Course: CS/SE 4347.003 Database Systems
Instructor: Jalal Omer
Date: 11/30/2021

University Course Registration System

By

Robert Jimenez - RXJ170012
Nasif Mahmood - NXM180057
Rakin Hasan - RXH180025
Antonio Ramaj - AXR190034
Patrick Drury - DPD190003

Table of Contents

Introduction

The Course Registration System is a state of the art program that allows users, whether they be students, professors, counselors, deans, or administrators, to monitor and change the courses a college offers, including the professor teaching the course, the students taking the course, and more.

The Course Registration System uses a relational database, managed by MySQL, to store persistent data related to users it must track, courses and their relationships to users, and much more.

The Course Registration System allows users to login, and then, depending on the type of user they are, as mentioned above, add/remove themselves to/from courses, add/remove others to/from courses, add/remove courses, et cetera. Each type of user is given a different view of what they can (and for security reasons) can not do. For example, a student can add/remove themselves to certain courses but they can’t remove other users from the system, that’s something only somebody logged in as an administrator can do.

What follows is the system requirements for the system, as well as a conceptual overlook of the database that allows the system to operate the way it does. Then we will show an overview of the actual application, show some complex queries that can be ran on the application, and finally how a user can use our application.

System Requirements

System Description

This system will allow students to register for college courses. The students will be able to view different classes and choose what to sign up for. If the student meets the prerequisites and the class is open, then they will be able to register for the course. Each student's course history and enrolled classes will be stored in a database. The info for each course such as the teacher, students enrolled, prerequisites will be stored in another database. A website will be used for naive users (students and instructors) to be able to interact with the DBMS. Database administrators will be able to interact with the DBMS directly using SQL to manipulate the databases. The functional requirements will allow manipulation of students and teachers with their courses.

The functions that students will be able to perform include viewing all the courses and sections available for the university and the specific sections they are signed for in a separate view, signing up for courses they are able to join based on their prerequisites and if they have room in their schedule, and remove themselves for classes they longer wish to be enrolled in.

The functions that instructors will be able to perform include viewing which students are enrolled in which of their sections and adding/removing students from their sections.

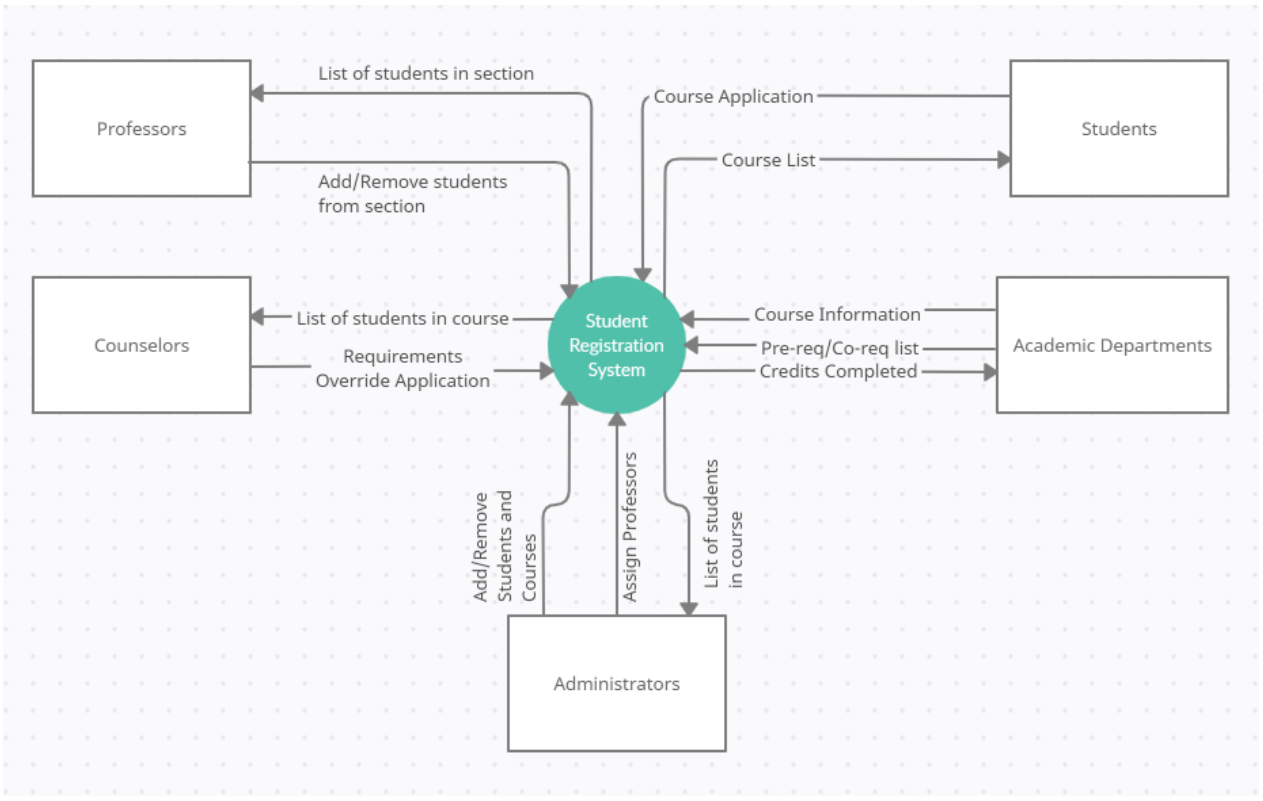
The functions that counselors will be able to perform include all the same functions that students and instructors are able to do but with a higher permission level that will allow them to override the students and instructors. The counselors will be able to do this for any student or instructor in the system.

Administrators will have the highest level of permissions allowing them to override any actions performed by students, instructors and counselors. The functions that administrators will be able to perform include viewing all the students' sections that they are signed up for and adding/removing the students from the sections, adding/removing courses and sections, and assigning professors to different sections of the same course.

Some of the non-functional requirements for the system are the users not having access to areas they are not allowed to, secure logins being handled by the SSO for the University, and multiple students will be able to register at the same time and will be enrolled based on a priority of who enrolled first. The non-functional requirements will focus on reliability, maintainability, safety, and usability.

Some interface requirements of the system are users being able to access the database through a website. This website shall have I/O based on the buttons the users press on the website. Different users will be able to interact with different parts of the website and their view of the web pages shall be different. The students will be able to see courses and enroll or drop. The professors shall be able to view the students enrolled in their courses and change the enrollment of their particular sections. The administrators will be able to manipulate professors' courses and students' enrollment.

Context Diagram



Professors
List of students in section
Add/Remove students from section
Course Application
Course List
Students
Counselors
List of students in course
Requirements Override Application
Student Registration System
Course Information
Pre-req/Co-req list
Credits Completed
Academic Departments
Add/Remove Students and Courses
Assign Professors
List of students in course
Administrators

Functional Requirements

At a high level, the system will keep track of a given university's administrators, counselors, students, professors, courses, and sections of courses in order to create a course registration system that can be used by students, professors, counselors, and university administrators.

The system will relate professors to courses they teach, as well as sections of said courses, counselors to subsets of students, administrators to everything, and students to the courses that they are able to sign up for as well as courses they are currently signed up for.

The system shall allow the addition/removal of students from sections of courses taught in the university. This functionality will be present for administrators, professors, counselors, and students, and the addition/removal of students from sections of courses will be based on the permission levels of each group of users, as well as a given student's credit hours and previously taken courses.

The system will also allow for the addition of courses/sections of courses and of removal/addition of professors from certain courses/sections.

The system will implement checks and balances in order to make sure the data in the system is logically correct at all times, including making sure that the number of a students in a given section of a course is at or under a certain limit, making sure that students cannot sign up for sections of courses at the same time, that professors can't be moved to teach sections of courses they don't teach or to sections that would present a time conflict, and more.

Functional requirements for each group of users:

Professors:

- Shall be able to see a view of all the students signed up for each section of each course they teach.
- Shall be able to add/remove students to their sections for each course they teach.
- Shall be able to view all the courses and their sections for the entire university.

Students:

- Shall be able to view all the courses and their sections for the entire university.
- Shall be able to view all courses/sections currently signed up for.
- Shall be able to add themselves to courses in accordance to the amount of hours they've accumulated, their prerequisites, and amount of credit hours they've signed up for so far.
- Shall be able to remove themselves from courses.

Counselors:

- Shall be able to see a view of all the students signed up for each section of each course.
- Shall be able to add/remove students to classes with the added ability to override prerequisites and credit hour limits.

Administrators:

- Shall be able to see a view of all the students signed up for each section of each course.
- Shall be able to switch professors to and from different sections of the same course.
- Shall be able to add/remove courses and sections.
- Shall be able to add/remove students from courses with the added ability to override prerequisites and credit hour limits.

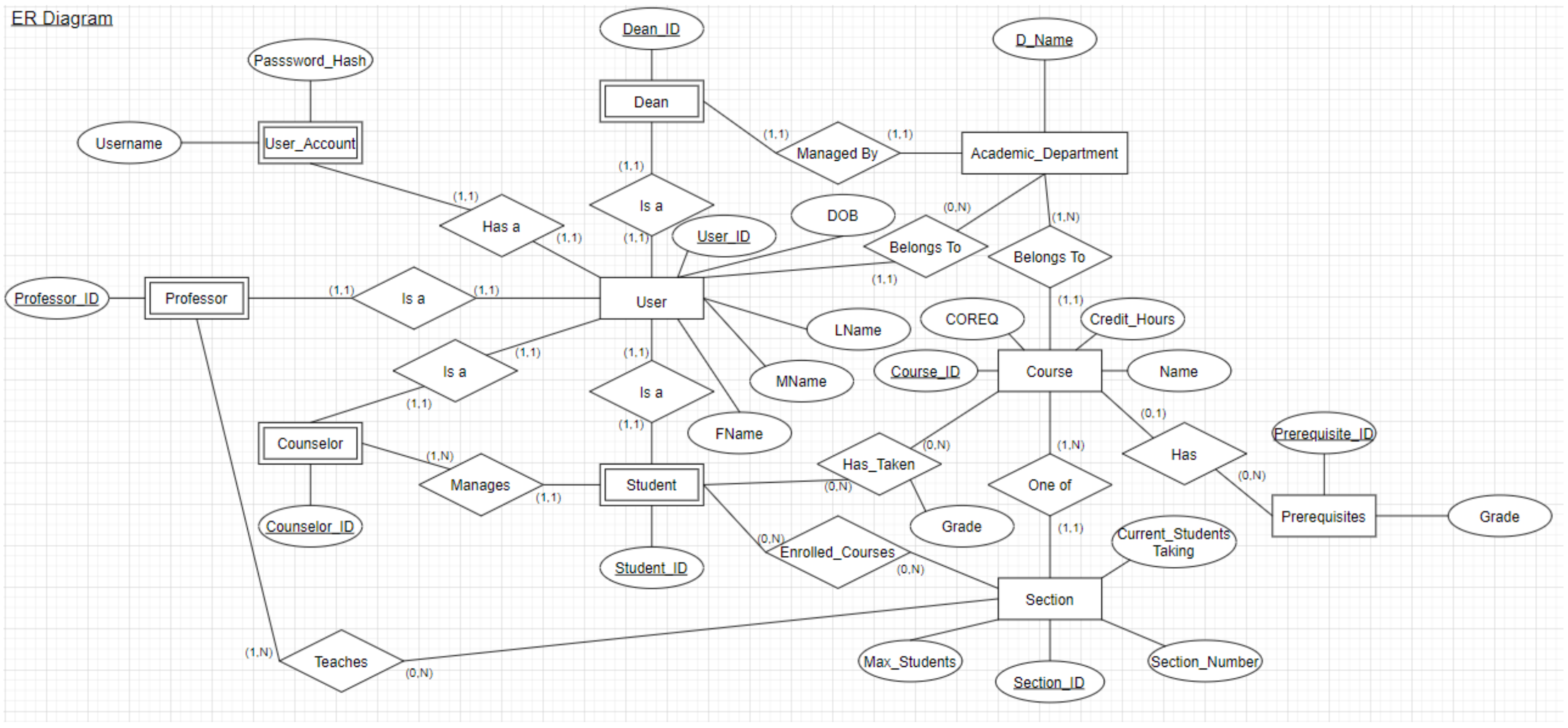
Non-functional Requirements

- Access to student information shall be restricted based on FERPA guidelines and the permissions of a given user
- Near real time updates should occur as changes are made to class enrollment and class creation, an update should take no longer then 30 seconds to appear
- Secure login will be handled by the Universities SSO
- Course registration will be verified and processed within 5 seconds of submission
- System uptime will remain between between 99-100%
- Multiple students should be allowed to register at the same time, registration priority will be determined based on who registered first
- Daily backups will be done to ensure suitable data redundancy
- On a registration attempt student prerequisites / corequisites will be verified
- Service will be accessible and performant globally
- A class search will be completed in less than 5 seconds

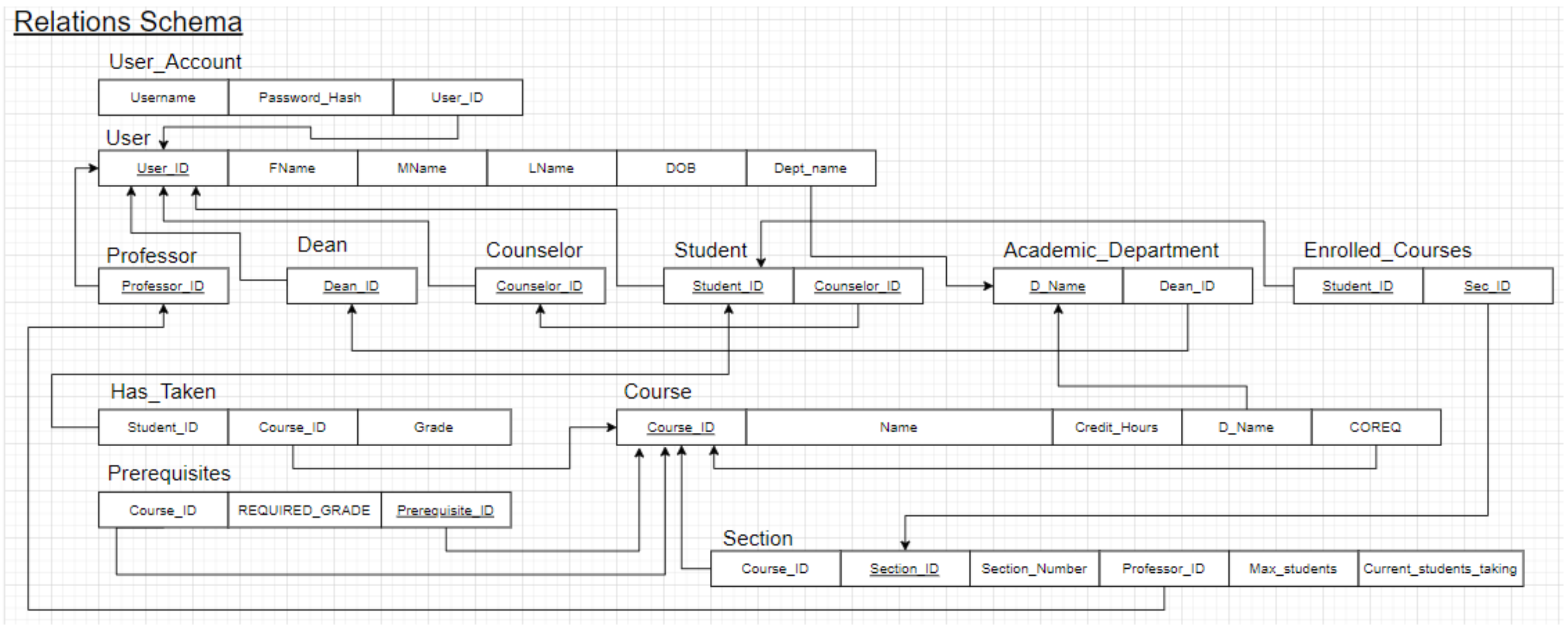
Interface Requirements

- Usernames/passwords shall be used for access to the database.
- The database shall be updated through a website.
- I/O shall be based on buttons and selection lists.
- There shall be a search engine that allows users to look up courses by name, subject, course number, and section.
- Professor/administrators shall be able to view all students enrolled in a course.
- Administrators shall be able to assign professors to the course.
- Students shall be able to see all their courses, as well as add and drop.
- No group of users shall be able to use a permission that is above their power. For example, a student shall not be able to assign a professor to a course, that shall be a power reserved for administrators only.
- The counsellors shall have a similar view, but they have certain special powers, such as overriding prerequisites for the student.

