MAGA (Make Advising Great Again)

A modern system for the University of Texas at El Paso’s Computer Science Advising Department.

Team 1

Table of Contents

1. Database Scope
2. Requirements & Assumptions
3. Entity-Relationship Model
4. Relational Model
5. Normalized Schema
6. Database Schema in MYSQL
7. Database Records
8. SQL queries
9. Views
10. Procedures & Triggers
11. Reports
12. Graphical User Interface
13. References
14. Appendix A

Database Scope

We will develop a system for the University of Texas at El Paso's Computer Science Department to aid in making advising easier for both advisors and their students. The goal is to eliminate the need and use of paper forms as well as easy access to updated information. This system will provide information pertaining to students such as student's full name, GPA, credits, username, entrance year, graduation year, classes taken, classes needed to graduate and others. The website will follow and meet UTEP's regular rules and regulations.

The system will provide a way for the Computer Science front desk (admins), advisors, and students to log in using their UTEP credentials to view their corresponding information. Admins will have access to view both advisors and students information. Advisors will have access to view only their assigned students information. Lastly, the students will have the lowest access being able to only view their advisors information and the resources needed in order to schedule their advising appointment.

The student will have a record of which courses they will take, have taken and have not taken. Advisors will be provided a section to create new notes about the advising session that could never be deleted. Once a student completes their advising session, their status will reflect that they have been advised therefore allowing admin to remove necessary holds. Furthermore, students will not schedule an appointment with advisors for approvals, but simply send an alert to the advisor who will either accept or deny their request.

Records will be stored and kept in the system even after students graduate but will no longer grant those students access to the system. The overall information will be used to generate reports on how many students have been advised, still need to get advised and how many students are assigned to each advisor for analytic purposes.

Requirements & Assumptions

Requirements:

Requirement Number	**Satisfied By**	**Implemented In**
1. The system shall be hosted within UTEP's IT department.	UTEP	GUI is hosted on cssrvlab/CS4342/Team1 The database is hosted on another UTEP server
2. The system shall provide the users (advisors, staff) with the ability to log in with their unique credentials.		GUI hosted on UTEP servers, provides logins for the three users (admin and advisor have a default username and password that has been hardcoded)
3. The system shall provide users with information about UTEP CS students (including the student's full name,GPA, credits, username, entrance year, graduation year, classes taken, classes needed to graduate, and more)		GUI has a button where only administrators can view one giant table with all student information. This data is pulled from the *Student* table in the MySQL database with a simple query that selects all information from the table and displays it using a PHP call.
4. System shall allow UTEP CS students to view information about		Within the GUI under the studentHome.php page, there is a table displaying all

advising, such as who their advisor is and where they can sign up for an advising appointment).		advisors along with their information and a button to schedule an appointment. When student presses button it takes them to advising_slip.php and then to make-appointment.php
5. Once the student has successfully completed their advising sheet, the system will allow them to schedule an appointment with the advisor.		SQL Queries within the back end will gather the information and implement a new addition into the database. Each advising slip will be given a unique ID to be used later.
6. The system shall keep track of each student's previous advising records, this record will be able to be updated as the semesters pass.		Within the database, the classes taken or will be taken will be indicated by a True while the classes not taken yet will be indicated with a False. Their records will be linked with their student ID and advisors/admins will be able to update the records as the student progresses through their degree.
7. The system shall alert users that a particular student has completed their advising session. It will send the student a copy of their advising slip and notify the front desk admins that the student is cleared to enroll when the time comes.		GUI, once the student has completed their advising, the advisor will then be able to approve or reject the student's advising slip. From there, on the GUI, the admin will then be able to process the final advising slip. When processed the SQL table will keep a record of it being processed.
8. The system shall provide a way for UTEP CS students to select which courses they would like to take the		GUI handles this by letting the students select their courses in the *appointments* webpage, here they fill out info about the course such

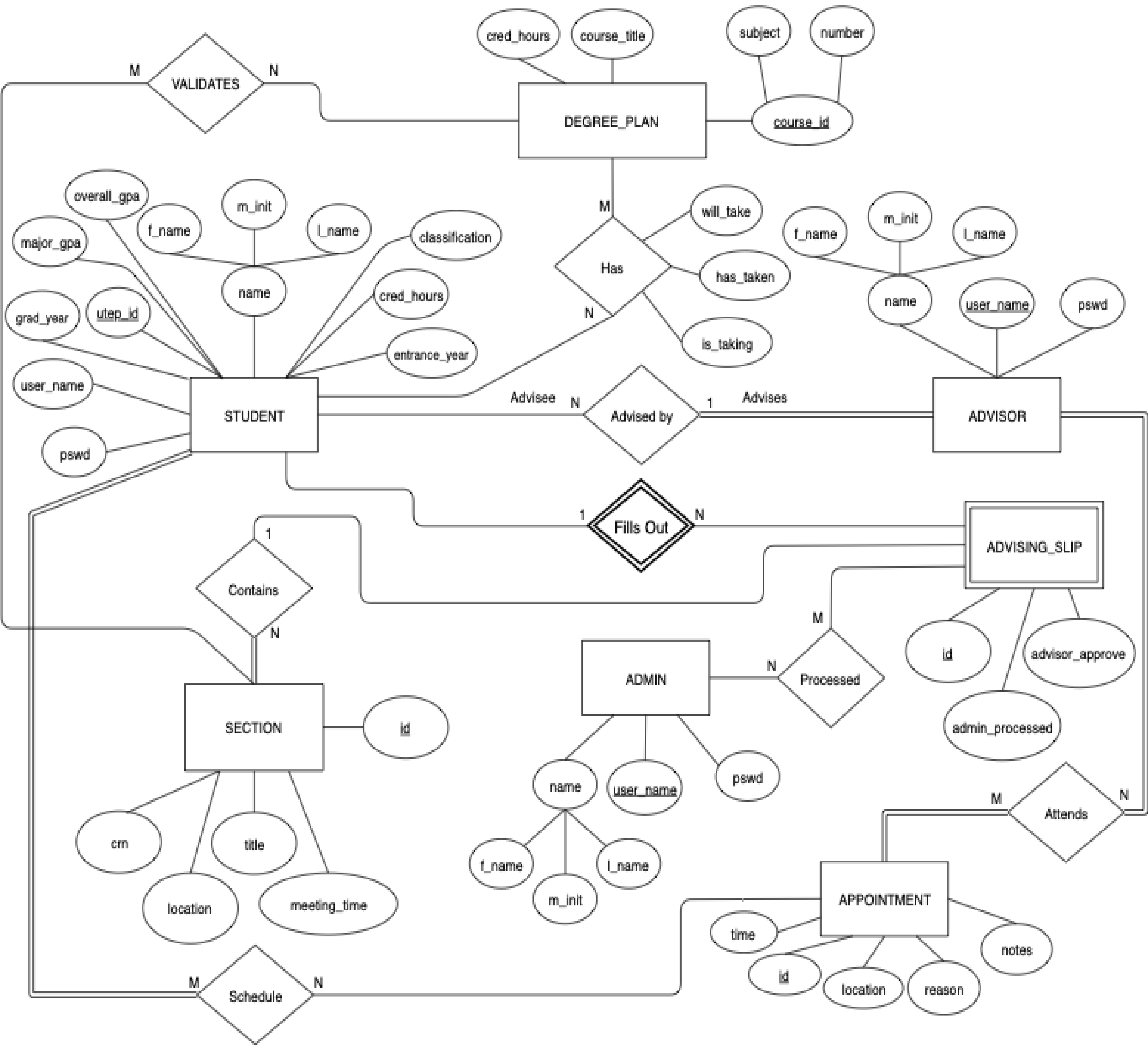
coming semester.		as CRN, title, time, etc. The PHP script will then send this information through POST to the database that will store this information in the *section* table, as well as generating a unique advising slip ID that advisors and admins can then keep track of.
9. The system shall allow advisors to see information about each of their students, but prevents students from seeing information about each other.		Only advisors/admins will have permission to view and write onto each student's records through the GUI while student's permissions are directly tied with their ID, so they can only see their own information.
10. The system shall store records of students even after they have graduated, but will no longer allow students access to these records.		After the student has graduated their permission to write will be revoked, but their records tied to their ID will remain within the database until removed by an admin through SQL Queries.
11. System shall have a way of generating reports with information such as how many students were advised, how many students each professor has been assigned, and how many students still need to be advised.		In the GUI under the admin page, there will be buttons that list what information they want and through the back end SQL Queries will retrieve the information and display it in a table on the GUI.
12. The system will provide a section for advisors to log information about the student advising session. Notes should not be deleted, but new		Within the GUI the advisor can add notes to a students advising sheet and using SQL Queries it will be stored within the database.

notes can be added continuously.		

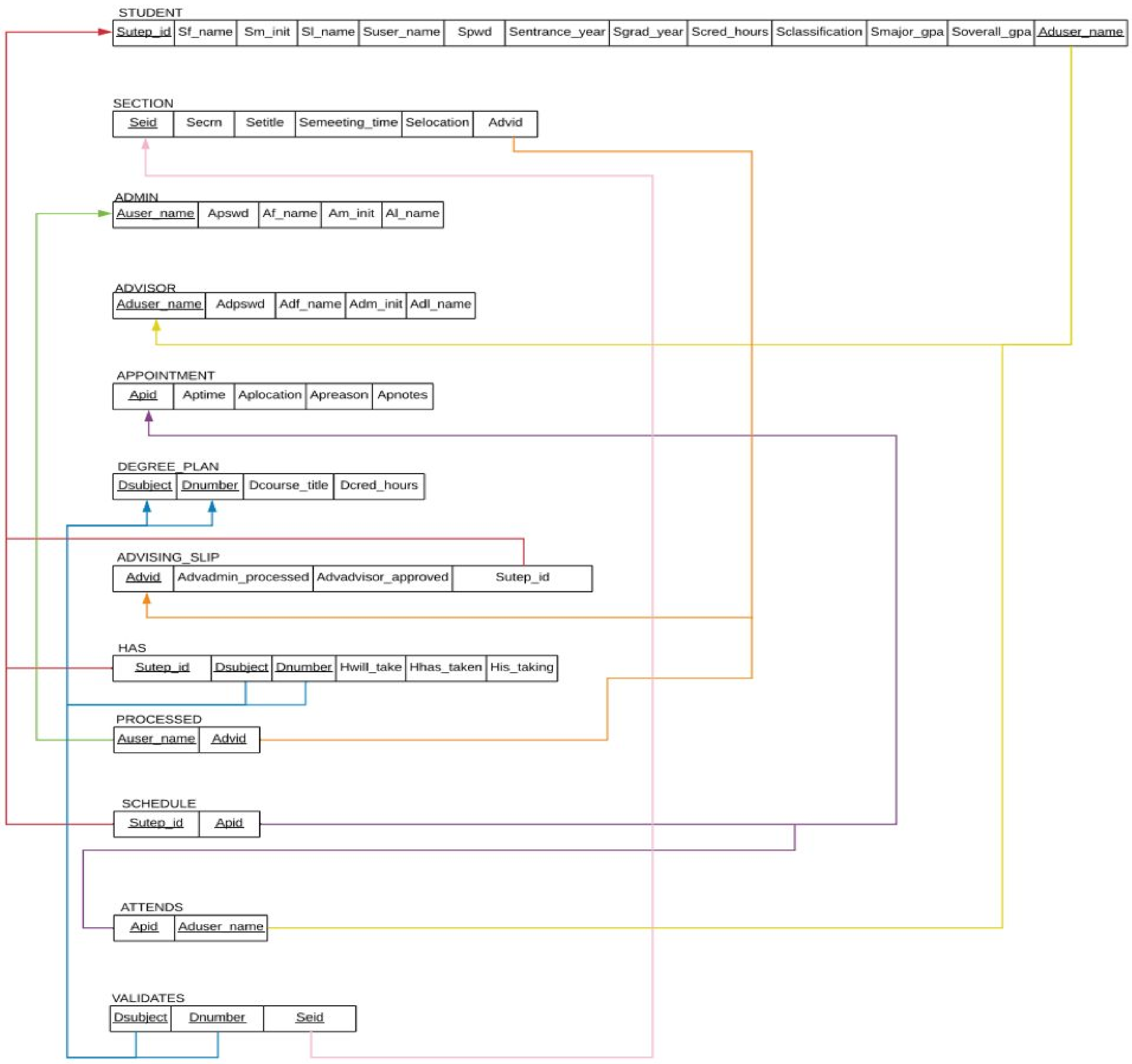
Assumptions:

1. We assume that users will have an internet connection and can browse the web.
2. We assume that the CS department will provide correct and accurate forms (advising, etc) and that wherever those files are stored will remain functional.
3. We assume that the information about students (GPA, classes taken) is accurate and does not need to be verified by our system.

