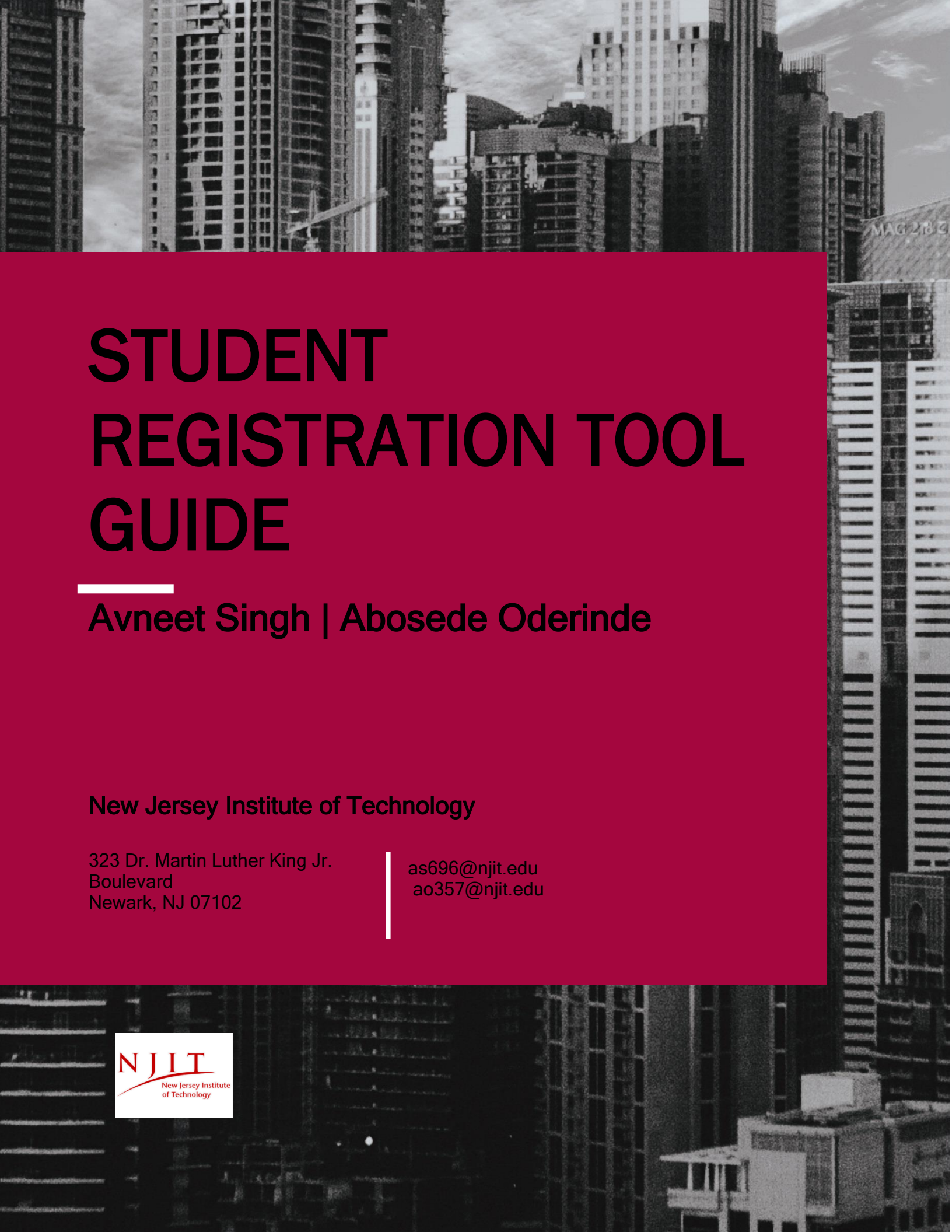

STUDENT REGISTRATION TOOL GUIDE

Avneet Singh | Abosede Oderinde

New Jersey Institute of Technology

323 Dr. Martin Luther King Jr. Boulevard
Newark, NJ 07102

as696@njit.edu
ao357@njit.edu

Table of Contents

Introduction

Welcome to NJIT's student registration tool guide! In this guide, we will be going over how to use our new tool to register for courses. The steps involved are the following: Installation, Setup, Getting Started as a New Student, Getting Started as a Returning Student, Fail Cases, and Future Additions.

Installation

In order to download our software, we've provided a link to a Google Drive folder containing both the Python file 'Assignment_5.py' and the SQL file 'Assignment_5.sql' below.

Click the link provided above and download the two files. To utilize our software, you need to complete a series of installations. The software required are listed below. We have also provided hyperlinks to the installation pages for each software.

- Python 3.10.4 – (This is the version of Python I use, but the code should work with a depreciated version. This can also be installed via the VS Code installation)
- Visual Studio Code, or a Python IDE of your choice
- MySQL
- MySQL Workbench - (Can be installed with the MySQL installer)

Providing steps for running through the installations will not be provided, as that is outside the scope of this guide. There are multiple guides on YouTube that will assist a new student in installing and setting up both VS Code and MySQL.

You also may need to install some Python packages to use the software. To install these packages, you'll need to run the following code in your Python terminal:

- pip install mysql.connector
- pip install datetime

The mysql.connector package allows us to connect to the MySQL server hosted on our local machine. The datetime package allows us to input the date of the registration into our database.

Getting Started

OPENING ASSIGNMENT_5.PY

Once you have the required software and packages installed in Python, we can start using the tool. First, we launch Visual Studio Code. Once VS Code is open, you will see a screen similar to Figure 1.