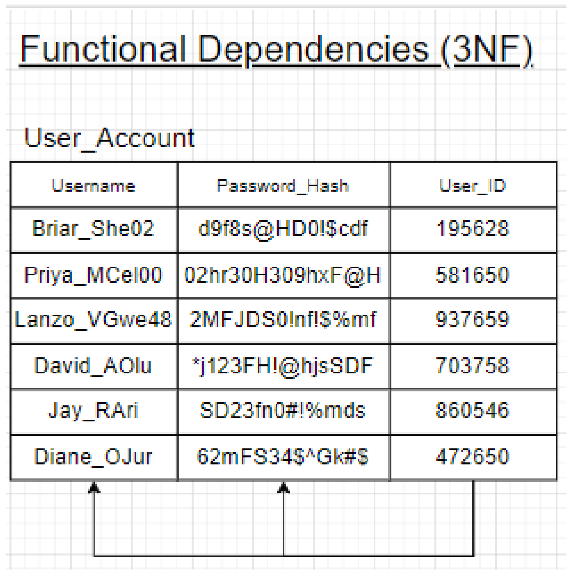
Entity Relationship Model



Relational Database Schema


Relations Schema
User_Account
Username | Password_Hash | User_ID
User
User_ID | FName | MName | LName | DOB | Dept_name
Professor
Professor_ID
Dean
Dean_ID
Counselor
Counselor_ID
Student
Student_ID | Counselor_ID
Academic_Department
D_Name | Dean_ID
Enrolled_Courses
Student_ID | Sec_ID
Has_Taken
Student_ID | Course_ID | Grade
Course
Course_ID | Name | Credit_Hours | D_Name | COREQ
Prerequisites
Course_ID | REQUIRED_GRADE | Prerequisite_ID
Section
Course_ID | Section_ID | Section_Number | Professor_ID | Max_students | Current_students_taking

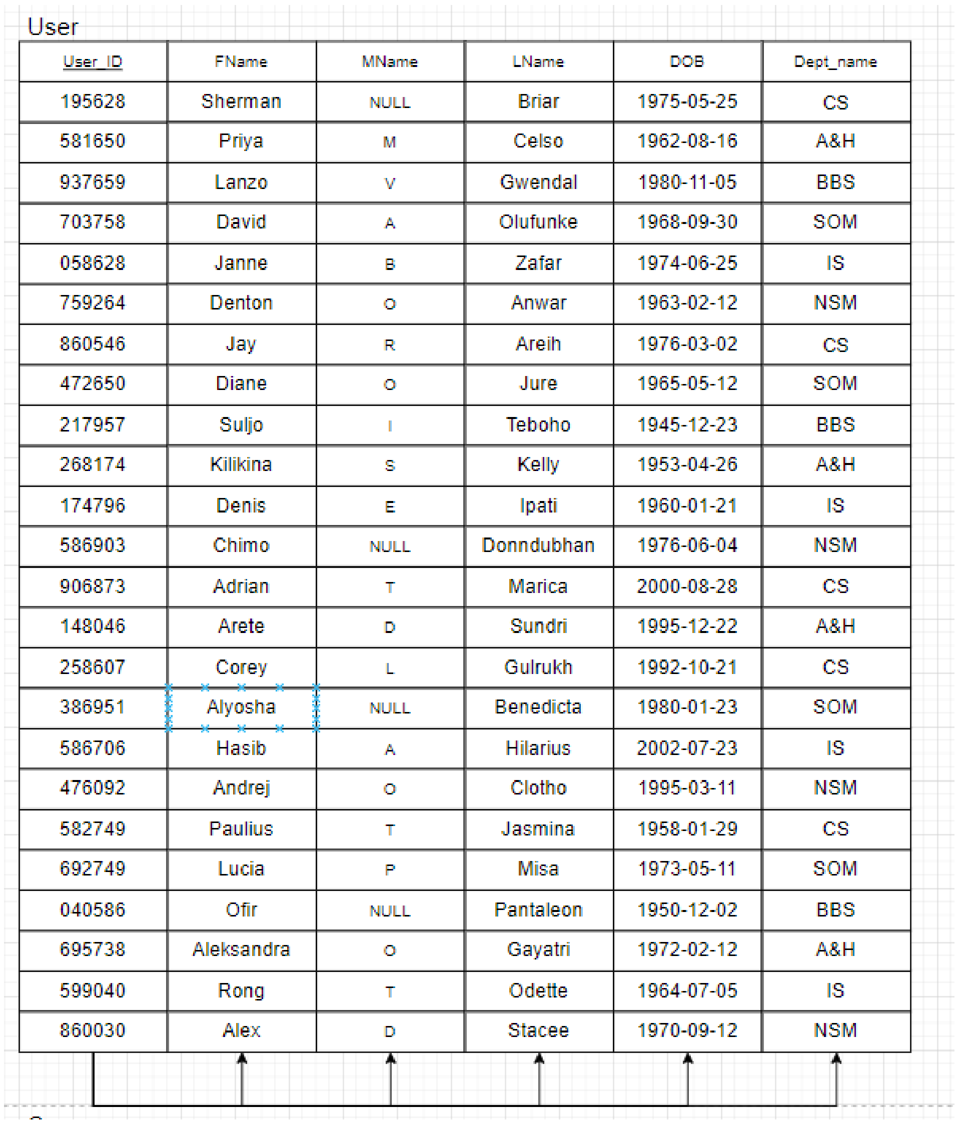
Functional Dependencies



Functional Dependencies (3NF)

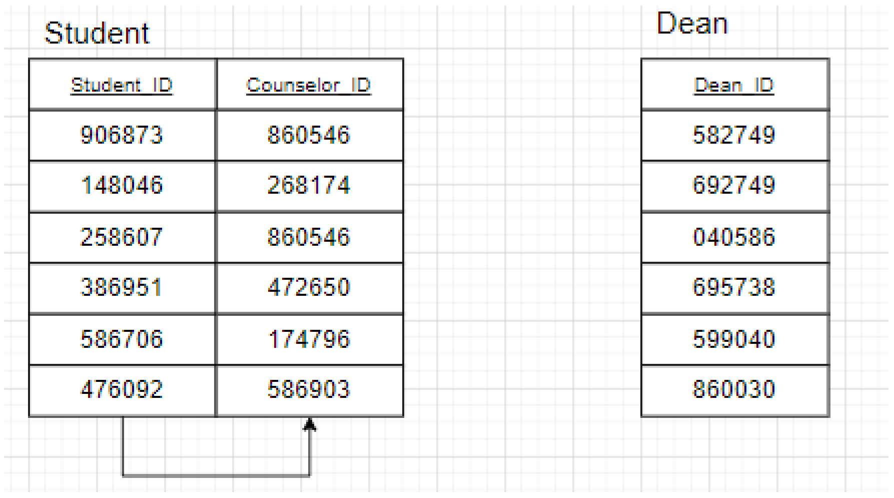
User_Account

Username	Password_Hash	User_ID
Briar_She02	d9f8s@HD0!$cdf	195628
Priya_MCel00	02hr30H309hxF@H	581650
Lanzo_VGwe48	2MFJDS0!nf!$%mf	937659
David_AOlu	*j123FH!@hjsSDF	703758
Jay_RAri	SD23fn0#!%mds	860546
Diane_OJur	62mFS34$^Gk#$	472650



User

User_ID	FName	MName	LName	DOB	Dept_name
195628	Sherman	NULL	Briar	1975-05-25	CS
581650	Priya	M	Celso	1962-08-16	A&H
937659	Lanzo	V	Gwendal	1980-11-05	BBS
703758	David	A	Olufunke	1968-09-30	SOM
058628	Janne	B	Zafar	1974-06-25	IS
759264	Denton	O	Anwar	1963-02-12	NSM
860546	Jay	R	Areih	1976-03-02	CS
472650	Diane	O	Jure	1965-05-12	SOM
217957	Suljo	I	Teboho	1945-12-23	BBS
268174	Kilikina	S	Kelly	1953-04-26	A&H
174796	Denis	E	Ipati	1960-01-21	IS
586903	Chimo	NULL	Donndubhan	1976-06-04	NSM
906873	Adrian	T	Marica	2000-08-28	CS
148046	Arete	D	Sundri	1995-12-22	A&H
258607	Corey	L	Gulrukh	1992-10-21	CS
386951	Alyosha	NULL	Benedicta	1980-01-23	SOM
586706	Hasib	A	Hilarius	2002-07-23	IS
476092	Andrej	O	Clotho	1995-03-11	NSM
582749	Paulius	T	Jasmina	1958-01-29	CS
692749	Lucia	P	Misa	1973-05-11	SOM
040586	Ofir	NULL	Pantaleon	1950-12-02	BBS
695738	Aleksandra	O	Gayatri	1972-02-12	A&H
599040	Rong	T	Odette	1964-07-05	IS
860030	Alex	D	Stacee	1970-09-12	NSM

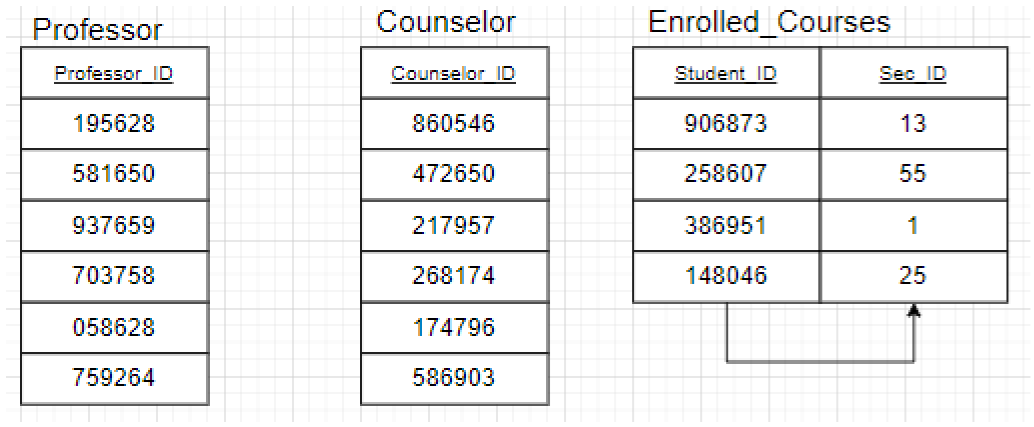


Student

Student_ID	Counselor_ID
906873	860546
148046	268174
258607	860546
386951	472650
586706	174796
476092	586903

Dean

Dean_ID
582749
692749
040586
695738
599040
860030



Professor

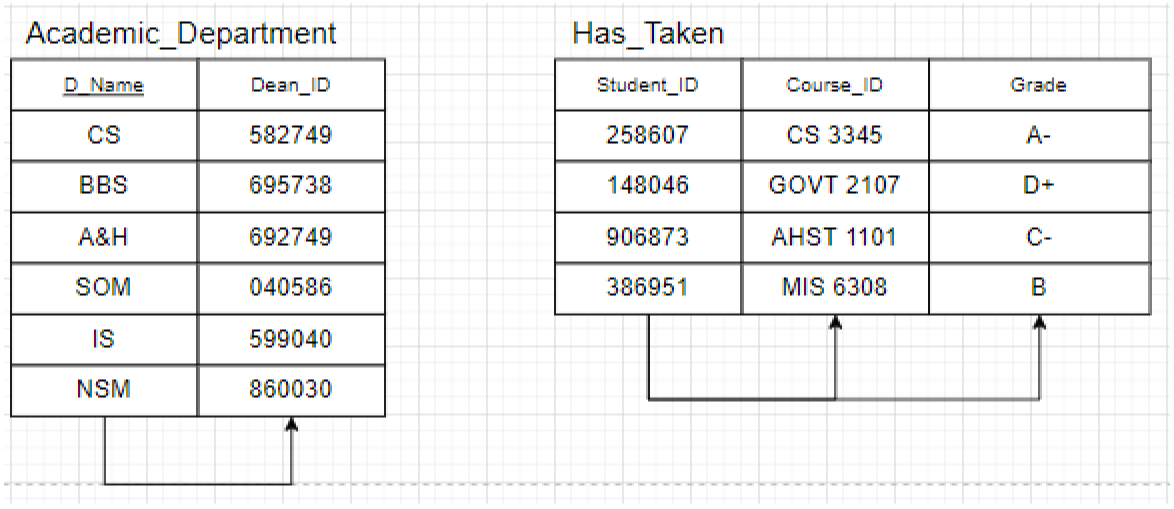
Professor_ID
195628
581650
937659
703758
058628
759264