Entity-Relationship Model

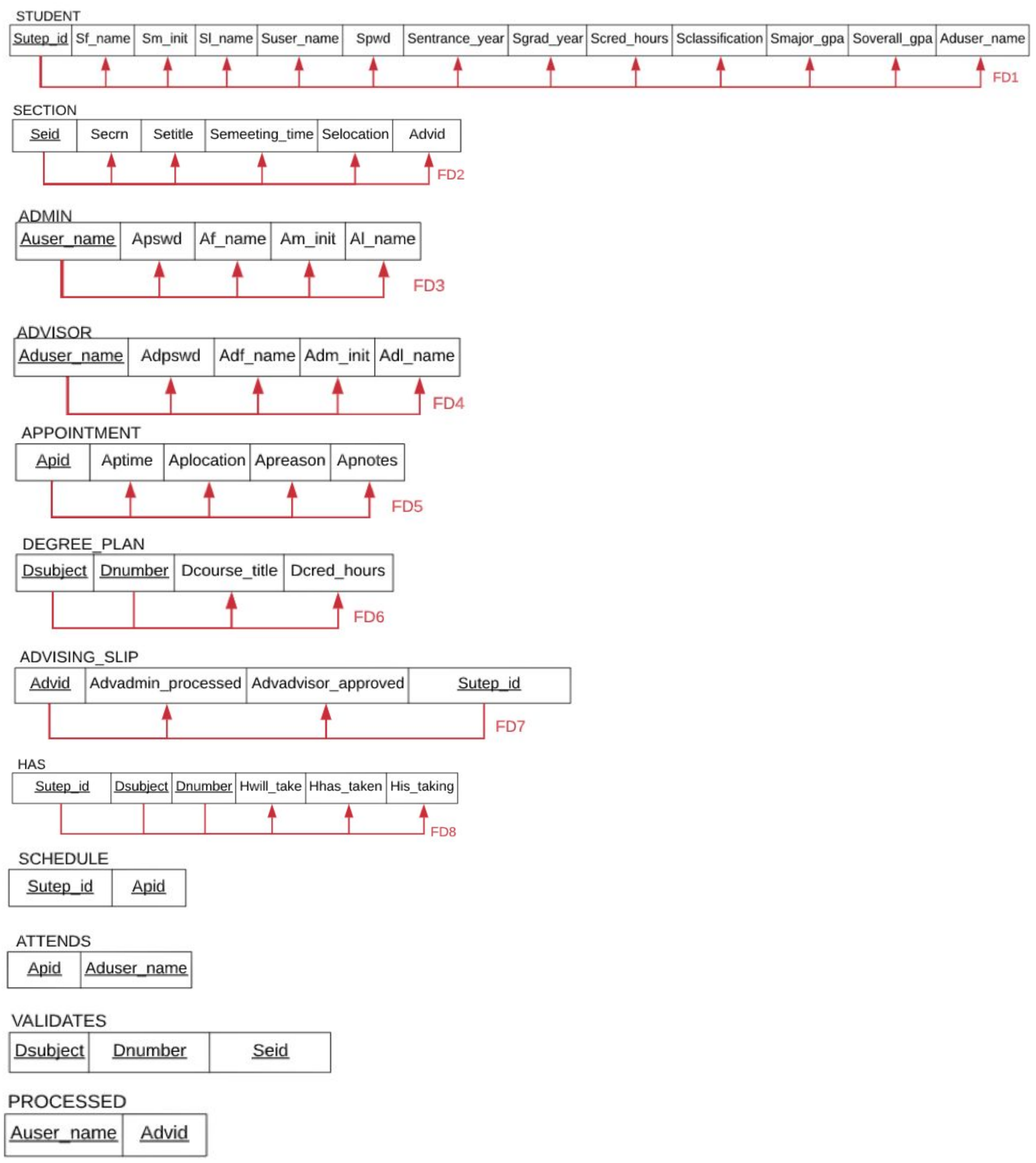


Relational Model


STUDENT
Sutep_id | Sf_name | Sm_init | Sl_name | Suser_name | Spwd | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name
SECTION
Seid | Secrn | Setitle | Semeeting_time | Selocation | Advid
ADMIN
Auser_name | Apswd | Af_name | Am_init | Al_name
ADVISOR
Aduser_name | Adpswd | Adf_name | Adm_init | Adl_name
APPOINTMENT
Apid | Aptime | Aplocation | Apreason | Apnotes
DEGREE_PLAN
Dsubject | Dnumber | Dcourse_title | Dcred_hours
ADVISING_SLIP
Advid | Advadmin_processed | Advadvisor_approved | Sutep_id
HAS
Sutep_id | Dsubject | Dnumber | Hwill_take | Hhas_taken | His_taking
PROCESSED
Auser_name | Advid
SCHEDULE
Sutep_id | Apid
ATTENDS
Apid | Aduser_name
VALIDATES
Dsubject | Dnumber | Seid

Normalized Schema

Functional Dependencies



1NF

- **STUDENT** is in 1NF given that all attributes are atomic.
- **SECTION** is in 1NF given that all attributes are atomic.
- **ADMIN** is in 1NF given that all attributes are atomic.
- **ADVISOR** is in 1NF given that all attributes are atomic.
- **APPOINTMENT** is in 1NF given that all attributes are atomic.
- **DEGREE_PLAN** is in 1NF given that all attributes are atomic.
- **ADVISING_SLIP** is in 1NF given that all attributes are atomic.
- **HAS** is in 1NF given that all attributes are atomic.
- **SCHEDULE** is in 1NF given that all attributes are atomic.
- **ATTENDS** is in 1NF given that all attributes are atomic.
- **PROCESSED** is in 1NF given that all attributes are atomic.
- **VALIDATES** is in 1NF given that all attributes are atomic.

2NF

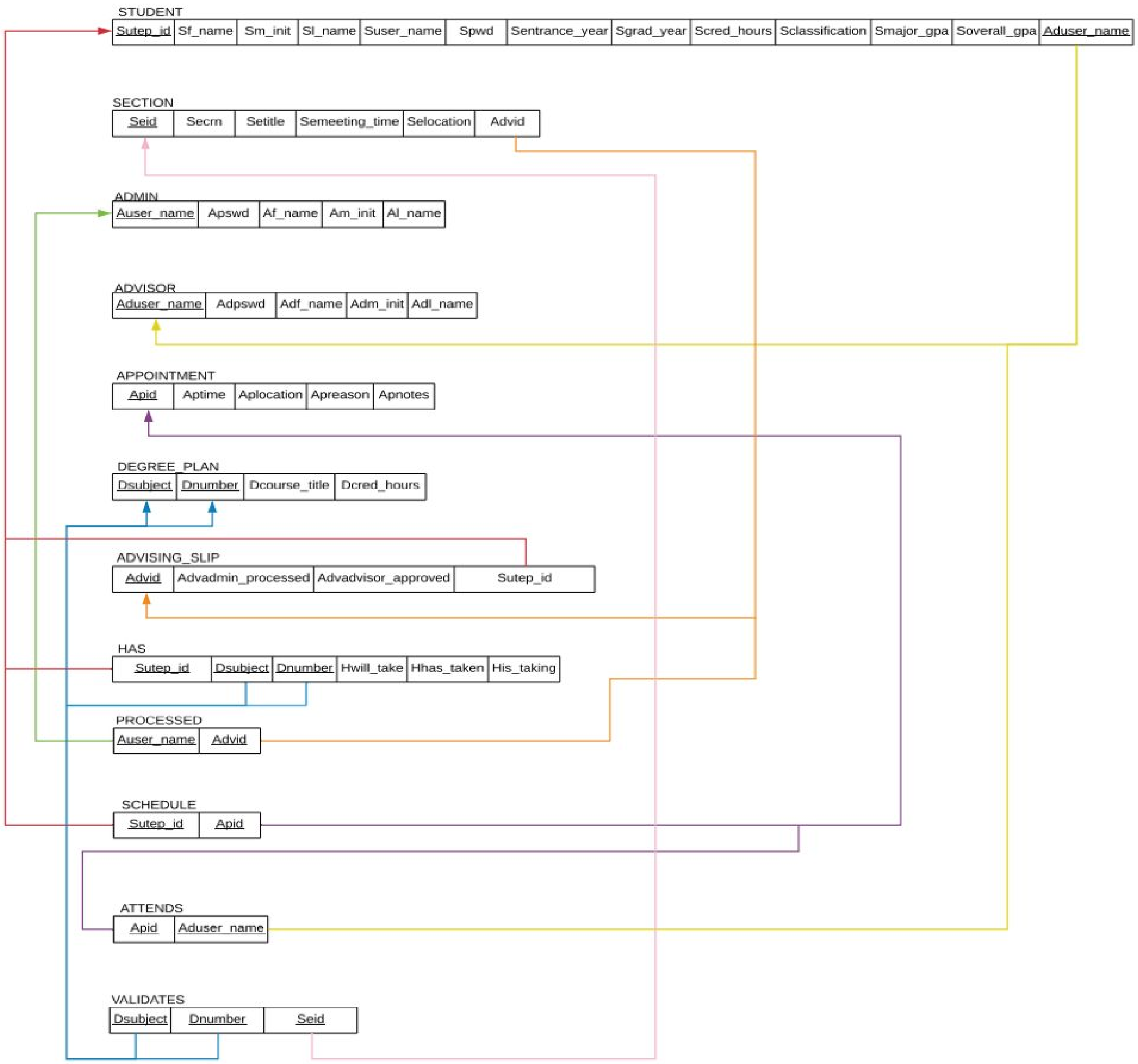
- The relation **STUDENT** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Sutep_id*.
- The relation **SECTION** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Seid*.
- The relation **ADMIN** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Auser_name*.
- The relation **ADVISOR** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Aduser_name*.
- The relation **APPOINTMENT** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Dsubject* and *Dnumber*.
- The relation **DEGREE_PLAN** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Stutep_id*.
- The relation **ADVISING_SLIP** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Advid*.
- The relation **HAS** is in 2NF because it is in 1NF and all its non-prime attributes fully-functionally depend on the primary key *Sutep_id, Dsubject and Dnumber*.
- The relation **SCHEDULE** is in 2NF because it is in 1NF and it does not contain non-prime attributes that can violate the form.
- The relation **ATTENDS** is in 2NF because it is in 1NF and it does not contain non-prime attributes that can violate the form.

- The relation **PROCESSED** is in 2NF because it is in 1NF and it does not contain non-prime attributes that can violate the form.
- The relation **VALIDATES** is in 2NF because it is in 1NF and it does not contain non-prime attributes that can violate the form.

3NF

- The relation **STUDENT** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **SECTION** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **ADMIN** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **ADVISOR** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **APPOINTMENT** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **DEGREE_PLAN** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **ADVISING_SLIP** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **HAS** is in 3NF because it is in 2NF and no non-prime attributes depends on another non-primary attribute.
- The relation **SCHEDULE** is in 3NF because it is in 2NF and it does not contain non-prime attributes that can violate the form.
- The relation **ATTENDS** is in 3NF because it is in 2NF and it does not contain non-prime attributes that can violate the form.
- The relation **PROCESSED** is in 3NF because it is in 2NF and it does not contain non-prime attributes that can violate the form.
- The relation **VALIDATES** is in 3NF because it is in 2NF and it does not contain non-prime attributes that can violate the form.