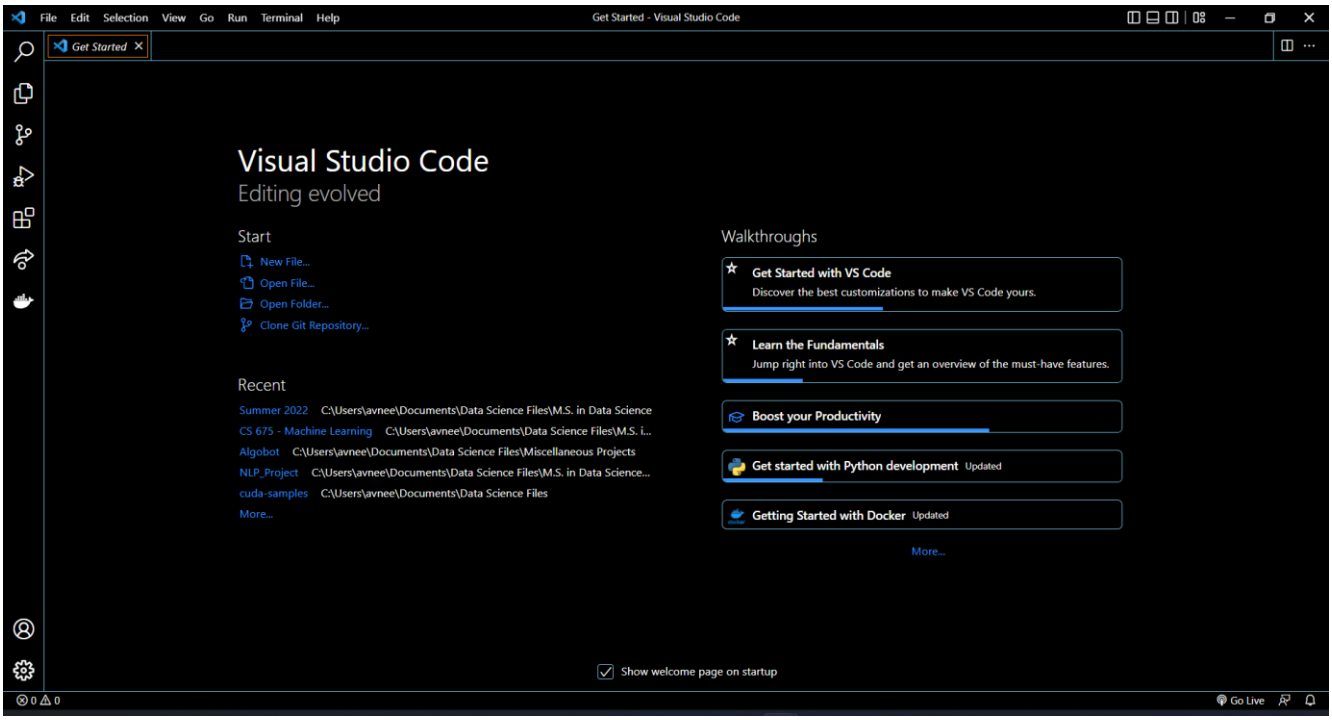

Figure 1

Click ‘Open File’ and select ‘Assignment_5.py’ from where you stored the Python file on your device. It will most likely be under the Downloads folder. The opened file should look similar to Figure 2.

Figure 2

OPENING ASSIGNMENT_5.SQL

Next, we must open the SQL file in MySQL Workbench. Ideally, NJIT would host this database on their server. For the scope of this project, we need to host the SQL database

locally. Once you open MySQL and set up your first localhost server via a YouTube tutorial, the application should look similar to Figure 3.

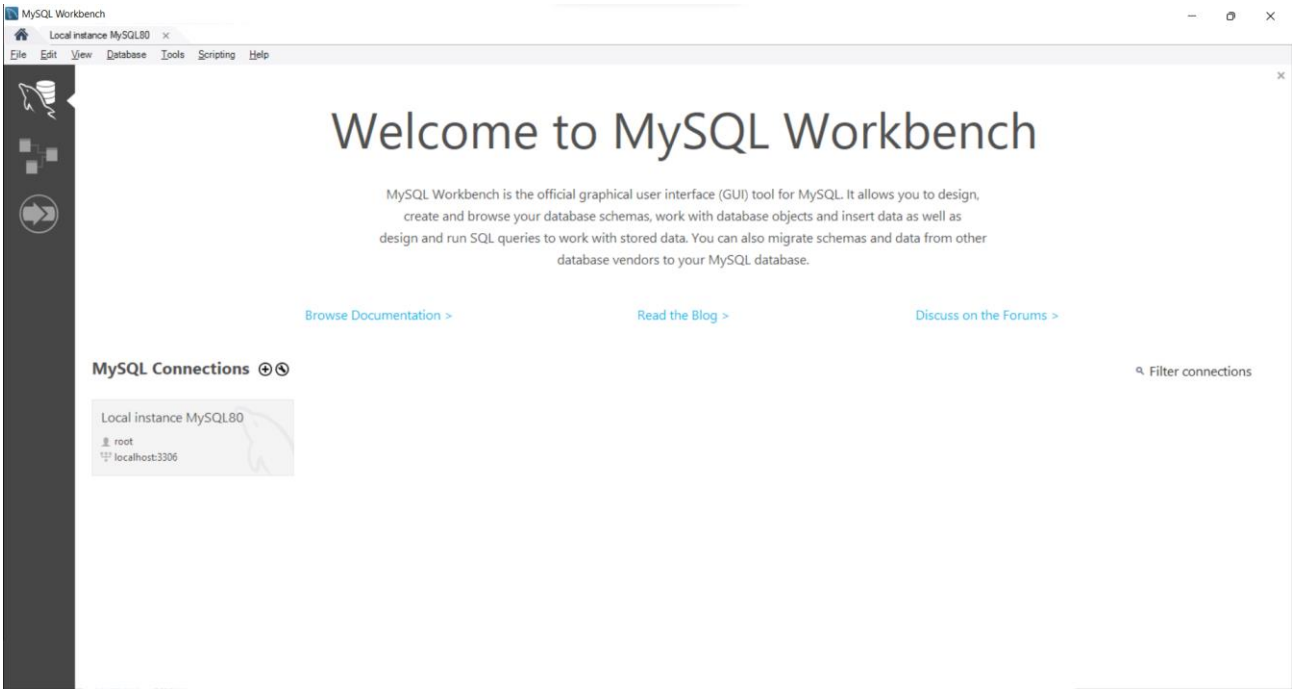

Figure 3

Navigate to “File” -> “Open Model” on the top of the screen, then open the “Assignment_5.sql” file by clicking on it. It should be located in your Downloads folder. You’ll need to connect to the local server, so once the SQL file is open, click on “Database” -> “Connect to Database.” You can keep most of this information the same if you kept the default settings when setting up your localhost server and click “OK.” Once your screen looks similar to Figure 4, you’re in good shape to start using the tool!