Counselor

Counselor_ID
860546
472650
217957
268174
174796
586903

Enrolled_Courses

Student_ID	Sec_ID
906873	13
258607	55
386951	1
148046	25

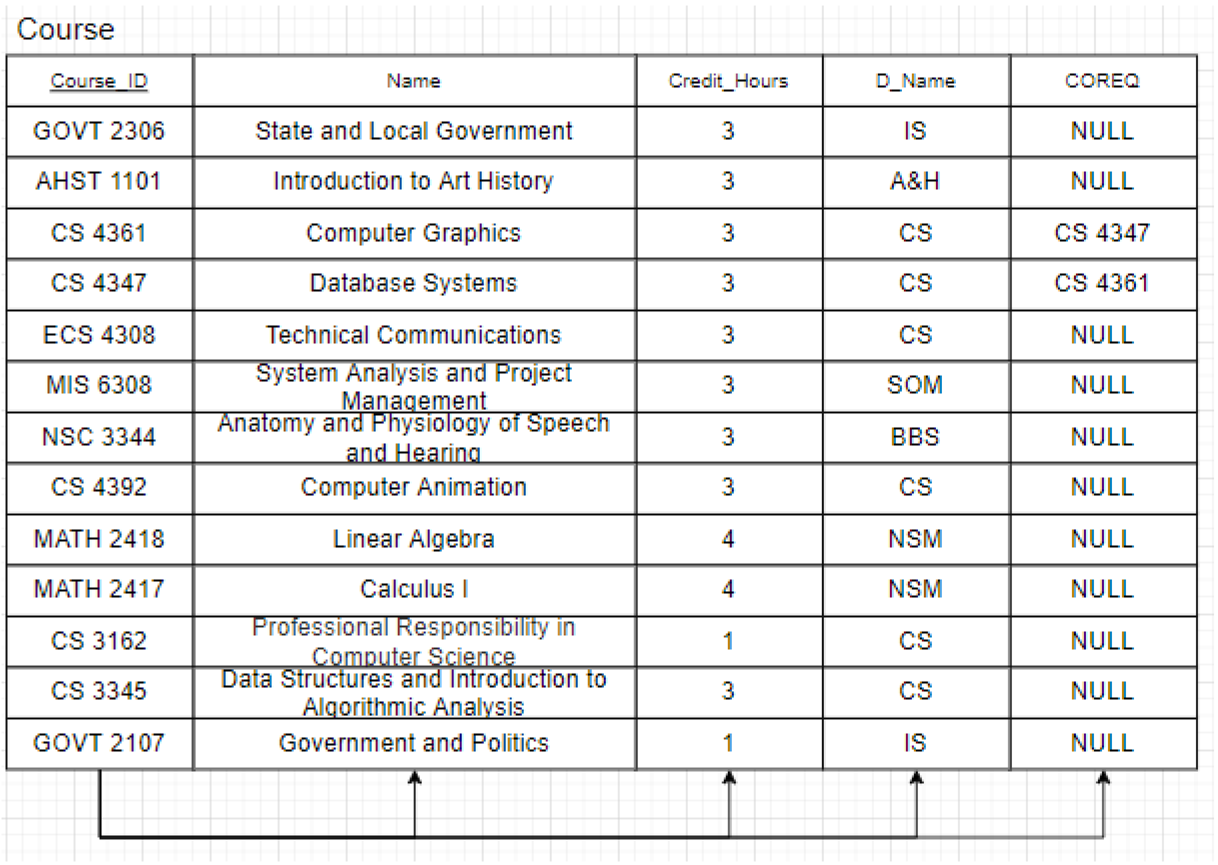


Academic_Department

D_Name	Dean_ID
CS	582749
BBS	695738
A&H	692749
SOM	040586
IS	599040
NSM	860030

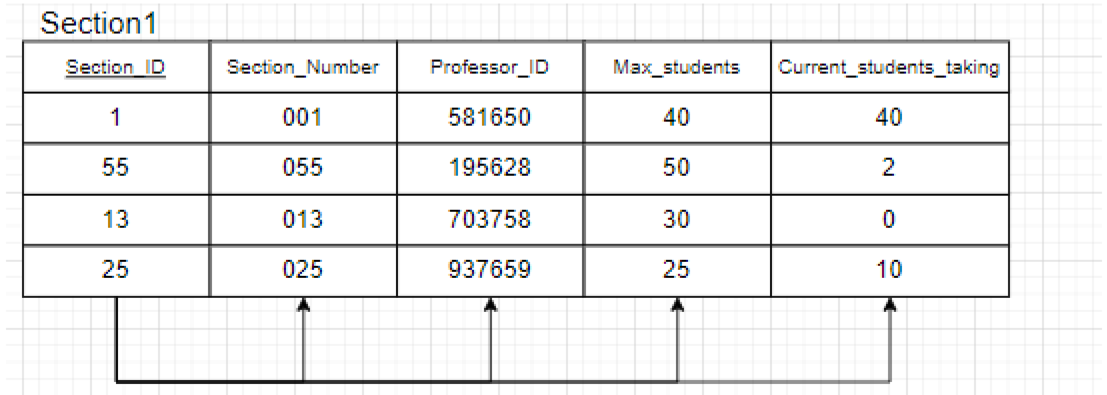
Has_Taken

Student_ID	Course_ID	Grade
258607	CS 3345	A-
148046	GOVT 2107	D+
906873	AHST 1101	C-
386951	MIS 6308	B



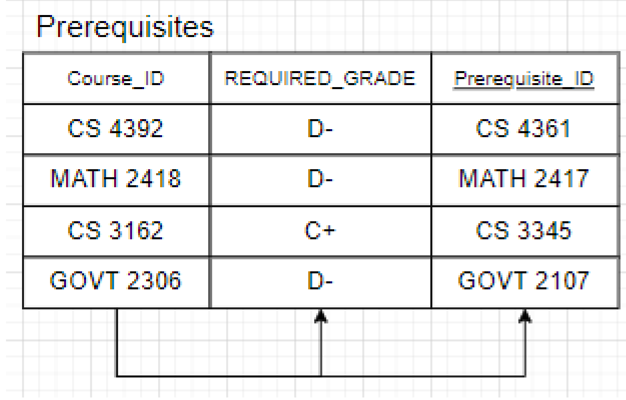
Course

Course_ID	Name	Credit_Hours	D_Name	COREQ
GOVT 2306	State and Local Government	3	IS	NULL
AHST 1101	Introduction to Art History	3	A&H	NULL
CS 4361	Computer Graphics	3	CS	CS 4347
CS 4347	Database Systems	3	CS	CS 4361
ECS 4308	Technical Communications	3	CS	NULL
MIS 6308	System Analysis and Project Management	3	SOM	NULL
NSC 3344	Anatomy and Physiology of Speech and Hearing	3	BBS	NULL
CS 4392	Computer Animation	3	CS	NULL
MATH 2418	Linear Algebra	4	NSM	NULL
MATH 2417	Calculus I	4	NSM	NULL
CS 3162	Professional Responsibility in Computer Science	1	CS	NULL
CS 3345	Data Structures and Introduction to Algorithmic Analysis	3	CS	NULL
GOVT 2107	Government and Politics	1	IS	NULL



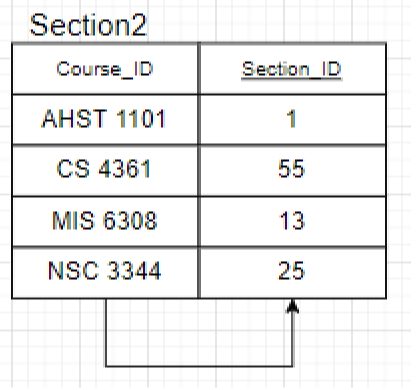
Section1

Section_ID	Section_Number	Professor_ID	Max_students	Current_students_taking
1	001	581650	40	40
55	055	195628	50	2
13	013	703758	30	0
25	025	937659	25	10



Prerequisites

Course_ID	REQUIRED_GRADE	Prerequisite_ID
CS 4392	D-	CS 4361
MATH 2418	D-	MATH 2417
CS 3162	C+	CS 3345
GOVT 2306	D-	GOVT 2107



Section2

Course_ID	Section_ID
AHST 1101	1
CS 4361	55
MIS 6308	13
NSC 3344	25

SQL Statements for Database Construction and Implementation

<u>CREATE TABLE Commands</u>

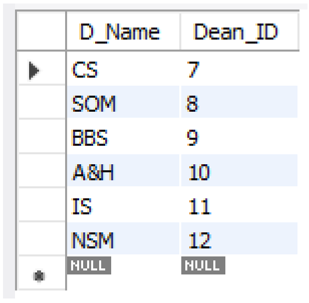
```
CREATE TABLE USER
        (User_ID                INT                     AUTO_INCREMENT NOT
NULL PRIMARY KEY,
         FName                  VARCHAR(50)             NOT NULL,
         MName                  VARCHAR(50)             DEFAULT “N/A”,
         LName                  VARCHAR(50)             NOT NULL,
         DOB                    DATE,
         Dept_Name              VARCHAR(4),
        FOREIGN KEY(Dept_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE SET NULL ON UPDATE CASCADE);

CREATE TABLE PROFESSOR
        (Professor_ID           INT                 PRIMARY KEY,
        FOREIGN KEY(Professor_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);



CREATE TABLE COUNSELOR
        (Counselor_ID           INT                 PRIMARY KEY,
        FOREIGN KEY(Counselor_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE STUDENT
        (Student_ID             INT             PRIMARY KEY,
        Counselor_ID            INT,
        FOREIGN KEY(Counselor_ID) REFERENCES COUNSELOR(Counselor_ID)
        ON DELETE SET NULL ON UPDATE CASCADE,
        FOREIGN KEY(Student_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE DEAN
        (Dean_ID                INT             PRIMARY KEY,
        FOREIGN KEY(Dean_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ENROLLED_COURSES
```

```
        (Student_ID                 INT             NOT NULL,
         Section_ID                 INT             NOT NULL,
        PRIMARY KEY (Student_ID, Section_ID),
        FOREIGN KEY(Section_ID) REFERENCES SECTION(Section_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Student_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE SECTION
        (Course_ID          VARCHAR(12) ,
         Section_ID                     INT             AUTO_INCREMENT NOT NULL
PRIMARY KEY,
        Section_Number      CHAR(3)             NOT NULL,
         Professor_ID                   INT,
         Max_students                   INT,
        Current_students_taking         INT,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE  CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Professor_ID) REFERENCES USER(User_ID)
ON DELETE SET NULL ON UPDATE CASCADE
        );



CREATE TABLE COURSE
        (Course_ID                  VARCHAR(12)             NOT NULL PRIMARY KEY,
         Name                       VARCHAR(50)             NOT NULL,
         Credit_Hours               INT                     NOT NULL,
         D_Name                     VARCHAR(4)              NOT NULL,
        Corequisite_ID              VARCHAR(12),
        FOREIGN KEY(Corequisite_ID) REFERENCES COURSE(Course_ID)
        ON DELETE  SET NULL ON UPDATE CASCADE,
        FOREIGN KEY(D_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE RESTRICT ON UPDATE CASCADE);

CREATE TABLE PREREQUISITES
        (Course_ID                          VARCHAR(12)             NOT NULL,
         REQUIRED_GRADE                     VARCHAR(2)              NOT NULL,
         Prerequisite_ID                    VARCHAR(12)             NOT NULL PRIMARY
KEY,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Prerequisite_ID) REFERENCES COURSE(Course_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ACADEMIC_DEPARTMENT
        (D_Name                 VARCHAR(4)              NOT NULL PRIMARY KEY,
        Dean_ID                 INT,
        FOREIGN KEY(Dean_ID) REFERENCES DEAN(Dean_ID)
        ON DELETE SET NULL
        ON UPDATE CASCADE);

CREATE TABLE HAS_TAKEN
        (Student_ID                     INT                     ,
        Course_ID                       VARCHAR(12)             ,
         Grade                  VARCHAR(2)              NOT NULL,
PRIMARY KEY(Student_ID, Course_ID),
        FOREIGN KEY(Student_ID) REFERENCES STUDENT(Student_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE RESTRICT ON UPDATE CASCADE);


CREATE TABLE USER_ACCOUNT
        (Username                       VARCHAR(15)             ,
         Password_Hash                  VARCHAR(100)            NOT NULL,
        User_ID                         INT,
        PRIMARY KEY(Username),
        FOREIGN KEY(User_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ADMINISTRATOR
(Admin_ID               INT             PRIMARY KEY,
PRIMARY KEY(Admin_ID),
FOREIGN KEY(Admin_ID) REFERENCES USER(User_ID)
ON DELETE CASCADE ON UPDATE CASCADE);
```

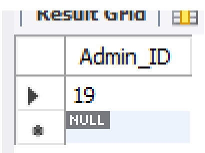
<u>ALTER Commands</u>

```
ALTER TABLE USER
ADD FOREIGN KEY(Dept_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE SET NULL
        ON UPDATE CASCADE
```

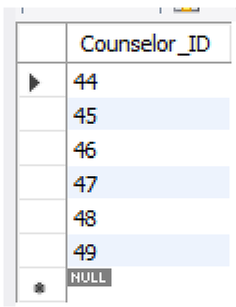
Snapshots of Tables with DBMS



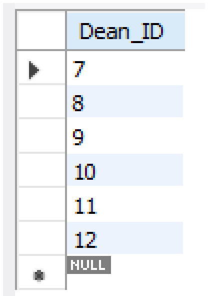
D_Name	Dean_ID
CS	7
SOM	8
BBS	9
A&H	10
IS	11
NSM	12
NULL	NULL



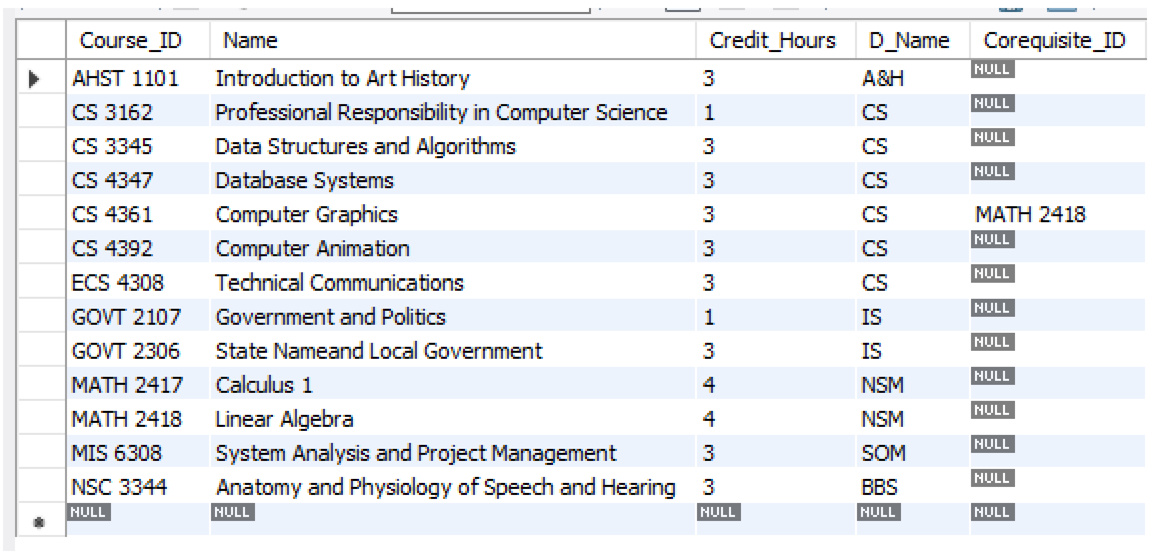
Admin_ID
19
NULL



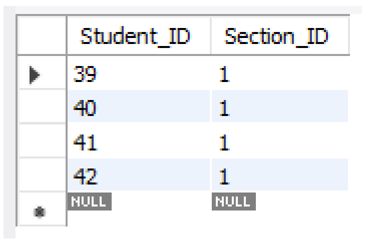
Counselor_ID
44
45
46
47
48
49
NULL



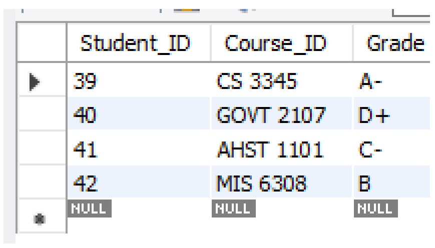
Dean_ID
7
8
9
10
11
12
NULL



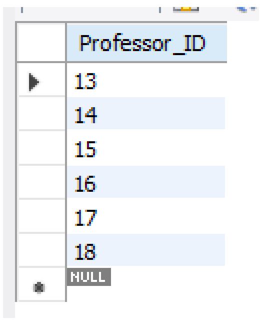
Course_ID	Name	Credit_Hours	D_Name	Corequisite_ID
AHST 1101	Introduction to Art History	3	A&H	NULL
CS 3162	Professional Responsibility in Computer Science	1	CS	NULL
CS 3345	Data Structures and Algorithms	3	CS	NULL
CS 4347	Database Systems	3	CS	NULL
CS 4361	Computer Graphics	3	CS	MATH 2418
CS 4392	Computer Animation	3	CS	NULL
ECS 4308	Technical Communications	3	CS	NULL
GOVT 2107	Government and Politics	1	IS	NULL
GOVT 2306	State Nameand Local Government	3	IS	NULL
MATH 2417	Calculus 1	4	NSM	NULL
MATH 2418	Linear Algebra	4	NSM	NULL
MIS 6308	System Analysis and Project Management	3	SOM	NULL
NSC 3344	Anatomy and Physiology of Speech and Hearing	3	BBS	NULL
NULL	NULL	NULL	NULL	NULL