Final Relations


STUDENT
Sutep_id | Sf_name | Sm_init | Sl_name | Suser_name | Spwd | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name
SECTION
Seid | Secrn | Setitle | Semeeting_time | Selocation | Advid
ADMIN
Auser_name | Apswd | Af_name | Am_init | Al_name
ADVISOR
Aduser_name | Adpswd | Adf_name | Adm_init | Adl_name
APPOINTMENT
Apid | Aptime | Aplocation | Apreason | Apnotes
DEGREE_PLAN
Dsubject | Dnumber | Dcourse_title | Dcred_hours
ADVISING_SLIP
Advid | Advadmin_processed | Advadvisor_approved | Sutep_id
HAS
Sutep_id | Dsubject | Dnumber | Hwill_take | Hhas_taken | His_taking
PROCESSED
Auser_name | Advid
SCHEDULE
Sutep_id | Apid
ATTENDS
Apid | Aduser_name
VALIDATES
Dsubject | Dnumber | Seid

Database Schema in MySQL

Creating the table for student information.

```
CREATE TABLE Student (
Sutep_id CHAR(8) NOT NULL PRIMARY KEY,
Sf_name CHAR(50) NOT NULL,
Sm_init CHAR(2),
Sl_name CHAR(50) NOT NULL,
Suser_name CHAR(50) NOT NULL,
Spswd CHAR(50) NOT NULL,
Sentrance_year YEAR NOT NULL,
Sgrad_year YEAR NOT NULL,
Scred_hours INT NOT NULL,
Sclassification CHAR(15) NOT NULL,
Smajor_gpa FLOAT NOT NULL,
Soverall_gpa FLOAT NOT NULL,
Aduser_name CHAR(50) NOT NULL,
FOREIGN KEY (Aduser_name) REFERENCES Advisor(Aduser_name))Engine=InnoDB;
```

Creating the table for course section information.

```
CREATE TABLE Section (
Seid INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
Secrn CHAR(5) NOT NULL,
Setitle CHAR(50) NOT NULL,
Semeeting_time CHAR(50),
Selocation CHAR(50),
Advid INT NOT NULL,
FOREIGN KEY (Advid) REFERENCES Advising_slip(Advid))Engine=InnoDB;
```

Creating the table for administrator information.

```
CREATE TABLE Admin (
Auser_name CHAR(50) NOT NULL PRIMARY KEY,
Apswd CHAR(50) NOT NULL,
Af_name CHAR(50) NOT NULL,
Am_init CHAR(2),
Al_name CHAR(50) NOT NULL)Engine=InnoDB;
```

Creating the table for administrator information.

```
CREATE TABLE Advisor (
Aduser_name CHAR(50) NOT NULL PRIMARY KEY,
Adpswd CHAR(50) NOT NULL,
Adf_name CHAR(50) NOT NULL,
Adm_init CHAR(2),
Adl_name CHAR(50) NOT NULL)Engine=InnoDB;
```

Creating the table for appointment information.

```
CREATE TABLE Appointment (
Apid INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
Aplocation CHAR(50) NOT NULL,
Aptime CHAR(50) NOT NULL,
Apreason VARCHAR(255),
Apnotes VARCHAR(255))Engine=InnoDB;
```

Creating the table for administrator information.

```
CREATE TABLE Degree_plan (
Dsubject CHAR(20) NOT NULL,
Dnumber CHAR(4),
Dcourse_title CHAR(50),
Dcred_hours INT NOT NULL,
PRIMARY KEY(Dsubject, Dnumber))Engine=InnoDB;
```

Creating the table for information about the advising slip.

```
CREATE TABLE Advising_slip (
Advid INT PRIMARY KEY AUTO_INCREMENT,
Advadmin_processed CHAR(1) NOT NULL,
Advadvisor_approved CHAR(1) NOT NULL,
Sutep_id CHAR(8) NOT NULL,
FOREIGN KEY (Sutep_id) REFERENCES Student(Sutep_id))Engine=InnoDB;
```

Creating the table for status information on courses student selects.

```
CREATE TABLE Has (
Sutep_id CHAR(8) NOT NULL,
Dsubject CHAR(20) NOT NULL,
Dnumber CHAR(4),
Hhas_taken CHAR(1) NOT NULL,
Hwill_take CHAR(1),
His_taking CHAR(1),
PRIMARY KEY(Sutep_id, Dsubject, Dnumber),
FOREIGN KEY (Sutep_id) REFERENCES Student(Sutep_id),
FOREIGN KEY (Dsubject, Dnumber) REFERENCES Degree_plan (Dsubject,
Dnumber))Engine=InnoDB;
```

Creating the table for keeping track of which stage of processing the advising slip is at.

```
CREATE TABLE Processed (
Auser_name CHAR(50) NOT NULL,
Advid INT NOT NULL,
PRIMARY KEY(Auser_name, Advid),
FOREIGN KEY (Auser_name) REFERENCES Admin(Auser_name),
FOREIGN KEY (Advid) REFERENCES Advising_slip(Advid))Engine=InnoDB;
```

Creating the table for a student scheduling an appointment with their advisor.

```
CREATE TABLE Schedule (
Sutep_id CHAR(8) NOT NULL,
Apid INT NOT NULL,
PRIMARY KEY(Sutep_id, Apid),
FOREIGN KEY (Sutep_id) REFERENCES Student(Sutep_id),
FOREIGN KEY (Apid) REFERENCES Appointment(Apid))Engine=InnoDB;
```

Creating the table for keeping track of which students went to advising appointments.

```
CREATE TABLE Attends (
Apid INT NOT NULL,
Aduser_name CHAR(50) NOT NULL,
PRIMARY KEY(Apid, Aduser_name),
FOREIGN KEY (Apid) REFERENCES Appointment(Apid),
FOREIGN KEY (Aduser_name) REFERENCES Advisor(Aduser_name))Engine=InnoDB;
```

Creating the table for verifying that the course the student wants is a valid option.

```
CREATE TABLE Validates (
Dsubject CHAR(20) NOT NULL,
Dnumber CHAR(4),
Seid INT NOT NULL,
PRIMARY KEY(Dsubject, Dnumber, Seid),
FOREIGN KEY (Dsubject, Dnumber) REFERENCES Degree_plan (Dsubject, Dnumber),
FOREIGN KEY (Seid) REFERENCES Section(Seid))Engine=InnoDB;
```

Database Records

ADMIN TABLE:

- INSERT INTO Admin (Auser_name, Apswd, Af_name, Am_init, Al_name) VALUES ('admin', 'admin1', 'super', 'J', 'adminLastName');

- INSERT INTO Admin (Auser_name, Apswd, Af_name, Am_init, Al_name) VALUES ('adminer', 'admin2', 'root', 'A', 'Rootier');

- INSERT INTO Admin (Auser_name, Apswd, Af_name, Am_init, Al_name) VALUES ('kreese', 'pwd90', 'Kyle', 'M', 'Reese');

ADVISOR TABLE:

- INSERT INTO Advisor VALUES ('nward', 'wardPass', 'Nigel', 'G', 'Ward');

- INSERT INTO Advisor VALUES ('freudy', 'msp430', 'Eric', 'L', 'Freudenthal');

- INSERT INTO Advisor (Aduser_name, Adpswd, Adf_name, Adl_name) VALUES ('ycheon', 'sup3rJ4V4', 'Yoonsik', 'Cheon');

STUDENT TABLE:

- INSERT INTO Student VALUES ('12345678', 'John', 'M', 'Smith', 'jmsmith', 'sfd879a^X', 2017, 2021, 94, ‘junior’, 3.6, 3.1, 'freudy');

- INSERT INTO Student VALUES ('89452854', 'Thor', 'K', 'Martinez', 'tmartinez', 'cs#*(0dd24)', 2019, 2023, 60, ‘freshman’, 4.0, 4.0, 'nward');

- INSERT INTO STUDENT VALUES ('85361773', 'Jason', 'A', 'Mamoa', 'jamamoa', '6/*X2hu', 2019, 2023, 61, 'Freshman', 2.0, 1.9, 'mceberio');

ADVISING_SLIP TABLE:

- INSERT INTO Advising_slip (Advadmin_processed, Advadvisor_approved, Sutep_id) VALUES ('F', ‘F’, ‘12345678’);

- INSERT INTO Advising_slip (Advadmin_processed, Advadvisor_approved, Sutep_id) VALUES ('T', ‘T’, ‘80579815’);

- INSERT INTO Advising_slip (Advadmin_processed, Advadvisor_approved, Sutep_id) VALUES ('F', ‘T’, ‘89452854’);

SECTION TABLE:

- INSERT INTO Section (Secrn, Setitle, Advid) VALUES ('26043', 'Software Engineering: Design & Implement', 1);

- INSERT INTO Section (Secrn, Setitle, Advid) VALUES ('27593', 'Probability & Statistics', 1);

- INSERT INTO Section (Secrn, Setitle, Advid) VALUES ('27347', 'Human-Computer Interaction', 1);

- INSERT INTO Section (Secrn, Setitle, Advid) VALUES ('28733', 'Cross-Platform App Development', 1);

APPOINTMENT TABLE:

- INSERT INTO Appointment (Aplocation, Aptime, Apreason) VALUES ('CCSB 3.0602', '10-12-2018 03:30', 'General advising for the upcoming semester');

- INSERT INTO Appointment (Aplocation, Aptime, Apreason) VALUES ('CCSB 1.0702', '11-12-2019 04:15', 'Graduate school advising');

- INSERT INTO Appointment (Aplocation, Aptime, Apreason) VALUES ('CCSB 3.0407', '04-08-2016 08:30', 'Tranfering credits questions');

DEGREE_PLAN TABLE:

- INSERT INTO degree_plan(Dsubject, Dnumber, Dcourse_title, Dcred_hours) VALUES('CS', '1301', 'Introduction to Computer Science', 3);

- INSERT INTO degree_plan(Dsubject, Dnumber, Dcourse_title, Dcred_hours) VALUES('CS', '1101', 'Introduction to Computer Science Lab', 1);

- INSERT INTO degree_plan(Dsubject, Dnumber, Dcourse_title, Dcred_hours) VALUES('CS', '2401', 'Elementary Data Structures', 4);

HAS TABLE:
- INSERT INTO Has(Sutep_id, Dsubject, Dnumber, Hhas_taken) VALUES ('80569035', 'CS', '1301', 'T');

- INSERT INTO Has(Sutep_id, Dsubject, Dnumber, Hhas_taken) VALUES ('80569035', 'CS', '1101', 'T');

- INSERT INTO Has(Sutep_id, Dsubject, Dnumber, Hhas_taken) VALUES ('80569035', 'CS', '2401', 'T');

PROCESSED TABLE:
- INSERT INTO Processed VALUES ('admin', 2);

- INSERT INTO Processed VALUES ('adminer', 2);

- INSERT INTO Processed VALUES ('kreese', 2);

SCHEDULE TABLE:
- INSERT INTO Schedule VALUES ('12345678', 1);

- INSERT INTO Schedule VALUES ('80579815', 2);

- INSERT INTO Schedule VALUES ('89452854', 3);

ATTENDS TABLE:
- INSERT INTO Attends VALUES (1, 'freudy');

- INSERT INTO Attends VALUES (2, 'nward');

- INSERT INTO Attends VALUES (3, 'nward');

VALIDATES TABLE:

- INSERT INTO Validates VALUES ('CS', '4311', 1);

- INSERT INTO Validates VALUES ('STAT', '3320', 2);

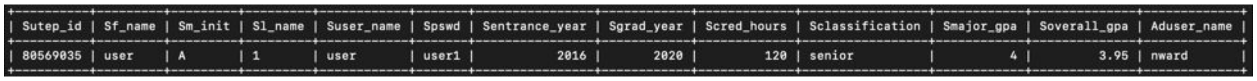
- INSERT INTO Validates VALUES ('CS', 'TBA4', 3);

- INSERT INTO Validates VALUES ('CS', 'TBA5', 4);

SQL queries

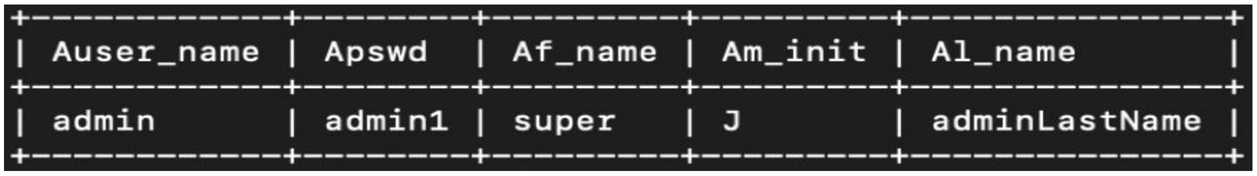
1. The system shall be hosted within UTEP's IT department.
2. The system shall provide the users (advisors, staff) with the ability to log in with their unique credentials.