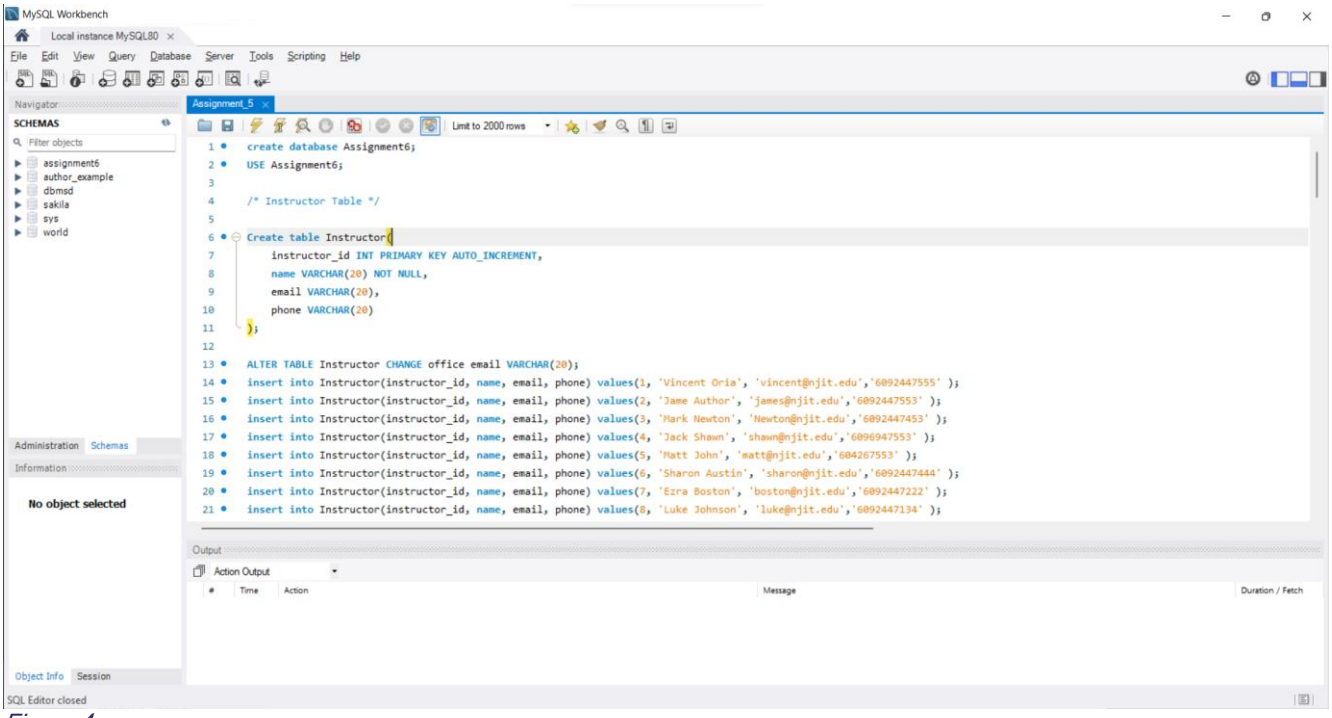

Figure 4

Registration – New Student

GETTING YOUR UCID

Let's get started! Navigate back to VS Code and click on the play icon in the top right corner. This will run the code and start the program. You should see something similar to Figure 5.

```
Hello, welcome to NJIT's student registration tool!

Please enter your UCID if you are a returning student.
If you are a new student, press 0.

 ENTRY:
```

Figure 5

Since you are a new student at NJIT, you need to type 0 in next to 'ENTRY". You will be asked to enter basic information, such as your name and address. Let's say your name is "John Smith" and you live in "Newark, NJ". Once we input this information, we will be given a brand new UCID, which will serve as your login token! You should see something similar to Figure 6.

```
Welcome to NJIT! Please provide some basic information for so we can update our database!

FULL NAME: John Smith
ADDRESS: Newark, NJ

Hi, John Smith! Your current address is Newark, NJ.

This information has been passed to our database!

 Your new UCID is: 52
```

Figure 6

CHOOSE YOUR FIRST COURSE

Now that you have your very own UCID, we can get you registered for your first course! The software will output a list of courses that you can choose from. There are 11 different courses to choose from! Once you decide which course you would like to register for, enter the integer value at the beginning of the course next to 'ENTRY'. A sample input would be any integer value from 1 to 11. Your options should look similar to Figure 7.

```
Now let's get you registered! Here are the available courses, along with the professors, sections, and number of spots left

(1) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 50 - 0 out of 5 seats available.
(2) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 60 - 2 out of 5 seats available.
(3) 601 - Web Development System - Professor Andrew Pole
        Section ID - 51 - 3 out of 5 seats available.
(4) 611 - Applied Statistics - Professor Ezra Boston
        Section ID - 56 - 3 out of 5 seats available.
(5) 621 - Artificial Intelligence - Professor Sharon Austin
        Section ID - 54 - 4 out of 5 seats available.
(6) 631 - Database Management - Professor Jack Shawn
        Section ID - 52 - 4 out of 5 seats available.
(7) 635 - Data Analytics for info - Professor Vincent Oria
        Section ID - 59 - 4 out of 5 seats available.
(8) 640 - IT - Professor Mark Newton
        Section ID - 55 - 2 out of 5 seats available.
(9) 656 - Data Analytics with R - Professor Matt John
        Section ID - 53 - 2 out of 5 seats available.
(10) 658 - Data Structure - Professor Vincent Oria
        Section ID - 58 - 2 out of 5 seats available.
(11) 664 - Big Data - Professor Luke Johnson
        Section ID - 57 - 2 out of 5 seats available.
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY:
```

Figure 7

Moving forward, let's say you've decided to enroll in IT, which is taught by Professor Mark Newton. We can see that there are only 2 slots left, so let's register quickly! Once we make our selection, the software should output text similar to Figure 8.

```
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY: 8

You've selected: IT - Section 55
Successfully registered student 52 for course number 640 - section number 55

You're all set! Register for more courses by restarting the program, inputting your new UCID, and selecting another course!
Remember! You can only register for a max of 3 courses.
```

Figure 8

In Figure 8, we can see the name of the course we've selected, as well as the section that we are registered to. The software also warns you that there is a course limit of 3 courses at NJIT, so you can't enroll for more.

