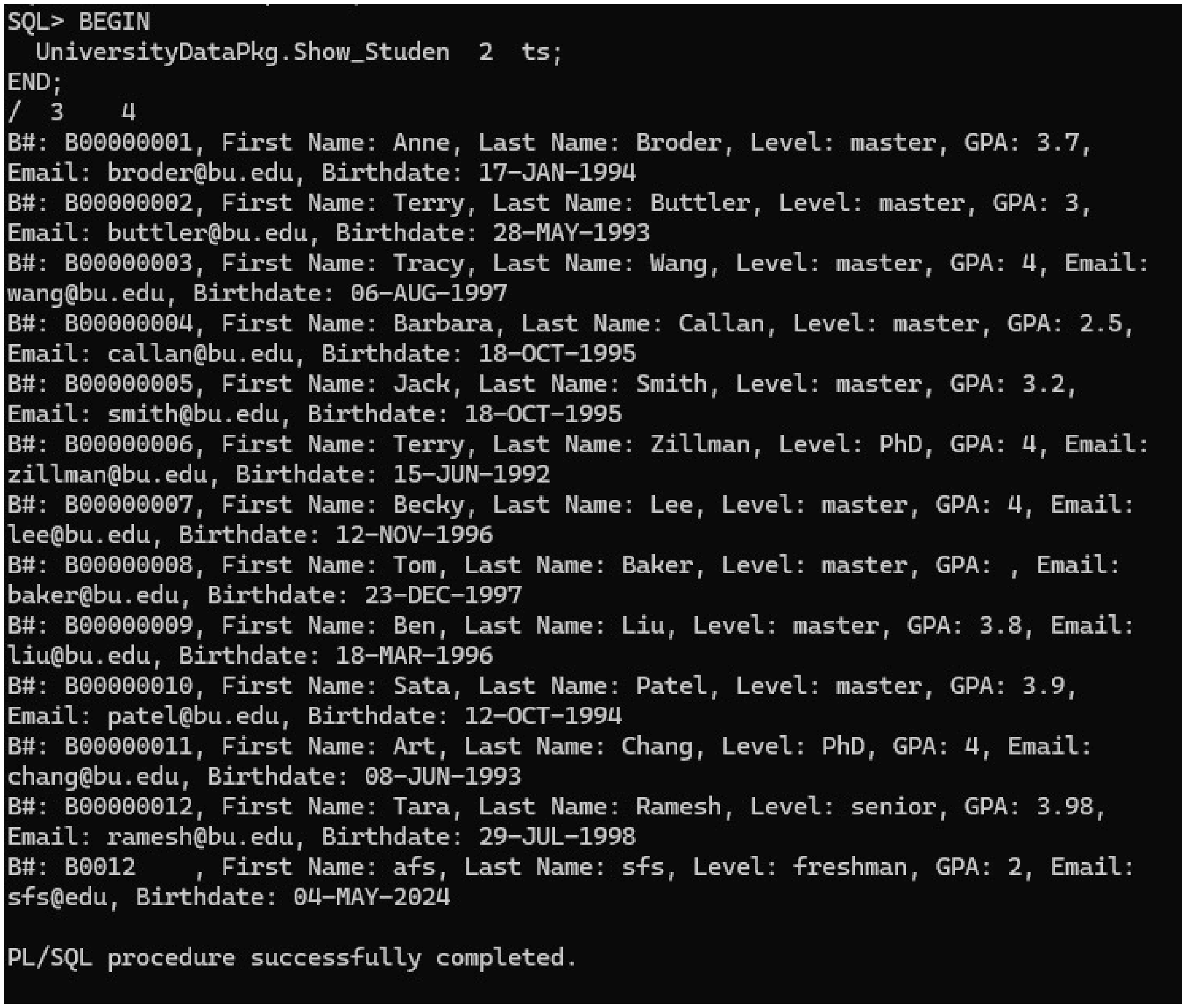
Student Registration System Report

Adam Cersosimo
Daniel Cersosimo
Rakesh Maddineni
Naga Himachand Pasupuleti

a.

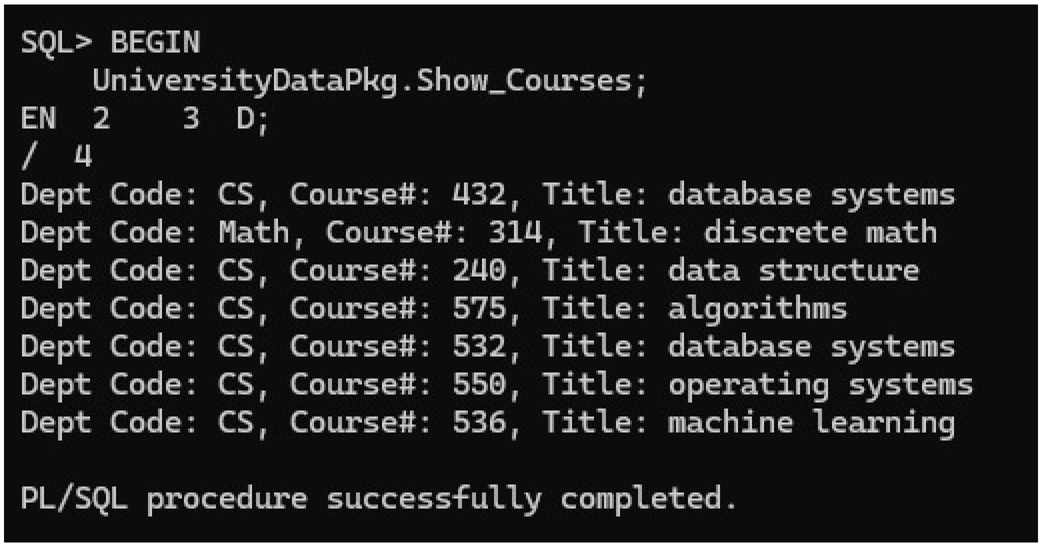
Here’s an overview of the functionalities implemented with screenshots of successful executions within the sqlplus terminal. We will demo how our application compliments constraints for each procedure in demo screenshots for enhanced brevity and clarity in our report.

- Procedures
    - `SHOW_STUDENTS`: We wrote this procedure to portray all student details including B#, name, level, GPA, email, and birthdate.



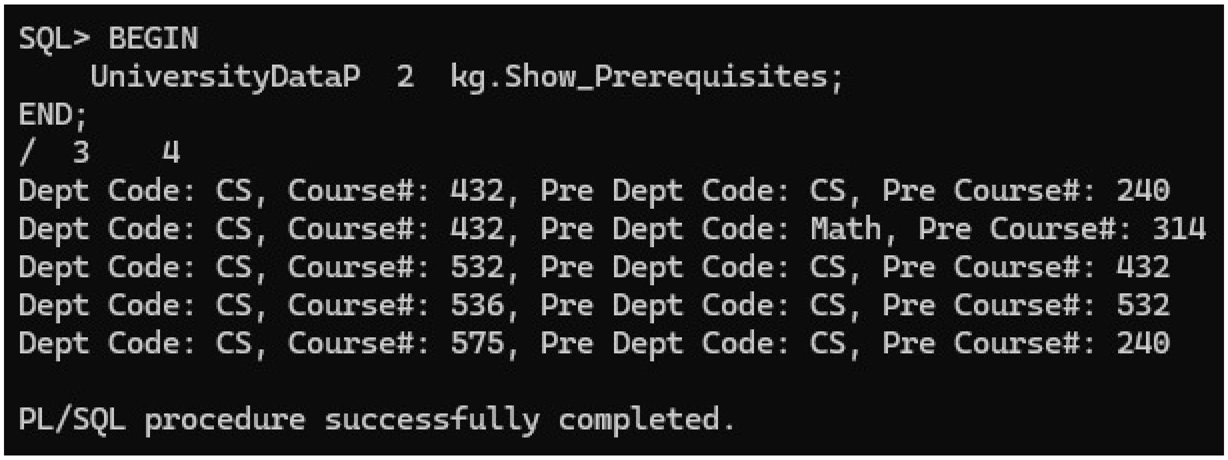
```
SQL> BEGIN
  UniversityDataPkg.Show_Studen  2  ts;
END;
/  3    4
B#: B00000001, First Name: Anne, Last Name: Broder, Level: master, GPA: 3.7,
Email: broder@bu.edu, Birthdate: 17-JAN-1994
B#: B00000002, First Name: Terry, Last Name: Buttler, Level: master, GPA: 3,
Email: buttler@bu.edu, Birthdate: 28-MAY-1993
B#: B00000003, First Name: Tracy, Last Name: Wang, Level: master, GPA: 4, Email:
wang@bu.edu, Birthdate: 06-AUG-1997
B#: B00000004, First Name: Barbara, Last Name: Callan, Level: master, GPA: 2.5,
Email: callan@bu.edu, Birthdate: 18-OCT-1995
B#: B00000005, First Name: Jack, Last Name: Smith, Level: master, GPA: 3.2,
Email: smith@bu.edu, Birthdate: 18-OCT-1995
B#: B00000006, First Name: Terry, Last Name: Zillman, Level: PhD, GPA: 4, Email:
zillman@bu.edu, Birthdate: 15-JUN-1992
B#: B00000007, First Name: Becky, Last Name: Lee, Level: master, GPA: 4, Email:
lee@bu.edu, Birthdate: 12-NOV-1996
B#: B00000008, First Name: Tom, Last Name: Baker, Level: master, GPA: , Email:
baker@bu.edu, Birthdate: 23-DEC-1997
B#: B00000009, First Name: Ben, Last Name: Liu, Level: master, GPA: 3.8, Email:
liu@bu.edu, Birthdate: 18-MAR-1996
B#: B00000010, First Name: Sata, Last Name: Patel, Level: master, GPA: 3.9,
Email: patel@bu.edu, Birthdate: 12-OCT-1994
B#: B00000011, First Name: Art, Last Name: Chang, Level: PhD, GPA: 4, Email:
chang@bu.edu, Birthdate: 08-JUN-1993
B#: B00000012, First Name: Tara, Last Name: Ramesh, Level: senior, GPA: 3.98,
Email: ramesh@bu.edu, Birthdate: 29-JUL-1998
B#: B0012    , First Name: afs, Last Name: sfs, Level: freshman, GPA: 2, Email:
sfs@edu, Birthdate: 04-MAY-2024

PL/SQL procedure successfully completed.
```

- `SHOW_COURSES`:This procedure was written to list all courses with department code, course number, and title.



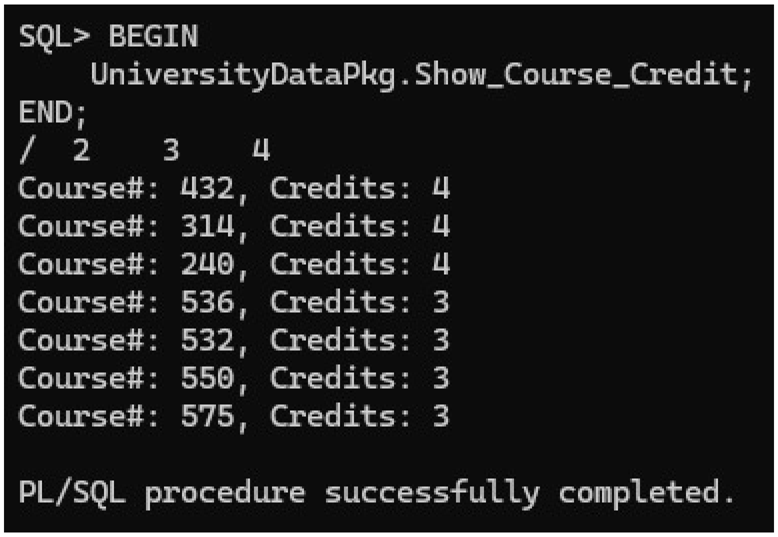
```
SQL> BEGIN
    UniversityDataPkg.Show_Courses;
EN  2    3  D;
/  4
Dept Code: CS, Course#: 432, Title: database systems
Dept Code: Math, Course#: 314, Title: discrete math
Dept Code: CS, Course#: 240, Title: data structure
Dept Code: CS, Course#: 575, Title: algorithms
Dept Code: CS, Course#: 532, Title: database systems
Dept Code: CS, Course#: 550, Title: operating systems
Dept Code: CS, Course#: 536, Title: machine learning

PL/SQL procedure successfully completed.
```

- `SHOW_PREREQUISITES`: We designed this procedure to display prerequisite relationships between courses.

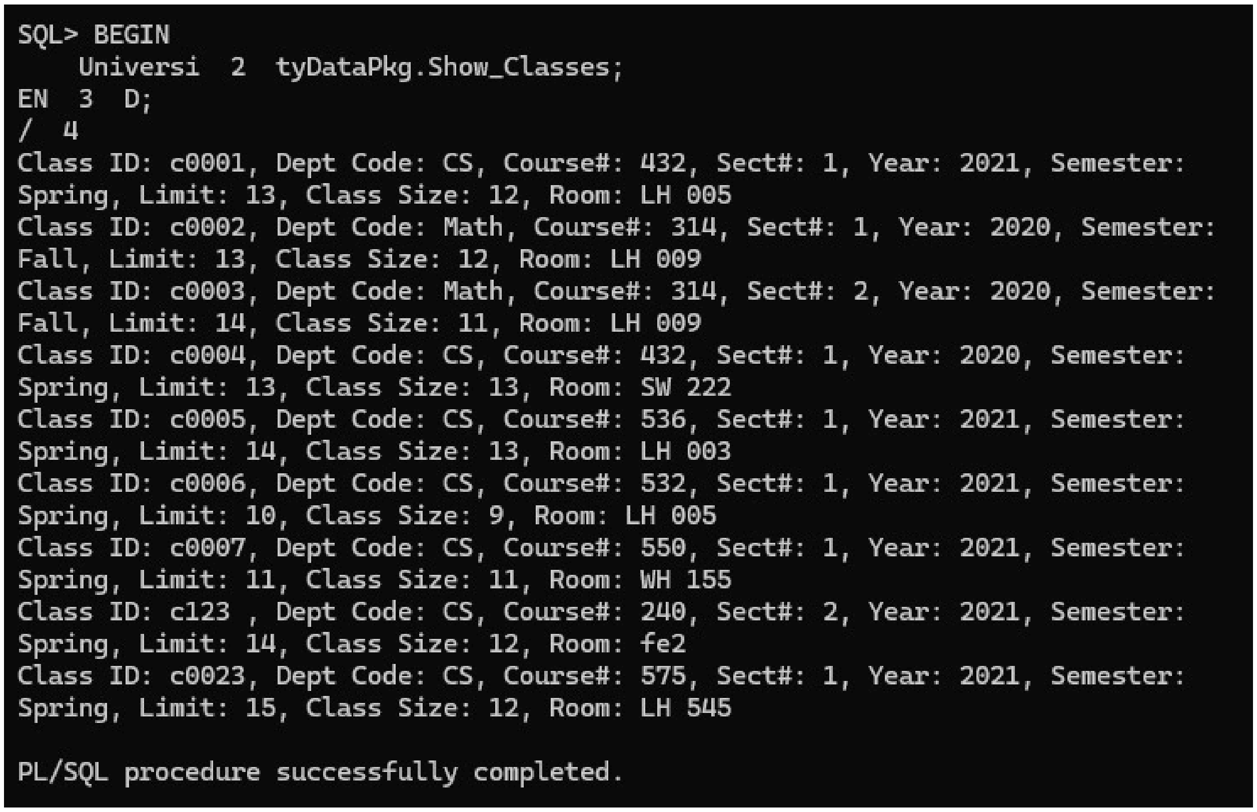


```
SQL> BEGIN
    UniversityDataP  2  kg.Show_Prerequisites;
END;
/  3    4
Dept Code: CS, Course#: 432, Pre Dept Code: CS, Pre Course#: 240
Dept Code: CS, Course#: 432, Pre Dept Code: Math, Pre Course#: 314
Dept Code: CS, Course#: 532, Pre Dept Code: CS, Pre Course#: 432
Dept Code: CS, Course#: 536, Pre Dept Code: CS, Pre Course#: 532
Dept Code: CS, Course#: 575, Pre Dept Code: CS, Pre Course#: 240

PL/SQL procedure successfully completed.
```