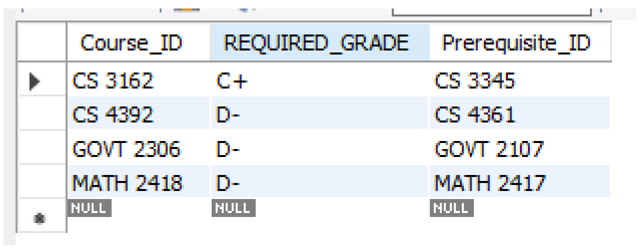
Student_ID	Section_ID
39	1
40	1
41	1
42	1
NULL	NULL



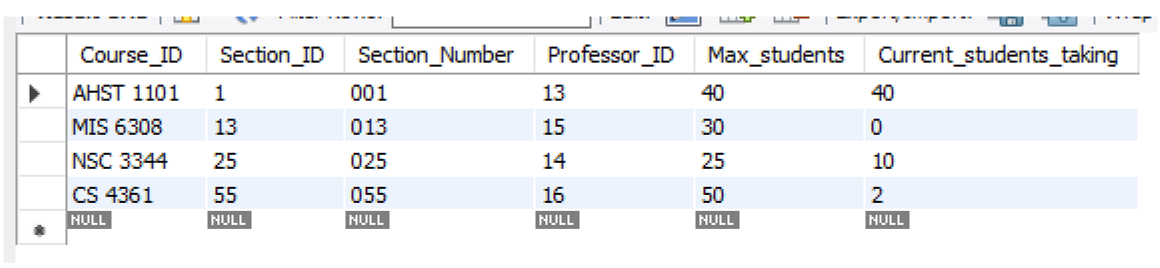
Student_ID	Course_ID	Grade
39	CS 3345	A-
40	GOVT 2107	D+
41	AHST 1101	C-
42	MIS 6308	B
NULL	NULL	NULL



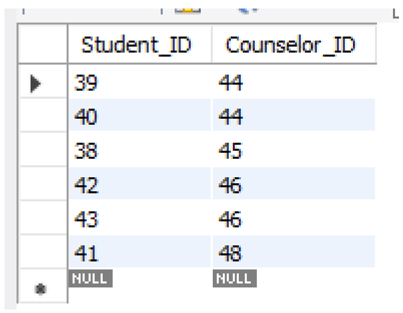
Professor_ID
13
14
15
16
17
18
NULL



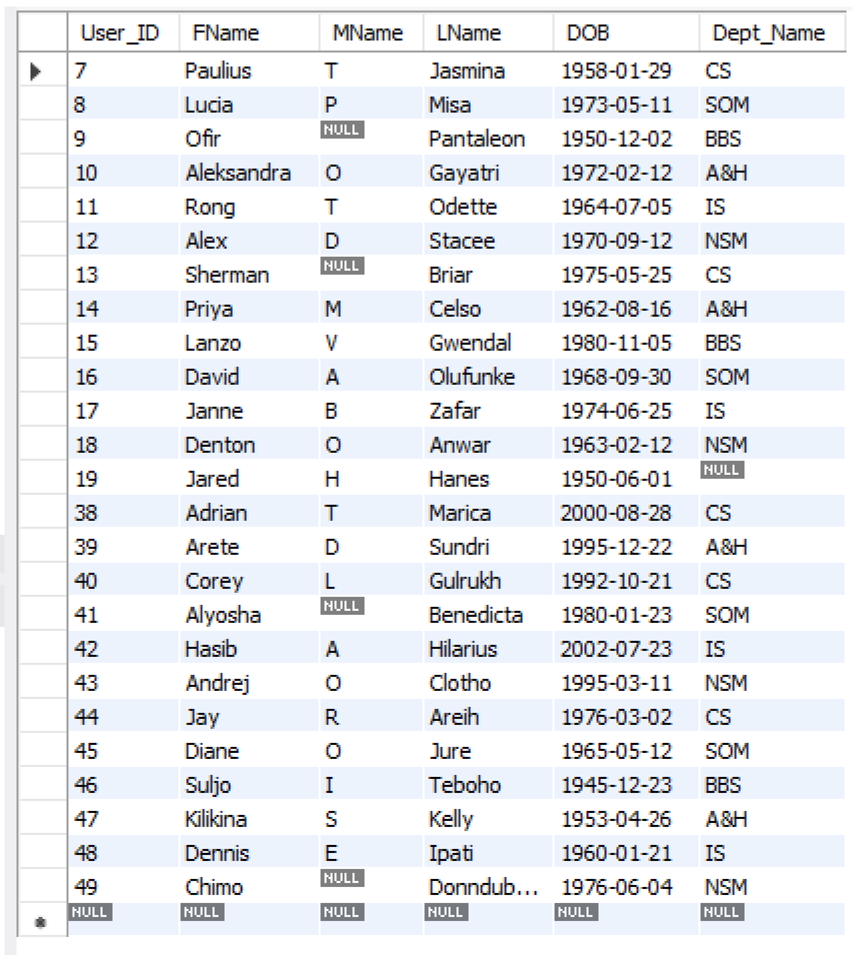
Course_ID	REQUIRED_GRADE	Prerequisite_ID
CS 3162	C+	CS 3345
CS 4392	D-	CS 4361
GOVT 2306	D-	GOVT 2107
MATH 2418	D-	MATH 2417
NULL	NULL	NULL



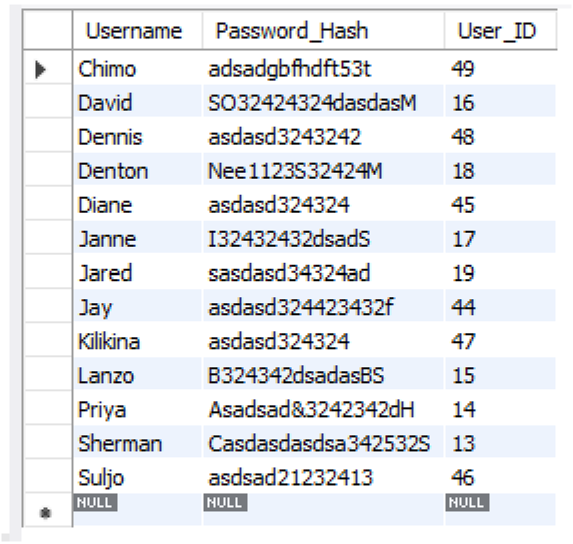
Course_ID	Section_ID	Section_Number	Professor_ID	Max_students	Current_students_taking
AHST 1101	1	001	13	40	40
MIS 6308	13	013	15	30	0
NSC 3344	25	025	14	25	10
CS 4361	55	055	16	50	2
NULL	NULL	NULL	NULL	NULL	NULL



Student_ID	Counselor_ID
39	44
40	44
38	45
42	46
43	46
41	48
NULL	NULL



User_ID	FName	MName	LName	DOB	Dept_Name
7	Paulius	T	Jasmina	1958-01-29	CS
8	Lucia	P	Misa	1973-05-11	SOM
9	Ofir	NULL	Pantaleon	1950-12-02	BBS
10	Aleksandra	O	Gayatri	1972-02-12	A&H
11	Rong	T	Odette	1964-07-05	IS
12	Alex	D	Stacee	1970-09-12	NSM
13	Sherman	NULL	Briar	1975-05-25	CS
14	Priya	M	Celso	1962-08-16	A&H
15	Lanzo	V	Gwendal	1980-11-05	BBS
16	David	A	Olufunke	1968-09-30	SOM
17	Janne	B	Zafar	1974-06-25	IS
18	Denton	O	Anwar	1963-02-12	NSM
19	Jared	H	Hanes	1950-06-01	NULL
38	Adrian	T	Marica	2000-08-28	CS
39	Arete	D	Sundri	1995-12-22	A&H
40	Corey	L	Gulrukh	1992-10-21	CS
41	Alyosha	NULL	Benedicta	1980-01-23	SOM
42	Hasib	A	Hilarius	2002-07-23	IS
43	Andrej	O	Clotho	1995-03-11	NSM
44	Jay	R	Areih	1976-03-02	CS
45	Diane	O	Jure	1965-05-12	SOM
46	Suljo	I	Teboho	1945-12-23	BBS
47	Kilikina	S	Kelly	1953-04-26	A&H
48	Dennis	E	Ipati	1960-01-21	IS
49	Chimo	NULL	Donndub...	1976-06-04	NSM
NULL	NULL	NULL	NULL	NULL	NULL



Username	Password_Hash	User_ID
Chimo	adsadgbfhdft53t	49
David	SO32424324dasdasM	16
Dennis	asdasd3243242	48
Denton	Nee1123S32424M	18
Diane	asdasd324324	45
Janne	I32432432dsadS	17
Jared	sasdasd34324ad	19
Jay	asdasd324423432f	44
Kilikina	asdasd324324	47
Lanzo	B324342dsadasBS	15
Priya	Asadsad&3242342dH	14
Sherman	Casdasdasdsa342532S	13
Suljo	asdsad21232413	46
NULL	NULL	NULL

Triggers

```
DELIMITER |

CREATE TRIGGER PREVENT_COREQ_CYCLE_INSERT
BEFORE INSERT
ON COURSE
FOR EACH ROW

BEGIN

IF EXISTS ( SELECT *
                FROM COURSE
                WHERE Corequisite_ID = NEW.Course_ID
                AND      NEW.Corequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
END IF;

END |

DELIMITER ;


DELIMITER |

CREATE TRIGGER PREVENT_COREQ_CYCLE_UPDATE
BEFORE UPDATE
ON COURSE
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
                FROM COURSE
                WHERE Corequisite_ID = NEW.Course_ID
                AND      NEW.Corequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
END IF;
END |

DELIMITER ;
DELIMITER |

CREATE TRIGGER PREVENT_PREREQ_CYCLE_INSERT
BEFORE INSERT
ON PREREQUISITES
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
                FROM PREREQUISITE
                WHERE Prerequsite_ID = NEW.Course_ID
                AND       NEW.Prerequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
        SET NEW.Prerequisite_ID = NULL;
END IF;
END |

DELIMITER ;


DELIMITER |

CREATE TRIGGER PREVENT_PREREQ_CYCLE_UPDATE
BEFORE UPDATE
ON PREREQUISITES
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
                FROM PREREQUISITES
                WHERE Prerequsite_ID = NEW.Course_ID
                AND       NEW.Prerequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
        SET NEW.Prerequisite_ID = NULL;
END IF;
END |

DELIMITER ;


DELIMITER |
CREATE TRIGGER TAKEN_PREREQS_INSERT
BEFORE INSERT
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF(
    NOT EXISTS ( SELECT *
                 FROM SECTION S, PREREQUISITES P, HAS_TAKEN HT
                 WHERE S.Section_ID = NEW.Section_ID
                 AND P.Course_ID = S.Course_ID
                 AND HT.Course_ID = P.Prerequisite_ID
                 AND HT.Grade >= P.REQUIRED_GRADE )

   AND

  EXISTS ( SELECT *
                 FROM SECTION S, PREREQUISITES P
                 WHERE S.Section_ID = NEW.Section_ID
                 AND P.Course_ID = S.Course_ID )
   )
THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF ;

END |

DELIMITER ;


DELIMITER |

CREATE TRIGGER TAKEN_PREREQS_UPDATE
BEFORE UPDATE
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
    NOT EXISTS ( SELECT *
                 FROM SECTION S, PREREQUISITES P, HAS_TAKEN HT
                 WHERE S.Section_ID = NEW.Section_ID
                 AND P.Course_ID = S.Course_ID
                 AND HT.Course_ID = P.Prerequisite_ID
```

```
                AND HT.Grade >= P.REQUIRED_GRADE
                )

   AND


EXISTS ( SELECT *
                FROM SECTION S, PREREQUISITES P
                WHERE S.Section_ID = NEW.Section_ID
                AND P.Course_ID = S.Course_ID )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF;

END |

DELIMITER ;


DELIMITER |

CREATE TRIGGER TAKEN_COREQS_UPDATE
BEFORE UPDATE
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
    NOT EXISTS ( SELECT *
                FROM SECTION S, COURSE C, HAS_TAKEN HT
                WHERE S.Section_ID = NEW.Section_ID
                AND C.Course_ID = S.Course_ID
                AND HT.Course_ID = C.Corequisite_ID )

   AND


EXISTS ( SELECT *
                FROM SECTION S, COURSE C
                WHERE S.Section_ID = NEW.Section_ID
                AND C.Course_ID = S.Course_ID
```

```
                  AND C.Corequisite_ID IS NULL )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF;

END |

DELIMITER ;

DELIMITER |

CREATE TRIGGER TAKEN_COREQS_INSERT
BEFORE INSERT
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
    NOT EXISTS ( SELECT *
                  FROM SECTION S, COURSE C, HAS_TAKEN HT
                  WHERE S.Section_ID = NEW.Section_ID
                  AND C.Course_ID = S.Course_ID
                  AND HT.Course_ID = C.Corequisite_ID )

   AND

  EXISTS ( SELECT *
                  FROM SECTION S, COURSE C
                  WHERE S.Section_ID = NEW.Section_ID
                  AND C.Course_ID = S.Course_ID
                  AND C.Corequisite_ID IS NULL )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;
END IF;

END |

DELIMITER ;
```