   SELECT * FROM student WHERE Suser_name='user' AND Spswd='user1';



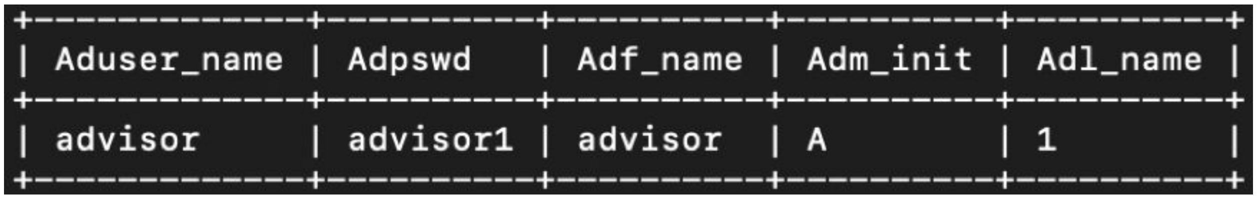
```
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| Sutep_id | Sf_name | Sm_init | Sl_name | Suser_name | Spswd | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name |
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| 80569035 | user    | A       | 1       | user       | user1 |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
```

   SELECT * FROM admin WHERE Auser_name='admin' AND Apswd='admin1';



```
+------------+--------+---------+---------+---------------+
| Auser_name | Apswd  | Af_name | Am_init | Al_name       |
+------------+--------+---------+---------+---------------+
| admin      | admin1 | super   | J       | adminLastName |
+------------+--------+---------+---------+---------------+
```

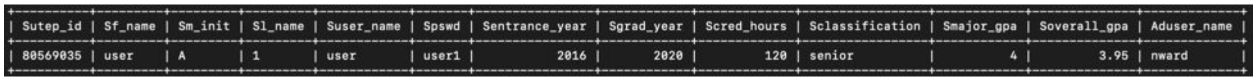
   SELECT * FROM advisor WHERE Aduser_name='advisor' AND Adpswd='advisor1';



```
+-------------+----------+----------+---------+---------+
| Aduser_name | Adpswd   | Adf_name | Adm_init | Adl_name |
+-------------+----------+----------+---------+---------+
| advisor     | advisor1 | advisor  | A        | 1        |
+-------------+----------+----------+---------+---------+
```

3. The system shall provide users with information about UTEP CS students (including the student's full name,GPA, credits, username, entrance year, graduation year, classes taken, classes needed to graduate, and more)

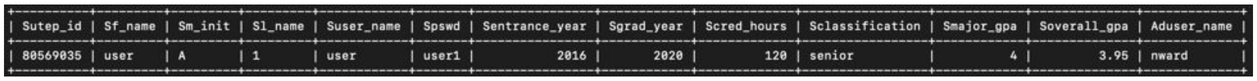
   SELECT * FROM student WHERE Suser_name='user' AND Spswd='user1';



```
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| Sutep_id | Sf_name | Sm_init | Sl_name | Suser_name | Spswd | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name |
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| 80569035 | user    | A       | 1       | user       | user1 |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
+----------+---------+---------+---------+------------+-------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
```

4. System shall allow UTEP CS students to view information about advising, such as who their advisor is.

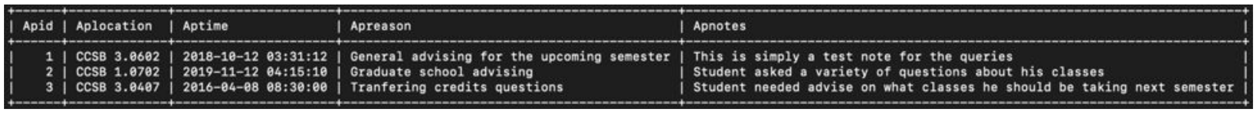
SELECT * FROM student WHERE Suser_name='user' AND Spswd='user1';



```
+----------+---------+---------+---------+------------+-------+----------------+-----------+------------+-----------------+-----------+---------------+-------------+
| Sutep_id | Sf_name | Sm_init | Sl_name | Suser_name | Spswd | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name |
+----------+---------+---------+---------+------------+-------+----------------+-----------+------------+-----------------+-----------+---------------+-------------+
| 80569035 | user    | A       | 1       | user       | user1 |           2016 |      2020 |        120 | senior          |         4 |          3.95 | nward       |
+----------+---------+---------+---------+------------+-------+----------------+-----------+------------+-----------------+-----------+---------------+-------------+
```

5. Once the student has successfully completed their advising sheet, the system will allow them to schedule an appointment with the advisor.

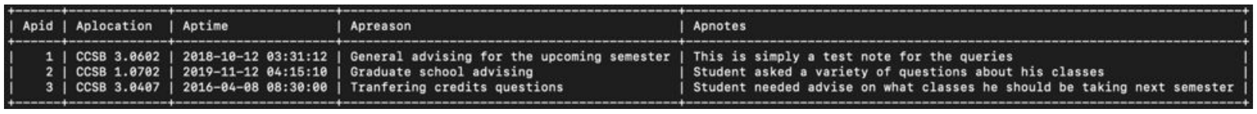
   INSERT INTO Appointment (Aplocation, Aptime, Apreason) VALUES ('CCSB 3.0602', '2018-10-12 03:31:12', 'General advising for the upcoming semester');



```
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
| Apid | Aplocation  | Aptime              | Apreason                                   | Apnotes                                                           |
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
|    1 | CCSB 3.0602 | 2018-10-12 03:31:12 | General advising for the upcoming semester | This is simply a test note for the queries                        |
|    2 | CCSB 1.0702 | 2019-11-12 04:15:10 | Graduate school advising                   | Student asked a variety of questions about his classes            |
|    3 | CCSB 3.0407 | 2016-04-08 08:30:00 | Tranfering credits questions               | Student needed advise on what classes he should be taking next semester |
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
```

6. The system shall keep track of each student's previous advising records, this record will be able to be updated as the semesters pass.

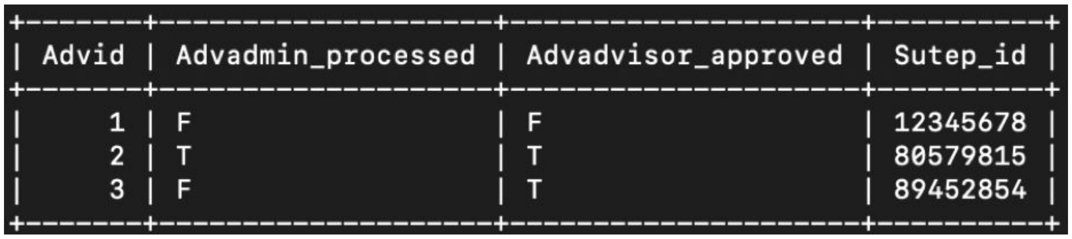
   SELECT * FROM appointment;



```
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
| Apid | Aplocation  | Aptime              | Apreason                                   | Apnotes                                                           |
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
|    1 | CCSB 3.0602 | 2018-10-12 03:31:12 | General advising for the upcoming semester | This is simply a test note for the queries                        |
|    2 | CCSB 1.0702 | 2019-11-12 04:15:10 | Graduate school advising                   | Student asked a variety of questions about his classes            |
|    3 | CCSB 3.0407 | 2016-04-08 08:30:00 | Tranfering credits questions               | Student needed advise on what classes he should be taking next semester |
+------+-------------+---------------------+--------------------------------------------+-------------------------------------------------------------------+
```

7. The system shall alert users that a particular student has completed their advising session. It will send the student a copy of their advising slip and notify the front desk admins that the student is cleared to enroll when the time comes.

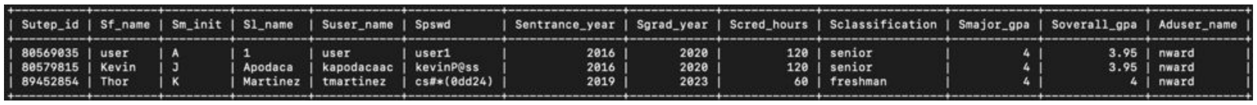
   SELECT * FROM advising_slip;



```
+-------+--------------------+---------------------+----------+
| Advid | Advadmin_processed | Advadvisor_approved | Sutep_id |
+-------+--------------------+---------------------+----------+
|     1 | F                  | F                   | 12345678 |
|     2 | T                  | T                   | 80579815 |
|     3 | F                  | T                   | 89452854 |
+-------+--------------------+---------------------+----------+
```

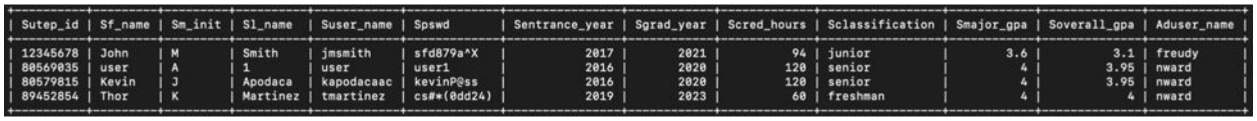
8. The system shall provide a way for UTEP CS students to select which courses they would like to take the coming semester.
9. The system shall allow advisors to see information about each of their students, but prevents students from seeing information about each other.

   SELECT * FROM student WHERE Aduser_name='nward';



```
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| Sutep_id | Sf_name | Sm_init | Sl_name  | Suser_name | Spswd       | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name |
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| 80569035 | user    | A       | 1        | user       | user1       |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
| 80579815 | Kevin   | J       | Apodaca  | kapodacaac | kevinP@ss   |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
| 89452854 | Thor    | K       | Martinez | tmartinez  | cs#*(0dd24) |           2019 |       2023 |          60 | freshman        |          4 |            4 | nward       |
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
```

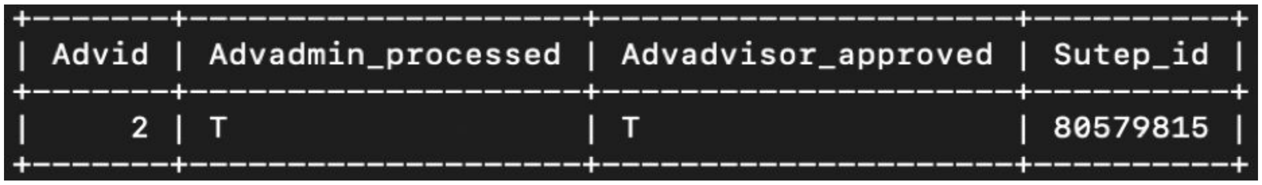
10. The system shall store records of students even after they have graduated, but will no longer allow students access to these records.
    SELECT * FROM student WHERE Sgrad_year >= 2019;



```
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| Sutep_id | Sf_name | Sm_init | Sl_name  | Suser_name | Spswd       | Sentrance_year | Sgrad_year | Scred_hours | Sclassification | Smajor_gpa | Soverall_gpa | Aduser_name |
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
| 12345678 | John    | M       | Smith    | jmsmith    | sfd879a^X   |           2017 |       2021 |          94 | junior          |        3.6 |          3.1 | freudy      |
| 80569035 | user    | A       | 1        | user       | user1       |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
| 80579815 | Kevin   | J       | Apodaca  | kapodacaac | kevinP@ss   |           2016 |       2020 |         120 | senior          |          4 |         3.95 | nward       |
| 89452854 | Thor    | K       | Martinez | tmartinez  | cs#*(0dd24) |           2019 |       2023 |          60 | freshman        |          4 |            4 | nward       |
+----------+---------+---------+----------+------------+-------------+----------------+------------+-------------+-----------------+------------+--------------+-------------+
```

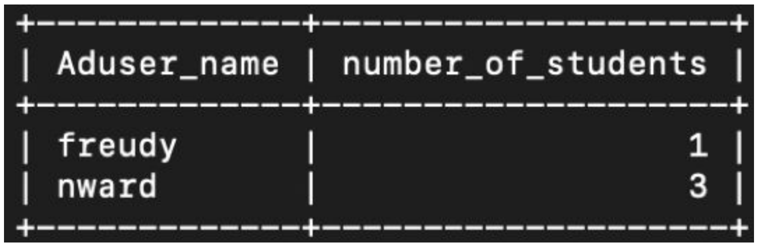
11. System shall have a way of generating reports with information such as how many students were advised, how many students each professor has been assigned, and how many students still need to be advised.
    How many students have been advised:

    SELECT * FROM advising_slip WHERE advadmin_processed='T';



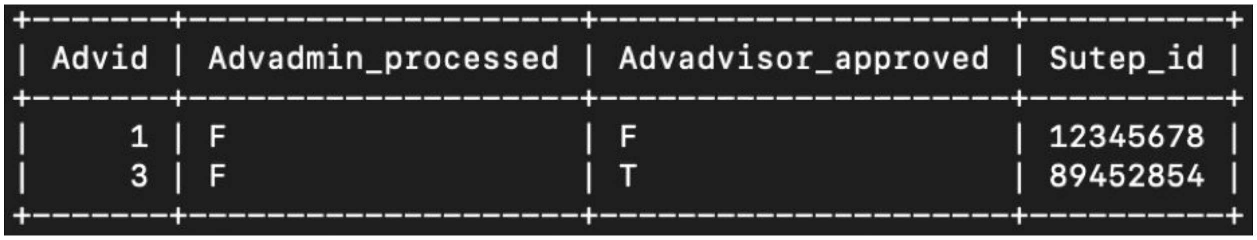
```
+-------+--------------------+---------------------+----------+
| Advid | Advadmin_processed | Advadvisor_approved | Sutep_id |
+-------+--------------------+---------------------+----------+
|     2 | T                  | T                   | 80579815 |
+-------+--------------------+---------------------+----------+
```

How many students each professor has:
SELECT Aduser_name, COUNT(*) AS number_of_students FROM student GROUP BY Aduser_name;



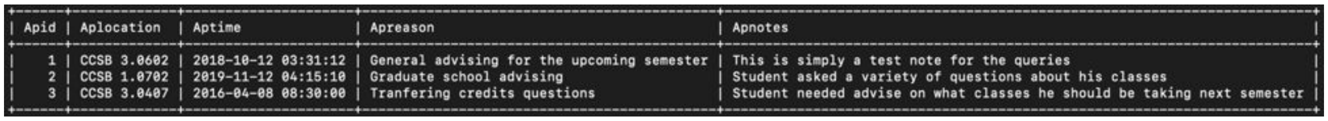
```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| freudy      |                  1 |
| nward       |                  3 |
+-------------+--------------------+
```