And that's it! We've successfully gotten you your student ID and registered you for your first course in IT. To register for another course, follow the next section of the guide.

Registration – Returning Student

RUNNING THE PROGRAM

Open the files based on the instructions in the "Installation" section of this guide. Once everything is open properly, you can run the Python file by clicking the play icon in the top right corner of VS Code. This will initialize the software. You should see something similar to Figure 9.

```
Hello, welcome to NJIT's student registration tool!

Please enter your UCID if you are a returning student.
If you are a new student, press 0.

 ENTRY:
```

Figure 9

REGISTERING FOR A COURSE

Let's walk through an example of how John Smith would use the Student Registration Tool to register for a course. Once John enters his UCID next to 'ENTRY,' we can see the courses he's enrolled in, as well as his basic information that he inputted while registering for the first time. It should look similar to Figure 10.

```
Welcome back, John Smith! Your current address is Newark, NJ

You are currently registered for 1 course.

Course:

IT
```

Figure 10

The software will output a list of courses that you can choose from. There are 11 different courses to choose from! Once you decide which course you would like to register for, enter the integer value at the beginning of the course next to 'ENTRY'. A sample input would be any integer value from 1 to 11. Your options should look similar to Figure 11.

```
Now let's get you registered for more! Here are the available courses, along with the professors, sections, and number of spots left

(1) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 50 - 0 out of 5 seats available.
(2) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 60 - 2 out of 5 seats available.
(3) 601 - Web Development System - Professor Andrew Pole
        Section ID - 51 - 3 out of 5 seats available.
(4) 611 - Applied Statistics - Professor Ezra Boston
        Section ID - 56 - 3 out of 5 seats available.
(5) 621 - Artificial Intelligence - Professor Sharon Austin
        Section ID - 54 - 4 out of 5 seats available.
(6) 631 - Database Management - Professor Jack Shawn
        Section ID - 52 - 4 out of 5 seats available.
(7) 635 - Data Analytics for info - Professor Vincent Oria
        Section ID - 59 - 4 out of 5 seats available.
(8) 640 - IT - Professor Mark Newton
        Section ID - 55 - 1 out of 5 seats available.
(9) 656 - Data Analytics with R - Professor Matt John
        Section ID - 53 - 2 out of 5 seats available.
(10) 658 - Data Structure - Professor Vincent Oria
        Section ID - 58 - 2 out of 5 seats available.
(11) 664 - Big Data - Professor Luke Johnson
        Section ID - 57 - 2 out of 5 seats available.
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY:
```

Figure 11

Let's say John wants to learn more about Business Intelligence. In this case, he would enter the integer 2 next to 'ENTRY' and press Enter. The software will output a confirmation message that looks similar to Figure 12, then exit.

```
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY: 2

You've selected: Business Intelligence - Section 60
Successfully registered student 52 for course number 600 - section number 60

You're all set! Register for more courses by restarting the program, inputting your new UCID, and selecting another course!
Remember! You can only register for a max of 3 courses.
```

Figure 12

And that's it! We've successfully registered you for a course at NJIT. To register for another course, repeat the steps in this section. Remember, you can only register for a maximum of 3 courses, so choose wisely!

Fail Cases

This section of the guide will go over some of the fail cases that our software handles elegantly. This will include when a student registers for a course that has no seats left, when a student registers for a course they're already registered to, and when a student tries to register for more than 3 courses.

#1 - STUDENT REGISTERS FOR FULL COURSES

Let's say that John Smith wants to enroll in Big Data. Once he launches the software and enters his UCID, he sees that the only available section for Big Data this semester is full! The course list looks like Figure 13.

```
Now let's get you registered! Here are the available courses, along with the professors, sections, and number of spots left

(1) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 50 - 0 out of 5 seats available.
(2) 600 - Business Intelligence - Professor Vincent Oria
        Section ID - 60 - 1 out of 5 seats available.
(3) 601 - Web Development System - Professor Andrew Pole
        Section ID - 51 - 3 out of 5 seats available.
(4) 611 - Applied Statistics - Professor Ezra Boston
        Section ID - 56 - 3 out of 5 seats available.
(5) 621 - Artificial Intelligence - Professor Sharon Austin
        Section ID - 54 - 4 out of 5 seats available.
(6) 631 - Database Management - Professor Jack Shawn
        Section ID - 52 - 4 out of 5 seats available.
(7) 635 - Data Analytics for info - Professor Vincent Oria
        Section ID - 59 - 4 out of 5 seats available.
(8) 640 - IT - Professor Mark Newton
        Section ID - 55 - 1 out of 5 seats available.
(9) 656 - Data Analytics with R - Professor Matt John
        Section ID - 53 - 2 out of 5 seats available.
(10) 658 - Data Structure - Professor Vincent Oria
        Section ID - 58 - 2 out of 5 seats available.
(11) 664 - Big Data - Professor Luke Johnson
        Section ID - 57 - 0 out of 5 seats available.
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY:
```

Figure 13

He figures he'll try his luck anyways and gets the error message displayed in Figure 14.

```
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY: 11

You've selected: Big Data - Section 57
The course you've selected is full, try another one!
```

Figure 14

The software warns him that the section is full and exits. John will have to figure out another course to take!

#2 - STUDENT REGISTERS FOR CLASS THEY'RE ALREADY IN

Let's go over an example where John can't recall which courses he's signed up for. He logs in to the software via UCID, checks the course list, and decides to enroll in Business Intelligence. He doesn't see that he's already registered and enters 2 into the entry line. The software outputs the following error displayed in Figure 15.

```
Select the section you wish to register for by inputting the number next to the name of the course you wish to register for!
ENTRY: 2

You've selected: Business Intelligence - Section 60
You're already registered for this course!
```

Figure 15

#3 - STUDENT REGISTERS FOR MORE THAN 3 COURSES

Let's go over an example where John tries to register for a course after he's already registered for 3 courses. Once John logs into the software via UCID, the software sees that he is registered for 3 courses already, so it outputs the message displayed in Figure 16.

```
Welcome back, John Smith! Your current address is Newark, NJ

You are currently registered for 3 courses.

Courses:

IT
Business Intelligence
Data Structure

Looks like you're registered for 3 courses, which is the maximum allowed at NJIT. Have a great day!
```

Figure 16

Entity-Relationship Design

Due to the time limitations of the summer semester at NJIT, this course was unable to properly pace the milestones of this project. Therefore, Professor Vincent Oria has provided the entity-relationship diagram for this project. This is displayed in Figure 17.

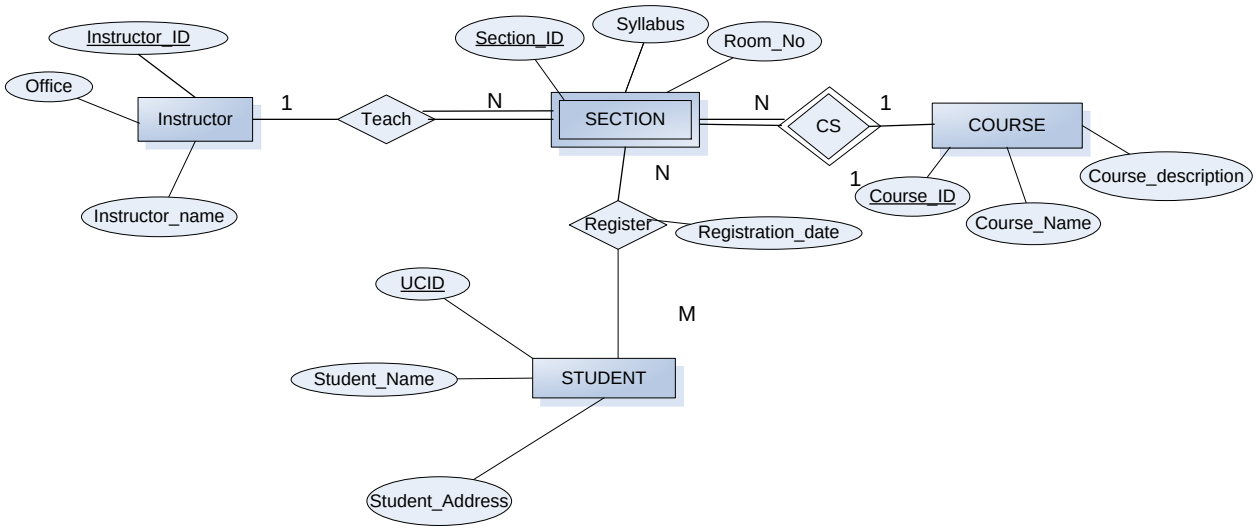

Figure 17

Relational Logical Database Design

RELATIONAL SCHEMA

Similarly to the Entity-Relationship Diagram, Professor Oria also provided us with a normalized Relational Schema, which this software utilizes to store the information. This schema is displayed below in Figure 18.

INSTRUCTOR(INSTRUCTOR_ID, INSTRUCTOR_NAME, INSTRUCTOR_OFFICE, INSTRUCTOR_PHONE)
COURSE(COURSE_ID, COURSE_NAME, COURSE_DESCRIPTION)
STUDENT (STUDENT_ID, STUDENT_NAME, STUDENT_ADDRESS)
SECTION(SECTION_ID, COURSE_ID, SECTION_SYLLABUS, ROOM, DAY, HOUR, INSTRUCTOR_ID)

The remaining TEACHING corresponds to
REGISTRATION (SECTION_ID, COURSE_ID, STUDENT_ID, REGISTRATION_DATE)

Figure 18

We actually had to change this schema slightly to better fit our needs. For example, we added a Registration_ID as the primary key of Registration for simplified querying.

SQL IMPLEMENTATION

We've implemented the relational schema shown above in SQL. In this section, we'll go over each table in SQL and what it looks like.

Instructor Table

The instructor table's implementation is shown in Figure 19.

```
Create table Instructor(
    instructor_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    email VARCHAR(20),
    phone VARCHAR(20)
);
```

Figure 19

We decided to change the office attribute from the original schema to email, but this does not affect the overall software because this information is only available in the database and cannot be changed via our product. We added the information based on the commands shown in Figure 20.

```
ALTER TABLE Instructor CHANGE office email VARCHAR(20);
insert into Instructor(instructor_id, name, email, phone) values(1, 'Vincent Oria', 'vincent@njit.edu','6092447555' );
insert into Instructor(instructor_id, name, email, phone) values(2, 'Jame Author', 'james@njit.edu','6092447553' );
insert into Instructor(instructor_id, name, email, phone) values(3, 'Mark Newton', 'Newton@njit.edu','6092447453' );
insert into Instructor(instructor_id, name, email, phone) values(4, 'Jack Shawn', 'shawn@njit.edu','6096947553' );
insert into Instructor(instructor_id, name, email, phone) values(5, 'Matt John', 'matt@njit.edu','604267553' );
insert into Instructor(instructor_id, name, email, phone) values(6, 'Sharon Austin', 'sharon@njit.edu','6092447444' );
insert into Instructor(instructor_id, name, email, phone) values(7, 'Ezra Boston', 'boston@njit.edu','6092447222' );
insert into Instructor(instructor_id, name, email, phone) values(8, 'Luke Johnson', 'luke@njit.edu','6092447134' );
insert into Instructor(instructor_id, name, email, phone) values(10, 'Denmark James', 'dens@njit.edu','6092447111' );
```