Data Population Commands

(Inserting Users)

```
INSERT INTO USER
VALUES
(NULL, "Sherman", NULL, "Briar", '1975-05-25', "CS"),
(NULL, "Priya", "M", "Celso", '1962-08-16', "A&H"),
(NULL, "Lanzo", "V", "Gwendal", '1980-11-05', "BBS"),
(NULL, "David", "A", "Olufunke", '1968-09-30', "SOM"),
(NULL, "Janne", "B", "Zafar", '1974-06-25', "IS"),
(NULL, "Denton", "O", "Anwar", '1963-02-12', "NSM")
(NULL, “Jared”, “H”, “Hanes”, 1950-06-01”, NULL);
(NULL, "Jay", 'R', 'Areih', '1976-03-02', 'CS'),
(NULL, 'Diane', 'O', 'Jure', '1965-05-12', 'SOM'),
(NULL, 'Suljo', 'I', 'Teboho', '1945-12-23', 'BBS'),
(NULL, 'Kilikina', 'S', 'Kelly', '1953-04-26', 'A&H'),
(NULL, 'Dennis', 'E', 'Ipati', '1960-01-21', 'IS'),
(NULL, 'Chimo', NULL, 'Donndubhan', '1976-06-04', 'NSM');
```

(Inserting Administrator)

```
INSERT INTO ADMINISTRATOR
VALUES
(19);
```

(Inserting professors)

```
INSERT INTO PROFESSOR
VALUES
(13),
(14),
(15),
(16),
(17),
(18);
```

(Inserting Counselors)

```
INSERT INTO COUNSELOR
VALUES (44), (45), (46), (47), (48), (49);
```

(Inserting Students)

```
INSERT INTO STUDENT
VALUES
```

```
(38, 45),
(39, 44),
(40, 44),
(41, 48),
(42, 46),
(43, 46);
```

(Inserting Deans)

```
INSERT INTO USER (FName, MName, LName, DOB, Dept_name)
VALUES ("Paulius", "T", "Jasmina", "1958-01-29", "CS"),
("Lucia", "P", "Misa", "1973-05-11", "SOM"),
("Ofir", NULL, "Pantaleon", "1950-12-02", "BBS"),
( "Aleksandra", "O", "Gayatri", "1972-02-12", "A&H"),
("Rong", "T", "Odette", "1964-07-05", "IS"),
("Alex", "D", "Stacee", "1970-09-12", "NSM");


INSERT INTO DEAN (Dean_ID)
VALUES (7),
(8),
(9),
(10),
(11),
(12);

UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=7
WHERE D_Name="CS";
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=8
WHERE D_Name="SOM";
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=9
WHERE D_Name="BBS";
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=10
WHERE D_Name="A&H";
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=11
WHERE D_Name="IS";
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=12
WHERE D_Name="NSM";
```

(Inserting Enrolled_Courses)

```
INSERT INTO ENROLLED_COURSES (Student_ID, Section_ID)
VALUES
(42, 1),
(39, 1),
(40, 1),
(41, 1);
```

(Inserting Course)

```
INSERT INTO COURSE (Course_ID, Name, Credit_Hours, D_Name, Corequisite_ID)
VALUES
("GOVT 2306", "State Nameand Local Government", 3, "IS", NULL),
("AHST 1101", "Introduction to Art History", 3, "A&H", NULL),
("CS 4347", "Database Systems", 3, "CS", NULL),
("ECS 4308", "Technical Communications", 3, "CS", NULL),
("MIS 6308", "System Analysis and Project Management", 3, "SOM", NULL),
("NSC 3344", "Anatomy and Physiology of Speech and Hearing", 3, "BBS", NULL),
("CS 4392", "Computer Animation", 3, "CS", NULL),
("MATH 2418", "Linear Algebra", 4, "NSM", NULL),
("CS 4361", "Computer Graphics", 3, "CS", "MATH 2418"),
("MATH 2417", "Calculus 1", 4, "NSM", NULL),
("CS 3162", "Professional Responsibility in Computer Science", 1, "CS", NULL),
("CS 3345", "Data Structures and Algorithms", 3, "CS", NULL),
("GOVT 2107", "Government and Politics", 1, "IS", NULL);
```

(Inserting Section)

```
INSERT INTO SECTION
VALUES
("AHST 1101", 1, "001", 13, 40, 40),
("CS 4361", 55,  "055", 16, 50, 2),
("MIS 6308", 13, "013", 15, 30, 0),
("NSC 3344", 25, "025", 14, 25, 10);
```

(Inserting Prerequisites)

```
INSERT INTO PREREQUISITES
VALUES
("CS 4392", "D-", "CS 4361"),
("MATH 2418", "D-", "MATH 2417"),
("CS 3162", "C+", "CS 3345"),
        ("GOVT 2306", "D-", "GOVT 2107");
```

(Inserting Academic_Department)

```
INSERT INTO Academic_Department (D_Name, Dean_ID)
VALUES (‘CS’, 582749),
        (‘BBS’, 695738),
        (‘A&H’, 692749),=
        (‘SOM’, 040586),
        (‘IS’, 599040),
        (‘NSM’, 860030);
```

(Inserting Has_Taken)

```
INSERT INTO HAS_TAKEN (Student_ID, Course_ID, Grade)
        VALUES
        (39, "CS 3345", "A-"),
        (40, "GOVT 2107", "D+"),
        (41, "AHST 1101", "C-"),
        (42, "MIS 6308", "B");
```

(Inserting User_Account)

```
INSERT INTO User_Account  (Username, Password_Hash, User_ID)
VALUES (‘Briar_She02’, ‘d9f8s@HD0!$cdf’, 195628),
        (‘Priya_MCel00’, ‘02hr30H309hxF@H’, 581650),
        (‘Lanzo_VGwe48’, ‘2MFJDS0!nf!$%mf’, 937659),
        (‘David_AOlu’, ‘*j123FH!@hjsSDF’, 703758),
        (‘Jay_RAri’, ‘SD23fn0#!%mds’,  860546),
        (‘Diane_OJur’, ‘62mFS34$^Gk#$’, 472650);
```

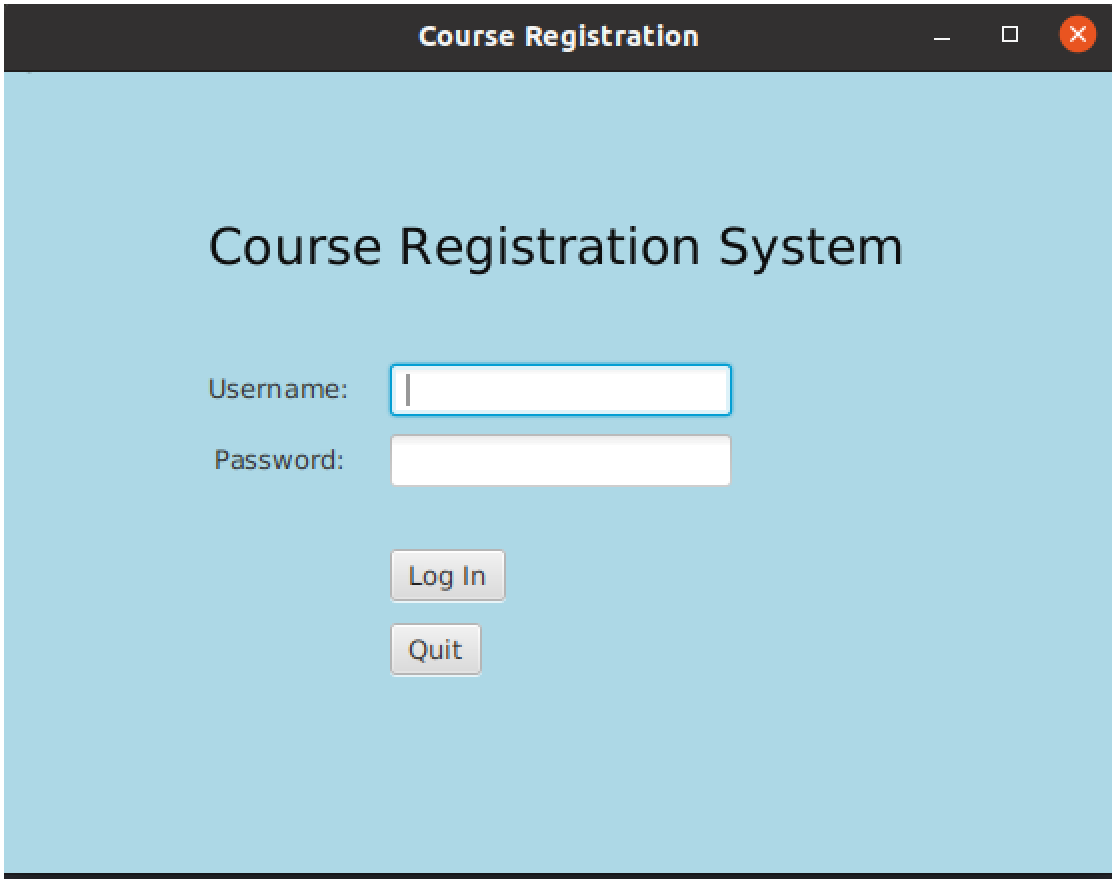
VIEW COMMANDS:

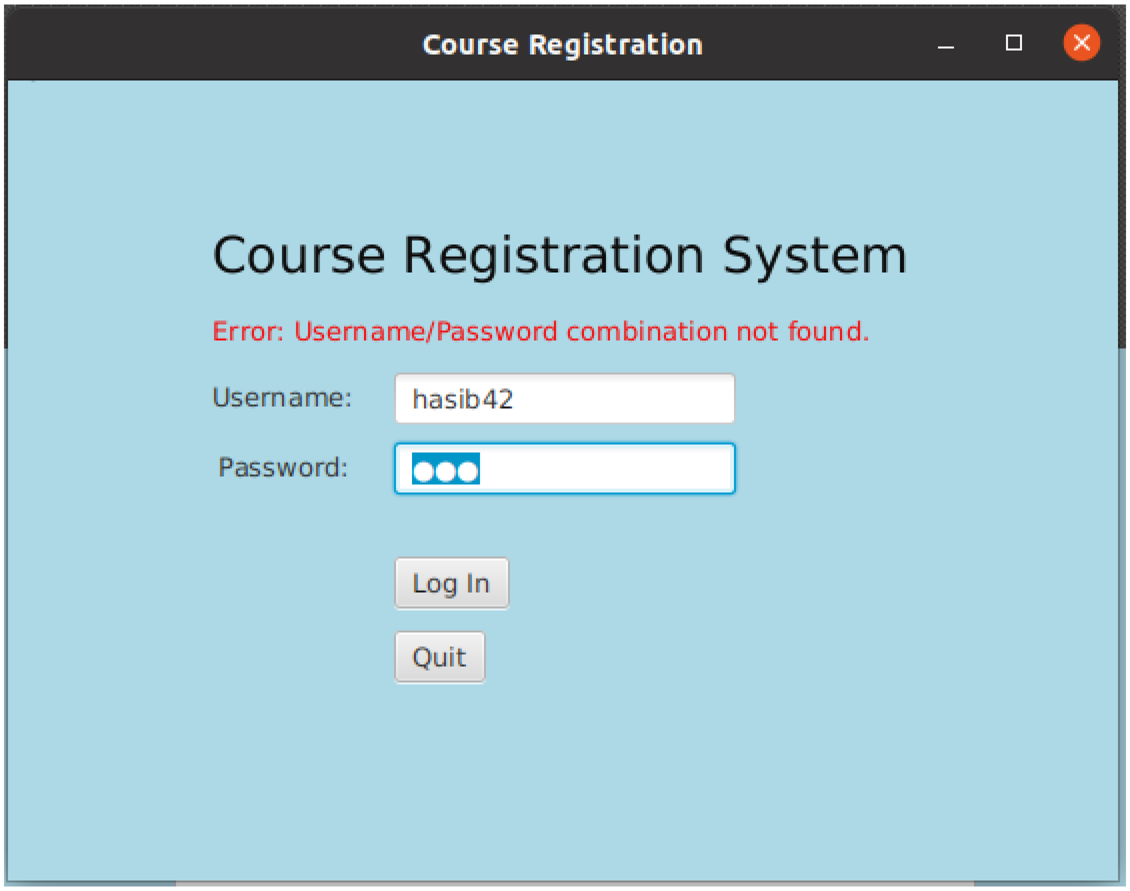
```
CREATE VIEW Students
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name, Counselor_ID FROM
USER,STUDENT
WHERE User_ID IN (SELECT Student_ID FROM STUDENT) AND Student_ID=User_ID;

CREATE VIEW Deans
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Dean_ID FROM DEAN);

CREATE VIEW Professors
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Professor_ID FROM PROFESSOR);

CREATE VIEW Counselors
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Counselor_ID FROM COUNSELOR)
```

Application/System Demonstration

The system is installed and ran with Java. The users of the system must have MySQL Connector for Java installed, and the appropriate libraries, as will be provided in the Appendix.