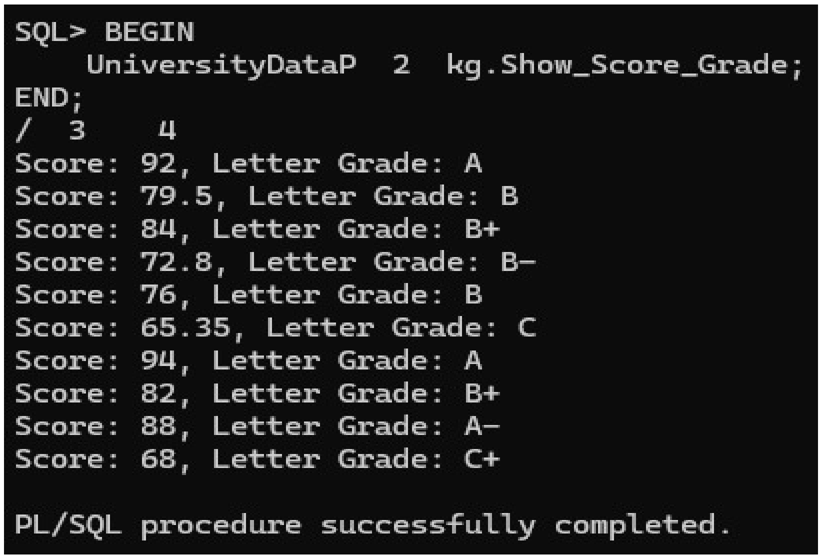
- `SHOW_COURSE_CREDIT`: This procedure was developed to show credit information for each course.


```
SQL> BEGIN
    UniversityDataPkg.Show_Course_Credit;
END;
/  2    3    4
Course#: 432, Credits: 4
Course#: 314, Credits: 4
Course#: 240, Credits: 4
Course#: 536, Credits: 3
Course#: 532, Credits: 3
Course#: 550, Credits: 3
Course#: 575, Credits: 3

PL/SQL procedure successfully completed.
```

- `SHOW_CLASSES`: Here we wrote a procedure which lists out all class sessions, including attributes such as class ID, department, course number, section, year, semester, class size, and room.

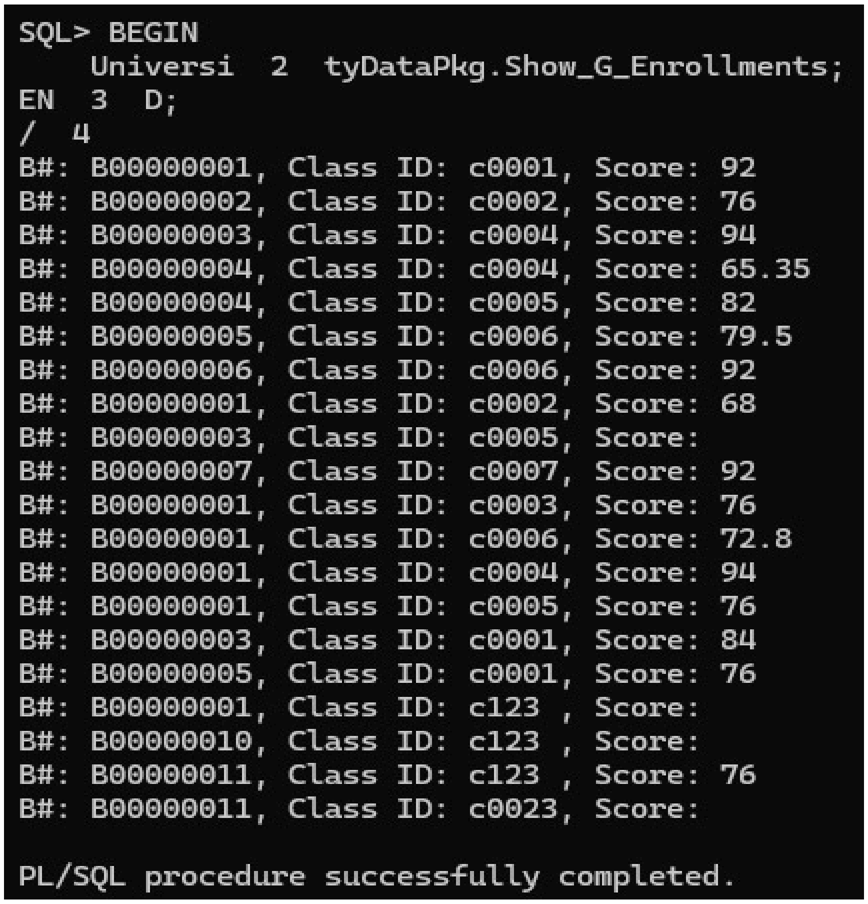

```
SQL> BEGIN
    Universi  2  tyDataPkg.Show_Classes;
EN  3  D;
/  4
Class ID: c0001, Dept Code: CS, Course#: 432, Sect#: 1, Year: 2021, Semester:
Spring, Limit: 13, Class Size: 12, Room: LH 005
Class ID: c0002, Dept Code: Math, Course#: 314, Sect#: 1, Year: 2020, Semester:
Fall, Limit: 13, Class Size: 12, Room: LH 009
Class ID: c0003, Dept Code: Math, Course#: 314, Sect#: 2, Year: 2020, Semester:
Fall, Limit: 14, Class Size: 11, Room: LH 009
Class ID: c0004, Dept Code: CS, Course#: 432, Sect#: 1, Year: 2020, Semester:
Spring, Limit: 13, Class Size: 13, Room: SW 222
Class ID: c0005, Dept Code: CS, Course#: 536, Sect#: 1, Year: 2021, Semester:
Spring, Limit: 14, Class Size: 13, Room: LH 003
Class ID: c0006, Dept Code: CS, Course#: 532, Sect#: 1, Year: 2021, Semester:
Spring, Limit: 10, Class Size: 9, Room: LH 005
Class ID: c0007, Dept Code: CS, Course#: 550, Sect#: 1, Year: 2021, Semester:
Spring, Limit: 11, Class Size: 11, Room: WH 155
Class ID: c123 , Dept Code: CS, Course#: 240, Sect#: 2, Year: 2021, Semester:
Spring, Limit: 14, Class Size: 12, Room: fe2
Class ID: c0023, Dept Code: CS, Course#: 575, Sect#: 1, Year: 2021, Semester:
Spring, Limit: 15, Class Size: 12, Room: LH 545

PL/SQL procedure successfully completed.
```

- `SHOW_SCORE_GRADE`: This procedure depicts scoring and grading standards.



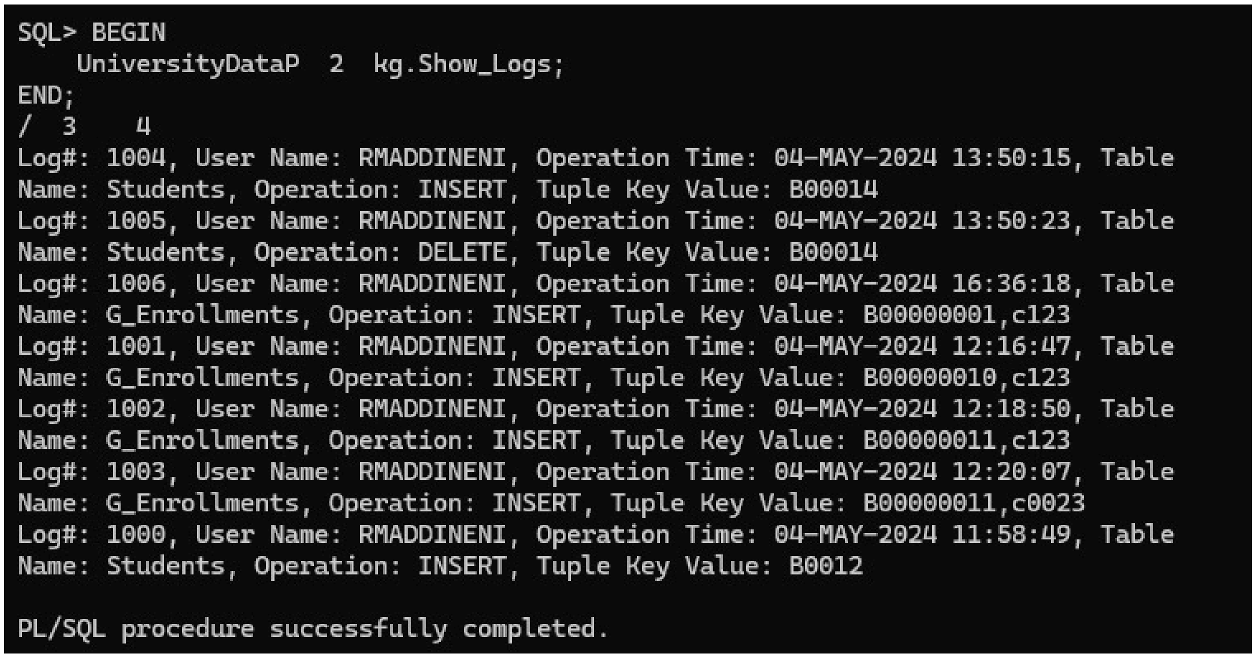
```
SQL> BEGIN
    UniversityDataP  2  kg.Show_Score_Grade;
END;
/  3    4
Score: 92, Letter Grade: A
Score: 79.5, Letter Grade: B
Score: 84, Letter Grade: B+
Score: 72.8, Letter Grade: B-
Score: 76, Letter Grade: B
Score: 65.35, Letter Grade: C
Score: 94, Letter Grade: A
Score: 82, Letter Grade: B+
Score: 88, Letter Grade: A-
Score: 68, Letter Grade: C+

PL/SQL procedure successfully completed.
```

- `SHOW_G_ENROLLMENTS`: We wrote this procedure to portray graduate student enrollments including student B#, class ID, and score.



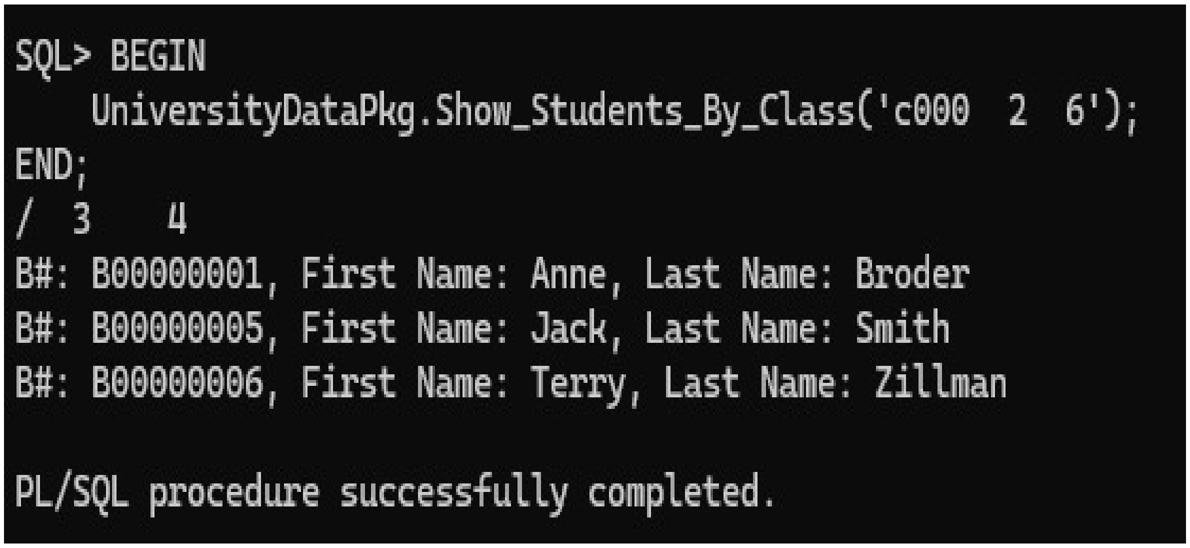
```
SQL> BEGIN
    Universi  2  tyDataPkg.Show_G_Enrollments;
EN  3  D;
/  4
B#: B00000001, Class ID: c0001, Score: 92
B#: B00000002, Class ID: c0002, Score: 76
B#: B00000003, Class ID: c0004, Score: 94
B#: B00000004, Class ID: c0004, Score: 65.35
B#: B00000004, Class ID: c0005, Score: 82
B#: B00000005, Class ID: c0006, Score: 79.5
B#: B00000006, Class ID: c0006, Score: 92
B#: B00000001, Class ID: c0002, Score: 68
B#: B00000003, Class ID: c0005, Score:
B#: B00000007, Class ID: c0007, Score: 92
B#: B00000001, Class ID: c0003, Score: 76
B#: B00000001, Class ID: c0006, Score: 72.8
B#: B00000001, Class ID: c0004, Score: 94
B#: B00000001, Class ID: c0005, Score: 76
B#: B00000003, Class ID: c0001, Score: 84
B#: B00000005, Class ID: c0001, Score: 76
B#: B00000001, Class ID: c123 , Score:
B#: B00000010, Class ID: c123 , Score:
B#: B00000011, Class ID: c123 , Score: 76
B#: B00000011, Class ID: c0023, Score:

PL/SQL procedure successfully completed.
```

- `SHOW_LOGS`: This procedure was included as a means of listing the entries from the logs table capturing user activities. This was a feature within the project requirements and is also present in the application which can be seen below.