Figure 20

The final instructor table is shown in Figure 21.

instructor_id	name	email	phone
5	Matt John	matt@njit.edu	604267553
6	Sharon Austin	sharon@njit.edu	6092447444
7	Ezra Boston	boston@njit.edu	6092447222
8	Luke Johnson	luke@njit.edu	6092447134
10	Denmark James	dens@njit.edu	6092447111
NULL	NULL	NULL	NULL

Figure 21

Course Table

The course table's implementation is shown in Figure 22.

```
Create table Course(
    course_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(40) NOT NULL,
    description VARCHAR(250)
);
```

Figure 22

The course table's main purpose is to hold the course ID and the course name. It also holds the department under the 'description' attribute, but this is not used within our software. We added the information based on the commands shown in Figure 23.

```
insert into Course(course_id, name, description) values(631, 'Database Management', 'Computer Science');
insert into Course(course_id, name, description) values(656, 'Data Analytics with R', 'Computer Science');
insert into Course(course_id, name, description) values(601, 'Web Development System', 'Computer Science');
insert into Course(course_id, name, description) values(640, 'IT', 'Computer Science');
insert into Course(course_id, name, description) values(621, 'Artificial Intelligence', 'Computer Science');
insert into Course(course_id, name, description) values(600, 'Business Intelligence', 'Computer Science');
insert into Course(course_id, name, description) values(635, 'Data Analytics for info', 'Computer Science');
insert into Course(course_id, name, description) values(611, 'Applied Statistics', 'Mathematics');
insert into Course(course_id, name, description) values(664, 'Big Data', 'Computer Science');
insert into Course(course_id, name, description) values(658, 'Data Structure', 'Computer Science');
```

Figure 23

The final instructor table is shown in Figure 24.

course_id	name	description
600	Business Intelligence	Computer Science
601	Web Development System	Computer Science
611	Applied Statistics	Mathematics
621	Artificial Intelligence	Computer Science
631	Database Management	Computer Science
635	Data Analytics for info	Computer Science

Figure 24

Student Table

The student table's implementation is shown in Figure 25.

```
Create table Student(
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(20) NOT NULL,
    address VARCHAR(20)
);
```

Figure 25

This table is used to contain all of NJIT's student names, addresses and UCIDs. The primary key student_id serves as the student's UCID. During implementation, we kept it as student_id for synchronization within the relational schema. Our software grabs the student's name and address from the Python file, then inserts this information into the Student table. We can query for the student_id to output the new student's UCID. We added the information based on the commands shown in Figure 26.

```
insert into Student(student_id, name, address) values(11, 'Mike Tyson', 'Trenton, NJ');
insert into Student(student_id, name, address) values(12, 'Matt Bason', 'Princeton, NJ');
insert into Student(student_id, name, address) values(13, 'Duke Badwin', 'Langhorne, PA');
insert into Student(student_id, name, address) values(14, 'Michelle Cook', 'Ewing, NJ');
insert into Student(student_id, name, address) values(15, 'Elizabeth Taylor', 'Elizabeth, NY');
insert into Student(student_id, name, address) values(16, 'Morenike Johnson', 'Monmonth, NJ');
insert into Student(student_id, name, address) values(17, 'Ademi Tope', 'Ewing, NJ');
insert into Student(student_id, name, address) values(18, 'Luke Tyson', 'Trenton, NJ');
insert into Student(student_id, name, address) values(19, 'Nike Apples', 'Trenton, NJ');
insert into Student(student_id, name, address) values(20, 'James Brown', 'Ewing, NJ');
insert into Student(student_id, name, address) values(21, 'Lin Lin', 'Trenton, NJ');
insert into Student(student_id, name, address) values(22, 'Aretha James', 'Trenton, NJ');
```

Figure 26

A snapshot of the final Student table is shown in Figure 27.

32	Avneet Singh	Hoboken, NJ
33	Thomas Cruise	Berlin, Germany
34	Steve Carrell	Scranton, PA
35	Lisa Monroe	Tuscany, Italy
36	Post Malone	Austin, Texas

Section Table

The section table's implementation is shown in Figure 27.

```
Create table Section(
    section_id INT PRIMARY KEY AUTO_INCREMENT,
    course_id INT,
    instructor_id INT,
    syllabus VARCHAR(250) NOT NULL,
    room VARCHAR(20),
    day VARCHAR(20),
    hour VARCHAR(20),
    address VARCHAR(20),
    foreign key(course_id) references Course(course_id) on delete set null,
    foreign key(instructor_id) references Instructor(instructor_id) on delete set null
```

Figure 27

This section contains the most attributes of the schema. However, we do not utilize most of these attributes due to time constraints. This table utilizes course_id and instructor_id as foreign keys that reference the Course and Instructor tables. The main use for this table is storing sections for all of the courses offered at NJIT. We added the information based on the commands shown in Figure 28.

```
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(50, 'CS600', 'B10', 'Monday','12:00','NJ',600,1);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(51, 'CS601', 'B5', 'Tuesday','16:00','NJ',601,2);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(52, 'CS631', 'B10', 'Friday','12:00','NJ',631,4);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(53, 'CS656', 'B12', 'Wednesday','18:00','NJ',656,5);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(54, 'CS621', 'B1', 'Monday','12:00','NJ',621,6);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(55, 'CS640', 'B10', 'Monday','12:00','NJ',640,3);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(56, 'CS611', 'B7', 'Saturday','12:00','NJ',611,7);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(57, 'CS664', 'B10', 'Monday','12:00','NJ',664,8);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(58, 'CS658', 'B5', 'Monday','12:00','NJ',658,1);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(59, 'CS635', 'B10', 'Monday','12:00','NJ',635,1);
insert into Section(section_id, syllabus, room, day, hour, address, course_id, instructor_id) values(60, 'CS600', 'B10', 'Monday','12:00','NJ',600,1);
```

Figure 28

A snapshot of the final Section table is shown in Figure 29.

section_id	course_id	instructor_id	syllabus	room	day	hour	address
50	600	1	CS600	B10	Monday	12:00	NJ
51	601	2	CS601	B5	Tuesday	16:00	NJ
52	631	4	CS631	B10	Friday	12:00	NJ
53	656	5	CS656	B12	Wednesday	18:00	NJ
54	621	6	CS621	B1	Monday	12:00	NJ
55	640	3	CS640	B10	Monday	12:00	NJ

Figure 29

Registration Table

The registration table's implementation is shown in Figure 27. As previously stated, we implemented a registration id as the primary key for simplified querying.

```
Create table Registration(
    registration_id INT PRIMARY KEY AUTO_INCREMENT,
    section_id INT,
    course_id INT,
    student_id INT,
    foreign key(section_id) references Section(section_id) on delete set null,
    foreign key(course_id) references Course(course_id) on delete set null,
    foreign key(student_id) references Student(student_id) on delete set null,
    registration_date DATE
);
```

Figure 30

The registration table is the most useful table in our schema, as this is the table that stores the registration of all students, along with the section_id of the section they signed up for, the course_id for the course they signed up for, and their student_id. The Python file uses datetime to push today's date to the SQL server.

```
insert into Registration(section_id, course_id, student_id, registration_date) values(50,600,11, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(51,601,12, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(52,631,14, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(53,656,25, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(54,621,16, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(55,640,23, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(56,611,17, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(57,664,28, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(58,658,11, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(59,631,21, '2022-05-01');
insert into Registration(section_id, course_id, student_id, registration_date) values(60,621,12, '2022-05-01');
```

Figure 31

A snapshot of the final Registration table is shown in Figure 31.

registration_id	section_id	course_id	student_id	registration_date
1	50	600	11	2022-05-01
2	51	601	12	2022-05-01
3	52	631	14	2022-05-01
4	53	656	25	2022-05-01
5	54	621	16	2022-05-01
6	55	640	23	2022-05-01

Figure 32

Application Program Design

In this section, we will step behind the curtain and show you how our software works. We've utilized Python to write the software.

IMPORTS

First, we import the required python packages. In this case, they are mysql.connector and datetime. We talked about this during the installation phase.

```
# Imports
import mysql.connector
from datetime import date
```

CONNECT TO THE MYSQL SERVER

Next, we can connect to our locally hosted server via the following code.

```
sql = mysql.connector.connect(user='root', password='password',
                              host='localhost',
                              database='assignment6')

cursor = sql.cursor()
```

FUNCTIONS

In this program, we've created 3 custom functions that help our software perform properly. We'll cover them in this section.

get_courses(UCID)

This function queries the database for the names of all the courses a student is taking given their UCID. This is primarily used when a student enters their UCID as a returning student and we would like to see which courses they are enrolled in.

```
def get_courses(UCID):
    res = []
    query = "SELECT name from course where course_id in (select course_id from
registration where student_id = " + str(
        UCID) + ");"
    for q in cursor.execute(query, multi=True):
        result = q.fetchall()
        for r in result:
            res.append(r[0])
        return res
```

registration(UCID)

This function is the meat and potatoes of the program. It queries our database and gets all course names, course IDs, course sections, instructor names, instructor IDs, and the amount of students registered for that course.

```
def registration(UCID):
    query = "SELECT C.name, C.course_id, I.name, I.instructor_id, S.section_id,
COUNT(R.student_id) as Count \
        FROM Course C, Section S, Instructor I, Registration R \
            WHERE C.course_id = S.course_id \
                AND S.instructor_id = I.instructor_id \
                    AND R.section_id = S.section_id \
                        GROUP BY S.section_id;"
```

Next, it loops through the results of the query and prints the readable output that we see in Figure 7. It appends the individual values to arrays that contain each attribute we've queried, but this is not utilized within our program.

```
for q in cursor.execute(query, multi=True):
        result = q.fetchall()
        cnt = 1
        for r in result:
            r = list(r)
            print("(" + str(cnt) + ") " + str(r[1]) + " - " +
                  str(r[0]) + " - Professor " + str(r[2]) +
                  "\n\t Section ID - " + str(r[4]) + " - " + str(5-int(r[5])) + " out
of 5 seats available.")
            cnt += 1

            # Appends
            c_name.append(r[0])
            c_id.append(r[1])
            i_name.append(r[2])
            i_id.append(r[3])
            s_id.append(r[4])
            all.append(r)
```

After printing the available courses and sections, the program asks the user which course and section they would like to sign up for, prints a confirmation message once selection has occurred, and exits.

```
choice = input(
            "Select the section you wish to register for by inputting the number next
to the name of the course you wish to register for!\nENTRY: ")
        print("\nYou've selected: " +
              str(all[int(choice)-1][0]) + " - Section " + str(all[int(choice)-1][4]))
        register(all[int(choice)-1][4], all[int(choice)-1][1], UCID)
        print("\nYou're all set! Register for more courses by restarting the program,
inputting your new UCID, and selecting another course!\nRemember! You can only
register for a max of 3 courses.")
        exit()
```

register(UCID)

This function first considers fail cases in which the program should not register a student for a course. First, the program checks to see if the student is already registered for the course they've chosen. If so, the program outputs an error message and quits.

```
def register(section_id, course_id, UCID):
    # CHECK IF THEY'RE ALREADY IN THE CLASS
    cursor.execute(
        "select student_id from registration where student_id = " + str(UCID) + " and
course_id = " + str(course_id) + "", multi=True)
    if cursor.fetchall() != []:
        print("You're already registered for this course!")
        exit()
```

Next, the program checks to see if the student is already registered for 3 courses before registering the student for another. If so, the program outputs an error message and quits.

```
# CHECK IF THEY'RE AT THE COURSE LIMIT
    cursor.execute(
        "select COUNT(course_id) from registration where student_id = " + str(UCID),
multi=True)
    if cursor.fetchall()[0][0] == 3:
        print("You're at the course limit!")
        exit()
```

The last fail case is to check if the course that the student selected is full. If this is the case, the program outputs an error message and quits.

```
# CHECK IF CLASS IS FULL
    cursor.execute(
        "select COUNT(student_id) from registration where section_id = " +
str(section_id), multi=True)
    if cursor.fetchall()[0][0] >= 5:
        print("The course you've selected is full, try another one!")
        exit()
```

If the function does not detect any of these fail conditions, then the program inserts the registration into the registration table in our SQL server and outputs a confirmation message.

```
else:
        cursor.execute("INSERT INTO Registration(section_id, course_id, student_id,
registration_date) VALUES(" +
                       str(section_id) + ", " + str(course_id) + ", " + str(UCID) + ",
'" + str(date.today()) + "');", multi=True)
        sql.commit()
        print("Successfully registered student " + str(UCID) +
              " for course number " + str(course_id) + " - section number " +
str(section_id))
```

START OF THE PROGRAM

After defining the functions, we print a welcome message and ask for the student's UCID. If they are a new student, they should enter 0.

```
# Start
UCID = input("\nHello, welcome to NJIT's student registration tool!\n"
             "\nPlease enter your UCID if you are a returning student.\nIf you are a
new student, press 0.\n\n ENTRY: ")

# Status for new or returning student
status = ''
```

This block of code queries the database for the student's name and address based on their UCID.

```
 Check if returning student and provide basic info
try:
    if int(UCID) != 0:
        select_stmt = "SELECT name, address FROM student WHERE student_id = " + \
            str(UCID)
        for q in cursor.execute(select_stmt, multi=True):
            result = list(q.fetchone())
            s_name = result[0]
            s_address = result[1]
            status = 'Returning'
```

Here, we check to see that the student entered a valid UCID. If the student enters a text value instead of an integer or a UCID that is not associated with a student, then the program will output an error message.

```
except:
    print("\nError! That is not a valid UCID. ")
```

Next, we check if the student is new. If so, the program prints a welcome message and asks for the basic information previously discussed.

```
else:
        # Get basic info if new student
        print("\nWelcome to NJIT! Please provide some basic information for so we can
update our database!\n")
        s_name = input("FULL NAME: ")
        s_address = input("ADDRESS: ")
        status = 'New'
```

If the student has a UCID, then the following code will run

```
if status == 'Returning':
    print("\nWelcome back, " + str(s_name) + "! " +
          "Your current address is " + str(s_address))

    # Get courses associated with this UCID
    res = get_courses(UCID)
```

This code prints a welcome message and prints the information previously queried from the SQL server. Then it runs the get_courses function to get the courses that the student is registered for. If the student isn't registered for any courses, the following code runs and goes through the registration function.

```
# If num_courses = 0
    if (res == None) | (len(res) == 0):
        print("\nLooks like you aren't registered for any classes yet! You can
register for a maximum of 3 courses per semester.")
        # RETURNING STUDENT COURSE REGISTRATION (0)
        print("\nNow let's get you registered! Here are the available courses, along
with the professors, sections, and number of spots left\n")
        registration(UCID)
        choice = input(
            "Select the section you wish to register for by inputting the number next
to the name of the course you wish to register for!\nENTRY: ")
        print("\nYou've selected: " +
              str(all[int(choice)-1][0]) + " - Section " + str(all[int(choice)-1][4]))
        register(all[int(choice)-1][4], all[int(choice)-1][1], UCID)
        print("\nYou're all set! Register for more courses by restarting the program,
inputting your new UCID, and selecting another course!\nRemember! You can only
register for a max of 3 courses.")
        exit()
```

If the student is registered for one course, then the following code will execute:

```
# If num_courses = 1
    elif len(res) == 1:
        print("\nYou are currently registered for 1 course. ")
        print("\nCourse: \n")
        print(res[0])
        # RETURNING STUDENT COURSE REGISTRATION (1)
        print("\nNow let's get you registered for more! Here are the available
courses, along with the professors, sections, and number of spots left\n")
        registration(UCID)
        choice = input(
            "Select the section you wish to register for by inputting the number next
to the name of the course you wish to register for!\nENTRY: ")
        print("\nYou've selected: " +
              str(all[int(choice)-1][0]) + " - Section " + str(all[int(choice)-1][4]))
        register(all[int(choice)-1][4], all[int(choice)-1][1], UCID)
        print("\nYou're all set! Register for more courses by restarting the program,
inputting your new UCID, and selecting another course!\nRemember! You can only
register for a max of 3 courses.")
```

This runs through similar steps as the previous code, but the difference lies in printing the word "course" instead of "courses", as the student is only enrolled in one course.

The next block of code covers if the student registered for 2 or more courses, then runs through similar steps as the registration function.

```
# If num_courses = 2+
    else:
        print("\nYou are currently registered for " +
              str(len(res)) + " courses. ")
        print("\nCourses: \n")
        # Gets names of courses student is taking
        for r in res:
            print(r)

        # CHECK IF REGISTERED FOR 3 COURSES
        if len(res) == 3:
            print("\nLooks like you're registered for 3 courses, which is the maximum
allowed at NJIT. Have a great day!")
            exit()
        # RETURNING STUDENT COURSE REGISTRATION (0)
        print("\nNow let's get you registered! Here are the available courses, along
with the professors, sections, and number of spots left\n")
        registration(UCID)
        choice = input(
            "Select the section you wish to register for by inputting the number next
to the name of the course you wish to register for!\nENTRY: ")

        print("\nYou've selected: " +
              str(all[int(choice)-1][0]) + " - Section " + str(all[int(choice)-1][4]))
        register(all[int(choice)-1][4], all[int(choice)-1][1], UCID)
        print("\nYou're all set! Register for more courses by restarting the program,
inputting your new UCID, and selecting another course!\nRemember! You can only
register for a max of 3 courses.")
```