How many students still need to be advised:
SELECT * FROM advising_slip WHERE advadmin_processed='F';



```
+-------+--------------------+----------------------+----------+
| Advid | Advadmin_processed | Advadvisor_approved  | Sutep_id |
+-------+--------------------+----------------------+----------+
|     1 | F                  | F                    | 12345678 |
|     3 | F                  | T                    | 89452854 |
+-------+--------------------+----------------------+----------+
```

12. The system will provide a section for advisors to log information about the student advising session. Notes should not be deleted, but new notes can be added continuously.
    SELECT * FROM appointment;

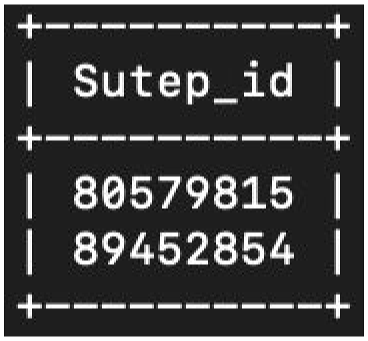


```
+------+---------------+---------------------+------------------------------------------+--------------------------------------------------------------+
| Apid | Aplocation    | Aptime              | Apreason                                 | Apnotes                                                      |
+------+---------------+---------------------+------------------------------------------+--------------------------------------------------------------+
|    1 | CCSB 3.0602 | 2018-10-12 03:31:12 | General advising for the upcoming semester | This is simply a test note for the queries                   |
|    2 | CCSB 1.0702 | 2019-11-12 04:15:10 | Graduate school advising                   | Student asked a variety of questions about his classes       |
|    3 | CCSB 3.0407 | 2016-04-08 08:30:00 | Tranfering credits questions               | Student needed advise on what classes he should be taking next semester |
+------+---------------+---------------------+------------------------------------------+--------------------------------------------------------------+
```

Views

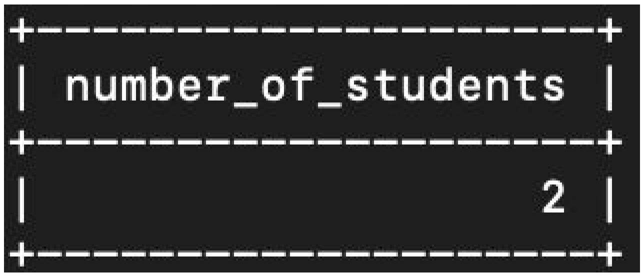
1. A view that will display the ID of all the current students that have completed their academic advising.

   CREATE VIEW studentsAdvised AS SELECT Sutep_id FROM advising_slip WHERE advadvisor_approved='T';



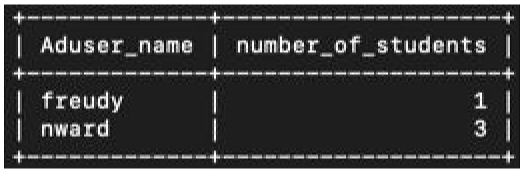
```
+----------+
| Sutep_id |
+----------+
| 80579815 |
| 89452854 |
+----------+
```

2. A view that will display a total count of how many students have been advised thus far.
   CREATE VIEW totalAdvised AS SELECT COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip WHERE Advadvisor_approved='T';


```
+----------------------+
| number_of_students |
+----------------------+
|                  2 |
+----------------------+
```

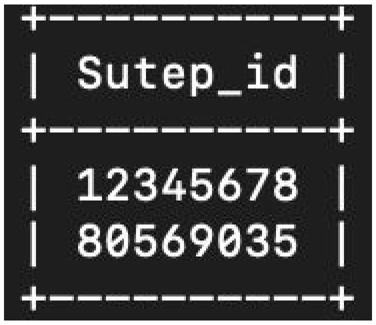
3. A view that will display how many students each professor has been assigned.

CREATE VIEW totalPerAdvisor AS SELECT Aduser_name, COUNT(*) AS number_of_students FROM student GROUP BY Aduser_name;


```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| freudy      |                  1 |
| nward       |                  3 |
+-------------+--------------------+
```

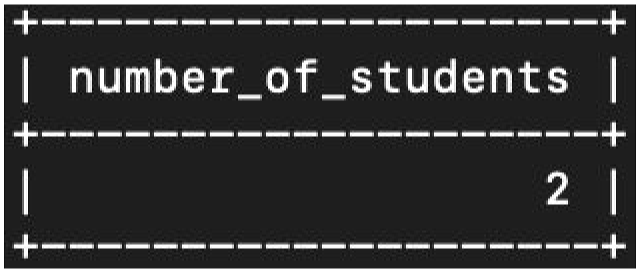
4. A view that will display information about which students that still need to be advised.

CREATE VIEW studentsNotAdvised AS SELECT Sutep_id FROM advising_slip WHERE advadvisor_approved='F';


```
+----------+
| Sutep_id |
+----------+
| 12345678 |
| 80569035 |
+----------+
```

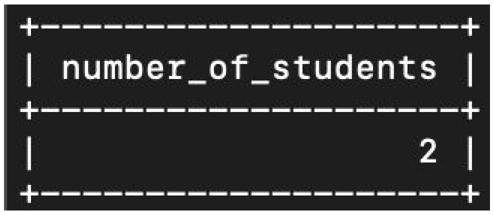
5. A view that will display a total count of how many students still need to be advised:

CREATE VIEW totalNotAdvised AS SELECT COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip WHERE Advadvisor_approved='F';



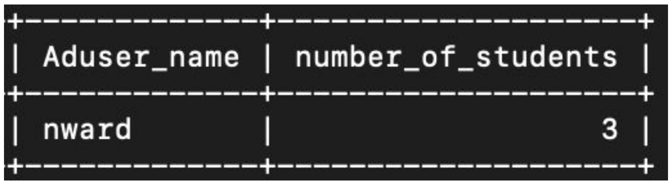
```
+--------------------+
| number_of_students |
+--------------------+
|                  2 |
+--------------------+
```

6. A view that displays the total count of how many students (out of the ones assigned to each advisor) have completed their academic advising.
   CREATE VIEW completedWithWard AS SELECT COUNT(Advadvisor_approved) AS
   number_of_students FROM advising_slip WHERE Advadvisor_approved='T' AND Sutep_id IN(SELECT Sutep_id FROM Student WHERE Aduser_name IN(SELECT Aduser_name FROM Advisor WHERE Aduser_name = 'nward'));



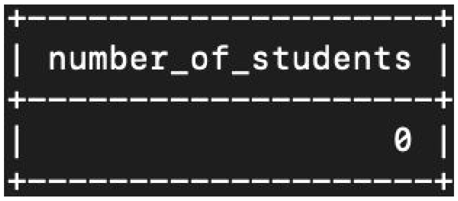
```
+--------------------+
| number_of_students |
+--------------------+
|                  2 |
+--------------------+
```

CREATE VIEW allFromWard AS SELECT Aduser_name, COUNT(*) AS total_students FROM student WHERE Aduser_name='nward';



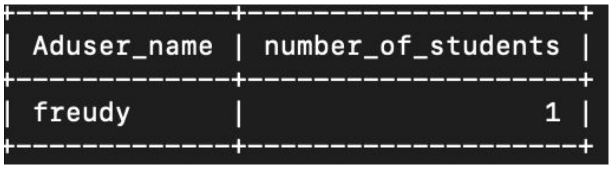
```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| nward       |                  3 |
+-------------+--------------------+
```

CREATE VIEW completedWithFreudy AS SELECT COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip WHERE Advadvisor_approved='T' AND Sutep_id IN(SELECT Sutep_id FROM Student WHERE Aduser_name IN(SELECT Aduser_name FROM Advisor WHERE Aduser_name = 'freudy'));



```
+--------------------+
| number_of_students |
+--------------------+
|                  0 |
+--------------------+
```

CREATE VIEW allFromFreudy AS SELECT Aduser_name, COUNT(*) AS total_students FROM student WHERE Aduser_name='freudy';



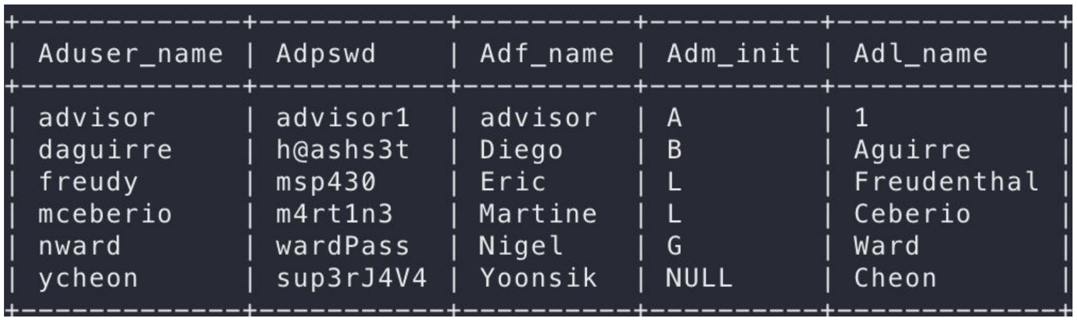
```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| freudy      |                  1 |
+-------------+--------------------+
```

Procedures & Triggers

Procedures

1. A procedure that will add a new advisor to the database, inserts all data stored from the advisor into the proper table.

```
DELIMITER //
	CREATE PROCEDURE insert_advisor(IN p_user_name char(50), IN p_pswd char(50), IN p_f_name char(50), IN p_m_init char(2), IN p_l_name char(50))
BEGIN
	INSERT INTO Advisor(Aduser_name, Adpswd, Adf_name, Adm_init, Adl_name) VALUES (p_user_name, p_pswd, p_f_name, p_m_init, p_l_name);
END //
DELIMITER ;
	CALL insert_advisor('mceberio', 'm4rt1n3', 'Martine', 'L', 'Ceberio');
```

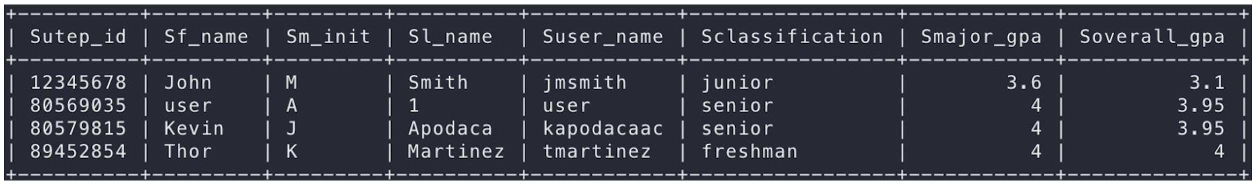


```
+-------------+-----------+----------+----------+------------+
| Aduser_name | Adpswd    | Adf_name | Adm_init | Adl_name   |
+-------------+-----------+----------+----------+------------+
| advisor     | advisor1  | advisor  | A        | 1          |
| daguirre    | h@ashs3t  | Diego    | B        | Aguirre    |
| freudy      | msp430    | Eric     | L        | Freudenthal |
| mceberio    | m4rt1n3   | Martine  | L        | Ceberio    |
| nward       | wardPass  | Nigel    | G        | Ward       |
| ycheon      | sup3rJ4V4 | Yoonsik  | NULL     | Cheon      |
+-------------+-----------+----------+----------+------------+
```

(a new advisor, Martine Ceberio, has been added to the Advisor's table)

2. A procedure that outputs potential applicants for scholarships. Staff can use this to find students that meet some GPA criteria to then email them about potential scholarships they can apply for.