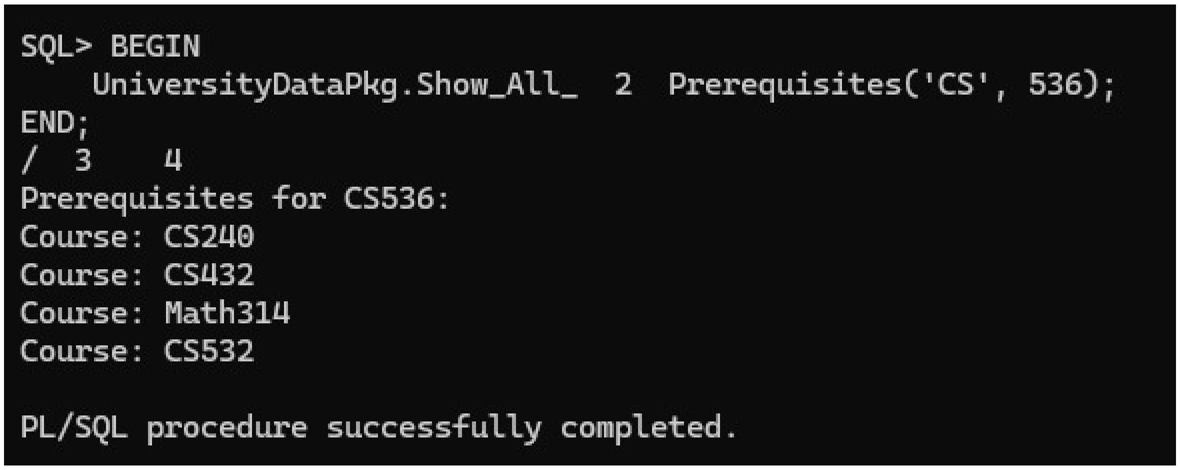
```
SQL> BEGIN
    UniversityDataP  2  kg.Show_Logs;
END;
/  3    4
Log#: 1004, User Name: RMADDINENI, Operation Time: 04-MAY-2024 13:50:15, Table
Name: Students, Operation: INSERT, Tuple Key Value: B00014
Log#: 1005, User Name: RMADDINENI, Operation Time: 04-MAY-2024 13:50:23, Table
Name: Students, Operation: DELETE, Tuple Key Value: B00014
Log#: 1006, User Name: RMADDINENI, Operation Time: 04-MAY-2024 16:36:18, Table
Name: G_Enrollments, Operation: INSERT, Tuple Key Value: B00000001,c123
Log#: 1001, User Name: RMADDINENI, Operation Time: 04-MAY-2024 12:16:47, Table
Name: G_Enrollments, Operation: INSERT, Tuple Key Value: B00000010,c123
Log#: 1002, User Name: RMADDINENI, Operation Time: 04-MAY-2024 12:18:50, Table
Name: G_Enrollments, Operation: INSERT, Tuple Key Value: B00000011,c123
Log#: 1003, User Name: RMADDINENI, Operation Time: 04-MAY-2024 12:20:07, Table
Name: G_Enrollments, Operation: INSERT, Tuple Key Value: B00000011,c0023
Log#: 1000, User Name: RMADDINENI, Operation Time: 04-MAY-2024 11:58:49, Table
Name: Students, Operation: INSERT, Tuple Key Value: B0012

PL/SQL procedure successfully completed.
```

- `SHOW_STUDENTS_BY_CLASS`: This procedure was penned to produce an output showing students enrolled in a specific class.



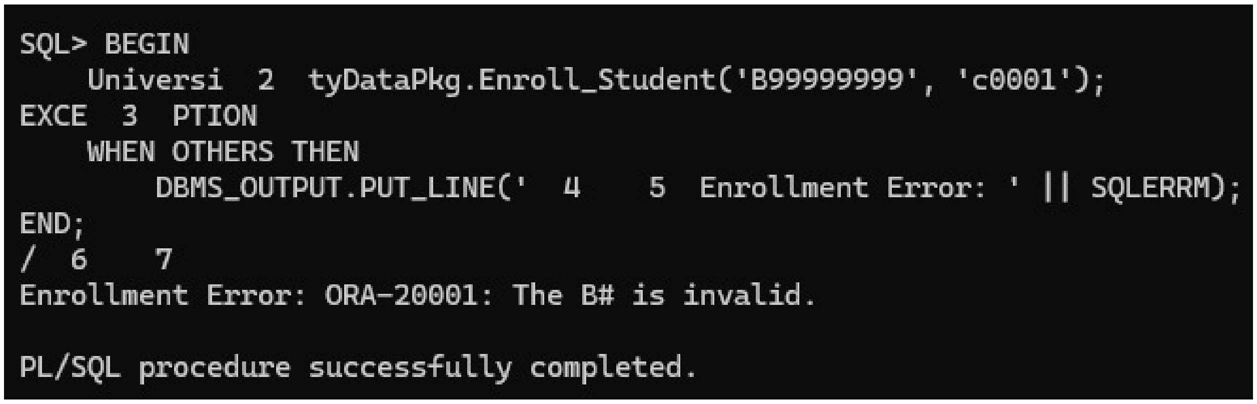
```
SQL> BEGIN
    UniversityDataPkg.Show_Students_By_Class('c000  2  6');
END;
/  3    4
B#: B00000001, First Name: Anne, Last Name: Broder
B#: B00000005, First Name: Jack, Last Name: Smith
B#: B00000006, First Name: Terry, Last Name: Zillman

PL/SQL procedure successfully completed.
```

- `SHOW_ALL_PREREQUISITES`: Here we wrote a procedure to effectively detail all prerequisites for a specific course, a component which bears a recursive element.

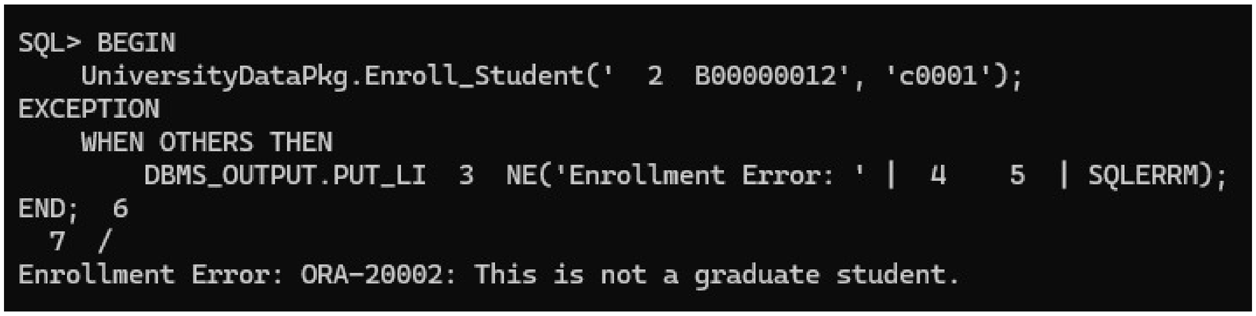


```
SQL> BEGIN
    UniversityDataPkg.Show_All_  2  Prerequisites('CS', 536);
END;
/  3    4
Prerequisites for CS536:
Course: CS240
Course: CS432
Course: Math314
Course: CS532

PL/SQL procedure successfully completed.
```

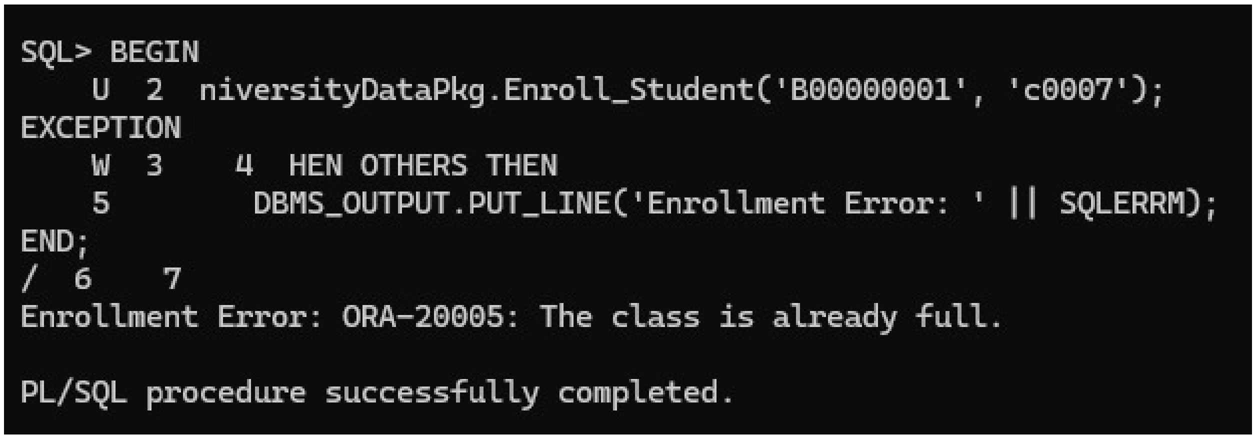
- `ENROLL_STUDENT`: This procedure is included to execute the enrollment of a graduate student in a class in a manner which effectively ensures integrity by checking prerequisites and availability. We will detail an array of constraint handling within this terminal view as they pertain to the enrolling a student functionality



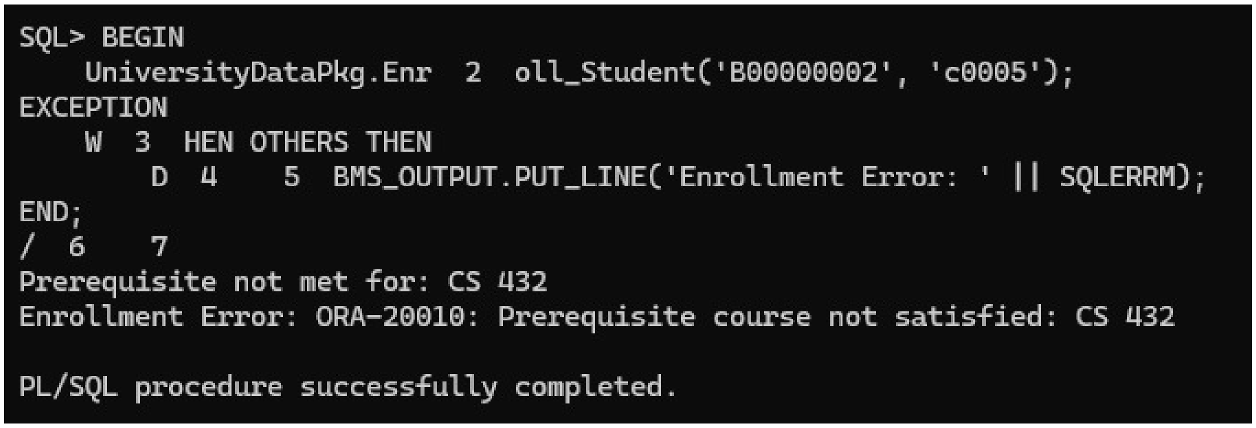
```
SQL> BEGIN
    Universi  2  tyDataPkg.Enroll_Student('B99999999', 'c0001');
EXCE  3  PTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('  4    5  Enrollment Error: ' || SQLERRM);
END;
/  6    7
Enrollment Error: ORA-20001: The B# is invalid.

PL/SQL procedure successfully completed.
```



```
SQL> BEGIN
    UniversityDataPkg.Enroll_Student('  2  B00000012', 'c0001');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LI  3  NE('Enrollment Error: ' |  4    5  | SQLERRM);
END;  6
  7  /
Enrollment Error: ORA-20002: This is not a graduate student.
```

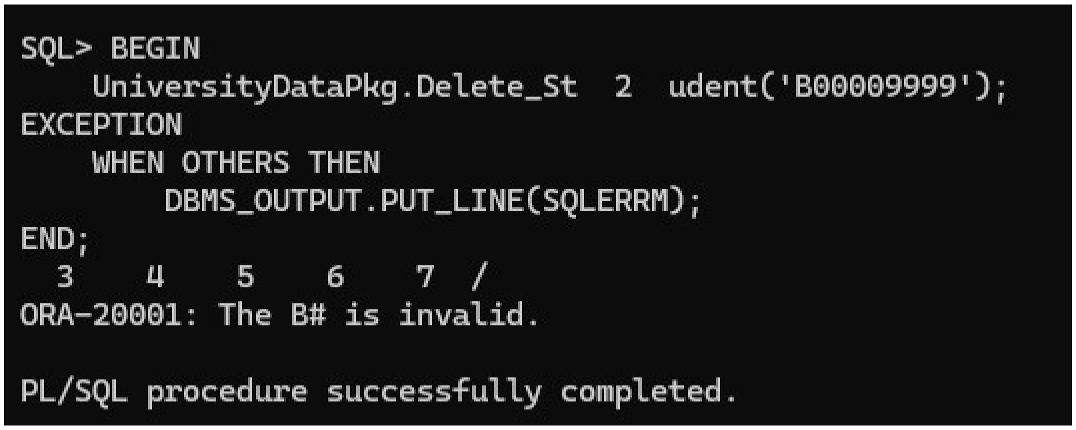


```
SQL> BEGIN
    UniversityDataPkg.Enroll_Student('B00000001', 'c0007');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Enrollment Error: ' || SQLERRM);
END;
/
Enrollment Error: ORA-20005: The class is already full.

PL/SQL procedure successfully completed.
```



```
SQL> BEGIN
    UniversityDataPkg.Enroll_Student('B00000002', 'c0005');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Enrollment Error: ' || SQLERRM);
END;
/
Prerequisite not met for: CS 432
Enrollment Error: ORA-20010: Prerequisite course not satisfied: CS 432

PL/SQL procedure successfully completed.
```

- `DELETE_STUDENT`: Another procedure which we developed which Deletes a student and all related enrollments which said student is taking part in. From a theoretical perspective, this essentially removes the presence of this student in the database in a cascading manner. We detail a scenario of an exception being properly handled below



```
SQL> BEGIN
    UniversityDataPkg.Delete_Student('B00009999');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
ORA-20001: The B# is invalid.

PL/SQL procedure successfully completed.
```

- Functions (Returning REF CURSORs for use with Java or other relevant clients)
    - Get_students, get_courses, get_prerequistes, get_course_credit, get_classes, get_score_grade, get_g_enrollments, get_logs are the functions implemented in the package.
    - Each of these functions opens a cursor to return results from their respective tables (STUDENTS, COURSES, PREREQUISITES, COURSE_CREDIT, CLASSES, SCORE_GRADE, G_ENROLLMENTS, LOGS) for further processing or display in client application.

Triggers

In order to execute the various functionalities which this system must support, when certain operations are executed we must ensure that integrity is maintained with respect to our database. In order to integrate features of this nature, we implement various triggers which, upon occurrence of certain operations, initiates reactionary action upon other aspects of the database. A few of the manners in which we utilize these triggers are as follows:

- Log Generation Triggers: Insert operations on students and enrollments, and delete operations on students and enrollments trigger log entries.
- Enrollment Count Adjustment: After inserting or deleting enrollment records, class sizes are updated accordingly.
- Cascade Delete: Automatically deletes enrollments when a student is deleted.

Sequences

The purpose of the use of sequences in this code is to generate values in the Log# column of the logs table. Our sequence starts at 1000 and then increments by 1 for every new row that is created in this table. This helps us do many things such as maintain uniqueness in this table which allows us to keep accurate records in an organized manner.

Summary of our PL/SQL

When considering the overall nature of our PL/SQL package and the myriad of procedures within, we can acknowledge how we implemented the various we can affirm the inclusion of a robust set of tools for managing a university database, with procedures for viewing and managing data, functions to integrate with external applications, and triggers to maintain data integrity and log activities automatically. The detailed implementation ensures consistency and maintains relationships between various entities such as students, courses, and classes. The usage of REF CURSORs allows for flexible integration with applications that require real-time data access.

Application code:

Here's an overview of the functionalities that are implemented in each component of the backend structure of our system:

- Model Classes
    - Class: Represents the class entity with properties such as class ID, course ID, instructor, schedule, etc.
    - Course: Defines a course with attributes like course ID, name, credits, and prerequisites.
    - Student: Represents a student, including properties such as student ID, name, major, and academic record.
    - Log: Used to record activities within the application, including user actions, timestamps, and the affected entities.
    - Enrollment: Models the relationship between students and classes, potentially including details like grades and enrollment status.

- Repositories
    - ClassRepository: Provides database access methods to manage class entities. This includes methods for finding, adding, updating, and deleting class records.
    - CourseRepository: Deals with persistence operations for course entities, including CRUD operations and possibly custom queries to fetch courses based on specific criteria like prerequisites.
    - StudentRepository: Manages database interactions for student entities, supporting basic CRUD operations and potentially complex queries related to student demographics or performance.
    - LogRepository: Used for accessing log entries from the database, which can include creating new logs or retrieving existing logs based on various filters.
    - EnrollmentRepository: Handles data access for enrollment records, providing methods to fetch and manage enrollment details, such as student-course relationships.