Login screen:



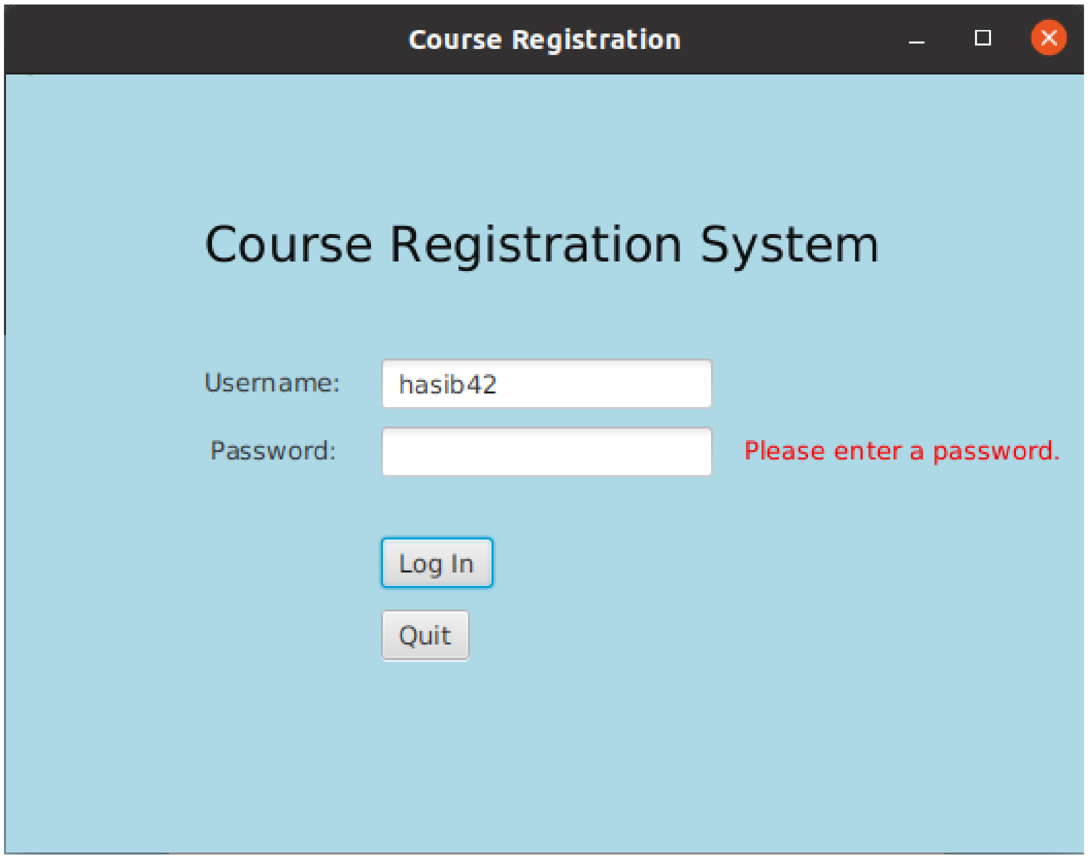

Course Registration

Course Registration System

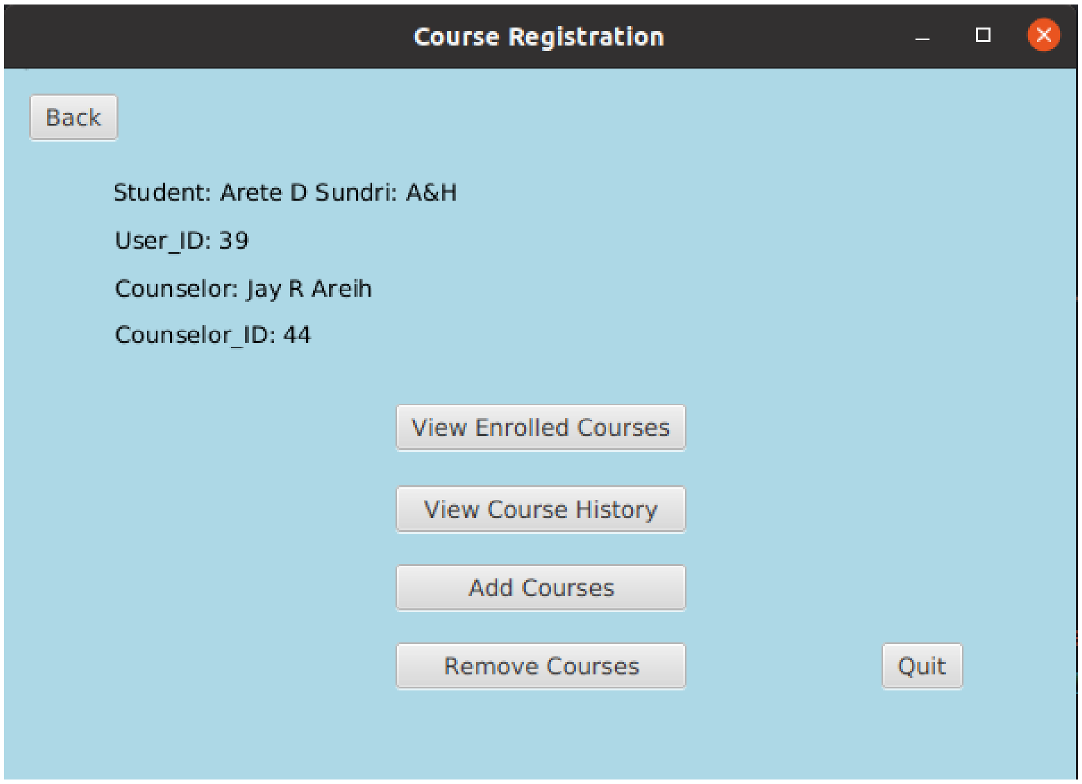
Error: Username/Password combination not found.

Username: hasib42

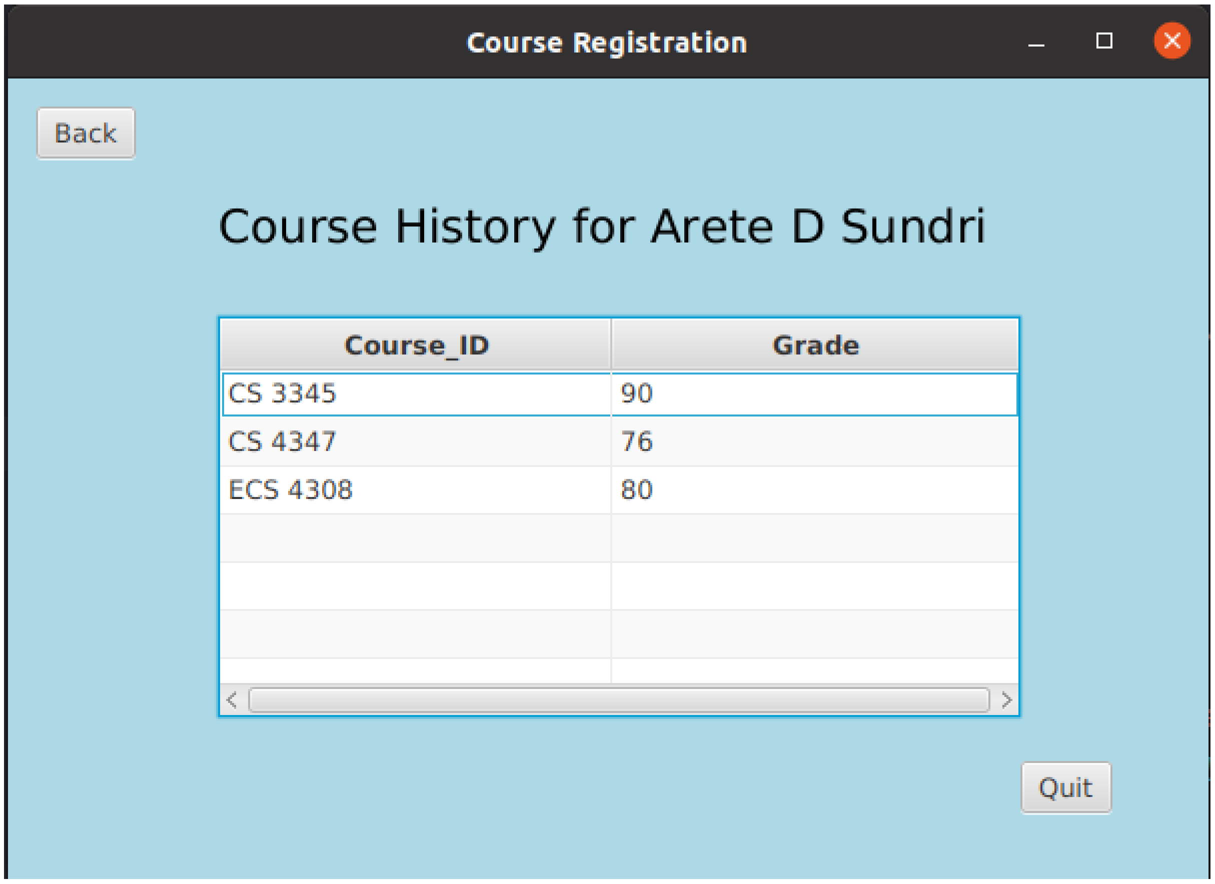
Password: ●●●



Student view:



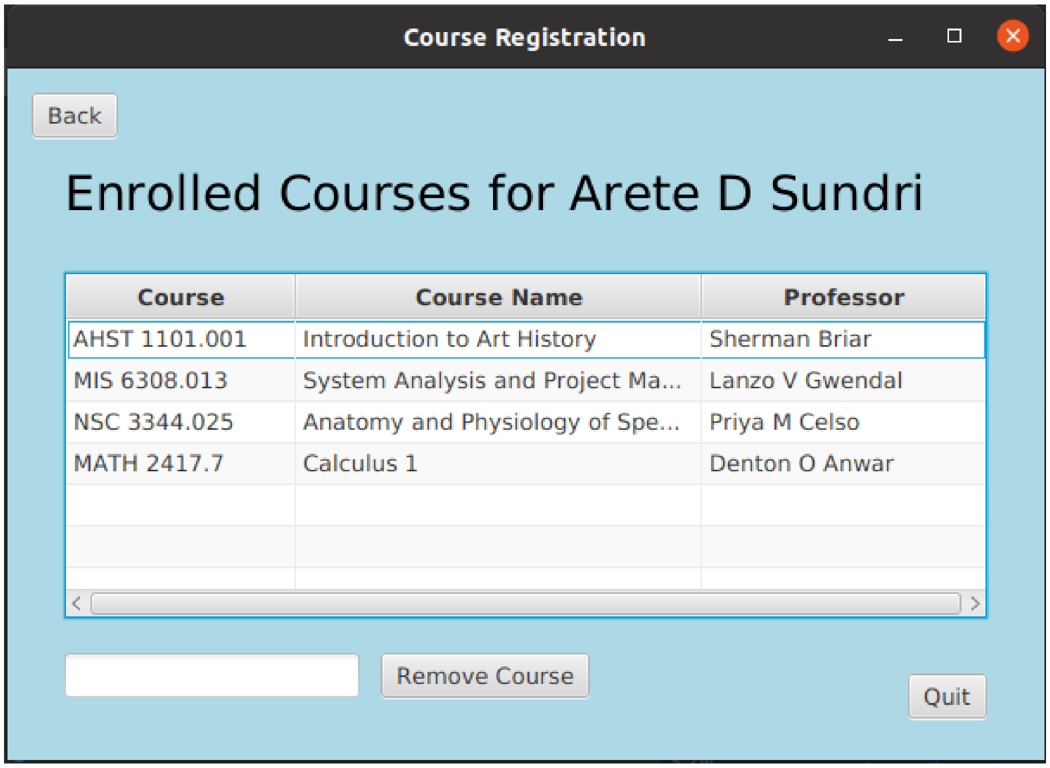
Course History:



Course History for Arete D Sundri

Course_ID	Grade
CS 3345	90
CS 4347	76
ECS 4308	80

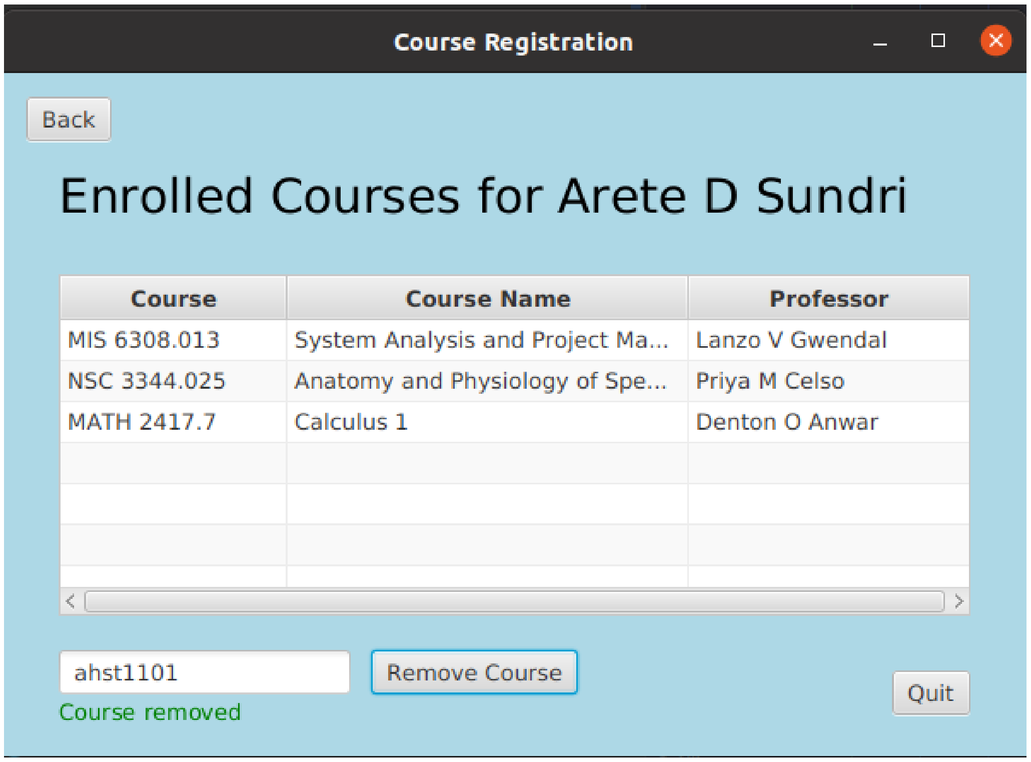
Removing courses:
Initial:



Enrolled Courses for Arete D Sundri

Course	Course Name	Professor
AHST 1101.001	Introduction to Art History	Sherman Briar
MIS 6308.013	System Analysis and Project Ma...	Lanzo V Gwendal
NSC 3344.025	Anatomy and Physiology of Spe...	Priya M Celso
MATH 2417.7	Calculus 1	Denton O Anwar

Removing AHST 1101.001



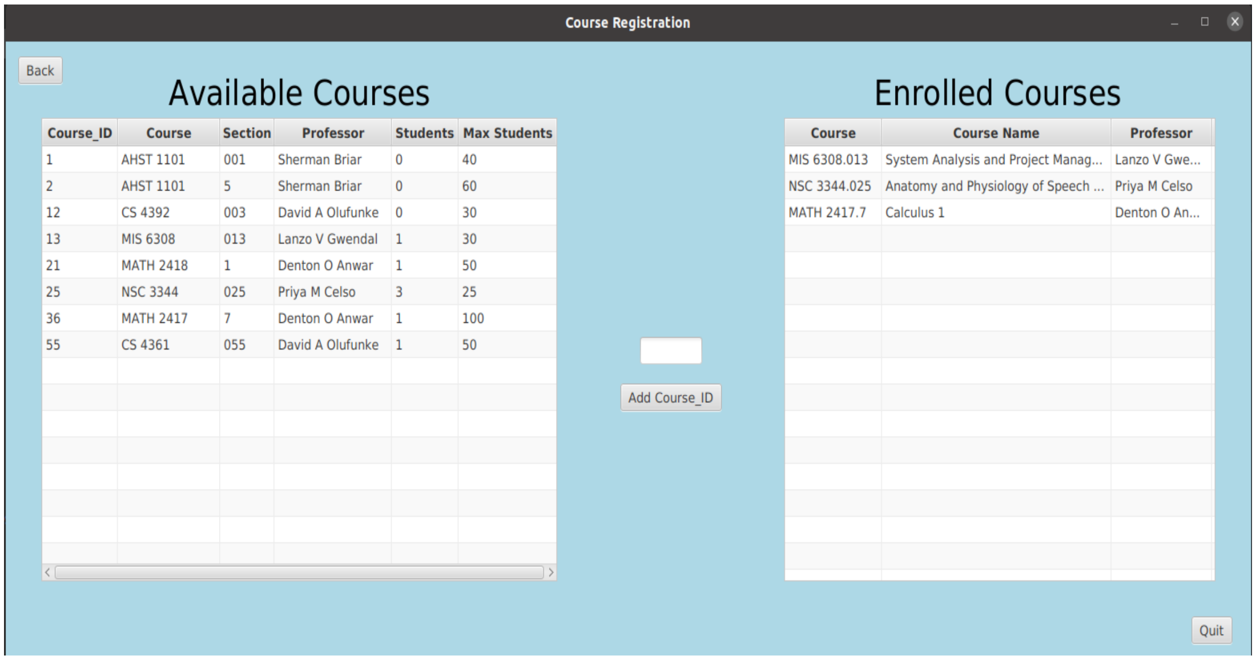
Enrolled Courses for Arete D Sundri

Course	Course Name	Professor
MIS 6308.013	System Analysis and Project Ma...	Lanzo V Gwendal
NSC 3344.025	Anatomy and Physiology of Spe...	Priya M Celso
MATH 2417.7	Calculus 1	Denton O Anwar

ahst1101

Course removed

Adding Courses:
Initial



Course Registration

Back

Available Courses

Course_ID	Course	Section	Professor	Students	Max Students
1	AHST 1101	001	Sherman Briar	0	40
2	AHST 1101	5	Sherman Briar	0	60
12	CS 4392	003	David A Olufunke	0	30
13	MIS 6308	013	Lanzo V Gwendal	1	30
21	MATH 2418	1	Denton O Anwar	1	50
25	NSC 3344	025	Priya M Celso	3	25
36	MATH 2417	7	Denton O Anwar	1	100
55	CS 4361	055	David A Olufunke	1	50

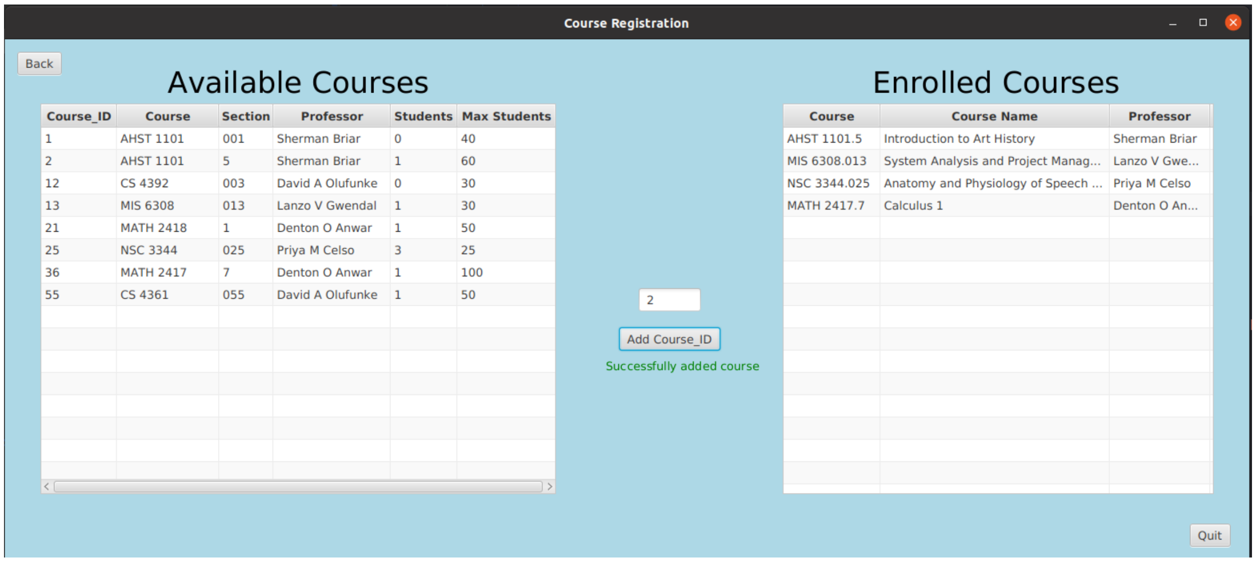
Add Course_ID

Enrolled Courses

Course	Course Name	Professor
MIS 6308.013	System Analysis and Project Manag...	Lanzo V Gwe...
NSC 3344.025	Anatomy and Physiology of Speech ...	Priya M Celso
MATH 2417.7	Calculus 1	Denton O An...

Quit

Adding Course_ID 2:



Course Registration

Back

Available Courses

Course_ID	Course	Section	Professor	Students	Max Students
1	AHST 1101	001	Sherman Briar	0	40
2	AHST 1101	5	Sherman Briar	1	60
12	CS 4392	003	David A Olufunke	0	30
13	MIS 6308	013	Lanzo V Gwendal	1	30
21	MATH 2418	1	Denton O Anwar	1	50
25	NSC 3344	025	Priya M Celso	3	25
36	MATH 2417	7	Denton O Anwar	1	100
55	CS 4361	055	David A Olufunke	1	50

2

Add Course_ID

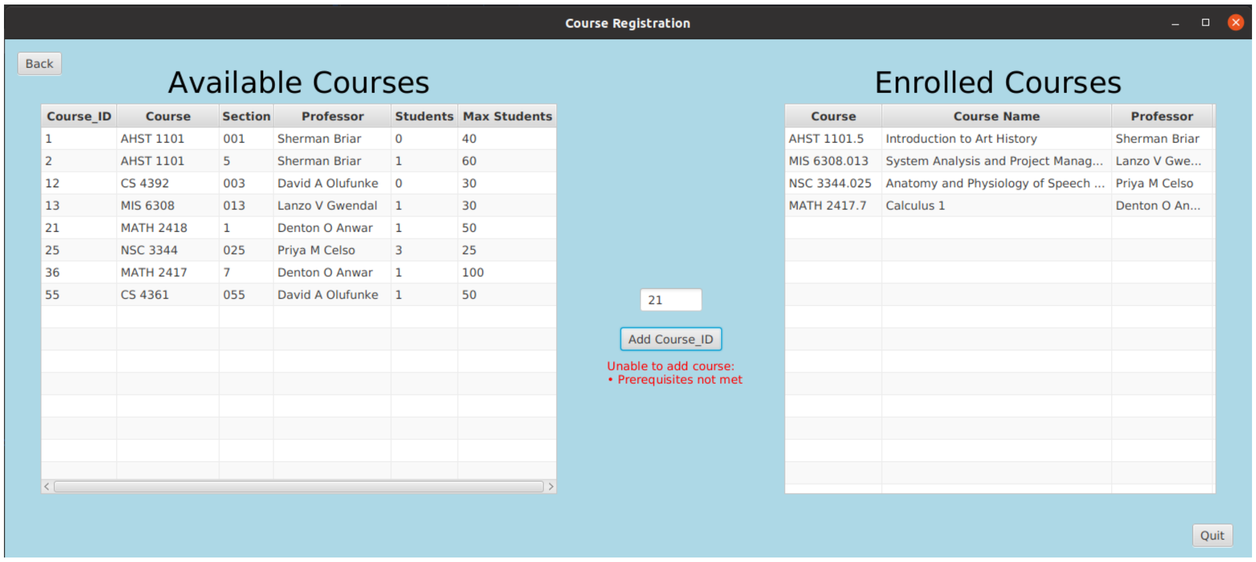
Successfully added course

Enrolled Courses

Course	Course Name	Professor
AHST 1101.5	Introduction to Art History	Sherman Briar
MIS 6308.013	System Analysis and Project Manag...	Lanzo V Gwe...
NSC 3344.025	Anatomy and Physiology of Speech ...	Priya M Celso
MATH 2417.7	Calculus 1	Denton O An...

Quit

Attempting to add course where prereqs not met:



Available Courses

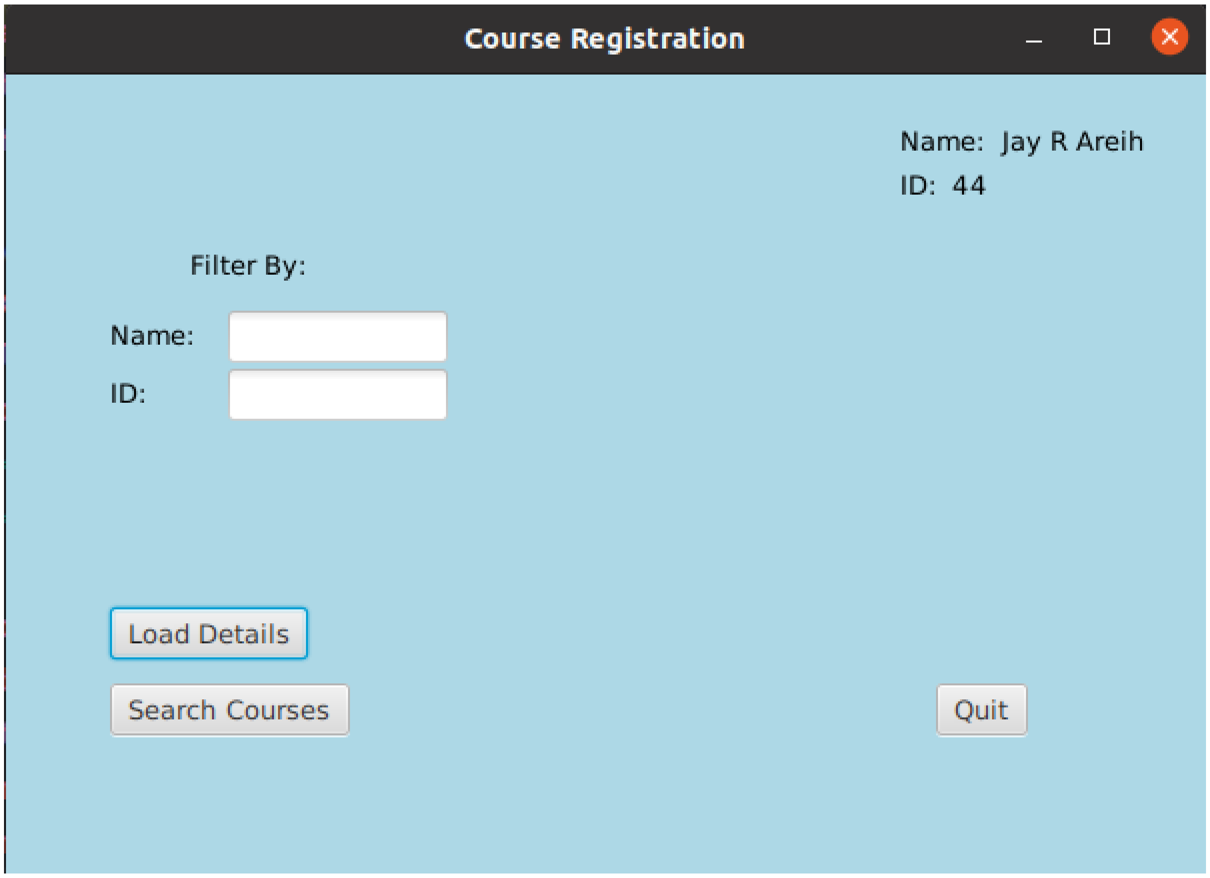
Course_ID	Course	Section	Professor	Students	Max Students
1	AHST 1101	001	Sherman Briar	0	40
2	AHST 1101	5	Sherman Briar	1	60
12	CS 4392	003	David A Olufunke	0	30
13	MIS 6308	013	Lanzo V Gwendal	1	30
21	MATH 2418	1	Denton O Anwar	1	50
25	NSC 3344	025	Priya M Celso	3	25
36	MATH 2417	7	Denton O Anwar	1	100
55	CS 4361	055	David A Olufunke	1	50

Unable to add course:
• Prerequisites not met

Enrolled Courses

Course	Course Name	Professor
AHST 1101.5	Introduction to Art History	Sherman Briar
MIS 6308.013	System Analysis and Project Manag...	Lanzo V Gwe...
NSC 3344.025	Anatomy and Physiology of Speech ...	Priya M Celso
MATH 2417.7	Calculus 1	Denton O An...

Counselor View:

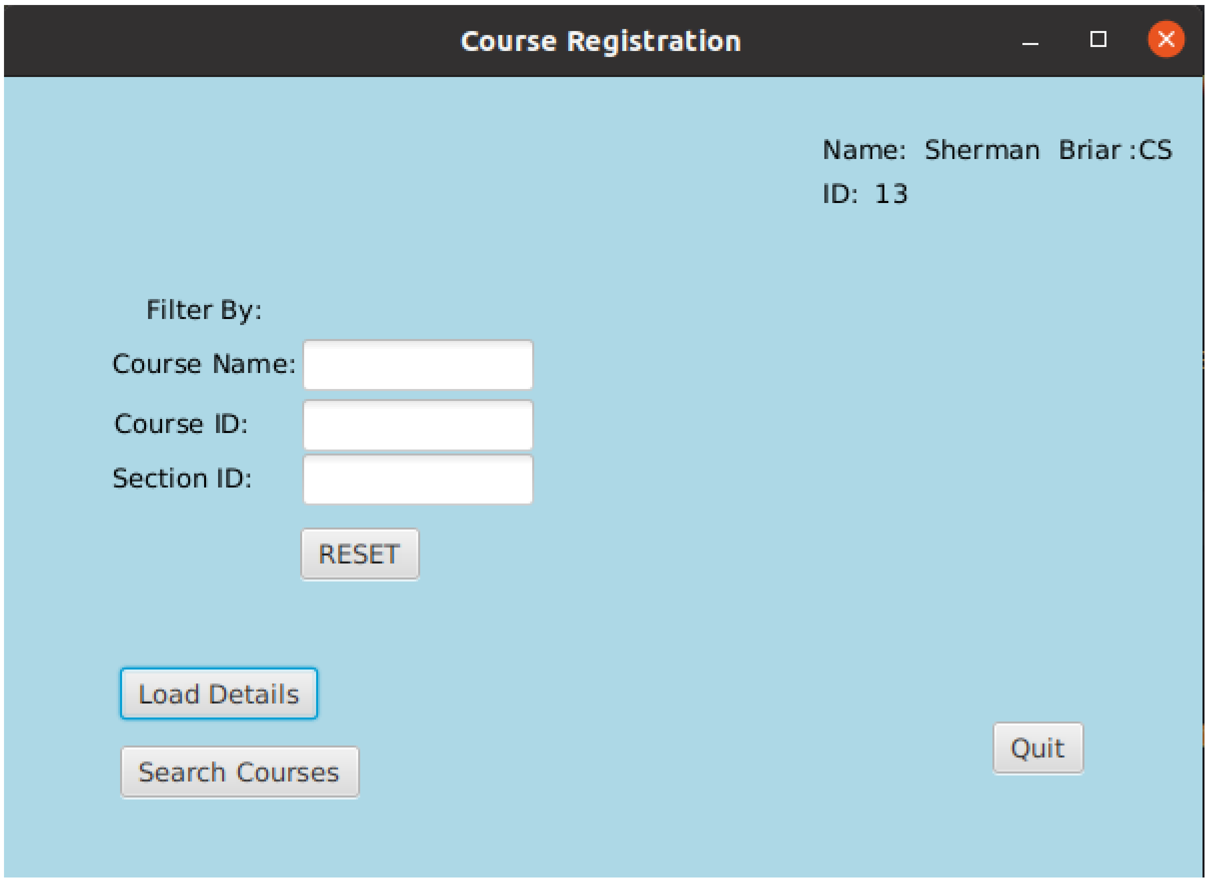


Name: Jay R Areih
ID: 44

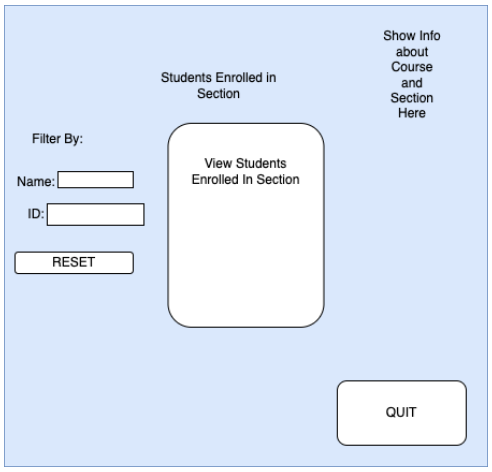
The search courses window will be similar to the add/drop courses window, but there will be no add/drop button. Clicking on a student in the scrollbar should bring up that student’s view.

The reset button will remove all search filters and restore the default layout of students in the scrollbar.

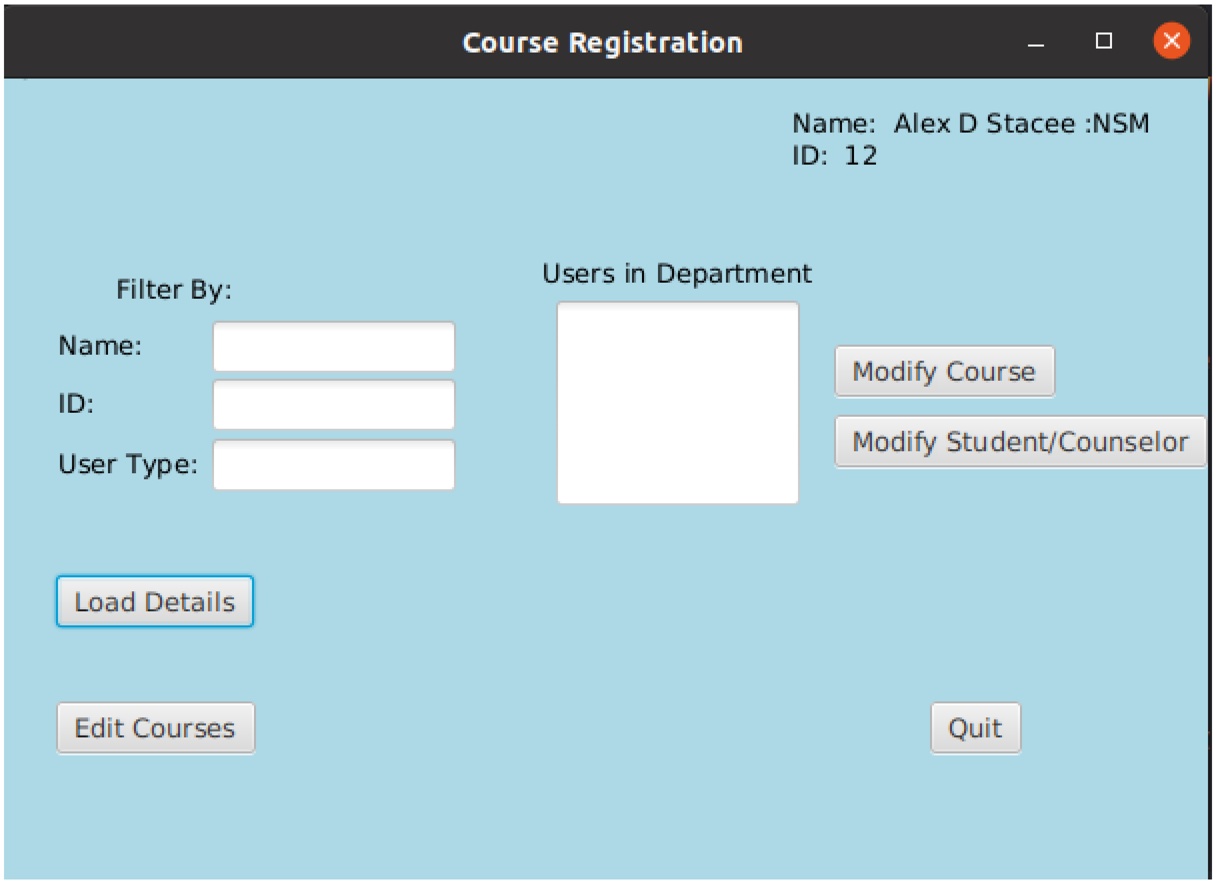
Professor View:



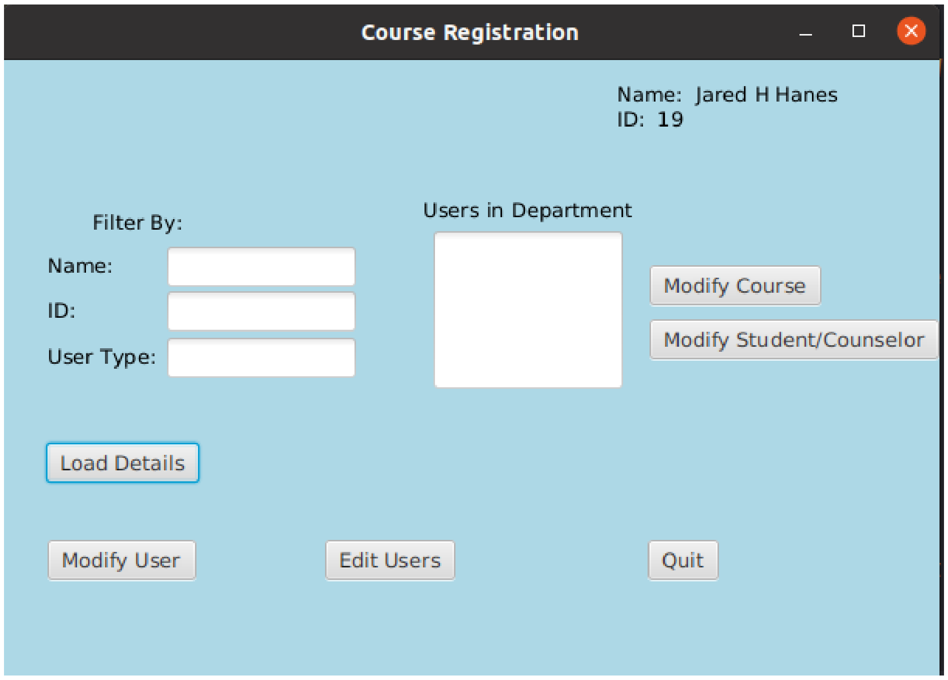
Section View:


Show Info about Course and Section Here
Students Enrolled in Section
Filter By:
Name:
ID:
RESET
View Students Enrolled In Section
QUIT

Dean View:


Course Registration
Name: Alex D Stacee :NSM
ID: 12
Filter By:
Name:
ID:
User Type:
Users in Department
Modify Course
Modify Student/Counselor
Load Details
Edit Courses
Quit

Admin:


Course Registration
Name: Jared H Hanes
ID: 19
Filter By:
Name:
ID:
User Type:
Users in Department
Modify Course
Modify Student/Counselor
Load Details
Modify User
Edit Users
Quit

User Application Interface

Options for menu for everyone:

1) Login
2) View info about themselves
   SELECT *
   FROM USER_ACCOUNT
   WHERE Username = (user’s username)
3) View information about courses (who teaches, how many students, etc.)
   SQL:
   SELECT FName, LName, Max_students, Current_students_taking, Course_ID, Section_Number
   FROM COURSE_REGISTRATION.SECTION, COURSE_REGISTRATION.USER
   WHERE Professor_ID = User_ID and Course_ID = "input course";

   Course id, Section number, professor name from user table, number of students taking, max students
4) Quit

After login if Student:

1) Register for course (search for a course based on input and choose to register or not)
   // ask user for their student id and section id of what they want to register for
   SQL:
   INSERT INTO ENROLLED_COURSES (Student_ID, Section_ID)
   VALUES (x, y)
   UPDATE SECTION
   SET Current_students_taking = Current_students_taking + 1
   WHERE SECTION.Section_ID = ENROLLED_COURSES.Section_ID;

2) View enrolled courses
   SQL:
   SELECT Course_ID, Section_Number, FName, LName
   FROM COURSE_REGISTRATION.USER,
   COURSE_REGISTRATION.SECTION,
   COURSE_REGISTRATION.ENROLLED_COURSES
   WHERE ENROLLED_COURSES.Student_ID = "input id" AND Professor_ID = User_ID AND SECTION.Section_ID = ENROLLED_COURSES.Section_ID;

   Course id, Section number, professor name
3) Unenroll from course (input is section number)
   DELETE FROM ENROLLED_COURSES WHERE Section_ID = (user input)

UPDATE SECTION
SET Current_students_taking = Current_students_taking - 1
WHERE SECTION.Section_ID = ENROLLED_COURSES.Section_ID;

4) View Course History
   SELECT * FROM HAS_TAKEN WHERE Student_ID = (current users id);

After login if Professor:

1) View courses and sections they teach
   SELECT Course_ID, Section_Number, Max_students, Current_students_taking
   FROM COURSE_REGISTRATION.SECTION
   WHERE Professor_ID = "input id";
   Course id, section number, number of students taking and maxpossible
2) Change a student's grade
   UPDATE HAS_TAKEN
   SET Grade = (Input from professor)
   WHERE Student_ID = (Input from professor) AND Course_ID = (Input from professor)
3) Create/Remove a section
   INSERT INTO SECTION (Section_ID, Section_Number, Max_Students, Current_Students_Taking)
   VALUES ((Input from professor))

   DELETE FROM SECTION
   WHERE Section_ID = (user input)

After login if Counselor:

1) View students
   SELECT Student_ID, FName, LName
   FROM COURSE_REGISTRATION.STUDENT,
   COURSE_REGISTRATION.USER
   WHERE STUDENT.Counselor_ID = "input id" AND Student_ID = User_ID;

   Student name and id
   (After picking a student) student info
2) Add or remove students from courses
   2a) Add
   INSERT INTO SECTION (Student_ID, Sec_ID)
   VALUES((Input from counselor))

   UPDATE SECTION
   SET Current_students_taking = Current_students_taking + 1
   WHERE SECTION.Section_ID = ENROLLED_COURSES.Section_ID;

2b) Remove
DELETE FROM STUDENT
WHERE Student_ID = (Input from counselor)

UPDATE SECTION
SET Current_students_taking = Current_students_taking - 1
WHERE SECTION.Section_ID = ENROLLED_COURSES.Section_ID;

3) Have all of the student options for the students that the counselor is in charge of.

After login if Dean:

1) View professors
SELECT Professor_ID, FName, LName
FROM COURSE_REGISTRATION.PROFESSOR,
COURSE_REGISTRATION.USER
WHERE USER.Dept_Name = "input dept" AND Professor_ID = User_ID;

Same stuff as professors can view after picking one
2) View students
SELECT Student_ID, FName, LName
FROM COURSE_REGISTRATION.STUDENT,
COURSE_REGISTRATION.USER
WHERE USER.Dept_Name = "input dept" AND Student_ID = User_ID;

Same stuff students can view after picking a student
3) View counselors
SELECT Counselor_ID, FName, LName
FROM COURSE_REGISTRATION.COUNSELOR,
COURSE_REGISTRATION.USER
WHERE USER.Dept_Name = "input dept" AND Counselor_ID = User_ID;

Counselor info
4) Change what professors teach
4a) Change the section's professor
UPDATE SECTION
SET Professor_ID = (new professor)
WHERE Section_ID = (Input from dean) AND Professor_ID = (old professor)
5) Change counselors for students
UPDATE STUDENT
SET Counselor_ID = (new counselor)
WHERE Student_ID = (input from dean)
6) Create new courses

INSERT INTO COURSE (Course_ID, Name, Credit_Hours, D_Name, Corequisite_ID)
VALUES (Input from dean)

7) Delete courses
   DELETE FROM COURSE WHERE Course_ID = (input)
8) Add new students //For 7-9 they must be input as users beforehand
   INSERT INTO STUDENT
   VALUES (student id and counselor id)
9) Delete student
   DELETE FROM STUDENT WHERE Student_ID = (input)
10) Add new counselors
   INSERT INTO COUNSELOR
   VALUES (counselor id)
11) Delete counselor
   DELETE FROM COUNSELOR WHERE Counselor_ID = (input)
12) Add new professors
   INSERT INTO PROFESSOR
   VALUES (professor id)
13) Delete professor
   DELETE FROM PROFESSOR WHERE Professor_ID = (input)

After login if Database admin

1) View users
   SELECT *
   FROM USER
   NATURAL JOIN USER_ACCOUNT
2) Add users
   INSERT INTO USER
   VALUES(input)
3) Delete users
   DELETE FROM USER WHERE User_ID = (input)

Views

```
CREATE VIEW Students
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name, Counselor_ID FROM
USER,STUDENT
WHERE User_ID IN (SELECT Student_ID FROM STUDENT) AND Student_ID=User_ID;

CREATE VIEW Deans
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Dean_ID FROM DEAN);

CREATE VIEW Professors
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Professor_ID FROM PROFESSOR);

CREATE VIEW Counselors
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Counselor_ID FROM COUNSELOR)
```

Complex Queries:

```
SELECT Section_ID, COUNT(*) AS Students
FROM SECTION
WHERE Course_ID =  'AHST 1101'
GROUP BY Section_ID;
```

Output:

```
Section_ID | Students
1                1
2                1
```

```
SELECT SUM(MAX_STUDENTS) AS Total_Students
FROM SECTION
WHERE Course_ID LIKE 'CS%';
```

Output:

```
Total_Students
80
```

```
SELECT Course_ID, SUM(MAX_STUDENTS) AS Max_Students
FROM SECTION
GROUP BY Course_ID;
```

Output:

```
Course_ID | Max_Students
AHST 1101   100
CS 4361     50
CS 4392     30
MATH 2417   100
MATH 2218   50
MIS 6308    30
NSC 3344    25
```

Conclusion/Future Work

The Course Registration System had to be built from the ground up. Realizing what the application had to do, a database was needed in order to fulfill the application’s needs. To build a database, we had to design a database. That has been the extent of this project summed up from a high level view. The Course Registration System now works as anybody would expect a system such as it to work, by utilizing a database storing long term, persistent data about the entities the Course Registration Systems needs to create, read, update, or delete.

Our group very much enjoyed working on this project, especially when we had the database ready to go and made an application utilizing it.

Some future work involving the project could include setting up a long term server to host it. We could also make the Course Registration System application multithreaded in order to speed up its work, particularly when running SQL statements over the Internet.

References

Oracle Corporation. “Getting Started with JDBC.” *https://docs.oracle.com/Javase/Tutorial/Jdbc/Basics/Gettingstarted.html*.

Redko, Alla. “JavaFX: TableView.” *Https://Docs.oracle.com/Javafx/2/ui_controls/Table-View.htm*.

https://stackoverflow.com/.

Appendix

https://github.com/aNtOnIoRaMaJ/CourseRegistration is the link to the repository containing the zip file containing all the work products.