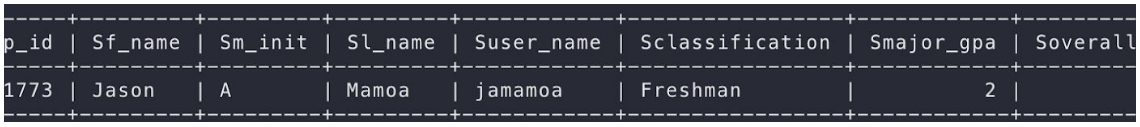
```
DELIMITER //
        CREATE PROCEDURE get_scholarship_applicants(IN target_GPA float)
BEGIN
        SELECT Sutep_id, Sf_name, Sm_init, Sl_name, Suser_name, Sclassification,
        Smajor_gpa, Soverall_gpa FROM Student WHERE Smajor_gpa >=
        target_GPA OR
        Soverall_GPA >= target_GPA;
END //
DELIMITER ;
        CALL get_scholarship_applicants('3.5');
```



```
+----------+---------+---------+----------+------------+----------------+------------+--------------+
| Sutep_id | Sf_name | Sm_init | Sl_name  | Suser_name | Sclassification | Smajor_gpa | Soverall_gpa |
+----------+---------+---------+----------+------------+----------------+------------+--------------+
| 12345678 | John    | M       | Smith    | jmsmith    | junior         |        3.6 |          3.1 |
| 80569035 | user    | A       | 1        | user       | senior         |          4 |         3.95 |
| 80579815 | Kevin   | J       | Apodaca  | kapodacaac | senior         |          4 |         3.95 |
| 89452854 | Thor    | K       | Martinez | tmartinez  | freshman       |          4 |            4 |
+----------+---------+---------+----------+------------+----------------+------------+--------------+
```

3. A procedure that outputs the relevant information about students that are considered on academic probation. Administration can use this to email those students to let them know the steps they can take to alleviate the problem so they are not in danger of losing their FAFSA or being dropped.

```
DELIMITER //
        CREATE PROCEDURE students_on_probation()
BEGIN
        SELECT Sutep_id, Sf_name, Sm_init, Sl_name, Suser_name, Sclassification,
        Smajor_gpa, Soverall_gpa FROM Student WHERE Smajor_gpa <= 2.0 OR
        Soverall_GPA <= 2.0;
END //
DELIMITER ;
        CALL students_on_probation();
```



```
----+---------+---------+---------+------------+-----------------+------------+---------
p_id | Sf_name | Sm_init | Sl_name | Suser_name | Sclassification | Smajor_gpa | Soverall
----+---------+---------+---------+------------+-----------------+------------+---------
1773 | Jason   | A       | Mamoa   | jamamoa    | Freshman        |          2 |
----+---------+---------+---------+------------+-----------------+------------+---------
```

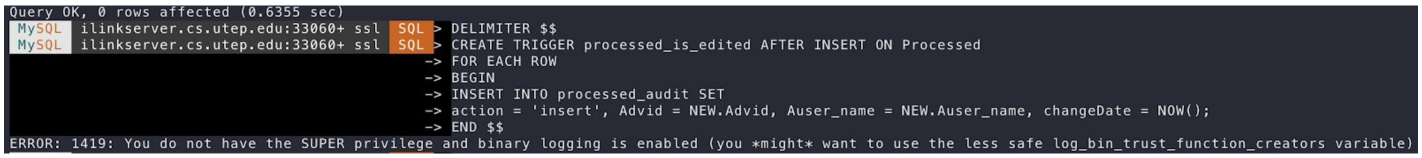
4. A procedure that handles SQL errors when trying to insert a new advisor into the Advisor table. If there is some exception thrown then the system will rollback to its previous state, otherwise it will insert the values into the database.

```
DELIMITER //

            CREATE PROCEDURE error_before_insert()

BEGIN

     DECLARE EXIT HANDLER FOR SQLEXCEPTION

     BEGIN

             ROLLBACK;

     END;

    START TRANSACTION;

     INSERT INTO Advisor VALUES (‘longpre’, L67x#1’, ‘Luc’, ‘M’, ‘Longpre’);

        COMMIT;

END //

DELIMITER ;

CALL error_before_insert();
```

Triggers

ISSUE: When attempting to create the triggers for this project we ran into the same issue that the class had when trying to do a previous homework. That is that we did not have permissions to run triggers on the database. This is the issue that was returned when triggers were created.

You do not have the SUPER privilege and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)



```
Query OK, 0 rows affected (0.6355 sec)
 MySQL  ilinkserver.cs.utep.edu:33060+ ssl  SQL > DELIMITER $$
 MySQL  ilinkserver.cs.utep.edu:33060+ ssl  SQL > CREATE TRIGGER processed_is_edited AFTER INSERT ON Processed
                                               -> FOR EACH ROW
                                               -> BEGIN
                                               -> INSERT INTO processed_audit SET
                                               -> action = 'insert', Advid = NEW.Advid, Auser_name = NEW.Auser_name, changeDate = NOW();
                                               -> END $$
ERROR: 1419: You do not have the SUPER privilege and binary logging is enabled (you *might* want to use the less safe log_bin_trust_function_creators variable)
```

1. A trigger that will keep a log of what updates have been made to the Advisor table. The client can use this every time there is a new advisor hired by the department that needs to be added to the database. This can also be used with the procedure that handles adding a new advisor.

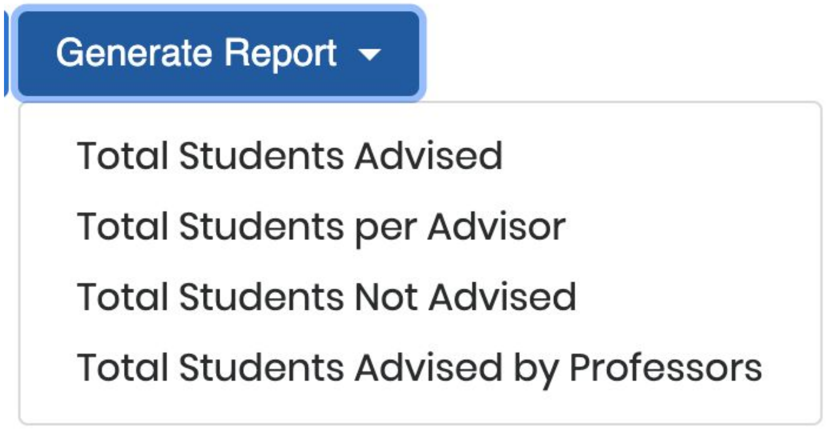
```
CREATE TABLE advisor_audit ( id INT AUTO_INCREMENT PRIMARY KEY,
        Aduser_name char(50) NOT NULL, Adf_name char(50), Adm_init char(50),
        Adl_name char(50), changeDate DATETIME DEFAULT NULL, action
        VARCHAR(50) DEFAULT NULL);

CREATE TRIGGER before_advisor_update
                BEFORE UPDATE ON Advisor
                FOR EACH ROW
INSERT INTO advisor_audit
SET action='update',
        Aduser_name = OLD.Aduser_name,
        Adf_name = OLD.Adf_name,
        Adm_init = OLD.Adm_init,
        Adl_name = OLD.Adl_name,
        changeDate = NOW();
```

2. A trigger that is used to keep track of who is making changes to the *processed* table. The client can use this to verify that the administration has processed the advising slip so that students are able to get their holds removed once registration time comes around.

```
CREATE Table processed_audit (Advid INT AUTO_INCREMENT PRIMARY KEY,

        Auser_name varchar(50) NOT NULL, changeDate DATETIME DEFAULT NULL,
        action varchar(50) DEFAULT NULL);

CREATE TRIGGER processed_is_edited AFTER INSERT ON Processed

FOR EACH ROW BEGIN INSERT INTO processed_audit

SET action = 'insert', Advid = NEW.Advid, Auser_name = NEW.Auser_name,

        changeDate = NOW();

END $$
```

Reports

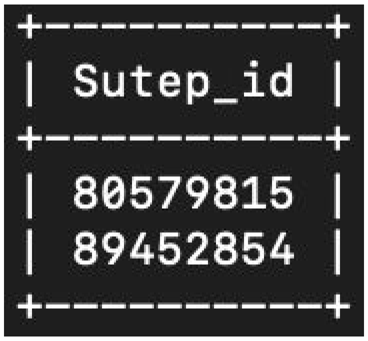
13. System shall have a way of generating reports with information such as how many students were advised, how many students each professor has been assigned, and how many students still need to be advised.


Generate Report
Total Students Advised
Total Students per Advisor
Total Students Not Advised
Total Students Advised by Professors

13a. Total Students Advised

All students that have been advised:

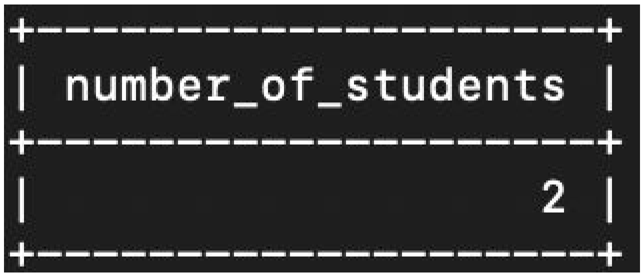
```
CREATE VIEW studentsAdvised AS SELECT Sutep_id FROM advising_slip
WHERE advadvisor_approved='T';
SELECT * FROM studentsAdvised;
```



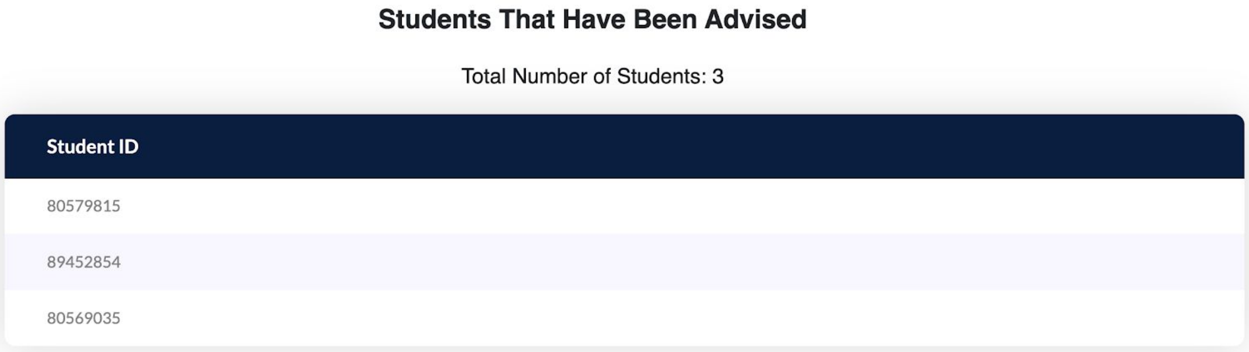
```
+----------+
| Sutep_id |
+----------+
| 80579815 |
| 89452854 |
+----------+
```

How many students have been advised:

CREATE VIEW totalAdvised AS SELECT COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip WHERE Advadvisor_approved='T';
SELECT * FROM totalAdvised;



```
+--------------------+
| number_of_students |
+--------------------+
|                  2 |
+--------------------+
```



Students That Have Been Advised

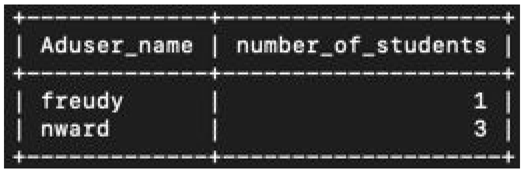
Total Number of Students: 3

Student ID
80579815
89452854
80569035

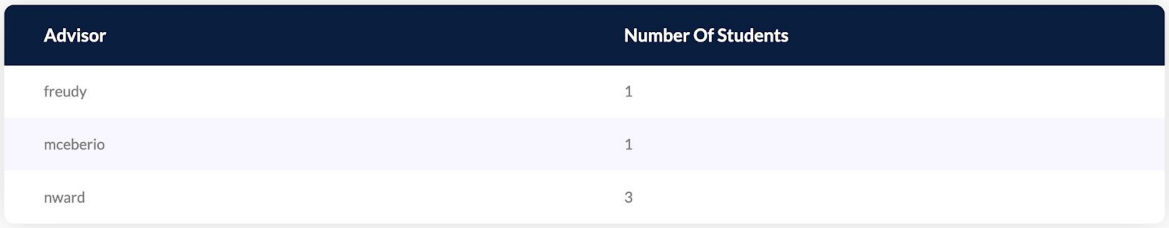
13b. Total Students Per Advisor

How many students each professor has:

CREATE VIEW totalPerAdvisor AS SELECT Aduser_name, COUNT(*) AS number_of_students FROM student GROUP BY Aduser_name;
SELECT * FROM totalPerAdvisor;



```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| freudy      |                  1 |
| nward       |                  3 |
+-------------+--------------------+
```



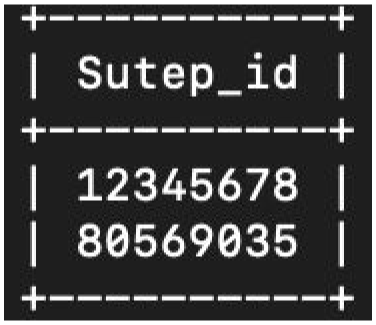
Total Number of Students per Advisor

Advisor	Number Of Students
freudy	1
mceberio	1
nward	3

13c. Total Students Not Advised

Students that still need to be advised:

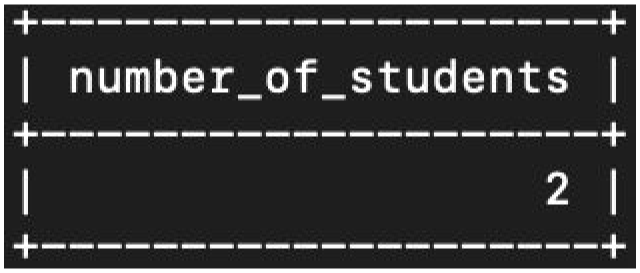
```
CREATE VIEW studentsNotAdvised AS SELECT Sutep_id FROM advising_slip
WHERE advadvisor_approved='F';
SELECT * FROM studentsNotAdvised;
```



