the MySQL Workbench. I learned how to link MySQL with my Java project and connect my schemas. I learned how to execute SQL statement in Java and open and close connections to the database. Overall, I have furthered my knowledge and experience in Java. I learned how to do web development. I have successfully learned how to create a proper, fully-functioning database application.

## **Susmita Goswami**

It was a bit difficult at first since I had to learn both SQL and JSP in order to work on this project. I've also had to learn how to use MySQL Workbench to test out if the project is actually updating the database. This has helped me to become a bit familiar with web development as I also had to learn that for this project. Since this was a team project, working on it allowed me to become a bit more familiar with using git to update my project files and pushing updates to our GitHub repository.

## **Anudeep Gogineni**

I have learned two three new concepts - SQL, servlets and JSP. I have learned new tools - Tomcat server, MySQL Workbench and IntelliJ web application development. I have also learned important tricks of debugging a web application. I spent a lot of time getting my installation ready and this gives me confidence to do it better in future projects. Also, I got a chance to polish my collaborative development skills using git.