- Services
    - ClassService: Contains business logic related to class management, interfacing with the ClassRepository to serve data to the controller.
    - CourseService: Implements business rules for course management, utilizing the CourseRepository to interact with the database.
    - StudentService: Manages business operations related to students, working with the StudentRepository to perform data handling and transformations.
    - LogService: Manages the retrieval and storage of log data, ensuring that data fetched through the LogRepository is processed according to business requirements.
    - EnrollmentService: Handles the logic for student enrollments in courses, including validations, creating new enrollments, and modifying existing ones

- Controllers
    - ClassController: Manages HTTP requests related to classes, such as listing all classes, retrieving details of a specific class, creating new classes, updating existing classes, and deleting classes.
    - CourseController: Handles operations related to courses, including listing courses, retrieving course details, adding, updating, and deleting courses.
    - StudentController: Deals with student-related actions, such as listing students, retrieving student details, enrolling in courses, dropping courses, and managing student profiles.
    - LogController: Likely manages logging or audit trails, showing logs related to different operations within the system.
    - EnrollmentController: Manages enrollment-specific functionalities, including registering students for classes, updating enrollment details, and possibly handling grading.

Overall Summary

These components together form a comprehensive system for managing a university's operations digitally. Controllers serve as the entry point for HTTP requests, interacting with services that encapsulate the logic which our system supports, from which it communicates with repositories for data persistence. The system is designed to handle all aspects of university management, from course and class scheduling to student enrollment and tracking operational logs for audit and troubleshooting purposes. In sum, we have successfully developed an application which provides the multitude of functionalities as discussed in the project requirements while producing an end product of a robust, efficient, and user friendly web page application.

b.

Project Meeting Schedule Breakdown:

April 10th - This was our initial meeting where we set a preliminary course of action regarding how we would plan on generally completing the project and its various extenuating components from a primarily temporal perspective. We also briefly discussed individual strengths and how this may be leveraged.

April 15th - In this meeting, we broke down the project description together, identifying key aspects of the requirements and outlining a viable course of action to complete each in alignment with out time constraints

April 22nd - This was the day of the mid project meeting and therefore, we utilized this gathering to have another meeting where we reflected on our current progress, assessed what must be fixed, aspects of the procedures not yet completed, and how to progress going forward with producing a working application harnessing our current progress.

May 4th - Final meeting to discuss any individual ideas pertaining to our consolidated application which we had completed. We went through the project requirements again in an effort to reaffirm our completed projects alignment with said criteria.

c.

Stemming from our above meetings and in conjunction with collaboration throughout the month, we designed and executed a plan to successfully develop this application. Our plan was to initially focus on working through the procedures as these serve as pillars which sustain the functionality of the entire application. Following this, our next goal was to work through the JDBC configuration to construct a menu driven interface which allows for a hypothetical user to conduct SRS operations by configuring their commands in said interface to call specific procedures as defined above. We sought to reach this point by the mid project meeting which we were able to achieve aside from some specific constraints not implemented yet with regards to a couple of their procedures. Moving forward, we cleaned up these misalignments and proceeded to develop a means of building a web application which provides an enhancement upon the menu driven interface, one which exhibited the same functionality however with improved aesthetic and user friendly design. As is typically expected and realized, execution often deviates in some capacity from a designed plan and ours was no exception with an instance of this occurring with regards to the procedures, an aspect of this project which took a sizable portion of time. Overall though, keeping potential occurrences of this nature in mind, we focused on keeping our plan flexible to be able to dynamically adjust as during execution.

d.

In our early meetings, we decided to opt for a collaborative approach with each of us working on each portion of the project in alignment with the plan. This deviates from “divide and conquer” approaches however we saw this as best due to our respective strengths and weaknesses not lending towards any substantial disparities in which tasks would be more suitable for any specific party. In addition, we felt that taking this collaborative approach would allow for each of us to remain in tune with each aspect of the progress for the project, effectively rendering each of us as quite knowledgeable in the myriad of components within our finalized application.

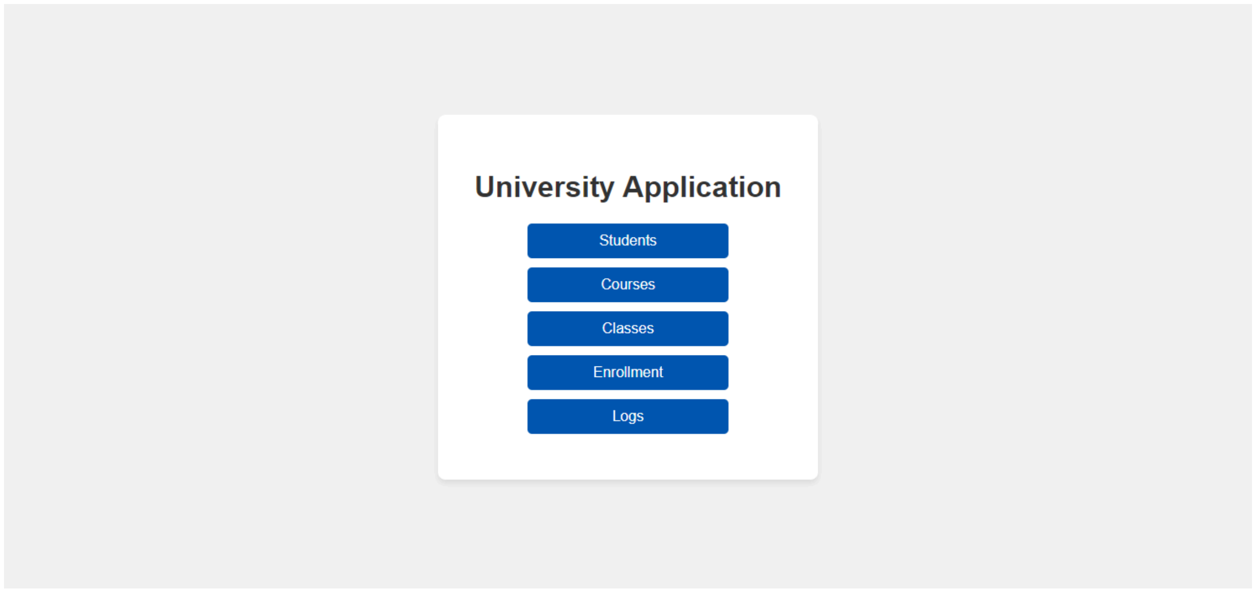
e.

We believe that a score of 1 would be suitable for our project as we communicated well, understood each other's unique situations/strengths, and collaboratively facilitated these acknowledgements in a manner which was conducive to an effective and efficiently planned/executed project. One point of note which produced some difficulty could be discrepancies in schedules and varying methods of balancing work between our courses. This is of course a natural situation which, due to our excellent communication, we were able to surpass to work well together and produce this application. Overall, we enjoyed working together and would definitely be open to future collaboration going forward.

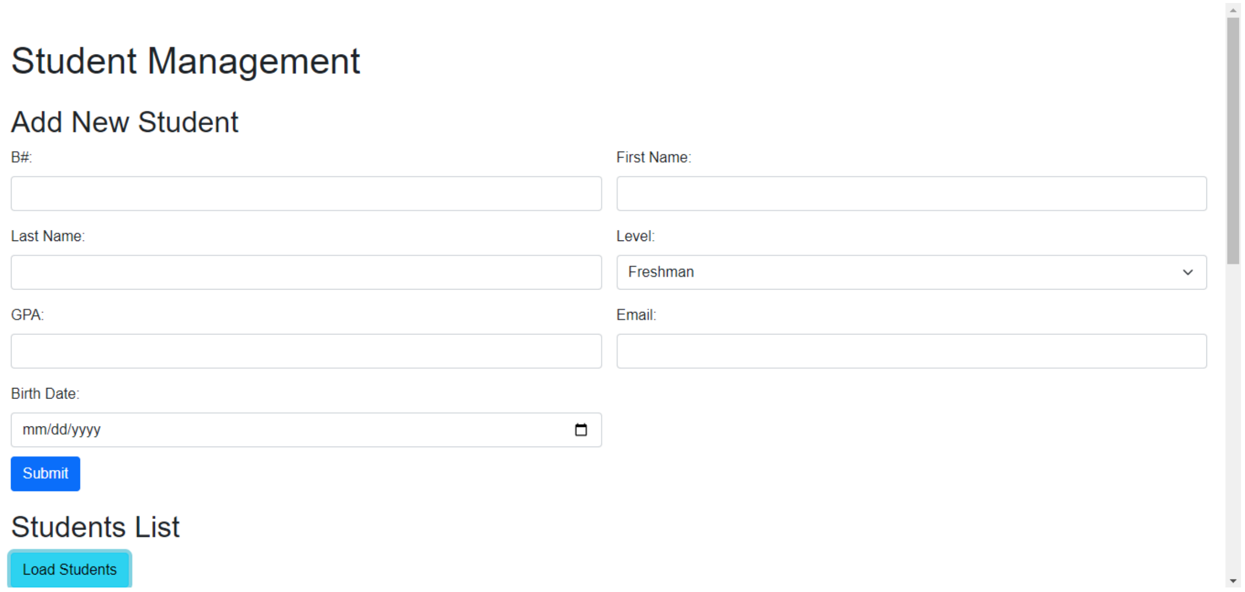
Demo Via Screenshots

- The following section will provide a thorough demonstration of our application through the usage of extensive screenshots and discussions pertaining to the array of functionalities and support for the intricacies of this system.

<u>Homepage:</u>



Students Page:



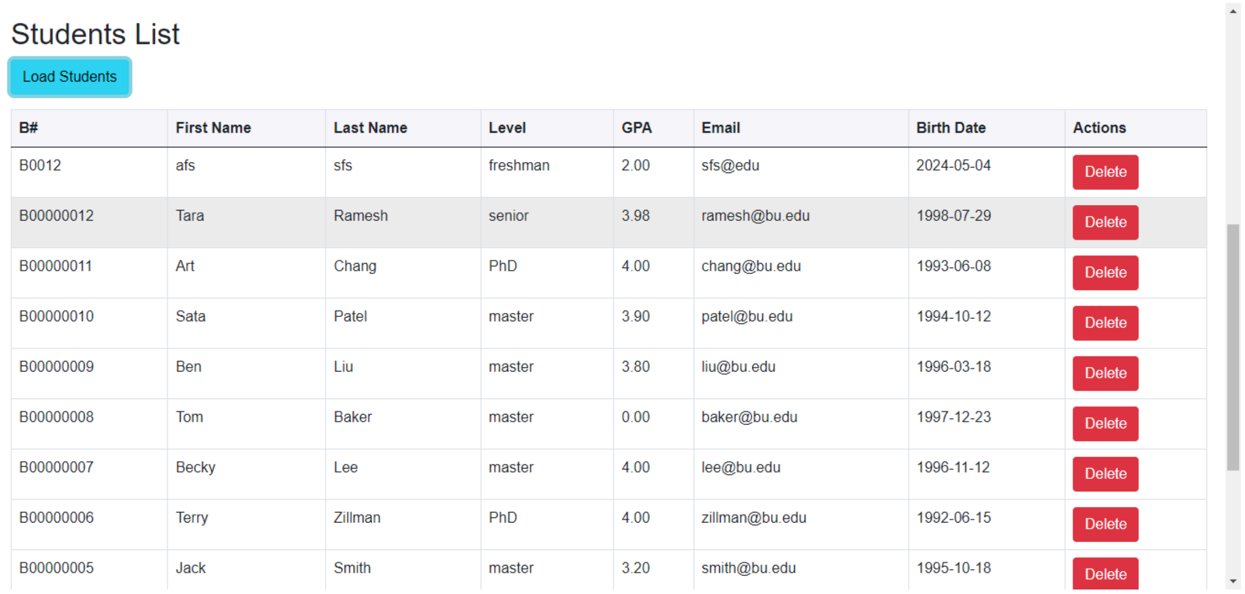
Student Management

Add New Student

B#:	First Name:
Last Name:	Level: Freshman
GPA:	Email:
Birth Date: mm/dd/yyyy	

Students List

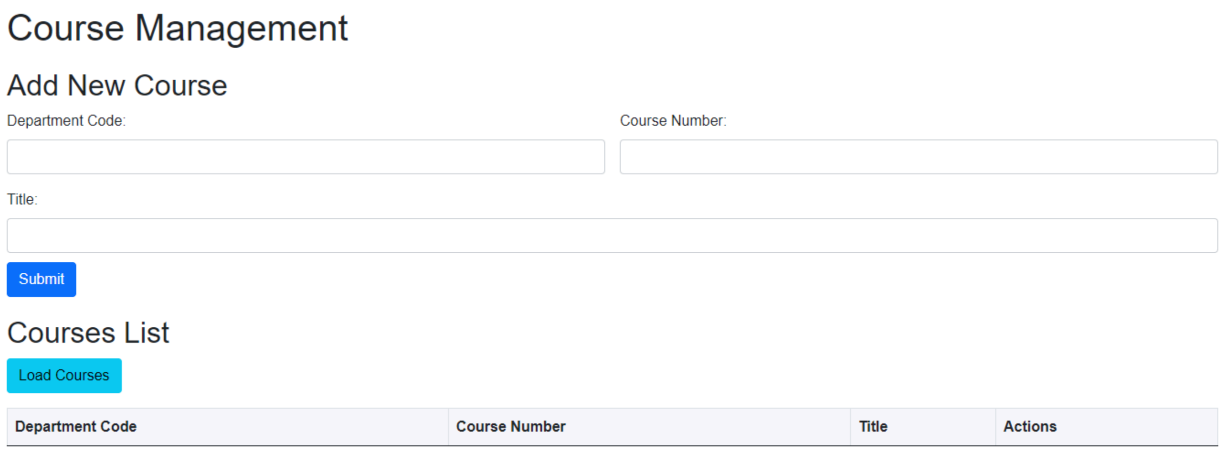
Loaded students on Student page:



Students List

B#	First Name	Last Name	Level	GPA	Email	Birth Date	Actions
B0012	afs	sfs	freshman	2.00	sfs@edu	2024-05-04	Delete
B00000012	Tara	Ramesh	senior	3.98	ramesh@bu.edu	1998-07-29	Delete
B00000011	Art	Chang	PhD	4.00	chang@bu.edu	1993-06-08	Delete
B00000010	Sata	Patel	master	3.90	patel@bu.edu	1994-10-12	Delete
B00000009	Ben	Liu	master	3.80	liu@bu.edu	1996-03-18	Delete
B00000008	Tom	Baker	master	0.00	baker@bu.edu	1997-12-23	Delete
B00000007	Becky	Lee	master	4.00	lee@bu.edu	1996-11-12	Delete
B00000006	Terry	Zillman	PhD	4.00	zillman@bu.edu	1992-06-15	Delete
B00000005	Jack	Smith	master	3.20	smith@bu.edu	1995-10-18	Delete

Courses page:



Course Management

Add New Course

Department Code:

Course Number:

Title:

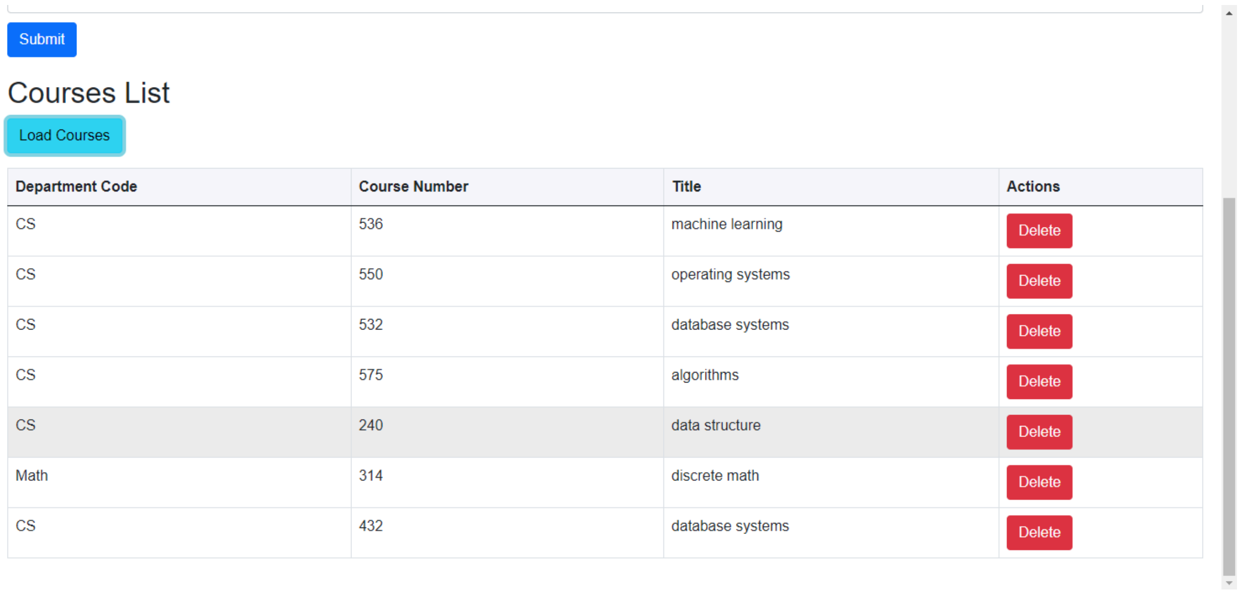
Submit

Courses List

Load Courses

Department Code	Course Number	Title	Actions

Loaded courses on the Courses page:



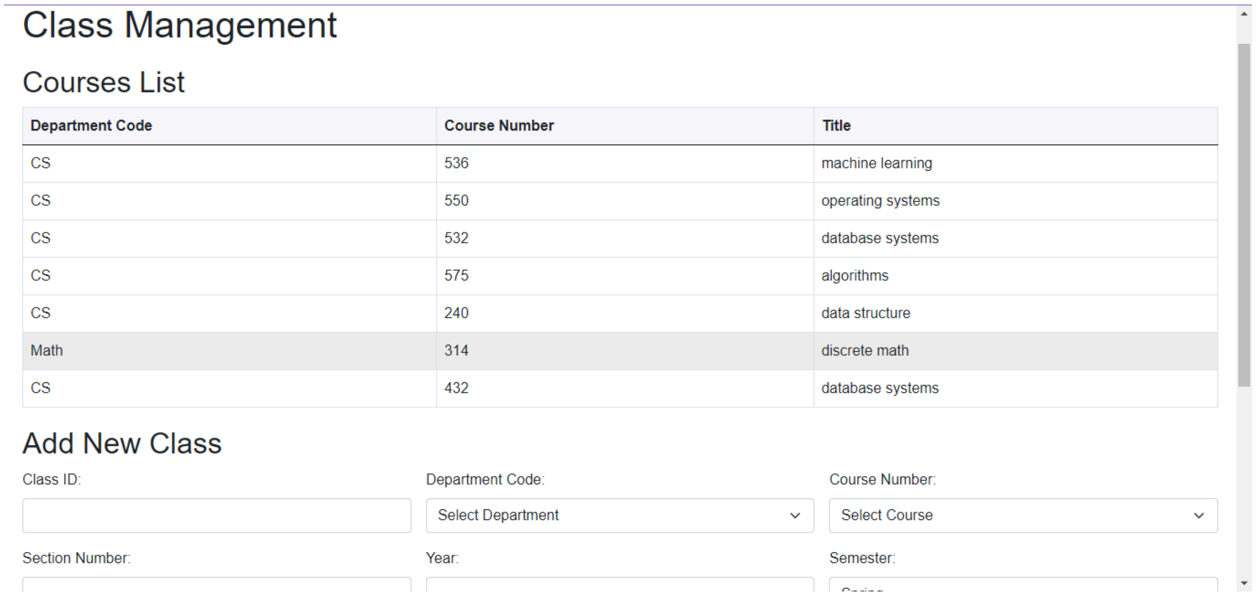
Submit

Courses List

Load Courses

Department Code	Course Number	Title	Actions
CS	536	machine learning	Delete
CS	550	operating systems	Delete
CS	532	database systems	Delete
CS	575	algorithms	Delete
CS	240	data structure	Delete
Math	314	discrete math	Delete
CS	432	database systems	Delete

Classes page:



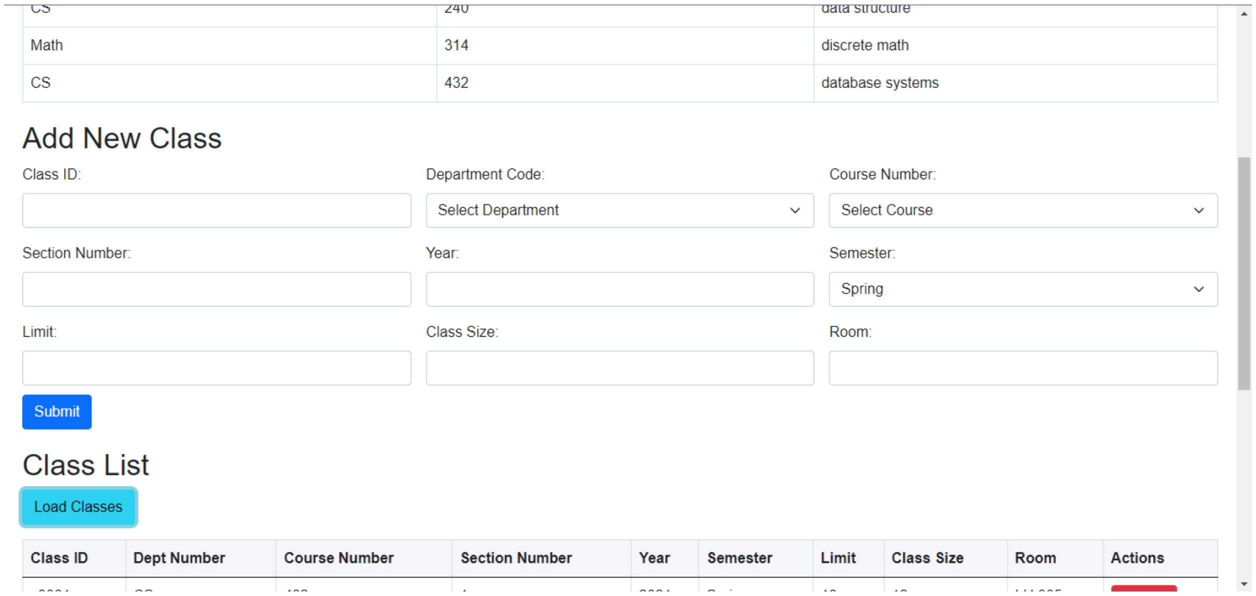
Class Management

Courses List

Department Code	Course Number	Title
CS	536	machine learning
CS	550	operating systems
CS	532	database systems
CS	575	algorithms
CS	240	data structure
Math	314	discrete math
CS	432	database systems

Add New Class

Class ID:	Department Code: Select Department	Course Number: Select Course
Section Number:	Year:	Semester:



Math	314	discrete math
CS	432	database systems

Add New Class

Class ID:	Department Code: Select Department	Course Number: Select Course
Section Number:	Year:	Semester: Spring
Limit:	Class Size:	Room:

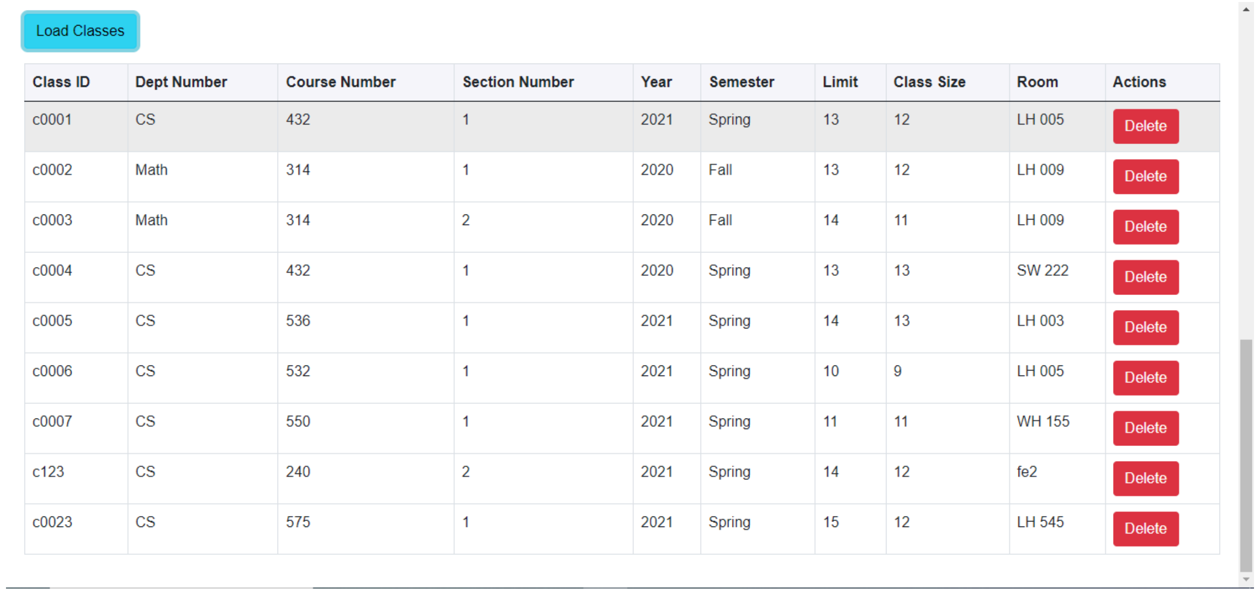
Submit

Class List

Load Classes

Class ID	Dept Number	Course Number	Section Number	Year	Semester	Limit	Class Size	Room	Actions

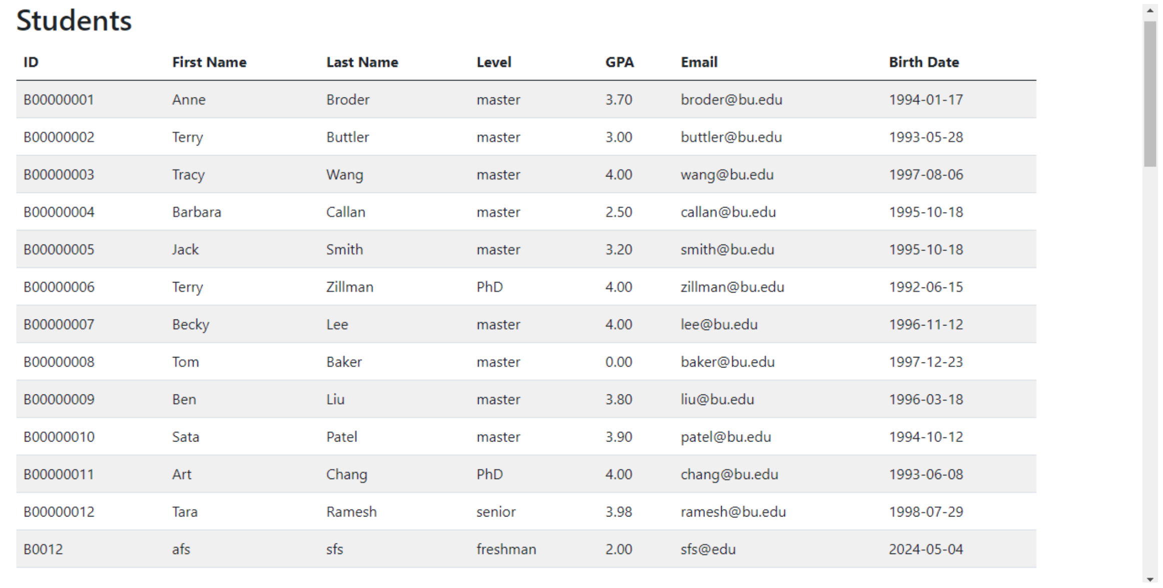
Loaded classes on Classes page:



Load Classes

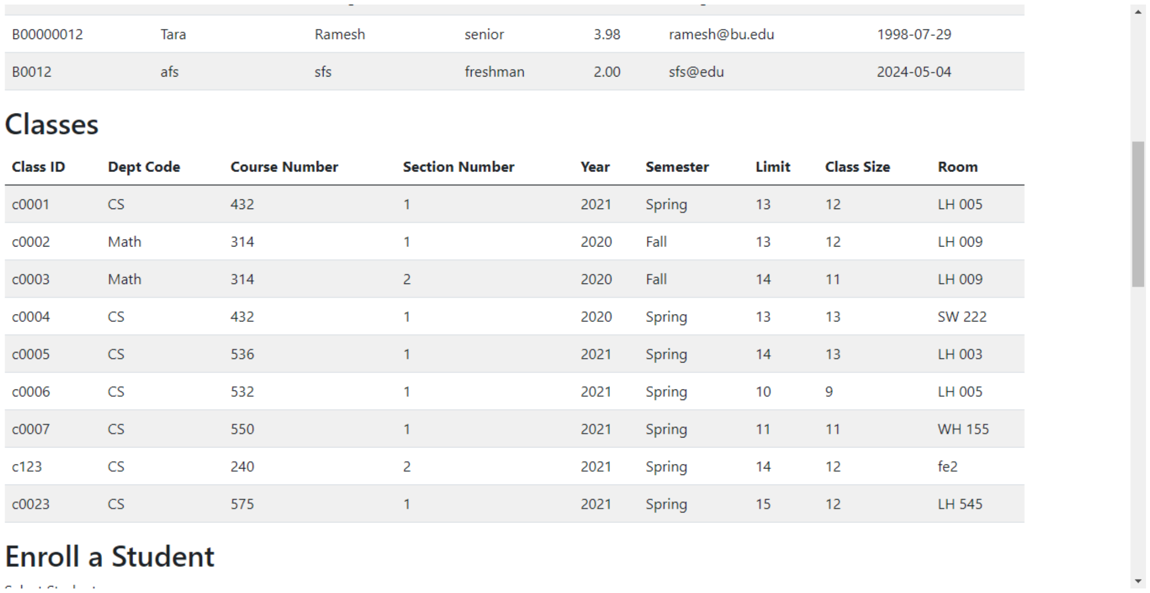
Class ID	Dept Number	Course Number	Section Number	Year	Semester	Limit	Class Size	Room	Actions
c0001	CS	432	1	2021	Spring	13	12	LH 005	Delete
c0002	Math	314	1	2020	Fall	13	12	LH 009	Delete
c0003	Math	314	2	2020	Fall	14	11	LH 009	Delete
c0004	CS	432	1	2020	Spring	13	13	SW 222	Delete
c0005	CS	536	1	2021	Spring	14	13	LH 003	Delete
c0006	CS	532	1	2021	Spring	10	9	LH 005	Delete
c0007	CS	550	1	2021	Spring	11	11	WH 155	Delete
c123	CS	240	2	2021	Spring	14	12	fe2	Delete
c0023	CS	575	1	2021	Spring	15	12	LH 545	Delete

Enrollment page:



Students

ID	First Name	Last Name	Level	GPA	Email	Birth Date
B00000001	Anne	Broder	master	3.70	broder@bu.edu	1994-01-17
B00000002	Terry	Buttler	master	3.00	buttler@bu.edu	1993-05-28
B00000003	Tracy	Wang	master	4.00	wang@bu.edu	1997-08-06
B00000004	Barbara	Callan	master	2.50	callan@bu.edu	1995-10-18
B00000005	Jack	Smith	master	3.20	smith@bu.edu	1995-10-18
B00000006	Terry	Zillman	PhD	4.00	zillman@bu.edu	1992-06-15
B00000007	Becky	Lee	master	4.00	lee@bu.edu	1996-11-12
B00000008	Tom	Baker	master	0.00	baker@bu.edu	1997-12-23
B00000009	Ben	Liu	master	3.80	liu@bu.edu	1996-03-18
B00000010	Sata	Patel	master	3.90	patel@bu.edu	1994-10-12
B00000011	Art	Chang	PhD	4.00	chang@bu.edu	1993-06-08
B00000012	Tara	Ramesh	senior	3.98	ramesh@bu.edu	1998-07-29
B0012	afs	sfs	freshman	2.00	sfs@edu	2024-05-04

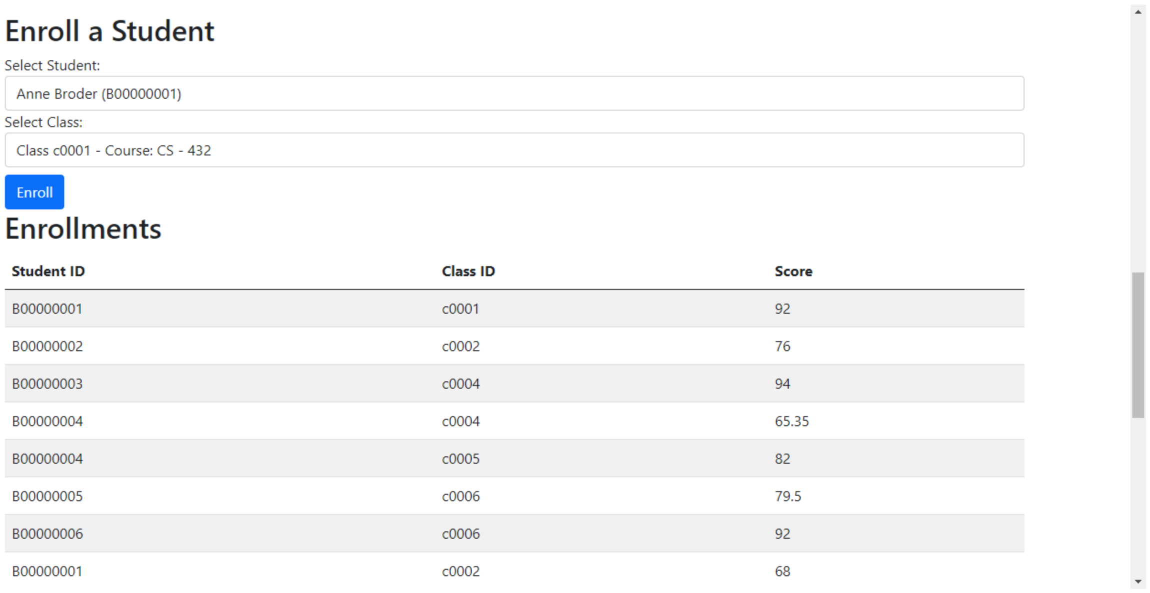


B00000012	Tara	Ramesh	senior	3.98	ramesh@bu.edu	1998-07-29
B0012	afs	sfs	freshman	2.00	sfs@edu	2024-05-04

Classes

Class ID	Dept Code	Course Number	Section Number	Year	Semester	Limit	Class Size	Room
c0001	CS	432	1	2021	Spring	13	12	LH 005
c0002	Math	314	1	2020	Fall	13	12	LH 009
c0003	Math	314	2	2020	Fall	14	11	LH 009
c0004	CS	432	1	2020	Spring	13	13	SW 222
c0005	CS	536	1	2021	Spring	14	13	LH 003
c0006	CS	532	1	2021	Spring	10	9	LH 005
c0007	CS	550	1	2021	Spring	11	11	WH 155
c123	CS	240	2	2021	Spring	14	12	fe2
c0023	CS	575	1	2021	Spring	15	12	LH 545

Enroll a Student



Enroll a Student

Select Student:

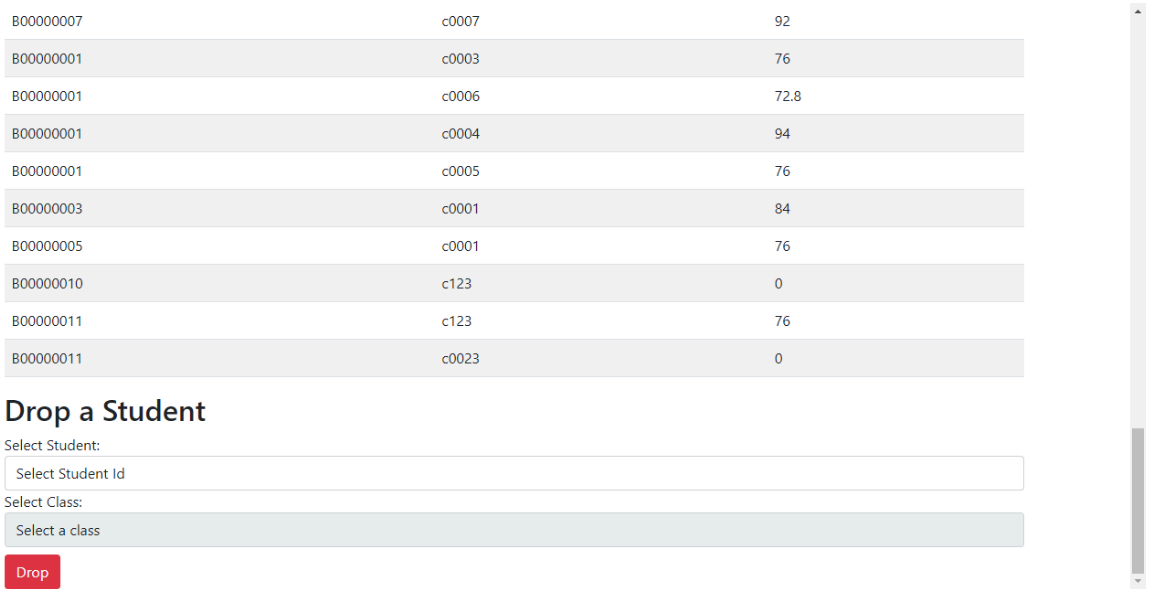
Anne Broder (B00000001)

Select Class:

Class c0001 - Course: CS - 432

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76
B00000003	c0004	94
B00000004	c0004	65.35
B00000004	c0005	82
B00000005	c0006	79.5
B00000006	c0006	92
B00000001	c0002	68



B00000007	c0007	92
B00000001	c0003	76
B00000001	c0006	72.8
B00000001	c0004	94
B00000001	c0005	76
B00000003	c0001	84
B00000005	c0001	76
B00000010	c123	0
B00000011	c123	76
B00000011	c0023	0

Drop a Student

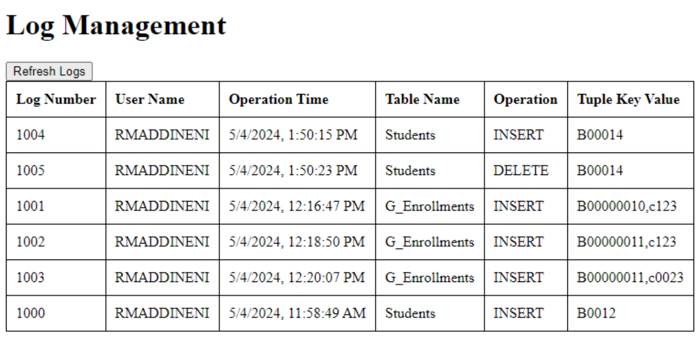
Select Student:

Select Student Id

Select Class:

Select a class

Log Management Page:



Log Management

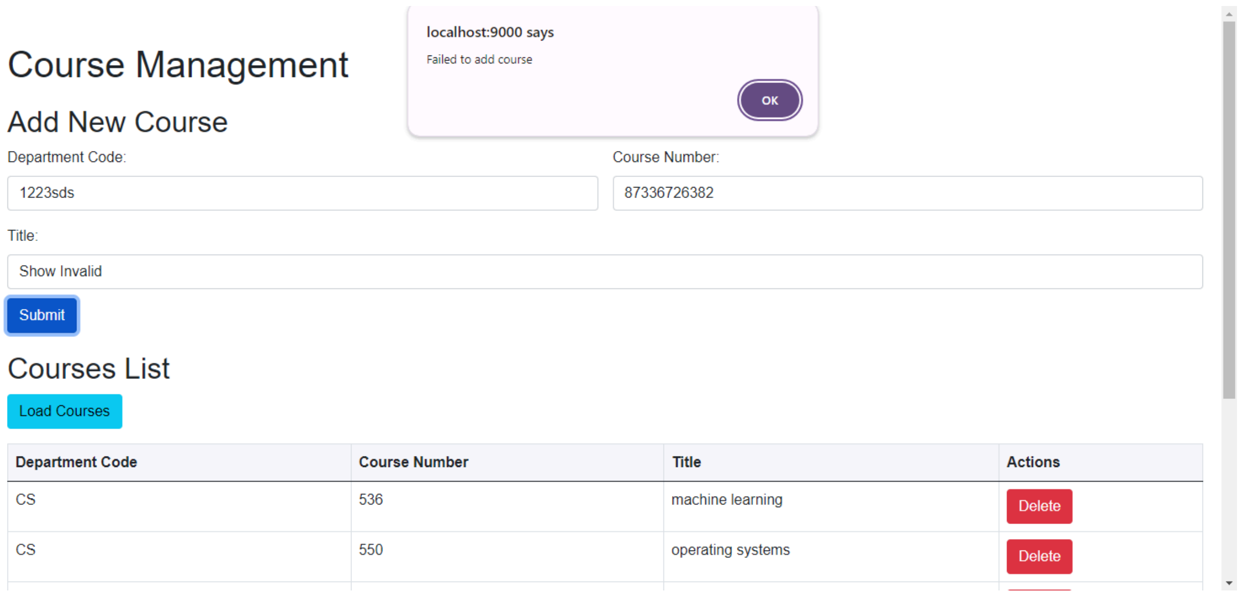
Log Number	User Name	Operation Time	Table Name	Operation	Tuple Key Value
1004	RMADDINENI	5/4/2024, 1:50:15 PM	Students	INSERT	B00014
1005	RMADDINENI	5/4/2024, 1:50:23 PM	Students	DELETE	B00014
1001	RMADDINENI	5/4/2024, 12:16:47 PM	G_Enrollments	INSERT	B00000010,c123
1002	RMADDINENI	5/4/2024, 12:18:50 PM	G_Enrollments	INSERT	B00000011,c123
1003	RMADDINENI	5/4/2024, 12:20:07 PM	G_Enrollments	INSERT	B00000011,c0023
1000	RMADDINENI	5/4/2024, 11:58:49 AM	Students	INSERT	B0012

Visuals showing front end results detailing the implementation of constraints:

Following the overview of the user interface above, we will now display how our application handles the constraints, exceptions, and errors to ensure a robust system which aligns with the requirements of the SRS. Each of these measures to implement these constraints for certain procedures will be shown as follows.

<u>Prerequisite Procedure Constraint Handling</u>

No invalid (dept_code, course#)



localhost:9000 says

Failed to add course

Department Code:	Course Number:
1223sds	87336726382

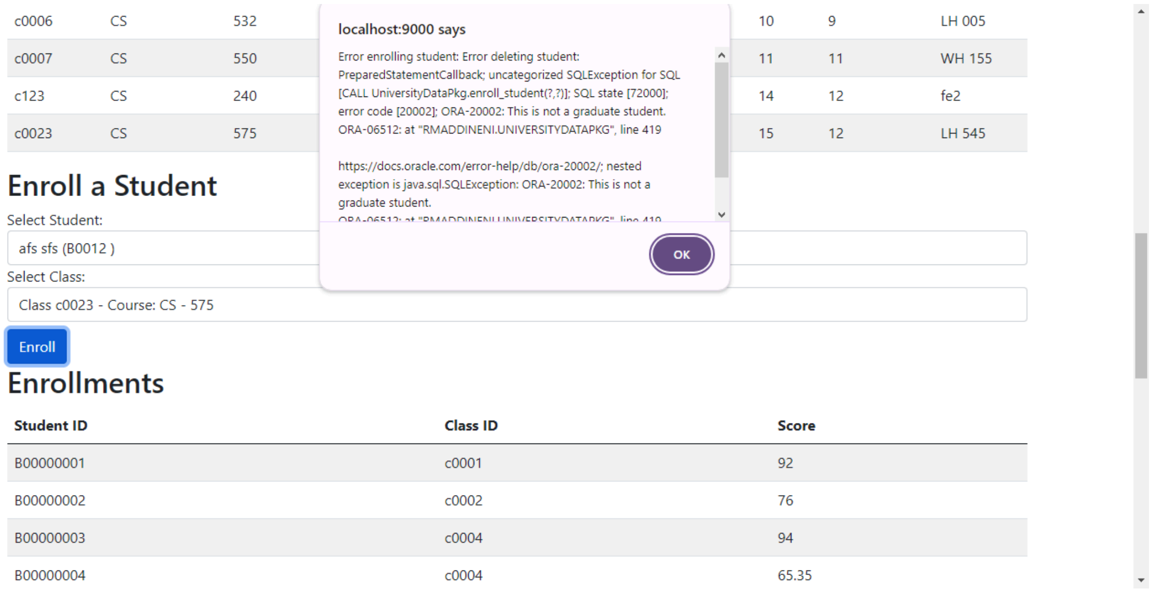
Title: Show Invalid

Department Code	Course Number	Title	Actions
CS	536	machine learning	Delete
CS	550	operating systems	Delete

<u>Enrollment Constraints:</u>

If B# entered does not appear in students then invalid - This is handled as we only allow for enrolling of students to occur from a dropdown of students who already registered to the students relation.

If the B# does not correspond to a graduate student, report "This is not a graduate student." -



c0006	CS	532	10	9	LH 005
c0007	CS	550	11	11	WH 155
c123	CS	240	14	12	fe2
c0023	CS	575	15	12	LH 545

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20002]; ORA-20002: This is not a graduate student.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 419

https://docs.oracle.com/error-help/db/ora-20002/; nested
exception is java.sql.SQLException: ORA-20002: This is not a
graduate student.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 419
```

Enroll a Student

Select Student: afs sfs (B0012)

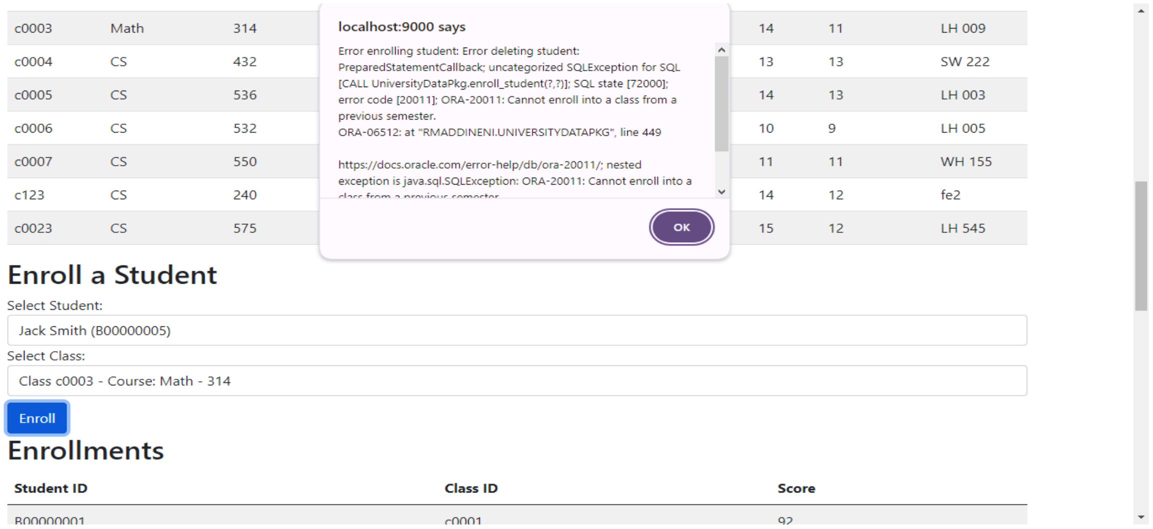
Select Class: Class c0023 - Course: CS - 575

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76
B00000003	c0004	94
B00000004	c0004	65.35

" If the classid is not in the classes table, report "The classid is invalid." - This constraint is incorporated in an analogous manner to that of the validity of the students as seen above in the sense that the only classes which can be selected are from a dropdown which only provides options for classes which exists in the classes relation

If the class is not offered in the current semester (suppose Spring 2021 is the current semester), reject the enrollment and report "Cannot enroll into a class from a previous semester." -



c0003	Math	314	14	11	LH 009
c0004	CS	432	13	13	SW 222
c0005	CS	536	14	13	LH 003
c0006	CS	532	10	9	LH 005
c0007	CS	550	11	11	WH 155
c123	CS	240	14	12	fe2
c0023	CS	575	15	12	LH 545

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20011]; ORA-20011: Cannot enroll into a class from a
previous semester.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 449

https://docs.oracle.com/error-help/db/ora-20011/; nested
exception is java.sql.SQLException: ORA-20011: Cannot enroll into a
class from a previous semester.
```

Enroll a Student

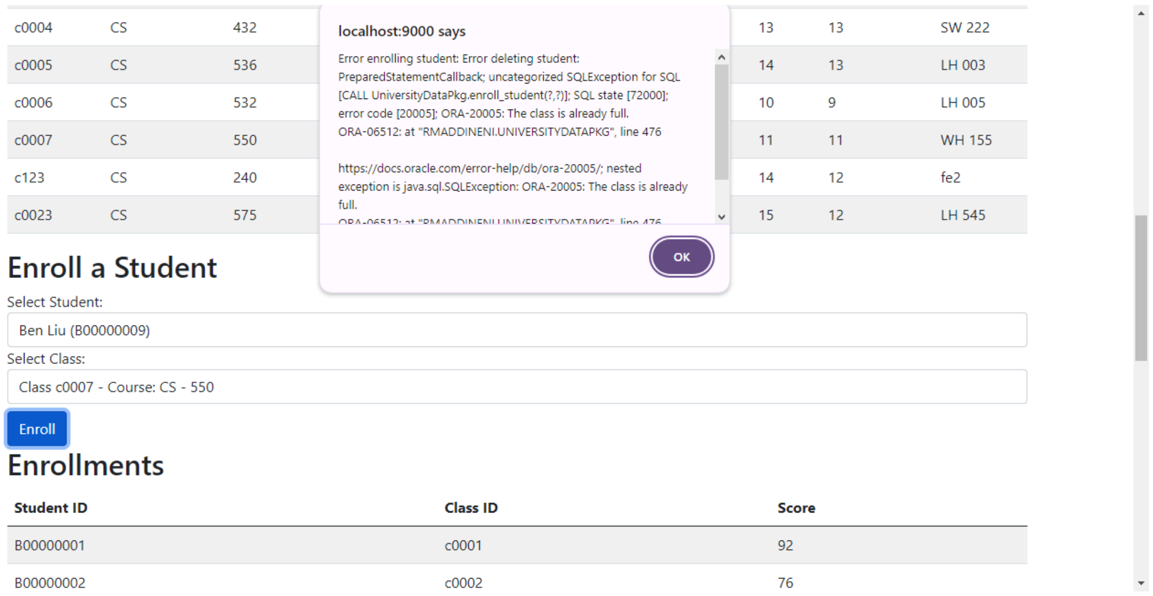
Select Student: Jack Smith (B00000005)

Select Class: Class c0003 - Course: Math - 314

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92

If the class is already full before the enrollment request, reject the enrollment request and report "The class is already full." -



c0004	CS	432	13	13	SW 222
c0005	CS	536	14	13	LH 003
c0006	CS	532	10	9	LH 005
c0007	CS	550	11	11	WH 155
c123	CS	240	14	12	fe2
c0023	CS	575	15	12	LH 545

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20005]; ORA-20005: The class is already full.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 476

https://docs.oracle.com/error-help/db/ora-20005/; nested
exception is java.sql.SQLException: ORA-20005: The class is already
full.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 476
```

Enroll a Student

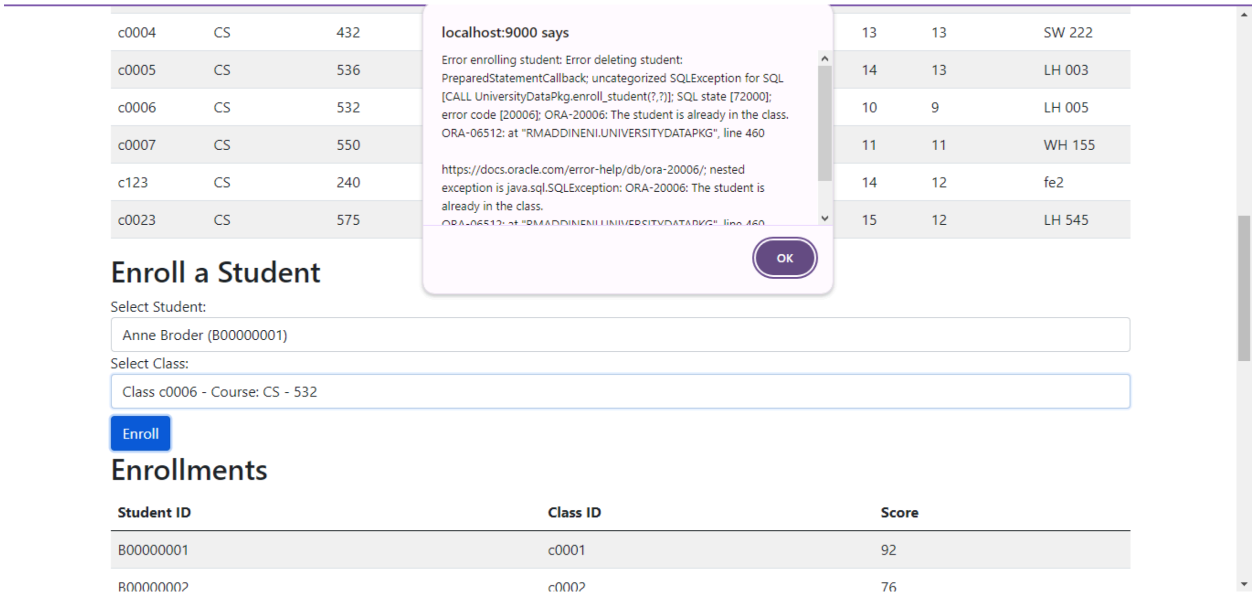
Select Student: Ben Liu (B00000009)

Select Class: Class c0007 - Course: CS - 550

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76

If the student is already in the class, report "The student is already in the class." -



c0004	CS	432	13	13	SW 222
c0005	CS	536	14	13	LH 003
c0006	CS	532	10	9	LH 005
c0007	CS	550	11	11	WH 155
c123	CS	240	14	12	fe2
c0023	CS	575	15	12	LH 545

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20006]; ORA-20006: The student is already in the class.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 460

https://docs.oracle.com/error-help/db/ora-20006/; nested
exception is java.sql.SQLException: ORA-20006: The student is
already in the class.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 460
```

Enroll a Student

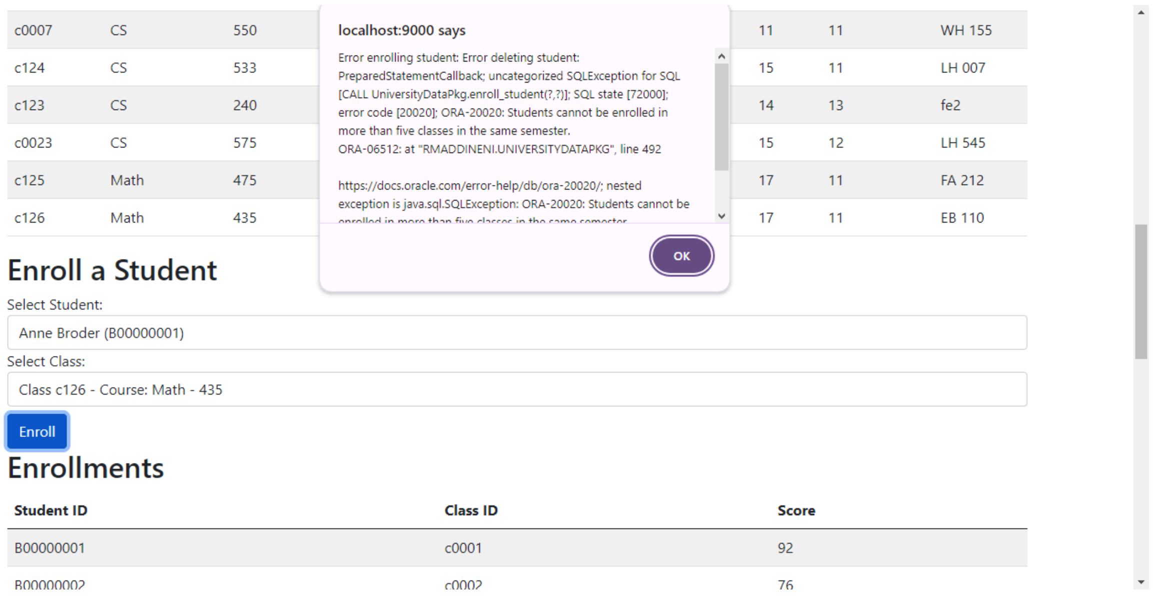
Select Student: Anne Broder (B00000001)

Select Class: Class c0006 - Course: CS - 532

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76

” If the student is already enrolled in five other classes in the same semester and the same year, report “Students cannot be enrolled in more than five classes in the same semester.” -



c0007	CS	550	11	11	WH 155
c124	CS	533	15	11	LH 007
c123	CS	240	14	13	fe2
c0023	CS	575	15	12	LH 545
c125	Math	475	17	11	FA 212
c126	Math	435	17	11	EB 110

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20020]; ORA-20020: Students cannot be enrolled in
more than five classes in the same semester.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 492

https://docs.oracle.com/error-help/db/ora-20020/; nested
exception is java.sql.SQLException: ORA-20020: Students cannot be
```

Enroll a Student

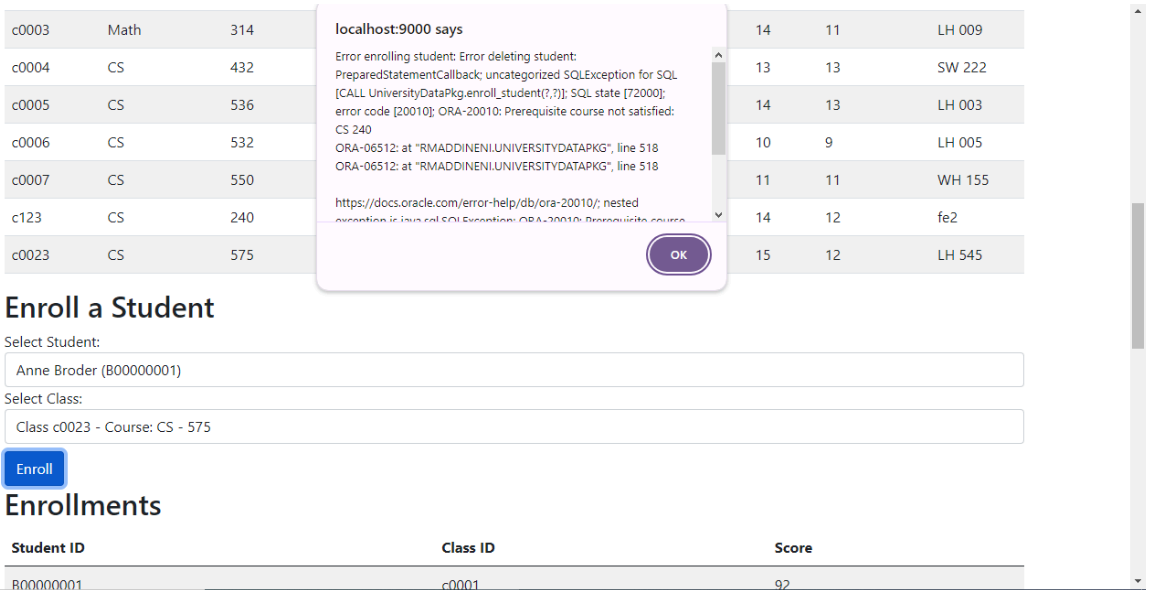
Select Student: Anne Broder (B00000001)

Select Class: Class c126 - Course: Math - 435

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76

. If the student has not completed the required prerequisite courses with at least a grade C, reject the enrollment and report “Prerequisite not satisfied.” -



c0003	Math	314	14	11	LH 009
c0004	CS	432	13	13	SW 222
c0005	CS	536	14	13	LH 003
c0006	CS	532	10	9	LH 005
c0007	CS	550	11	11	WH 155
c123	CS	240	14	12	fe2
c0023	CS	575	15	12	LH 545

```
localhost:9000 says

Error enrolling student: Error deleting student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.enroll_student(?,?)]; SQL state [72000];
error code [20010]; ORA-20010: Prerequisite course not satisfied:
CS 240
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 518
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 518

https://docs.oracle.com/error-help/db/ora-20010/; nested
```

Enroll a Student

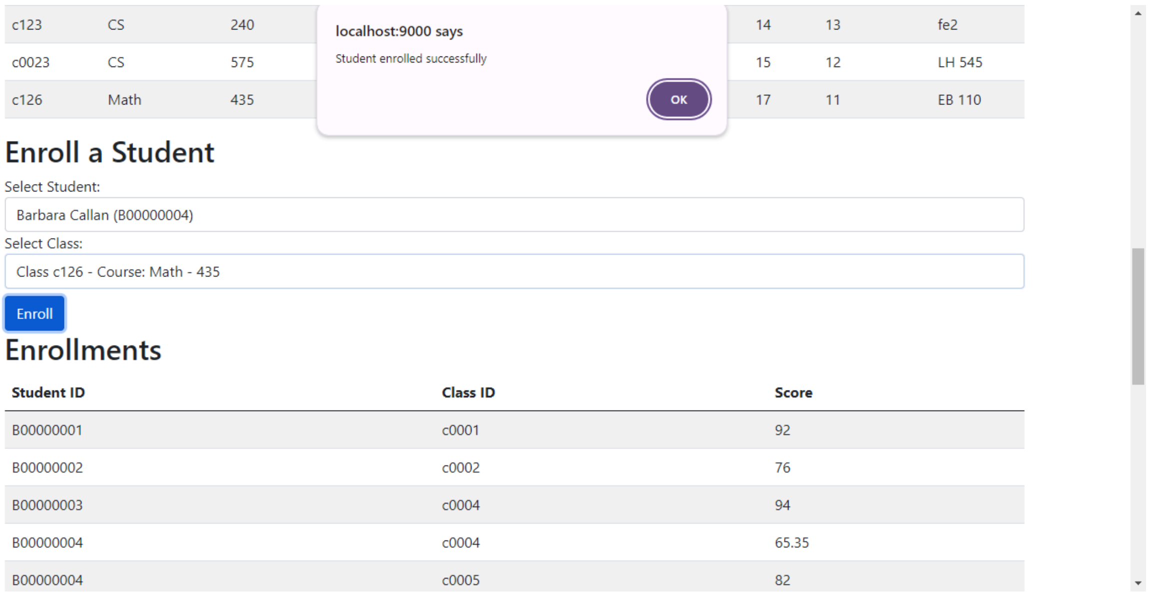
Select Student: Anne Broder (B00000001)

Select Class: Class c0023 - Course: CS - 575

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92

Sample successful enrollment-



c123	CS	240	14	13	fe2
c0023	CS	575	15	12	LH 545
c126	Math	435	17	11	EB 110

localhost:9000 says

Student enrolled successfully

Enroll a Student

Select Student: Barbara Callan (B00000004)

Select Class: Class c126 - Course: Math - 435

Enrollments

Student ID	Class ID	Score
B00000001	c0001	92
B00000002	c0002	76
B00000003	c0004	94
B00000004	c0004	65.35
B00000004	c0005	82

<u>Drop graduate student from class:</u>

If the student is not in the Students table, report "The B# is invalid."-

This is handled as the only means of dropping a student stems from a dropdown where the options are all valid B#s for students in the Students relation

If the B# does not correspond to a graduate student, report "This is not a graduate student." -

In an analogous manner to above, the only options for B#s to drop also only pertain to graduate students, effectively handling this constraint

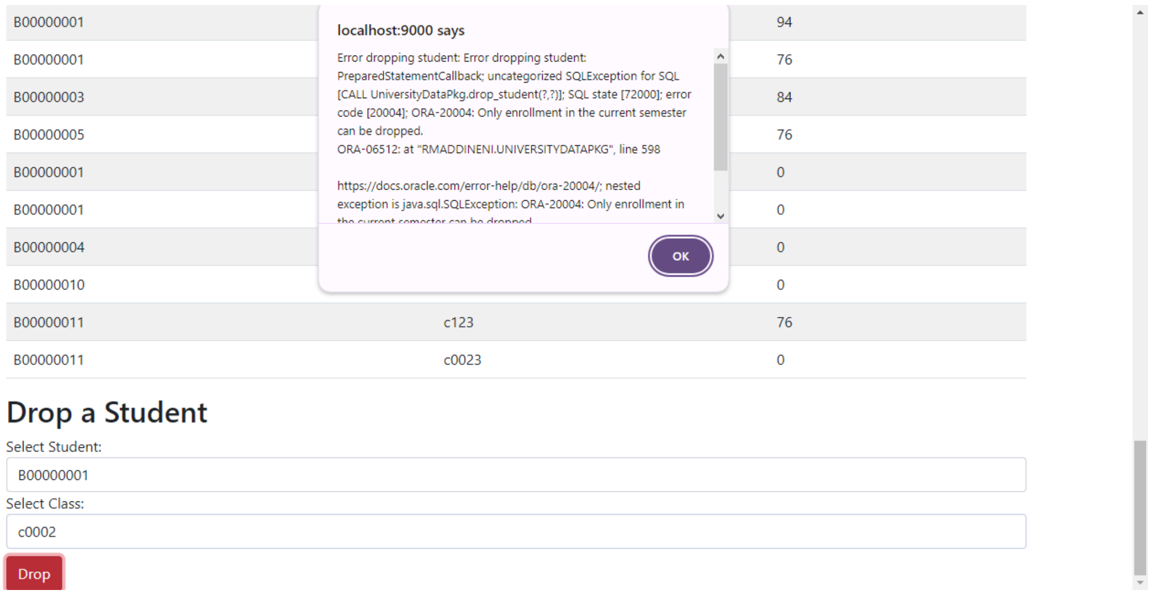
If the classid is not in the Classes table, report "The classid is invalid." -

The classes dropdown only has classes which the student, which must be selected first, is enrolled in which in turn guarantees the validity of the classes being dropped

If the student is not enrolled in the class, report "The student is not enrolled in the class."

This is handled by the dropdown as well as our dropping students feature requires a user to first choose the student which then gives access to the classes dropdown which only contains options for the classes which the chosen student is enrolled in.

” If the class is not offered in Spring 2021, reject the drop attempt and report “Only enrollment in the current semester can be dropped.” -



B00000001		94
B00000001		76
B00000003		84
B00000005		76
B00000001		0
B00000001		0
B00000004		0
B00000010		0
B00000011	c123	76
B00000011	c0023	0

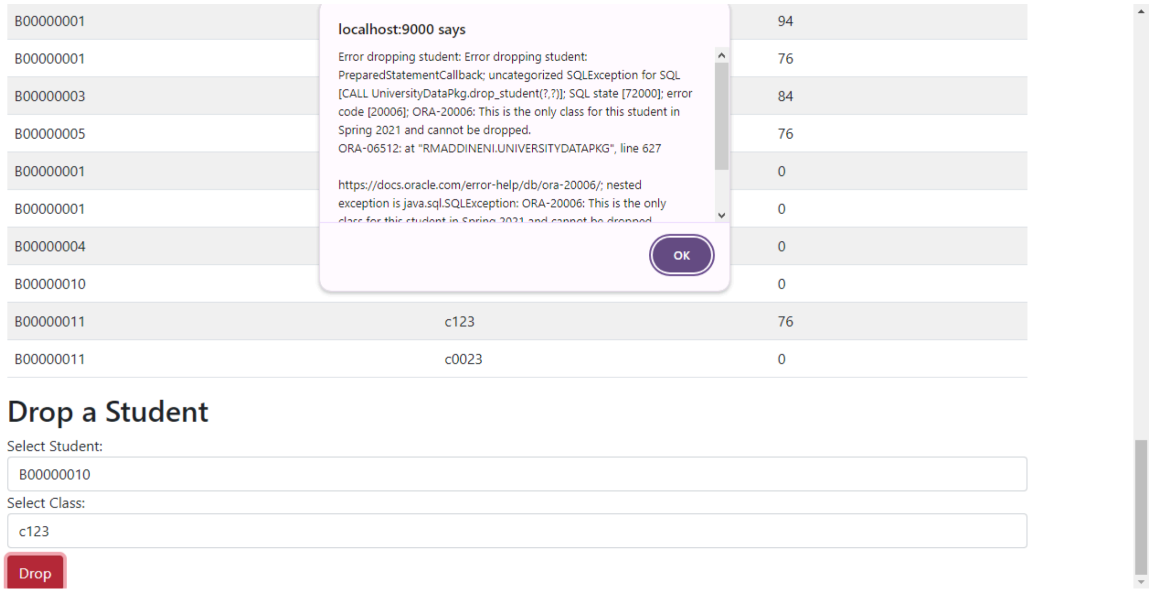
```
localhost:9000 says

Error dropping student: Error dropping student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.drop_student(?,?)]; SQL state [72000]; error
code [20004]; ORA-20004: Only enrollment in the current semester
can be dropped.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 598

https://docs.oracle.com/error-help/db/ora-20004/; nested
exception is java.sql.SQLException: ORA-20004: Only enrollment in
the current semester can be dropped.
```

Drop a Student

Select Student: B00000001

Select Class: c0002

If the class is the last class for the student in Spring 2021, reject the drop request and report “This is the only class for this student in Spring 2021 and cannot be dropped.” -



B00000001		94
B00000001		76
B00000003		84
B00000005		76
B00000001		0
B00000001		0
B00000004		0
B00000010		0
B00000011	c123	76
B00000011	c0023	0

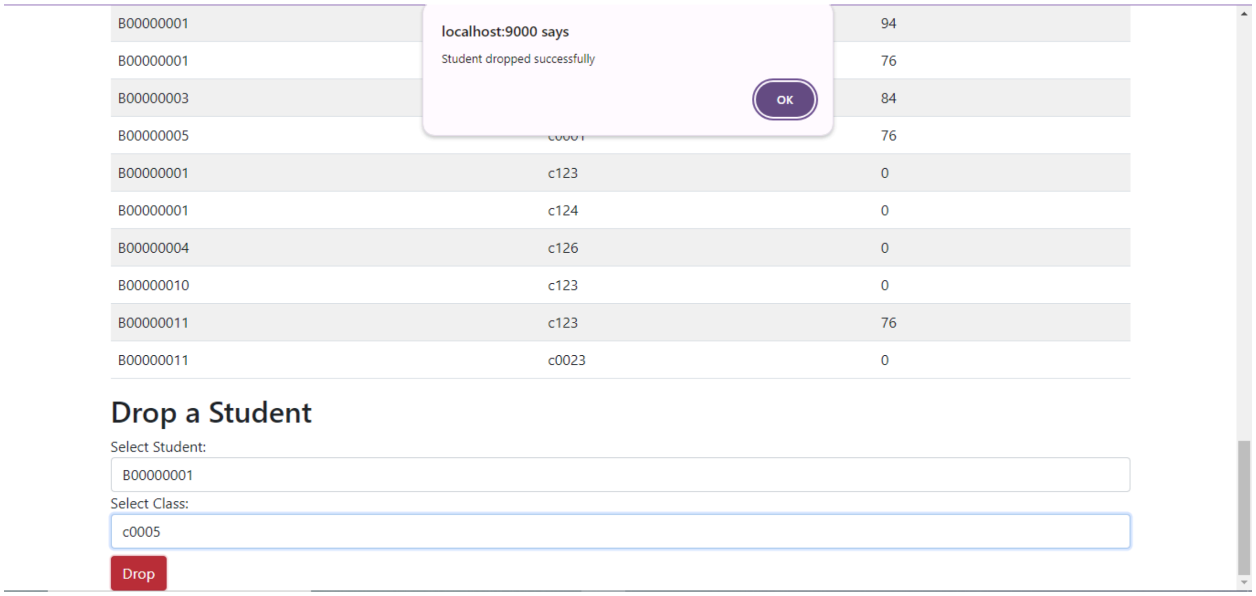
```
localhost:9000 says

Error dropping student: Error dropping student:
PreparedStatementCallback; uncategorized SQLException for SQL
[CALL UniversityDataPkg.drop_student(?,?)]; SQL state [72000]; error
code [20006]; ORA-20006: This is the only class for this student in
Spring 2021 and cannot be dropped.
ORA-06512: at "RMADDINENI.UNIVERSITYDATAPKG", line 627

https://docs.oracle.com/error-help/db/ora-20006/; nested
exception is java.sql.SQLException: ORA-20006: This is the only
class for this student in Spring 2021 and cannot be dropped.
```

Drop a Student

Select Student: B00000010

Select Class: c123

Sample Successful Dropping of a Student-



localhost:9000 says

Student dropped successfully

OK

B00000001		94
B00000001		76
B00000003		84
B00000005	c0001	76
B00000001	c123	0
B00000001	c124	0
B00000004	c126	0
B00000010	c123	0
B00000011	c123	76
B00000011	c0023	0

Drop a Student

Select Student: B00000001

Select Class: c0005

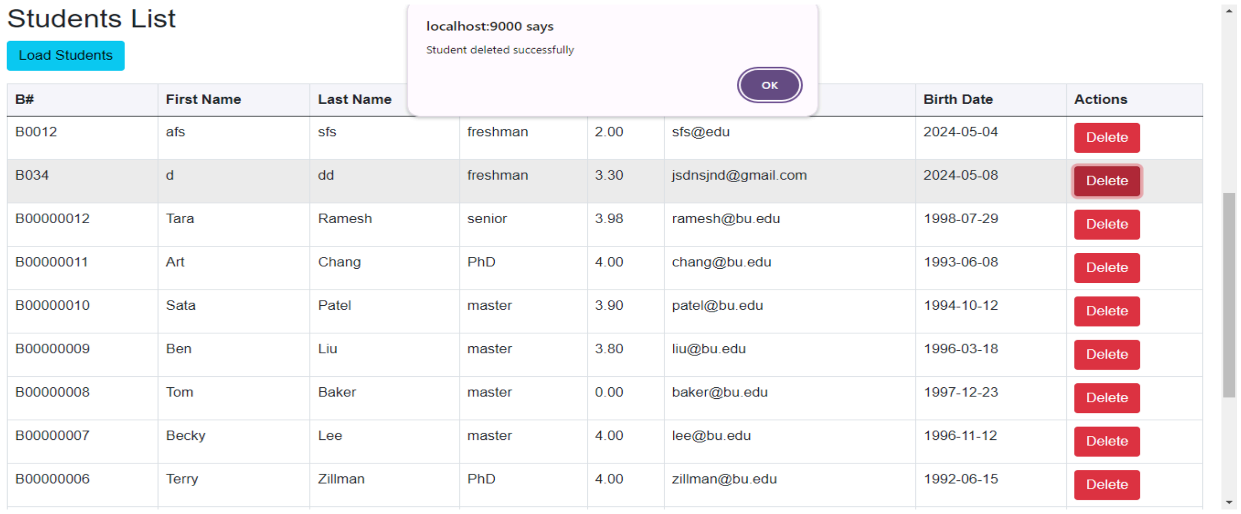
Drop

<u>Delete a Student from the Students Table:</u>

If the student is not in the Students table, report "The B# is invalid." -

Our delete feature for students is a unique option for each student in the table, as a result, it is guaranteed that only existing students can be dropped from the students relation

Sample Successful Deletion -



Students List

Load Students

localhost:9000 says

Student deleted successfully

OK

B#	First Name	Last Name				Birth Date	Actions
B0012	afs	sfs	freshman	2.00	sfs@edu	2024-05-04	Delete
B034	d	dd	freshman	3.30	jsdnsjnd@gmail.com	2024-05-08	Delete
B00000012	Tara	Ramesh	senior	3.98	ramesh@bu.edu	1998-07-29	Delete
B00000011	Art	Chang	PhD	4.00	chang@bu.edu	1993-06-08	Delete
B00000010	Sata	Patel	master	3.90	patel@bu.edu	1994-10-12	Delete
B00000009	Ben	Liu	master	3.80	liu@bu.edu	1996-03-18	Delete
B00000008	Tom	Baker	master	0.00	baker@bu.edu	1997-12-23	Delete
B00000007	Becky	Lee	master	4.00	lee@bu.edu	1996-11-12	Delete
B00000006	Terry	Zillman	PhD	4.00	zillman@bu.edu	1992-06-15	Delete