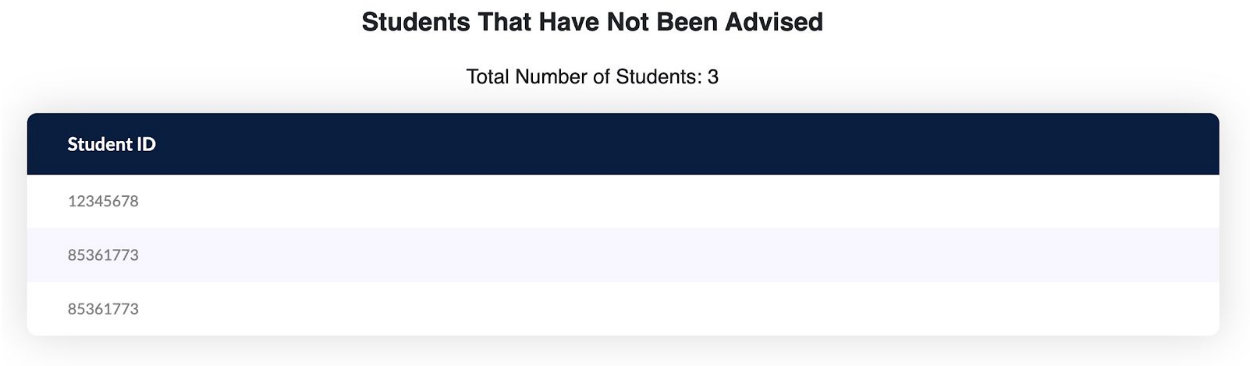
```
+----------+
| Sutep_id |
+----------+
| 12345678 |
| 80569035 |
+----------+
```

How many students still need to be advised:

```
CREATE VIEW totalNotAdvised AS SELECT COUNT(Advadvisor_approved) AS
number_of_students FROM advising_slip WHERE Advadvisor_approved='F';
SELECT * FROM totalNotAdvised;
```



```
+--------------------+
| number_of_students |
+--------------------+
|                  2 |
+--------------------+
```



Students That Have Not Been Advised

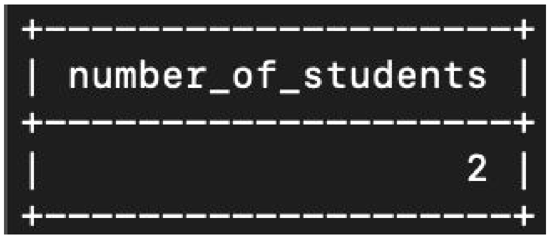
Total Number of Students: 3

Student ID
12345678
85361773
85361773

13d. Total Students Advised By Professor

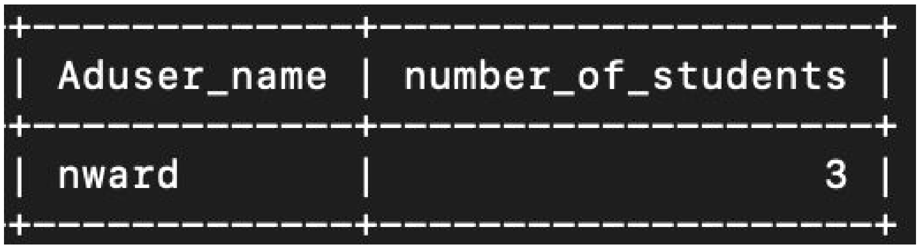
How many students out of the ones assigned to each advisor completed advising for that semester:

```
CREATE VIEW completedWithWard AS SELECT
COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip
WHERE Advadvisor_approved='T' AND Sutep_id IN(SELECT Sutep_id FROM
Student WHERE Aduser_name IN(SELECT Aduser_name FROM Advisor
WHERE Aduser_name = 'nward'));
SELECT * FROM completedWithWard;
```



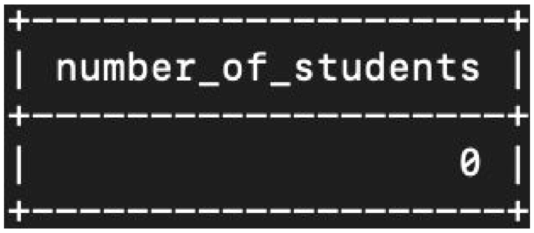
```
+--------------------+
| number_of_students |
+--------------------+
|                  2 |
+--------------------+
```

```
CREATE VIEW allFromWard AS SELECT Aduser_name, COUNT(*) AS
total_students FROM student WHERE Aduser_name='nward';
SELECT * FROM allFromWard;
```



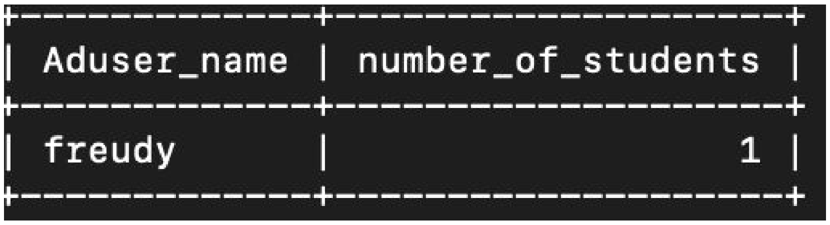
```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| nward       |                  3 |
+-------------+--------------------+
```

CREATE VIEW completedWithFreudy AS SELECT COUNT(Advadvisor_approved) AS number_of_students FROM advising_slip WHERE Advadvisor_approved='T' AND Sutep_id IN(SELECT Sutep_id FROM Student WHERE Aduser_name IN(SELECT Aduser_name FROM Advisor WHERE Aduser_name = 'freudy'));
SELECT * FROM completedWithFreudy;

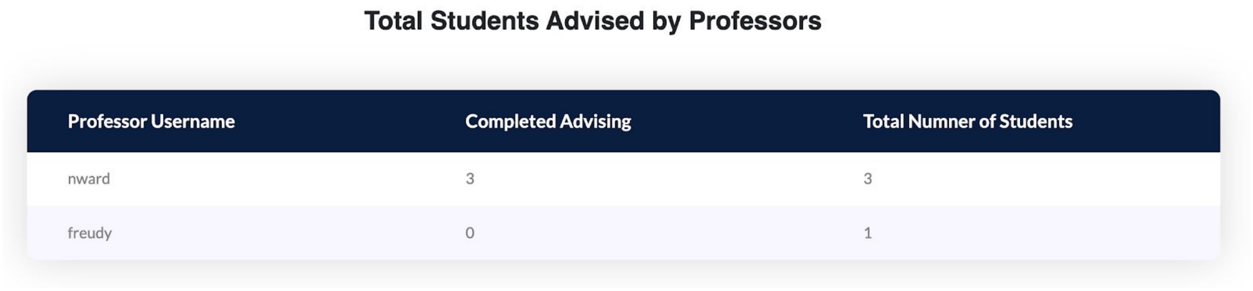


```
+--------------------+
| number_of_students |
+--------------------+
|                  0 |
+--------------------+
```

CREATE VIEW allFromFreudy AS SELECT Aduser_name, COUNT(*) AS total_students FROM student WHERE Aduser_name='freudy';
SELECT * FROM allFromFreudy;



```
+-------------+--------------------+
| Aduser_name | number_of_students |
+-------------+--------------------+
| freudy      |                  1 |
+-------------+--------------------+
```

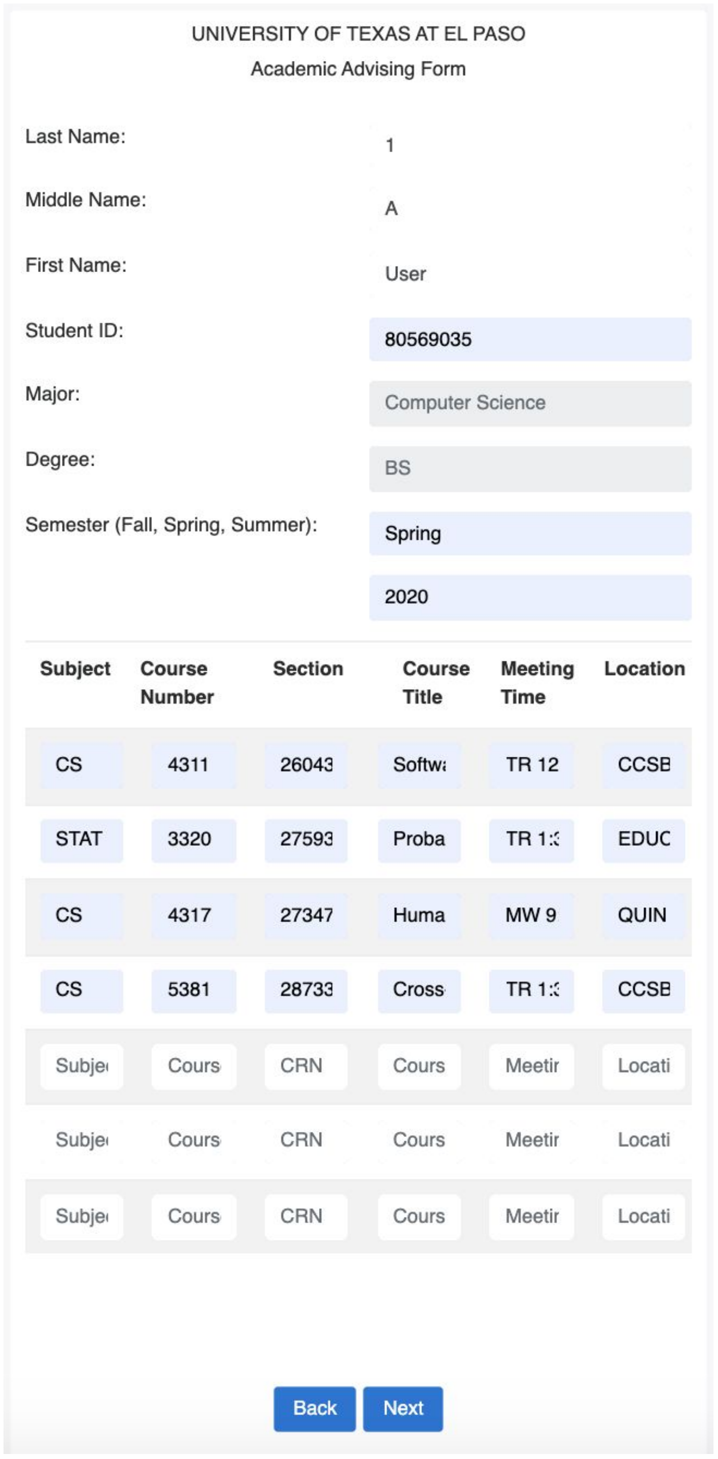


Total Students Advised by Professors

Professor Username	Completed Advising	Total Numner of Students
nward	3	3
freudy	0	1

Graphical User Interface

1. The students can insert information regarding their advising slip into the user interface shown below and the information will be inserted into the database. A new advising slip record will be created and inserted in the advising_slip table that will be tied to the student. A new section will be inserted for every section the student fills out in the section table that will also be tied to the student.



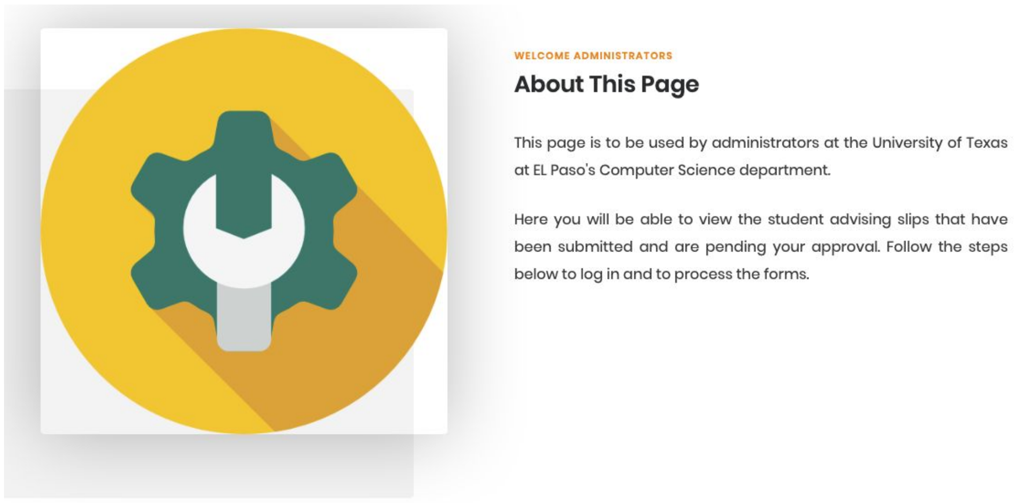
UNIVERSITY OF TEXAS AT EL PASO
Academic Advising Form

Last Name:	1
Middle Name:	A
First Name:	User
Student ID:	80569035
Major:	Computer Science
Degree:	BS
Semester (Fall, Spring, Summer):	Spring
	2020

Subject	Course Number	Section	Course Title	Meeting Time	Location
CS	4311	26043	Softw	TR 12	CCSB
STAT	3320	27593	Proba	TR 1:3	EDUC
CS	4317	27347	Huma	MW 9	QUIN
CS	5381	28733	Cross	TR 1:3	CCSB
Subje	Cours	CRN	Cours	Meetir	Locati
Subje	Cours	CRN	Cours	Meetir	Locati
Subje	Cours	CRN	Cours	Meetir	Locati

2. When an administrator is logged onto the system, they may select a dropdown to generate some specific report (in this case to see the total number of students that have been advised thus far) this will return both a count of students as well as a list of the UTEP

IDs of the students that have been advised. The images below show these operations being carried out on the GUI. Views were created to generate these reports as listed under the views section in this report and a selection on all the data for each pertaining view is queried to retrieve and display results.

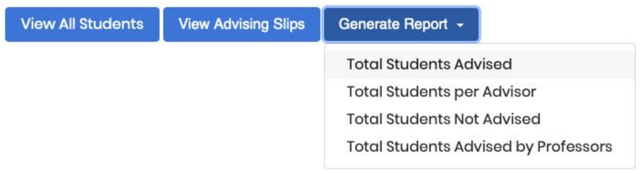


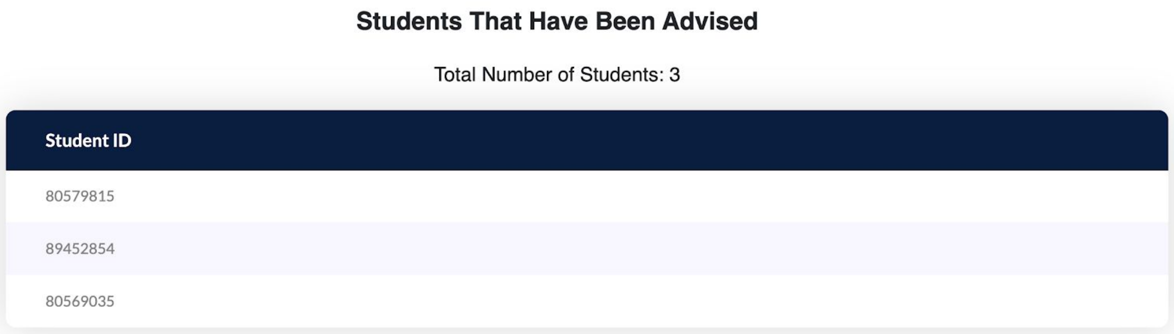
WELCOME ADMINISTRATORS

About This Page

This page is to be used by administrators at the University of Texas at EL Paso's Computer Science department.

Here you will be able to view the student advising slips that have been submitted and are pending your approval. Follow the steps below to log in and to process the forms.


View All Students
View Advising Slips
Generate Report
Total Students Advised
Total Students per Advisor
Total Students Not Advised
Total Students Advised by Professors



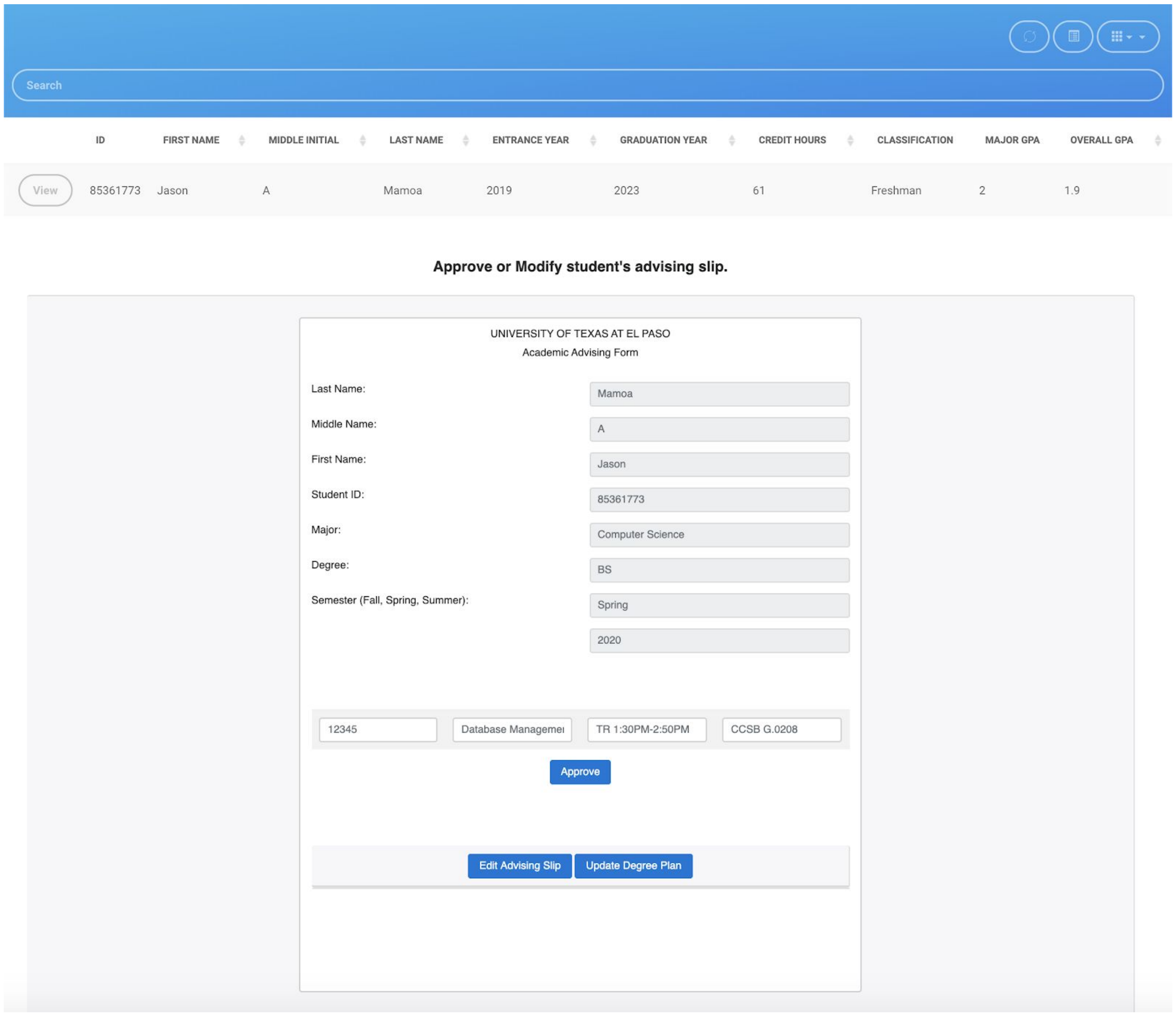
Students That Have Been Advised

Total Number of Students: 3

Student ID
80579815
89452854
80569035

3. When an advisor is logged onto the system, they may select to view a student's advising slip. The images below show these operations being carried out on the GUI. A professor

can approve which updates the advadmin_approved field in the database. A professor can also modify the courses which will update the section table in the database.


Search

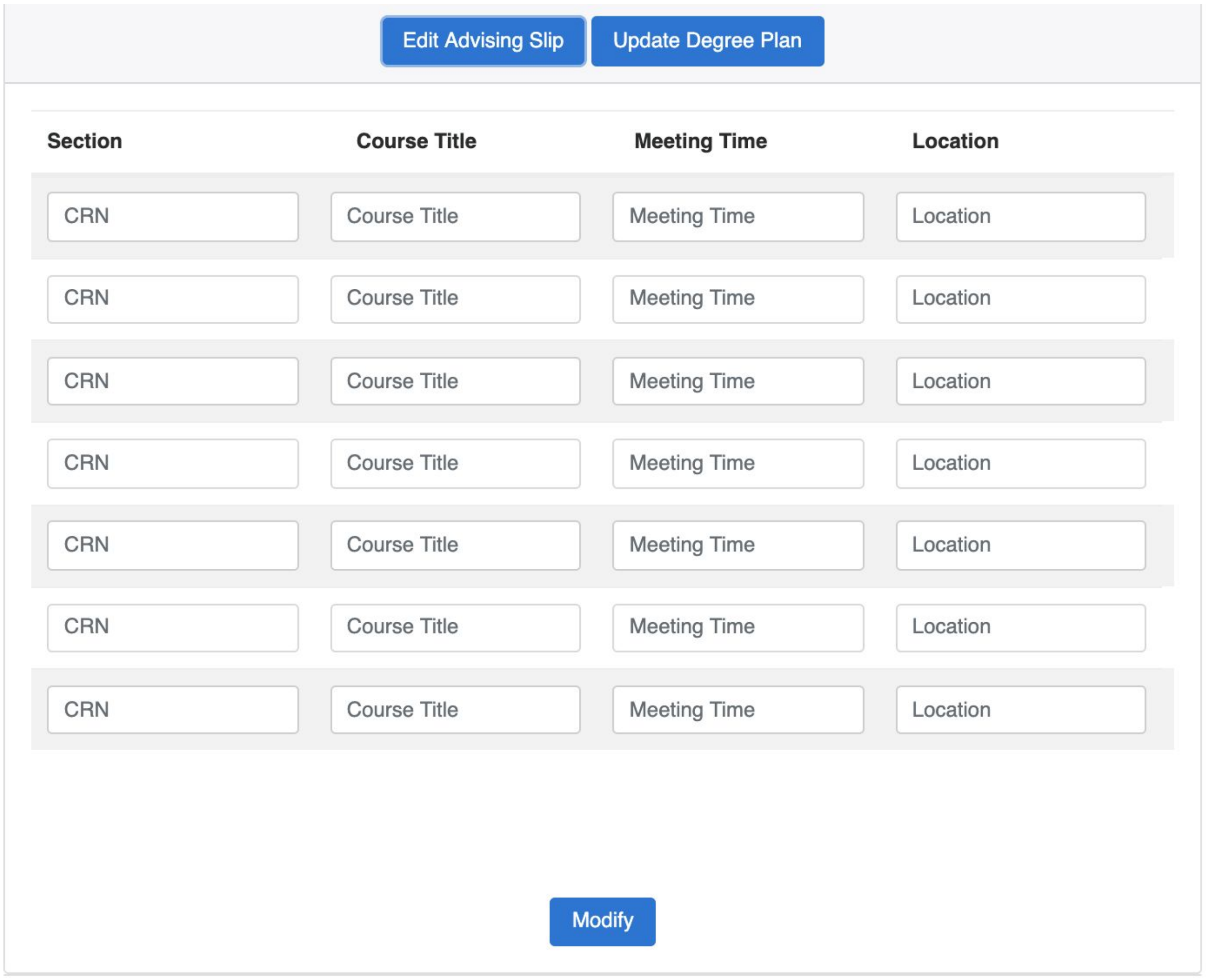
	ID	FIRST NAME	MIDDLE INITIAL	LAST NAME	ENTRANCE YEAR	GRADUATION YEAR	CREDIT HOURS	CLASSIFICATION	MAJOR GPA	OVERALL GPA
View	85361773	Jason	A	Mamoa	2019	2023	61	Freshman	2	1.9

Approve or Modify student's advising slip.

UNIVERSITY OF TEXAS AT EL PASO
Academic Advising Form

Last Name:	Mamoa
Middle Name:	A
First Name:	Jason
Student ID:	85361773
Major:	Computer Science
Degree:	BS
Semester (Fall, Spring, Summer):	Spring
	2020

12345	Database Manageme	TR 1:30PM-2:50PM	CCSB G.0208



Section	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location
CRN	Course Title	Meeting Time	Location

References

[1] Textbook: Elmasri, R., & Navathe, S. (2011). *Database systems* (Vol. 9). Boston, MA: Pearson Education.

[2] Jennifer and Ashley (2019, September 10). Class interview.

Appendix

1.3 Database Scope:

1.4 Requirements & Assumptions:

1.5 Updated Entity-Relationship Model:

1.6 Updated Relational Model:

1.7 Updated Normalized Schema:

1.8 Updated Database Schema in MySQL:

1.9 Updated Database Records:

1.10 SQL Queries:

1.11 Views:

1.12 Procedures, Triggers:

1.13: Reports:

1.14 Graphical User Interface:

1.15: References:

1.16: Appendix A:

Signatures: