Information Technology Institute

2024

UNIVERSITY CASE STUDY

Prepared by :
Ibrahim El-Nwasany

I. Database Design

Overview:

The University Database manages comprehensive data about university students, departments, courses, grades, and enroll. The database structureensures proper organization and relationships between various entities.

Requirements:

**Entities and Attributes: **

1. Students:
- Student-ID (Primary Key)
- FirstName
- LastName
- Address
- Date Of Birth
- Gender
- Enrollment-Date
- Email
-GPA

2. Courses:
- Course-ID (Primary Key)
- Course-Name
- Start-Date
- End-Date
- Capacity
- Description
- Credit-Hours

3. Departments:
- Department-ID (Primary Key)
- Department-Name

- Location
- Description
- Contact-Email
- Contact-Phone
- Student-Count
- Establishment-Date

4. Grades:
- Grade-ID (Primary Key)
- Grade-Value
- Evaluation-Date
- PassFailStatus
-Student ID (fk)
-Course ID (fk)

5-Students_Courses:
-Student ID (fk)
-Course ID (fk)
-both are composite pk

Relationships, Cardinality, and Participation:

1. Students - Courses Relationship:
- Verb: "Enroll in"
- Cardinality: Many-to-Many (M:N)
- Participation:
  - Each student can enroll in multiple courses.
  - Each course can have multiple enrolled students.

2. Students - Grades Relationship:
- Verb: "Receive"
- Cardinality: One-to-Many (1:N)

- Participation:
- Each student may receive multiple grades.
- Each grade must be associated with a student.

3. Courses - Grades Relationship:

- Verb: "Have"
- Cardinality: One-to-Many (1:N)
- Participation:
  - Each course may have multiple grades.
  - Each grade must be associated with a course.

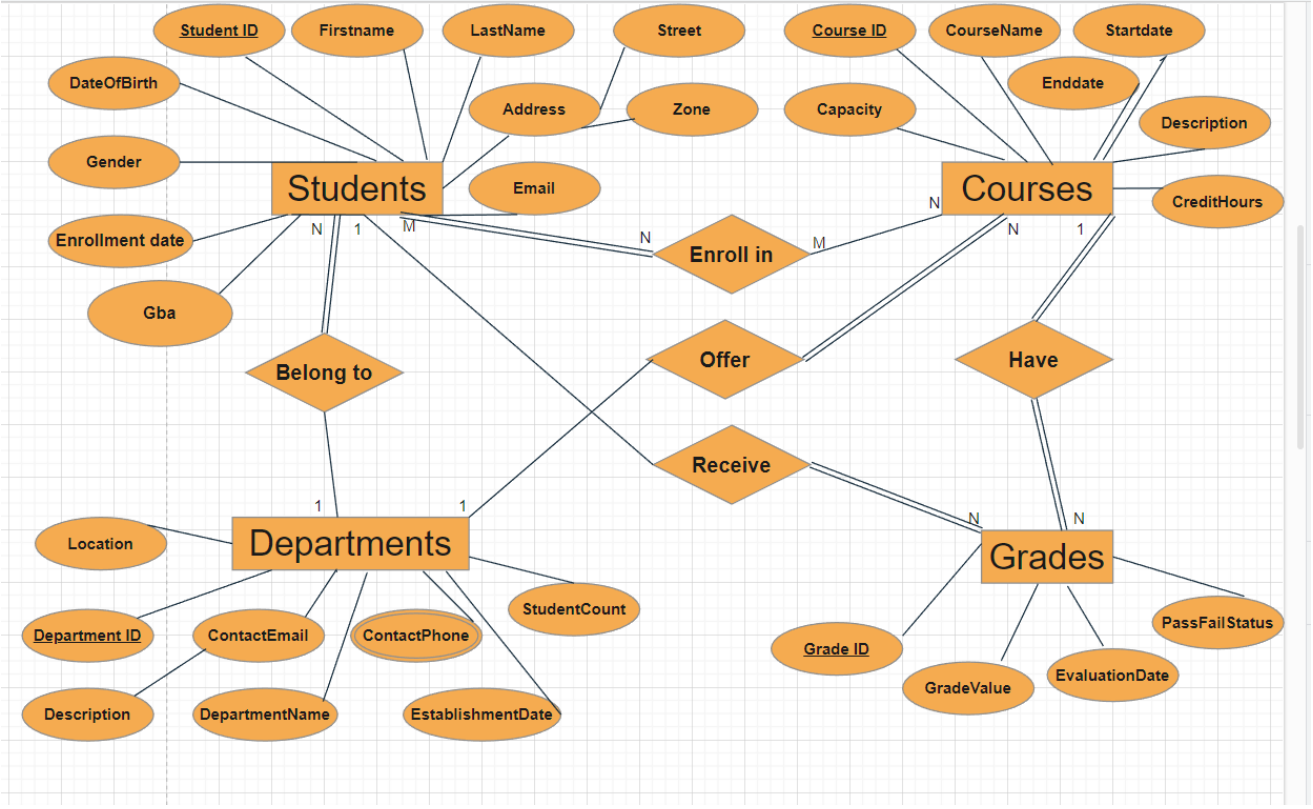
4. Departments - Courses Relationship:

- Verb: "Offer"
- Cardinality: One-to-Many (1:N)
- Participation:
  - Each department may offer multiple courses.
  - Each course must be associated with a department.

5. Students - Departments Relationship:

- Verb: "Belong to"
- Cardinality: One-to-Many (1:M)
- Participation:
  - Each student must belong to exactly one department.
  - Each department may have multiple students.

The ER-Diagram:



Mapping and Normalization:

Students(Student_id (pk) , firstname, lastname , gender , email, enrollmentdate , dateofbitrh , street , zone , department_id(fk))

Courses(course_id(pk),coursename,startdate,enddate,capacity,descripti on,CreditHours,department_id(fk))

Departments(department_id(pk), departmentname, description , location , contactemail, studentcount ,establishmentdate)

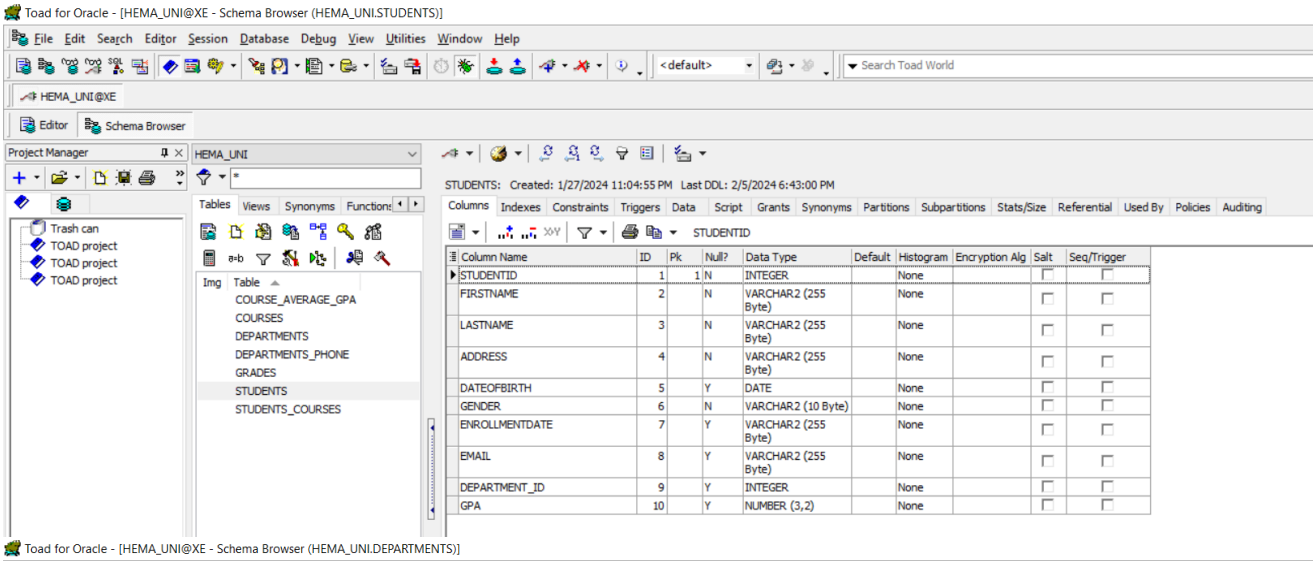
Grades(Grade_id,Gardevalue,Evaluationdate,passfailstatus,student_id(fk),course_id(fk)

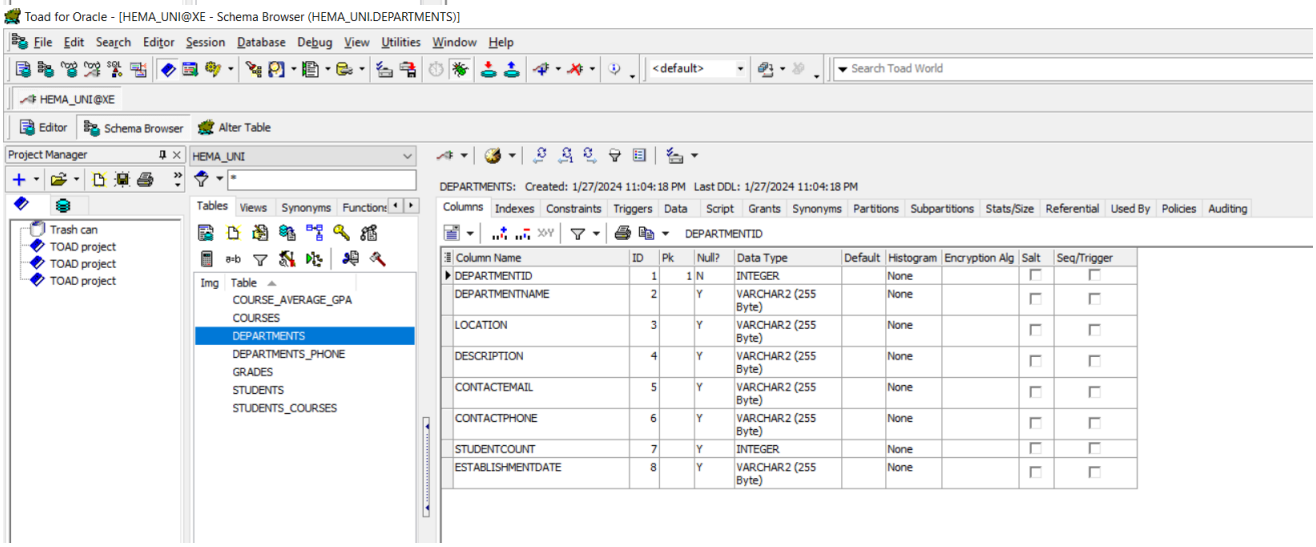
Departments_phone(department_id(fk),contactphone)

Students_Courses (Student_ID(fk),Course_ID(fk))

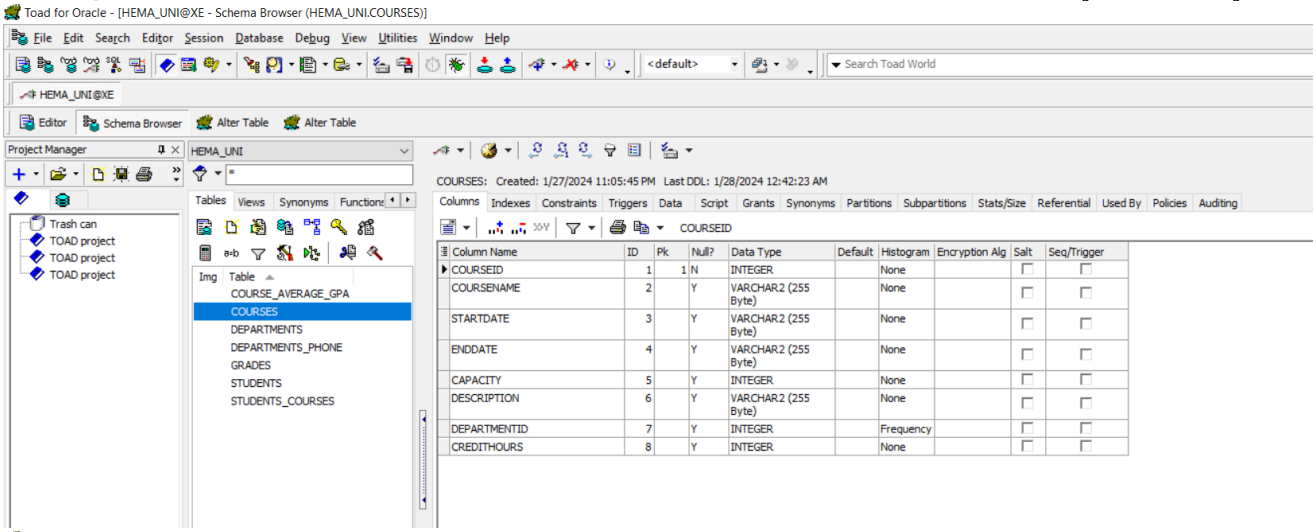
II. SQL Implementation

Creation of Tables and constraints:





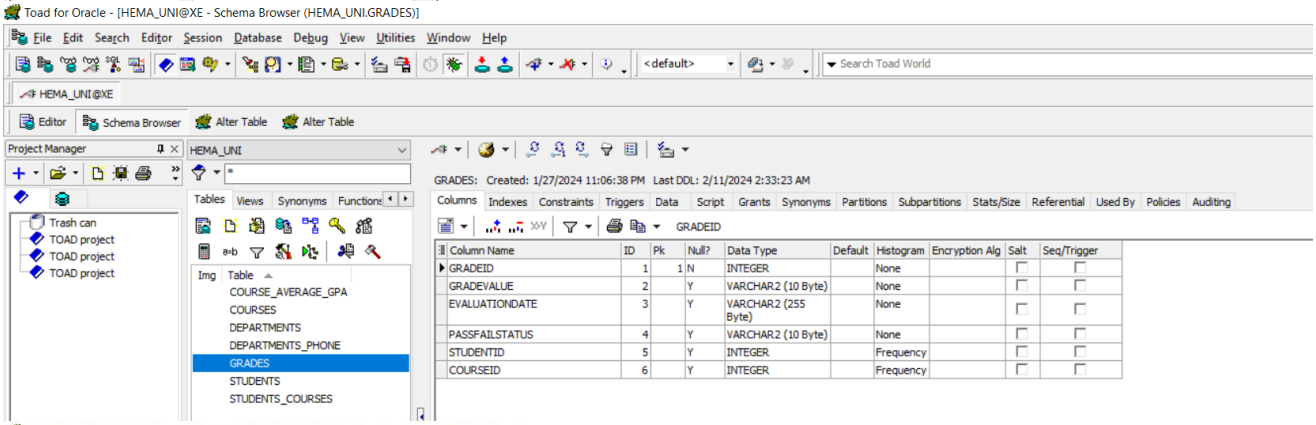
Toad for Oracle - [HEMA_UNI@XE - Schema Browser (HEMA_UNI.COURSES)]



COURSES: Created: 1/27/2024 11:05:45 PM Last DDL: 1/28/2024 12:42:23 AM

Column Name	ID	Pk	Null?	Data Type	Default	Histogram	Encryption Alg	Salt	Seq/Trigger
COURSEID	1	1	N	INTEGER		None			
COURSENAME	2		Y	VARCHAR2 (255 Byte)		None			
STARTDATE	3		Y	VARCHAR2 (255 Byte)		None			
ENDDATE	4		Y	VARCHAR2 (255 Byte)		None			
CAPACITY	5		Y	INTEGER		None			
DESCRIPTION	6		Y	VARCHAR2 (255 Byte)		None			
DEPARTMENTID	7		Y	INTEGER		Frequency			
CREDITHOURS	8		Y	INTEGER		None			

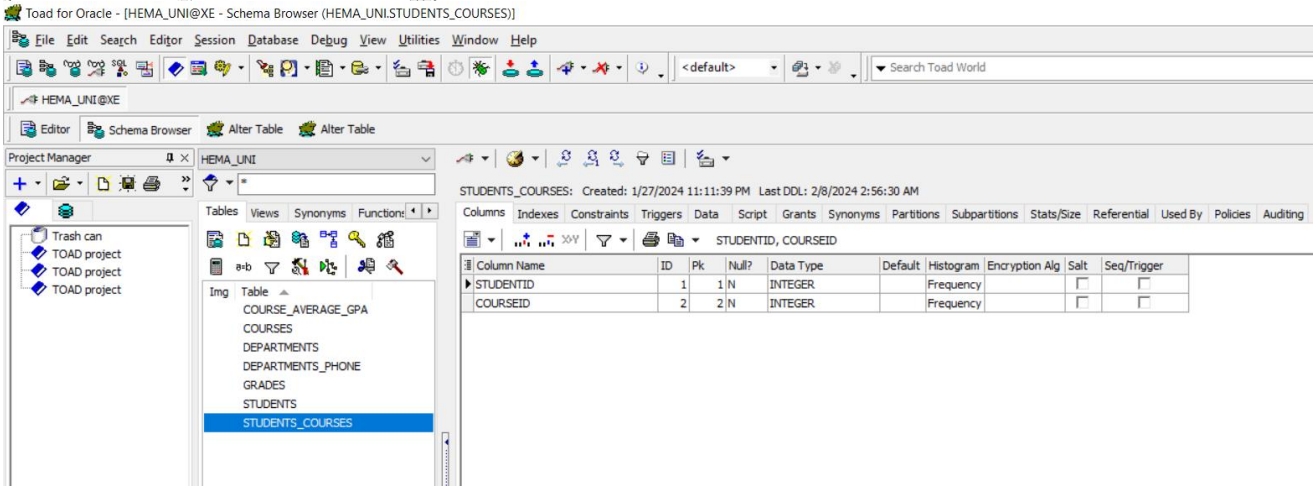
Toad for Oracle - [HEMA_UNI@XE - Schema Browser (HEMA_UNI.GRADES)]



GRADES: Created: 1/27/2024 11:06:38 PM Last DDL: 2/11/2024 2:33:23 AM

Column Name	ID	Pk	Null?	Data Type	Default	Histogram	Encryption Alg	Salt	Seq/Trigger
GRADEID	1	1	N	INTEGER		None			
GRADEVALUE	2		Y	VARCHAR2 (10 Byte)		None			
EVALUATIONDATE	3		Y	VARCHAR2 (255 Byte)		None			
PASSFAILSTATUS	4		Y	VARCHAR2 (10 Byte)		None			
STUDENTID	5		Y	INTEGER		Frequency			
COURSEID	6		Y	INTEGER		Frequency			

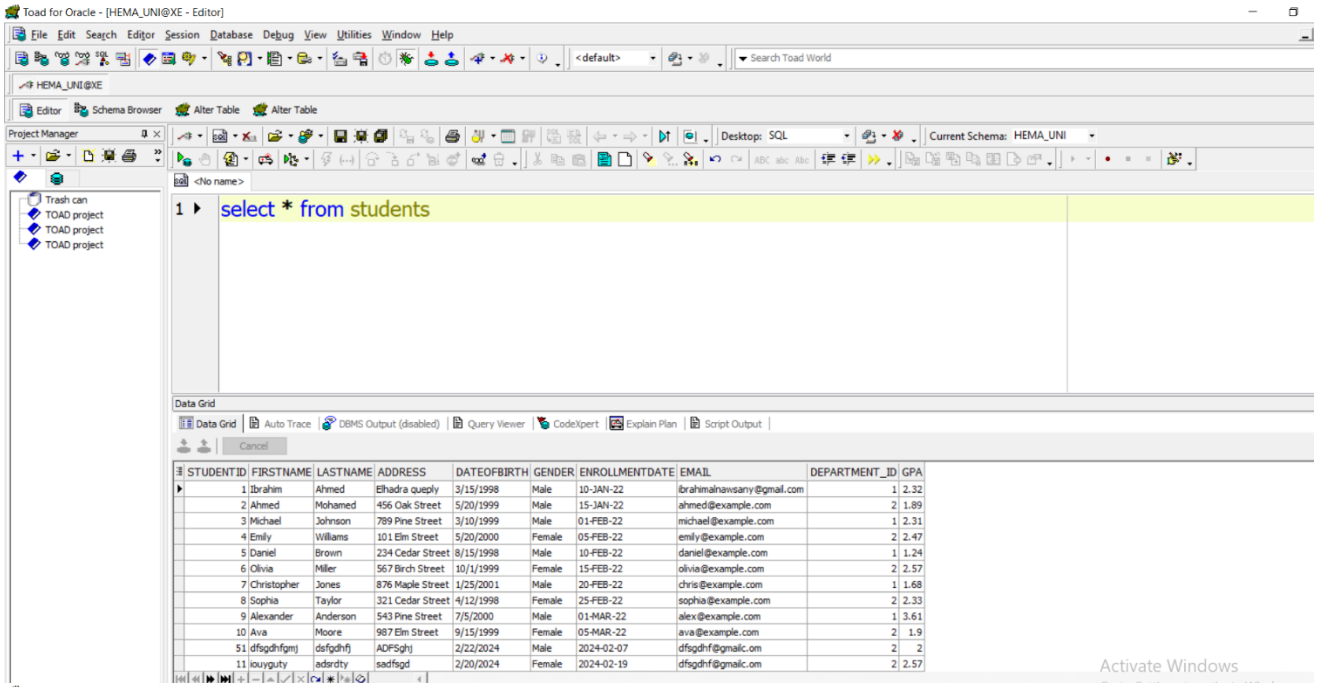
Toad for Oracle - [HEMA_UNI@XE - Schema Browser (HEMA_UNI.STUDENTS_COURSES)]



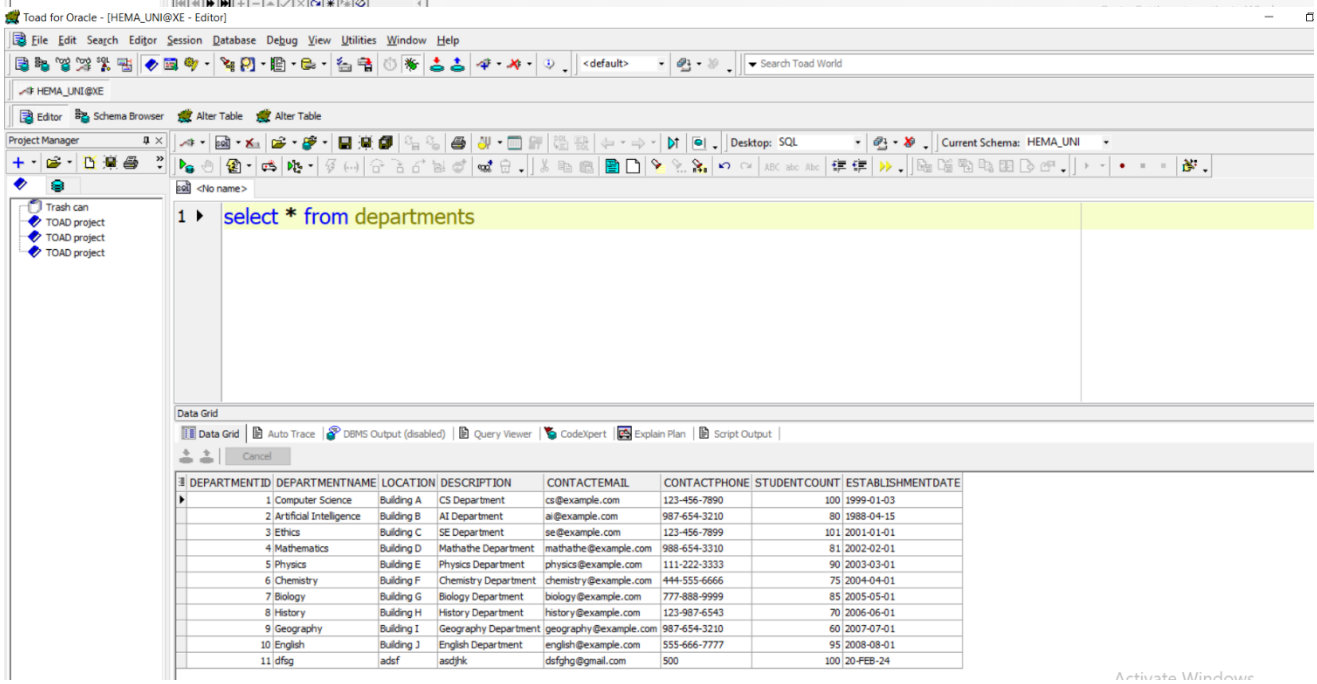
STUDENTS_COURSES: Created: 1/27/2024 11:11:39 PM Last DDL: 2/8/2024 2:56:30 AM

Column Name	ID	Pk	Null?	Data Type	Default	Histogram	Encryption Alg	Salt	Seq/Trigger
STUDENTID	1	1	N	INTEGER		Frequency			
COURSEID	2	2	N	INTEGER		Frequency			

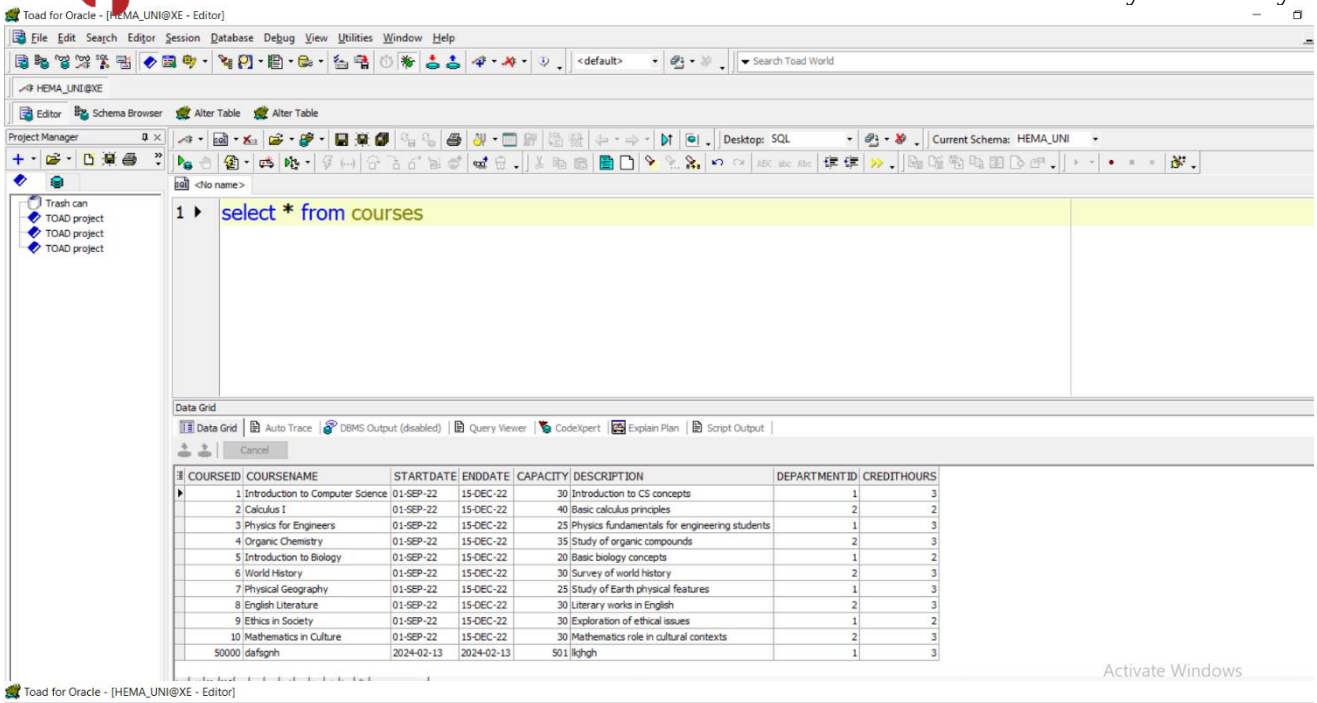
Populating Sample Data

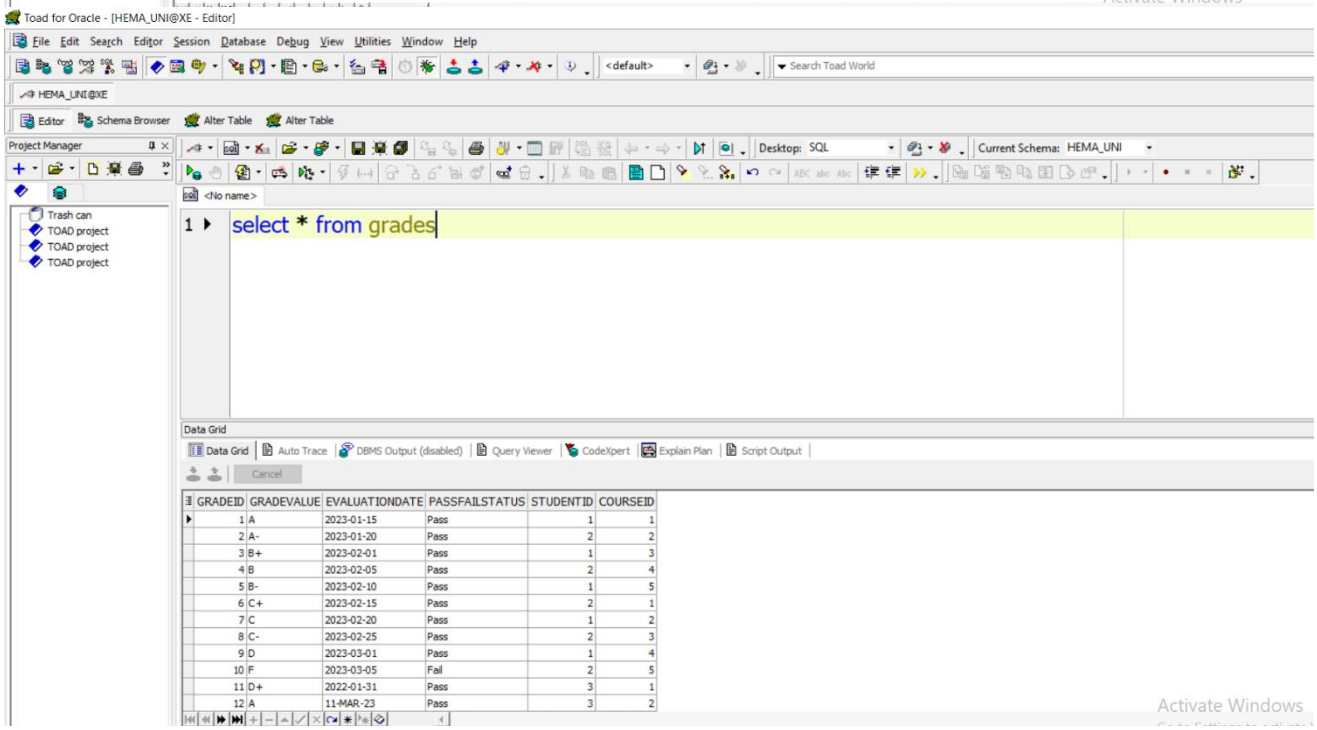


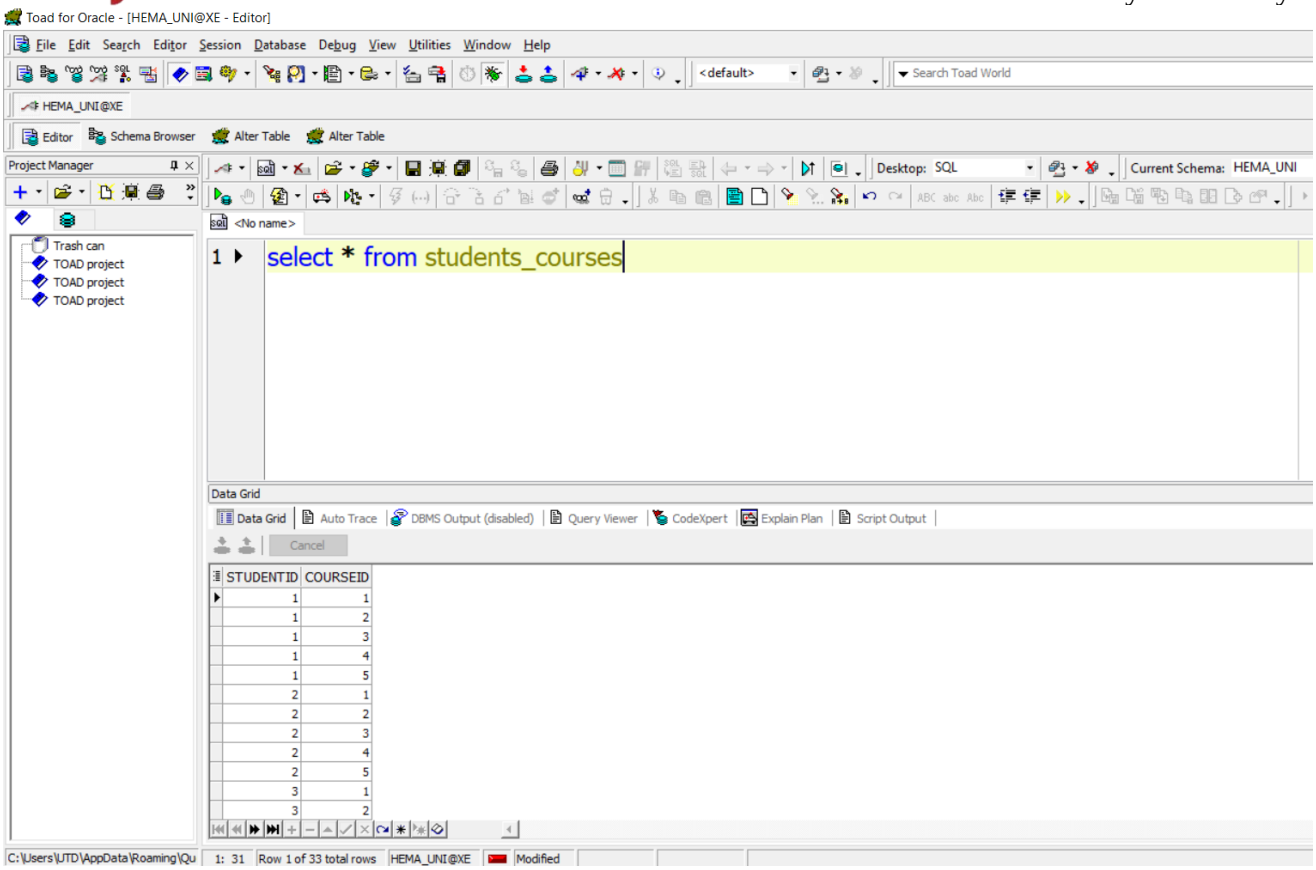
STUDENTID	FIRSTNAME	LASTNAME	ADDRESS	DATEOFBIRTH	GENDER	ENROLLMENTDATE	EMAIL	DEPARTMENT_ID	GPA
1	Ibrahim	Ahmed	Elhadra queply	3/15/1998	Male	10-JAN-22	ibrahimalnawsany@gmail.com	1	2.32
2	Ahmed	Mohamed	456 Oak Street	5/20/1999	Male	15-JAN-22	ahmed@example.com	2	1.89
3	Michael	Johnson	789 Pine Street	3/10/1999	Male	01-FEB-22	michael@example.com	1	2.31
4	Emily	Williams	101 Elm Street	5/20/2000	Female	05-FEB-22	emily@example.com	2	2.47
5	Daniel	Brown	234 Cedar Street	8/15/1998	Male	10-FEB-22	daniel@example.com	1	1.24
6	Olivia	Miller	567 Birch Street	10/1/1999	Female	15-FEB-22	olivia@example.com	2	2.57
7	Christopher	Jones	876 Maple Street	1/25/2001	Male	20-FEB-22	chris@example.com	1	1.68
8	Sophia	Taylor	321 Cedar Street	4/12/1998	Female	25-FEB-22	sophia@example.com	2	2.33
9	Alexander	Anderson	543 Pine Street	7/5/2000	Male	01-MAR-22	alex@example.com	1	3.61
10	Ava	Moore	987 Elm Street	9/15/1999	Female	05-MAR-22	ava@example.com	2	1.9
51	dfsgdhfgmj	dsfgdhfj	ADFSghj	2/22/2024	Male	2024-02-07	dfsgdhf@gmailc.om	2	2
11	iouyguty	adsrdty	sadfsgd	2/20/2024	Female	2024-02-19	dfsgdhf@gmailc.om	2	2.57



DEPARTMENTID	DEPARTMENTNAME	LOCATION	DESCRIPTION	CONTACTEMAIL	CONTACTPHONE	STUDENTCOUNT	ESTABLISHMENTDATE
1	Computer Science	Building A	CS Department	cs@example.com	123-456-7890	100	1999-01-03
2	Artificial Intelligence	Building B	AI Department	ai@example.com	987-654-3210	80	1988-04-15
3	Ethics	Building C	SE Department	se@example.com	123-456-7899	101	2001-01-01
4	Mathematics	Building D	Mathathe Department	mathathe@example.com	988-654-3310	81	2002-02-01
5	Physics	Building E	Physics Department	physics@example.com	111-222-3333	90	2003-03-01
6	Chemistry	Building F	Chemistry Department	chemistry@example.com	444-555-6666	75	2004-04-01
7	Biology	Building G	Biology Department	biology@example.com	777-888-9999	85	2005-05-01
8	History	Building H	History Department	history@example.com	123-987-6543	70	2006-06-01
9	Geography	Building I	Geography Department	geography@example.com	987-654-3210	60	2007-07-01
10	English	Building J	English Department	english@example.com	555-666-7777	95	2008-08-01
11	dfsg	adsf	asdjhk	dsfghg@gmail.com	500	100	20-FEB-24

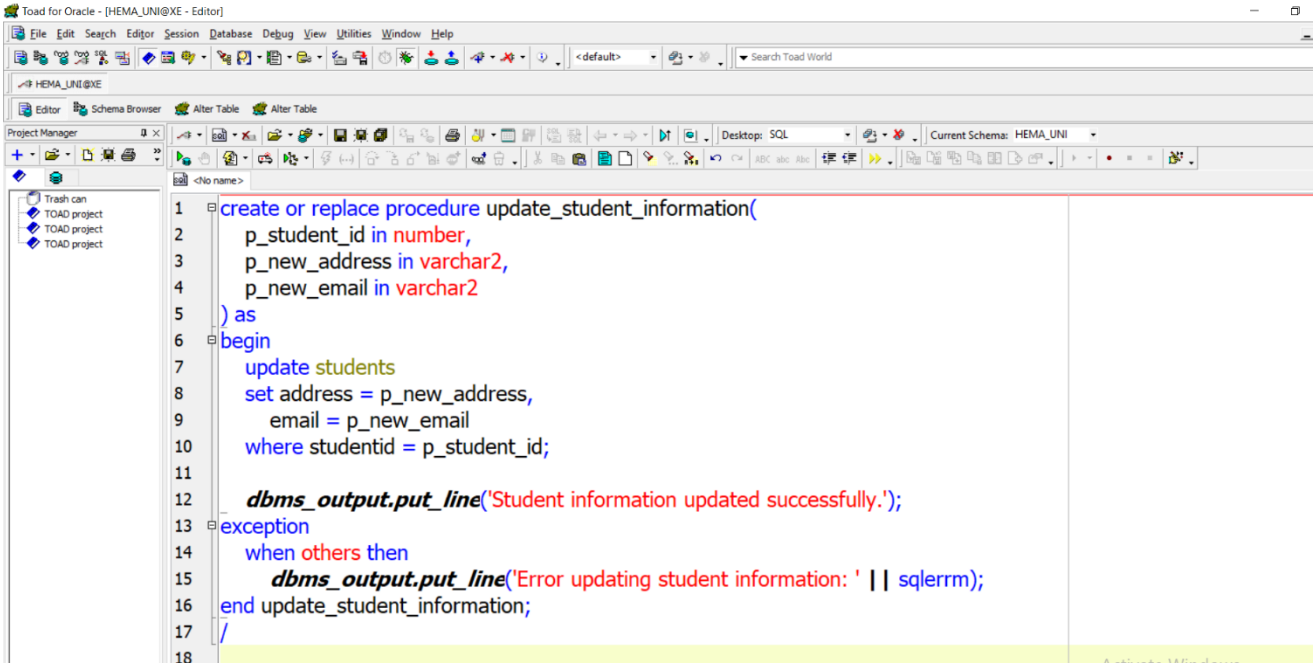

Toad for Oracle - [HEMA_UNI@XE - Editor]
File Edit Search Editor Session Database Debug View Utilities Window Help
HEMA_UNI@XE
Editor Schema Browser Alter Table Alter Table
Project Manager
Trash can
TOAD project
TOAD project
TOAD project
<No name>
1 select * from courses
Data Grid
Data Grid Auto Trace DBMS Output (disabled) Query Viewer CodeXpert Explain Plan Script Output
Cancel
COURSEID COURSENAME STARTDATE ENDDATE CAPACITY DESCRIPTION DEPARTMENTID CREDITHOURS
1 Introduction to Computer Science 01-SEP-22 15-DEC-22 30 Introduction to CS concepts 1 3
2 Calculus I 01-SEP-22 15-DEC-22 40 Basic calculus principles 2 2
3 Physics for Engineers 01-SEP-22 15-DEC-22 25 Physics fundamentals for engineering students 1 3
4 Organic Chemistry 01-SEP-22 15-DEC-22 35 Study of organic compounds 2 3
5 Introduction to Biology 01-SEP-22 15-DEC-22 20 Basic biology concepts 1 2
6 World History 01-SEP-22 15-DEC-22 30 Survey of world history 2 3
7 Physical Geography 01-SEP-22 15-DEC-22 25 Study of Earth physical features 1 3
8 English Literature 01-SEP-22 15-DEC-22 30 Literary works in English 2 3
9 Ethics in Society 01-SEP-22 15-DEC-22 30 Exploration of ethical issues 1 2
10 Mathematics in Culture 01-SEP-22 15-DEC-22 30 Mathematics role in cultural contexts 2 3
50000 dafsgnh 2024-02-13 2024-02-13 501 lighgh 1 3
Activate Windows


Toad for Oracle - [HEMA_UNI@XE - Editor]
File Edit Search Editor Session Database Debug View Utilities Window Help
HEMA_UNI@XE
Editor Schema Browser Alter Table Alter Table
Project Manager
Trash can
TOAD project
TOAD project
TOAD project
<No name>
1 select * from grades
Data Grid
Data Grid Auto Trace DBMS Output (disabled) Query Viewer CodeXpert Explain Plan Script Output
Cancel
GRADEID GRADEVALUE EVALUATIONDATE PASSFAILSTATUS STUDENTID COURSEID
1 A 2023-01-15 Pass 1 1
2 A- 2023-01-20 Pass 2 2
3 B+ 2023-02-01 Pass 1 3
4 B 2023-02-05 Pass 2 4
5 B- 2023-02-10 Pass 1 5
6 C+ 2023-02-15 Pass 2 1
7 C 2023-02-20 Pass 1 2
8 C- 2023-02-25 Pass 2 3
9 D 2023-03-01 Pass 1 4
10 F 2023-03-05 Fail 2 5
11 D+ 2022-01-31 Pass 3 1
12 A 11-MAR-23 Pass 3 2
Activate Windows


Toad for Oracle - [HEMA_UNI@XE - Editor]
File Edit Search Editor Session Database Debug View Utilities Window Help
HEMA_UNI@XE
Editor Schema Browser Alter Table Alter Table
Project Manager
Trash can
TOAD project
TOAD project
TOAD project
<No name>
1 select * from students_courses
Data Grid
Data Grid Auto Trace DBMS Output (disabled) Query Viewer CodeXpert Explain Plan Script Output
Cancel
STUDENTID COURSEID
1 1
1 2
1 3
1 4
1 5
2 1
2 2
2 3
2 4
2 5
3 1
3 2
C:\Users\UTD\AppData\Roaming\Qu
1: 31 Row 1 of 33 total rows HEMA_UNI@XE Modified

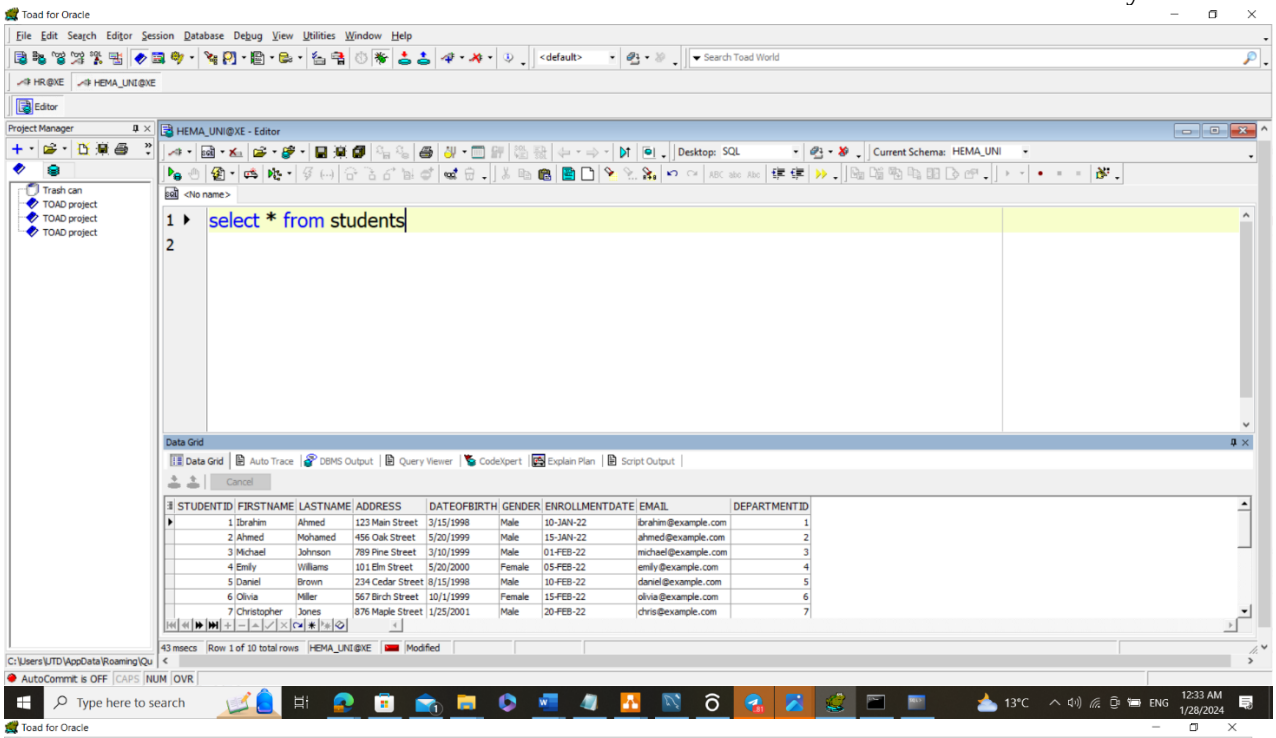
III. PLSQL Implementation

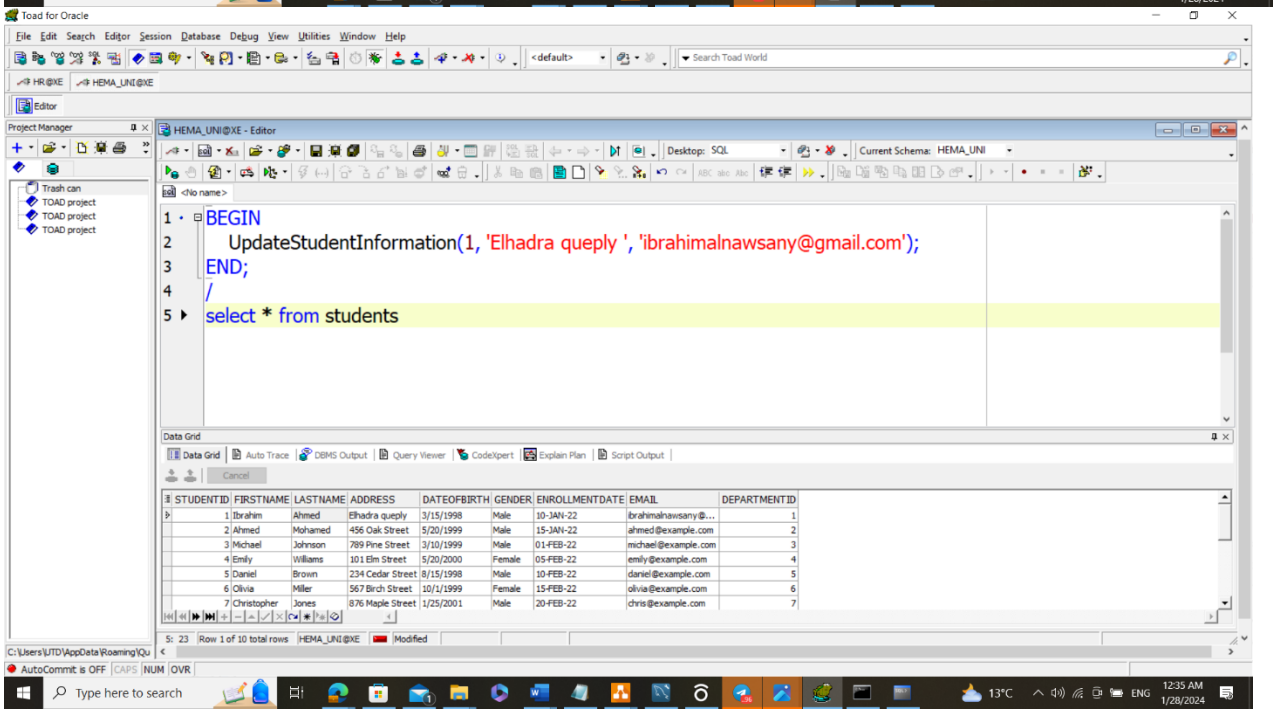
PlSQL Procedure for Updating Student Info:



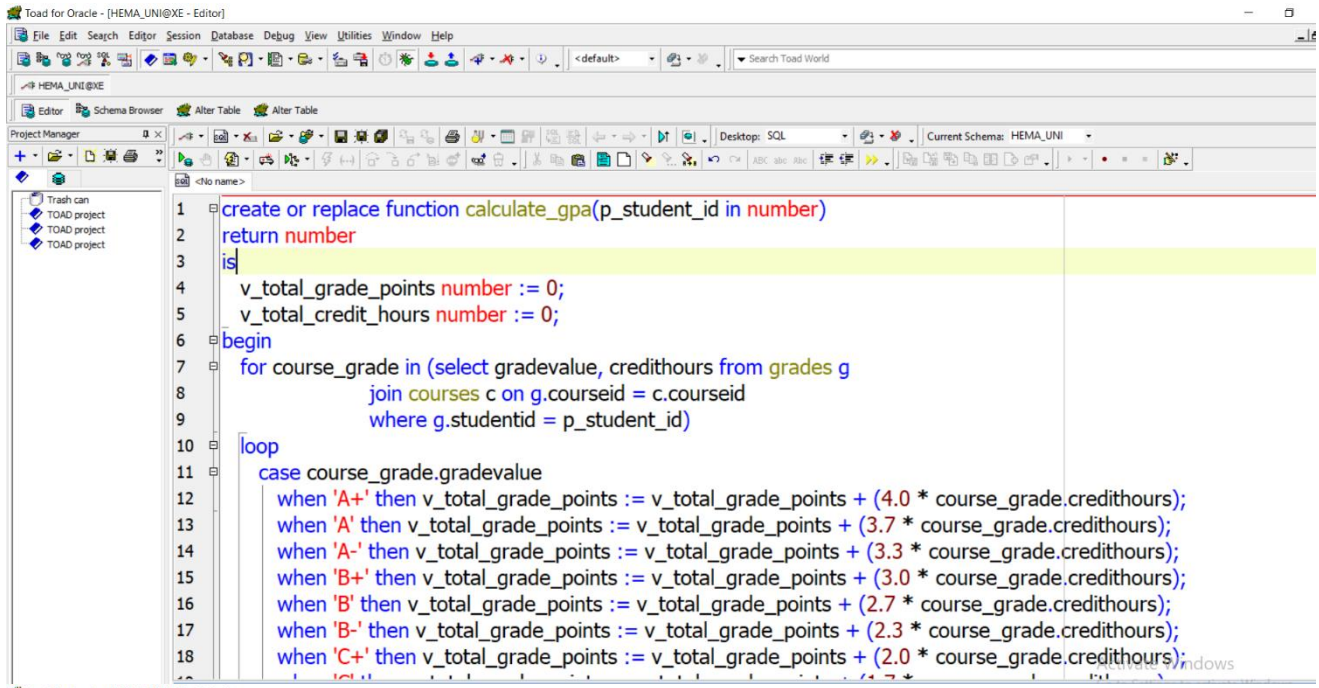
```
create or replace procedure update_student_information(
    p_student_id in number,
    p_new_address in varchar2,
    p_new_email in varchar2
) as
begin
    update students
    set address = p_new_address,
        email = p_new_email
    where studentid = p_student_id;

    dbms_output.put_line('Student information updated successfully.');
exception
    when others then
        dbms_output.put_line('Error updating student information: ' || sqlerrm);
end update_student_information;
/
```

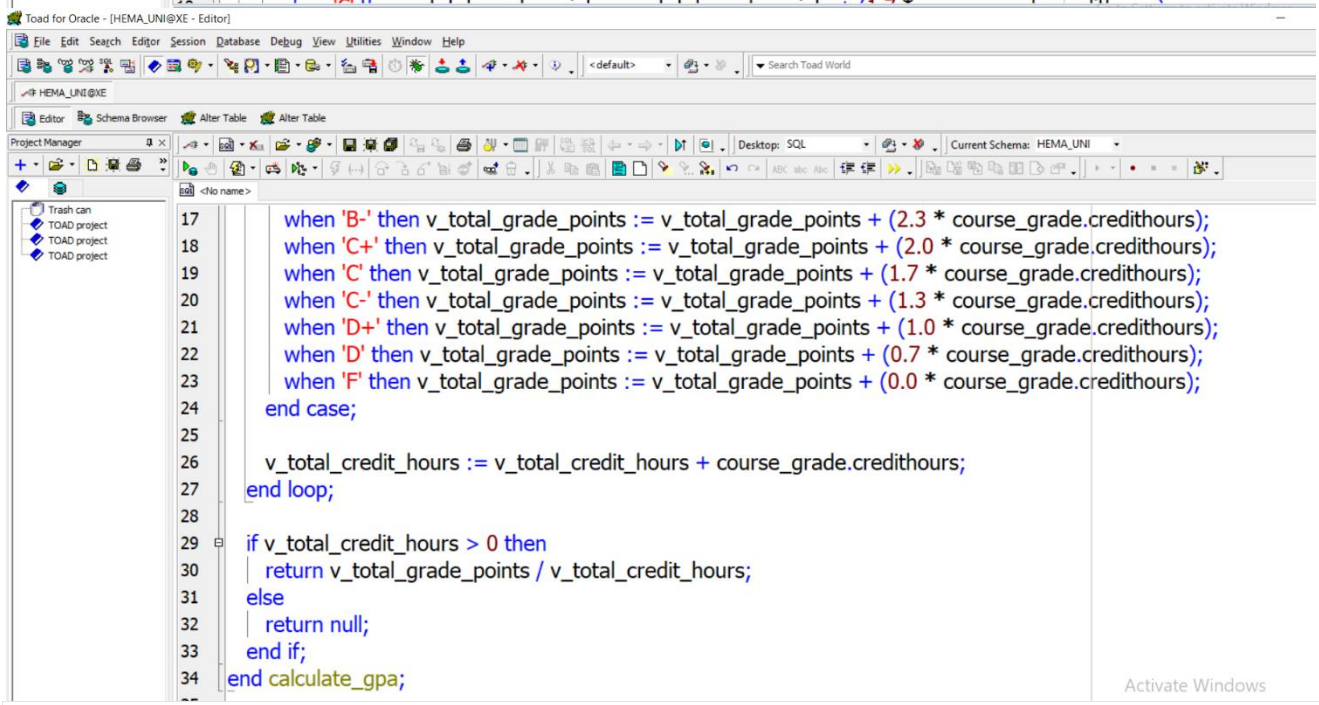
This PL/SQL procedure named **update_student_information** takes three parameters: **p_student_id**, **p_new_address**, and **p_new_email**. It updates the **address** and **email** columns in the **students** table with the new address and email provided as input, respectively, for the specified student ID. If the update is successful, it outputs a success message. If any error occurs during the update process, it catches the exception and outputs an error message along with the error details using **dbms_output.put_line**. This procedure facilitates the updating of student information in the database, handling errors gracefully and providing feedback on the operation's success or failure.


Toad for Oracle
HEMA_UNI@XE - Editor
select * from students
Data Grid
STUDENTID FIRSTNAME LASTNAME ADDRESS DATEOFBIRTH GENDER ENROLLMENTDATE EMAIL DEPARTMENTID
1 Ibrahim Ahmed 123 Main Street 3/15/1998 Male 10-JAN-22 ibrahim@example.com 1
2 Ahmed Mohamed 456 Oak Street 5/20/1999 Male 15-JAN-22 ahmed@example.com 2
3 Michael Johnson 789 Pine Street 3/10/1999 Male 01-FEB-22 michael@example.com 3
4 Emily Williams 101 Elm Street 5/20/2000 Female 05-FEB-22 emily@example.com 4
5 Daniel Brown 234 Cedar Street 8/15/1998 Male 10-FEB-22 daniel@example.com 5
6 Olivia Miller 567 Birch Street 10/1/1999 Female 15-FEB-22 olivia@example.com 6
7 Christopher Jones 876 Maple Street 1/25/2001 Male 20-FEB-22 chris@example.com 7

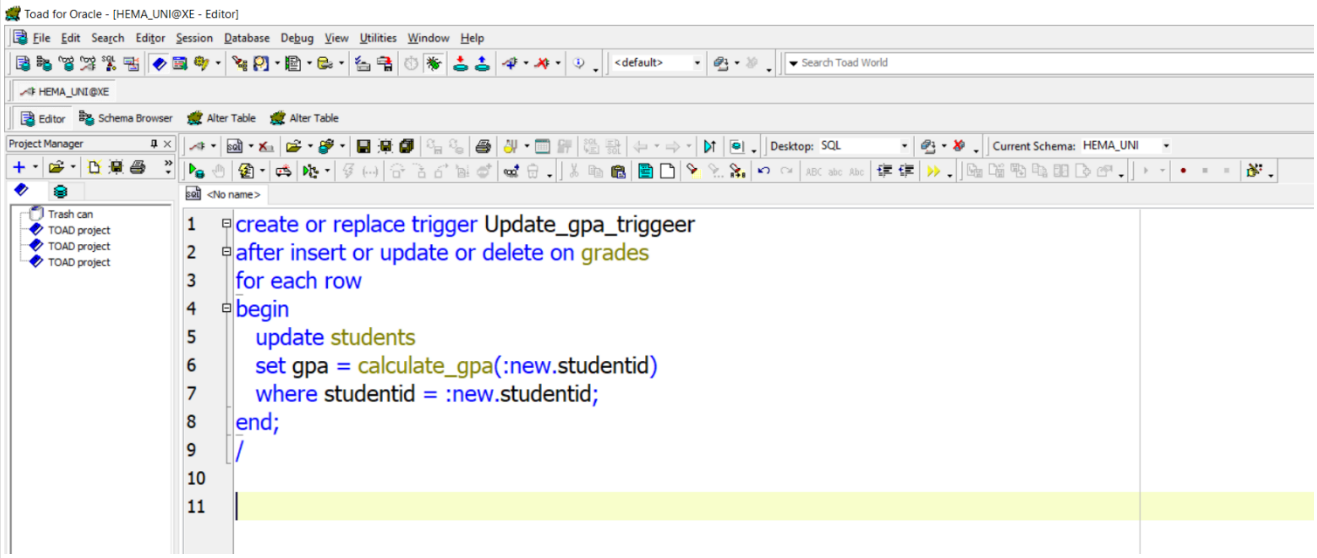

Toad for Oracle
HEMA_UNI@XE - Editor
BEGIN
UpdateStudentInformation(1, 'Elhadra queply ', 'ibrahimalnawsany@gmail.com');
END;
/
select * from students
Data Grid
STUDENTID FIRSTNAME LASTNAME ADDRESS DATEOFBIRTH GENDER ENROLLMENTDATE EMAIL DEPARTMENTID
1 Ibrahim Ahmed Elhadra queply 3/15/1998 Male 10-JAN-22 ibrahimalnawsany@... 1
2 Ahmed Mohamed 456 Oak Street 5/20/1999 Male 15-JAN-22 ahmed@example.com 2
3 Michael Johnson 789 Pine Street 3/10/1999 Male 01-FEB-22 michael@example.com 3
4 Emily Williams 101 Elm Street 5/20/2000 Female 05-FEB-22 emily@example.com 4
5 Daniel Brown 234 Cedar Street 8/15/1998 Male 10-FEB-22 daniel@example.com 5
6 Olivia Miller 567 Birch Street 10/1/1999 Female 15-FEB-22 olivia@example.com 6
7 Christopher Jones 876 Maple Street 1/25/2001 Male 20-FEB-22 chris@example.com 7

PlSQL Function For Calculating Student GPA:



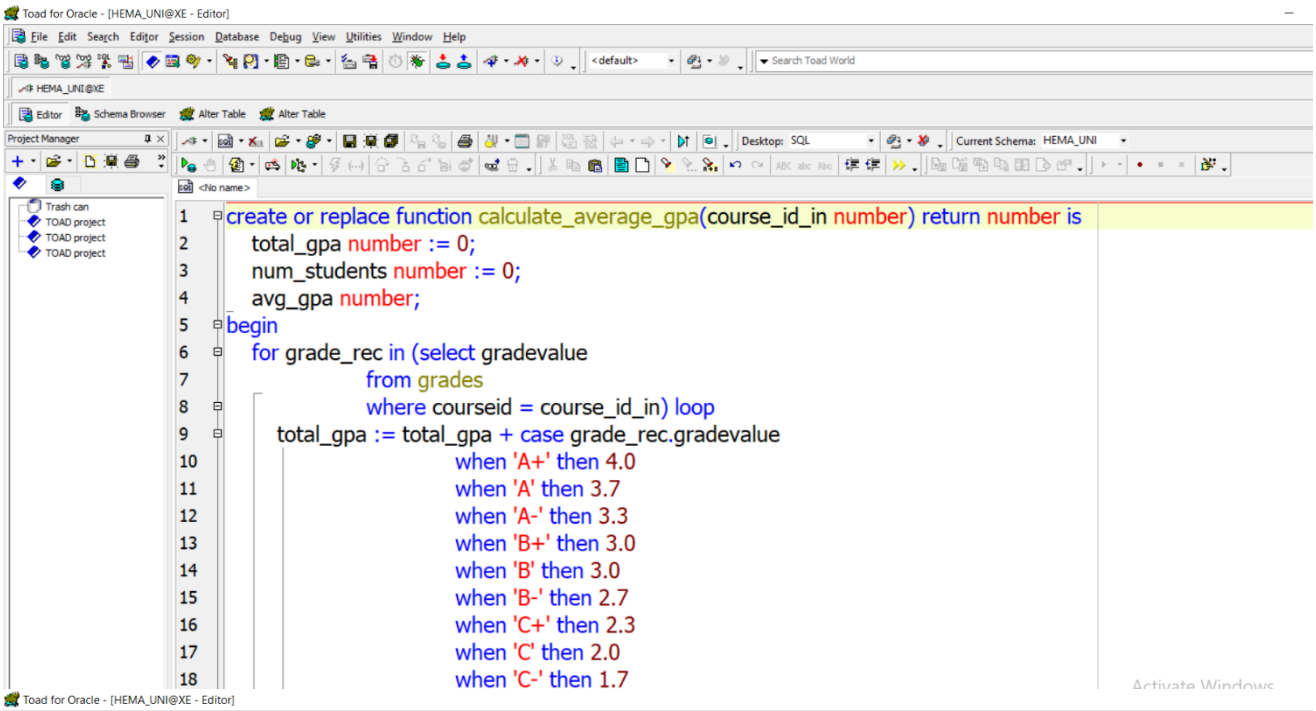


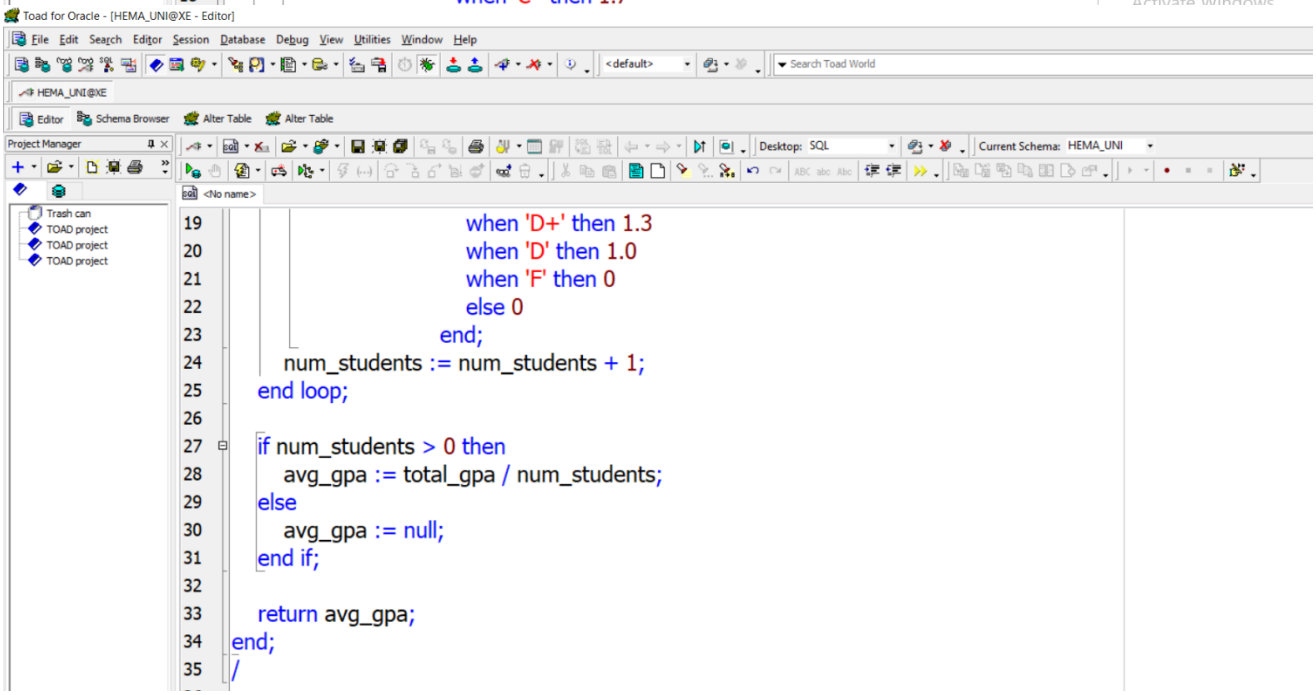
This PL/SQL function named **calculate_gpa** takes a student ID as input and calculates the GPA (Grade Point Average) for that student. It does so by retrieving grades and credit hours for courses associated with the given student ID from the database and then iterating through each course grade. For each grade, it assigns a numerical value based on a predefined scale and calculates the total grade points by multiplying the grade value with the credit hours for the course. It also accumulates the total credit hours. Finally, it computes the GPA by dividing the total grade points by the total credit hours. If no credit hours are found, it returns null. This function can be useful for academic institutions or systems managing student records to automatically compute GPAs based on course grades and credit hours.



This PL/SQL trigger named **Update_gpa_trigger** is fired after an insert, update, or delete operation on the **grades** table. For each affected row, it updates the **gpa** column in the **students** table by invoking the **calculate_gpa** function with the student ID of the affected row. This trigger ensures that whenever there's a change in grades for a student, their GPA is automatically recalculated and updated in the **students** table. It helps in maintaining up-to-date GPA records for students without manual intervention.

PlSQL Function For Calculating Average GPA:





To help me in Java report This PL/SQL function named **calculate_average_gpa** takes a course ID as input and calculates the average GPA of students enrolled in that course. It iterates through each grade recorded for the specified course, converting each grade to its corresponding numerical GPA value based on a predefined scale. It accumulates the total GPA and counts the number of students. Finally, it computes the average GPA by dividing the total GPA by the number of students. If there are no students enrolled in the course, it returns null. This function is useful for determining the overall performance of students in a particular course and can assist in academic analysis and decision-making

IV. Automation Scripts

Bash script for database backup.

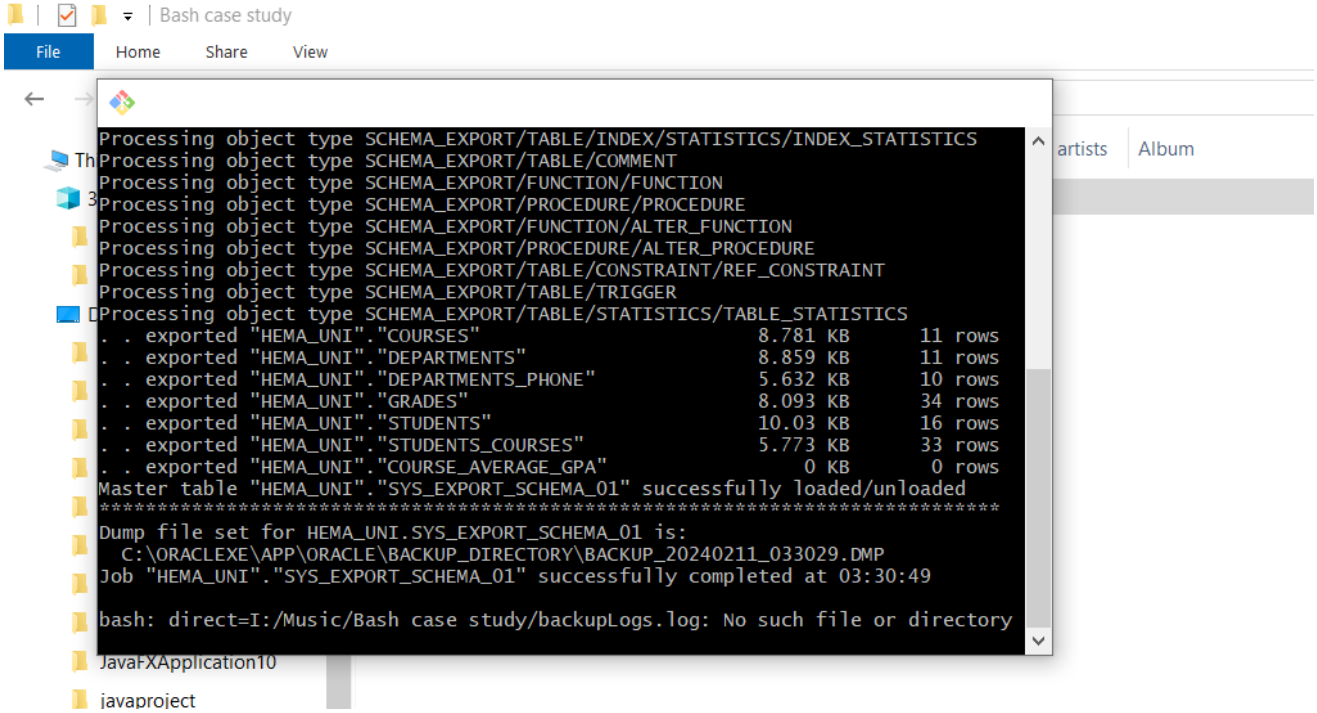
I:\Music\Bash case study\backup_script.sh - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

tnsnames.ora | change.log | grades.fxml | backup_script.sh | disk_monitor_script.sh

```
#!/bin/bash

db_owner=HEMA_UNI
pass=123
db=XE
direct=direct="I:/Music/Bash case study"
d=$(date +"%Y%m%d_%H%M%S")
file_name="backup_${d}.dmp"

expdp ${db_owner}/${pass}@${db} DUMPFILE=${file_name}  DIRECTORY=backup_directory;

if [ $? -eq 0 ]; then
    echo " Make Backup Successfully " >> "${direct}/backupLogs.log"
else
    echo " ERROR : Make Backup Failed " >> "${direct}/backupLogs.log"
fi
```

On Running the Script

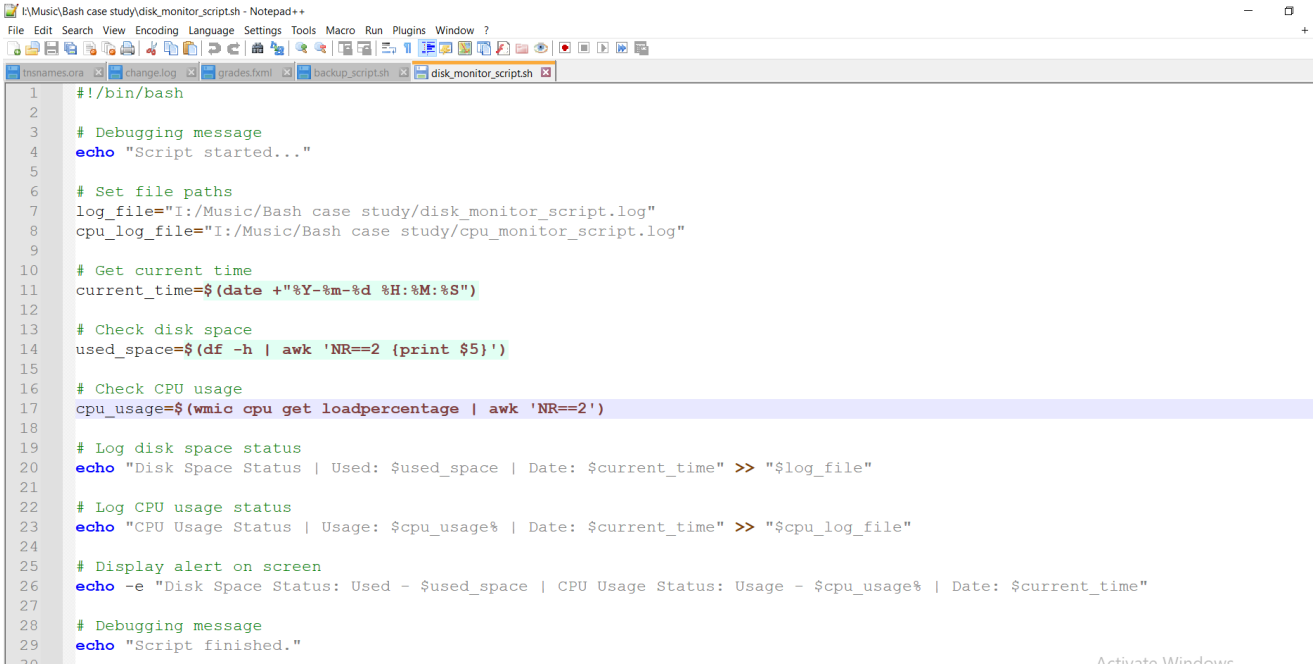


The provided Bash script automates the process of exporting data from an Oracle database using Oracle Data Pump. It initializes variables for the database owner, password, database name, directory path for storing backups, and generates a timestamp for the backup file name.

Upon execution, it invokes the **expdp** command with appropriate parameters to export the data to a dump file in the specified directory. Subsequently, it checks the exit status of the **expdp** command and logs the outcome, either indicating successful backup completion or failure, to a designated log file.

Overall, the script streamlines the backup process, enhancing operational efficiency and ensuring data integrity through automated logging of backup outcomes.

Bash script for monitoring disk space and sending alerts.



MINGW64:/i/Music/Bash case study

```
UTD@Nwasaaaaaanyyyyyyy MINGW64 ~
$ cd "I:/Music/Bash case study"

UTD@Nwasaaaaaanyyyyyyy MINGW64 /i/Music/Bash case study
$ bash disk_monitor_script.sh
Script started...
Disk Space Status: Used - 45G | CPU Usage Status: Usage - 8                % | Da
te: 2024-02-11 20:44:14
Script finished.

UTD@Nwasaaaaaanyyyyyyy MINGW64 /i/Music/Bash case study
$
```

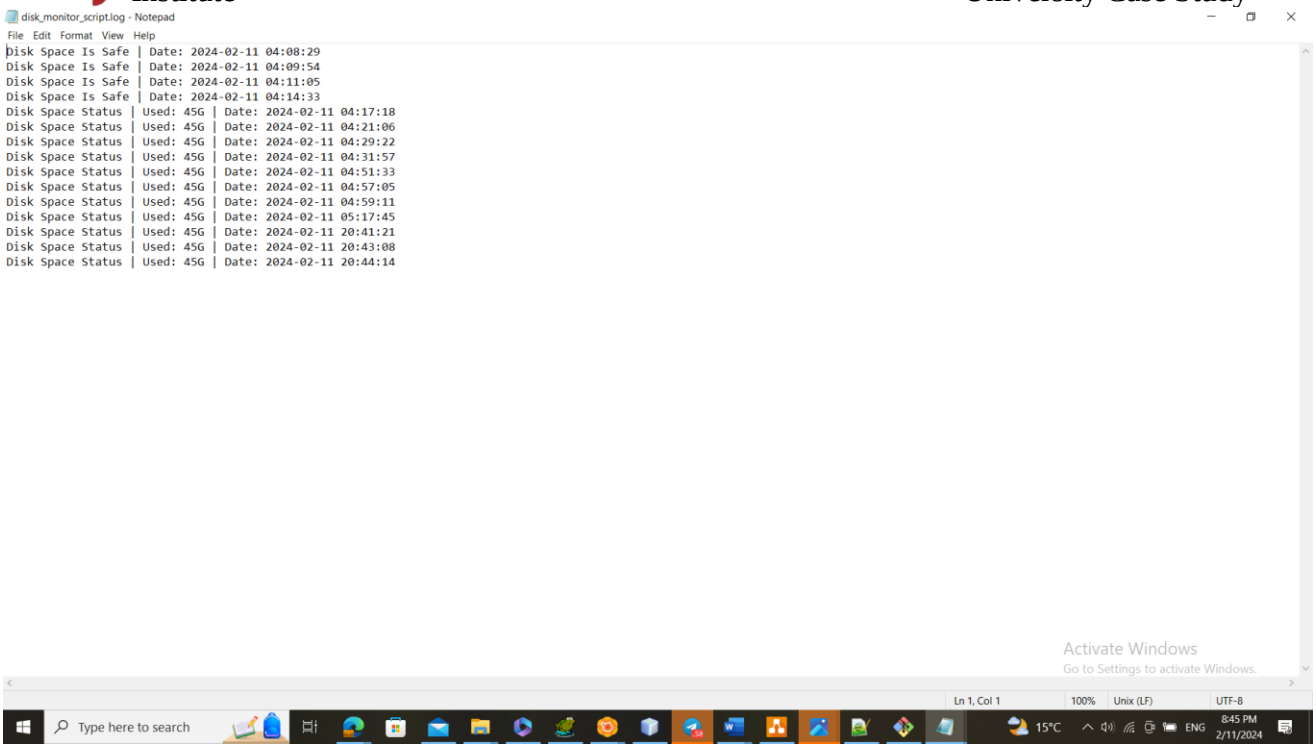
cpu_monitor_script.log - Notepad

File Edit Format View Help

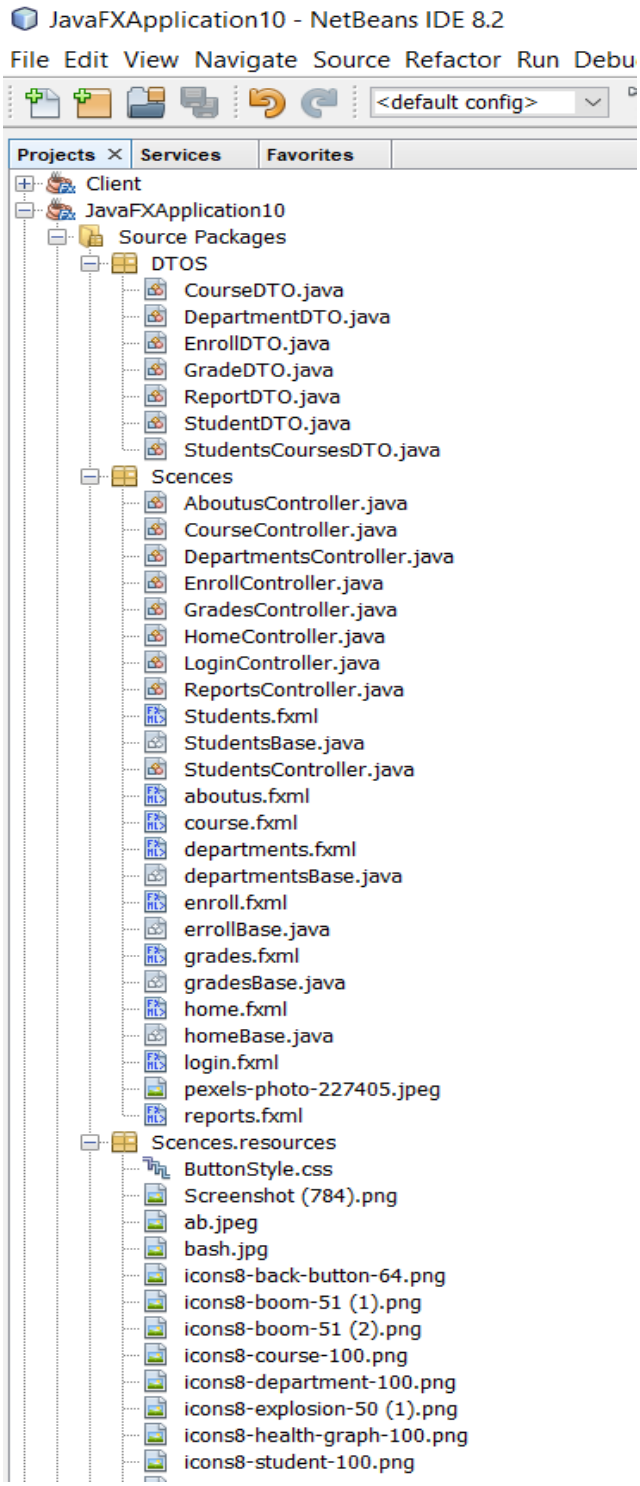
```
CPU Usage Is Safe | Date: 2024-02-11 04:08:29
CPU Usage Is Safe | Date: 2024-02-11 04:09:54
CPU Usage Is Safe | Date: 2024-02-11 04:11:05
CPU Usage Is Safe | Date: 2024-02-11 04:14:33
CPU Usage Status | Usage: 15              % | Date: 2024-02-11 04:17:18
CPU Usage Status | Usage: 24              % | Date: 2024-02-11 04:21:06
CPU Usage Status | Usage: 30              % | Date: 2024-02-11 04:29:22
CPU Usage Status | Usage: 11              % | Date: 2024-02-11 04:31:57
CPU Usage Status | Usage: 63              % | Date: 2024-02-11 04:51:33
CPU Usage Status | Usage: 8               % | Date: 2024-02-11 04:57:05
CPU Usage Status | Usage: 27              % | Date: 2024-02-11 04:59:11
CPU Usage Status | Usage: 21              % | Date: 2024-02-11 05:17:45
CPU Usage Status | Usage: 14              % | Date: 2024-02-11 20:41:21
CPU Usage Status | Usage: 5               % | Date: 2024-02-11 20:43:08
CPU Usage Status | Usage: 8               % | Date: 2024-02-11 20:44:14
```

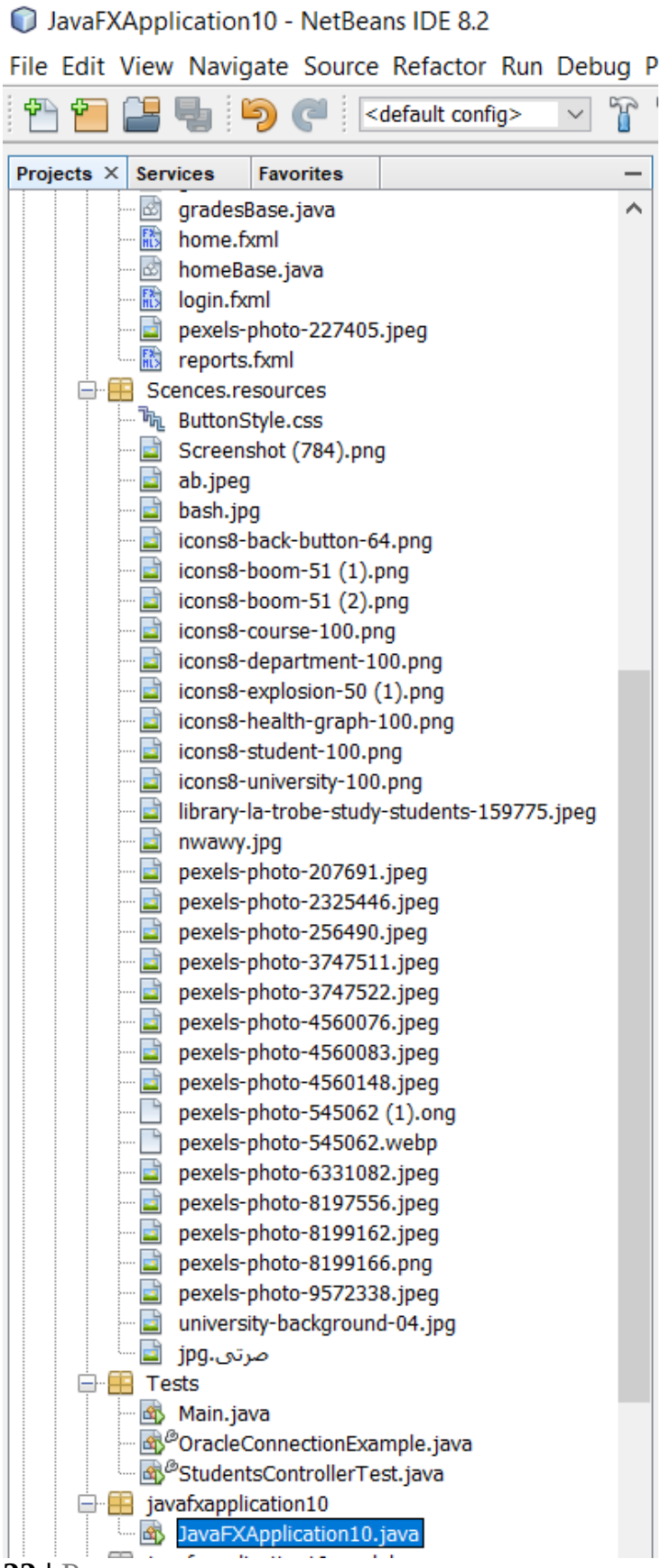

Ln 1, Col 1 | 100% | Unix (LF) | UTF-8

15°C ENG 8:45 PM 2/11/2024


disk_monitor_script.log - Notepad
File Edit Format View Help
Disk Space Is Safe | Date: 2024-02-11 04:08:29
Disk Space Is Safe | Date: 2024-02-11 04:09:54
Disk Space Is Safe | Date: 2024-02-11 04:11:05
Disk Space Is Safe | Date: 2024-02-11 04:14:33
Disk Space Status | Used: 45G | Date: 2024-02-11 04:17:18
Disk Space Status | Used: 45G | Date: 2024-02-11 04:21:06
Disk Space Status | Used: 45G | Date: 2024-02-11 04:29:22
Disk Space Status | Used: 45G | Date: 2024-02-11 04:31:57
Disk Space Status | Used: 45G | Date: 2024-02-11 04:51:33
Disk Space Status | Used: 45G | Date: 2024-02-11 04:57:05
Disk Space Status | Used: 45G | Date: 2024-02-11 04:59:11
Disk Space Status | Used: 45G | Date: 2024-02-11 05:17:45
Disk Space Status | Used: 45G | Date: 2024-02-11 20:41:21
Disk Space Status | Used: 45G | Date: 2024-02-11 20:43:08
Disk Space Status | Used: 45G | Date: 2024-02-11 20:44:14
Activate Windows
Go to Settings to activate Windows.
Ln 1, Col 1
100%
Unix (LF)
UTF-8
Type here to search
15°C
ENG
8:45 PM
2/11/2024

V. Java Application Development

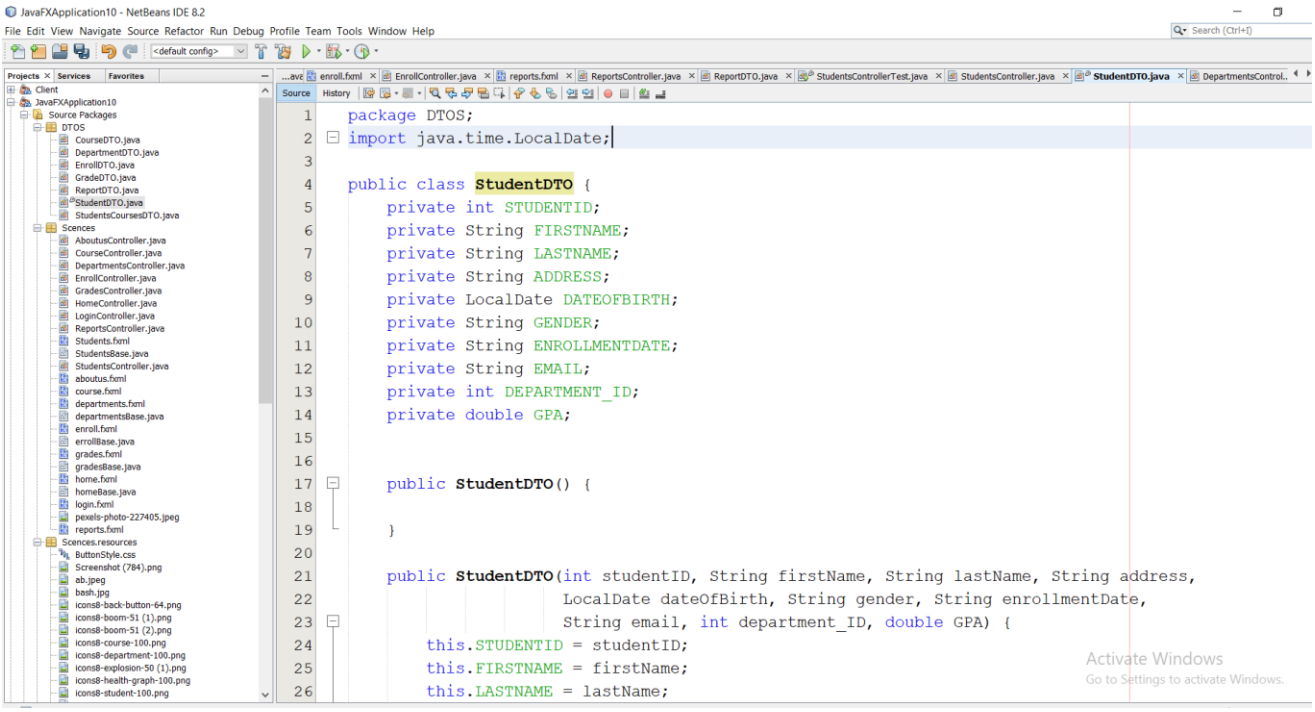



JavaFXApplication10 - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug P
<default config>
Projects
Services
Favorites
gradesBase.java
home.fxml
homeBase.java
login.fxml
pexels-photo-227405.jpeg
reports.fxml
Scences.resources
ButtonStyle.css
Screenshot (784).png
ab.jpeg
bash.jpg
icons8-back-button-64.png
icons8-boom-51 (1).png
icons8-boom-51 (2).png
icons8-course-100.png
icons8-department-100.png
icons8-explosion-50 (1).png
icons8-health-graph-100.png
icons8-student-100.png
icons8-university-100.png
library-la-trobe-study-students-159775.jpeg
nwawy.jpg
pexels-photo-207691.jpeg
pexels-photo-2325446.jpeg
pexels-photo-256490.jpeg
pexels-photo-3747511.jpeg
pexels-photo-3747522.jpeg
pexels-photo-4560076.jpeg
pexels-photo-4560083.jpeg
pexels-photo-4560148.jpeg
pexels-photo-545062 (1).ong
pexels-photo-545062.webp
pexels-photo-6331082.jpeg
pexels-photo-8197556.jpeg
pexels-photo-8199162.jpeg
pexels-photo-8199166.png
pexels-photo-9572338.jpeg
university-background-04.jpg
jpg.صرتى
Tests
Main.java
OracleConnectionExample.java
StudentsControllerTest.java
javafxapplication10
JavaFXApplication10.java

DTOS Package:

- Contain Java classes for each object including attributes, constructor , setters and getters.
- We use these classes to extract object information and to fill the table views
- Each object will represent a scene in the application UI.

Ex: Students DTO (other DTOS with the same way found them in java code folder)



JavaFXApplication10 Package

Components:

JavaFXApplication10.java class responsible for initializing and configuring the JavaFX application, stat from login , enter the home fxml , loading the user interface layout from an FXML file, and displayingthe application window.

```
public class JavaFXApplication10 extends Application {


    @Override
    public void start(Stage stage) throws Exception {
        // Load the FXML file using an absolute path
        Parent root = FXMLLoader.load(getClass().getResource("/Scences/login.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

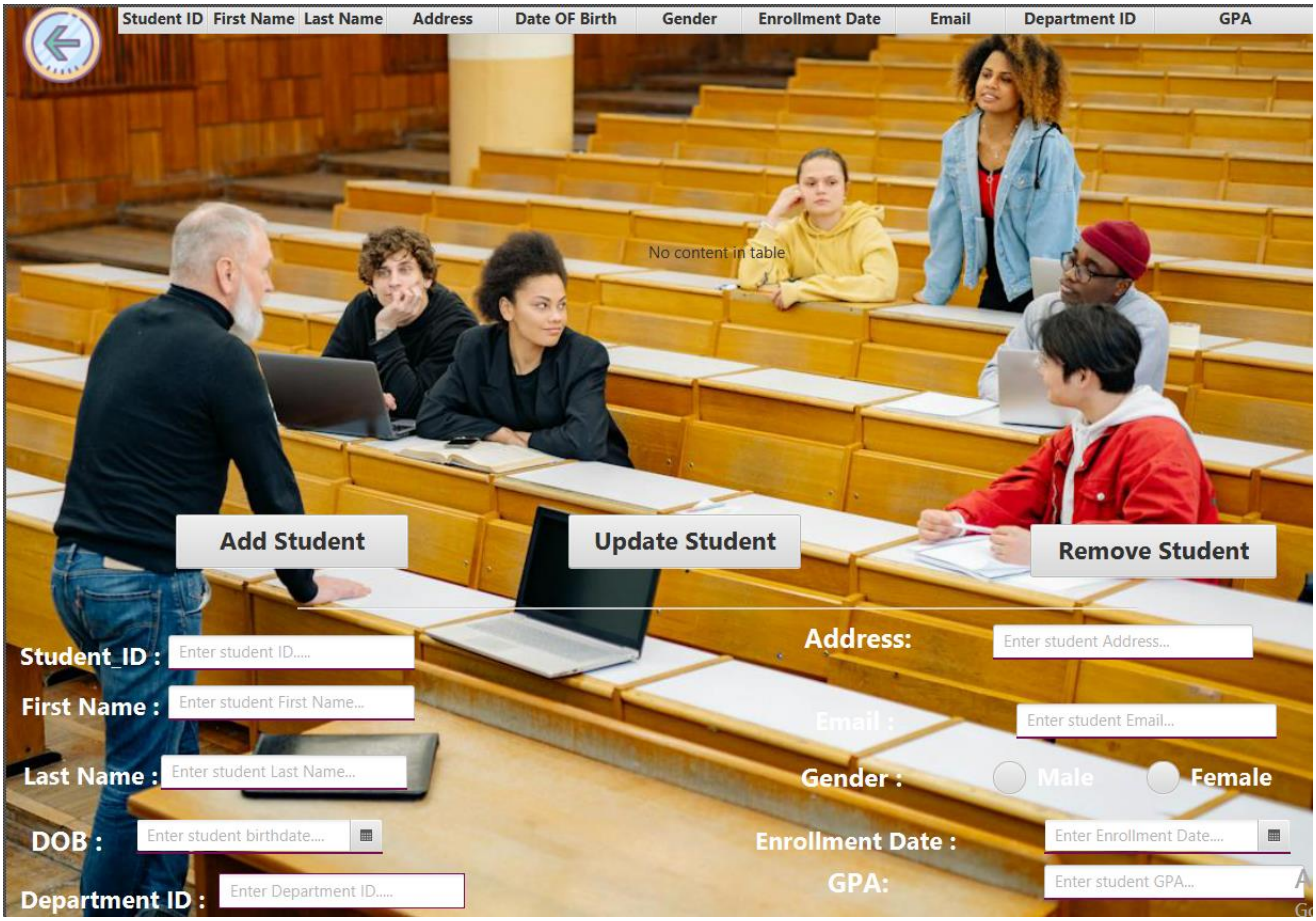
The Application Scenes:

Login Scene:

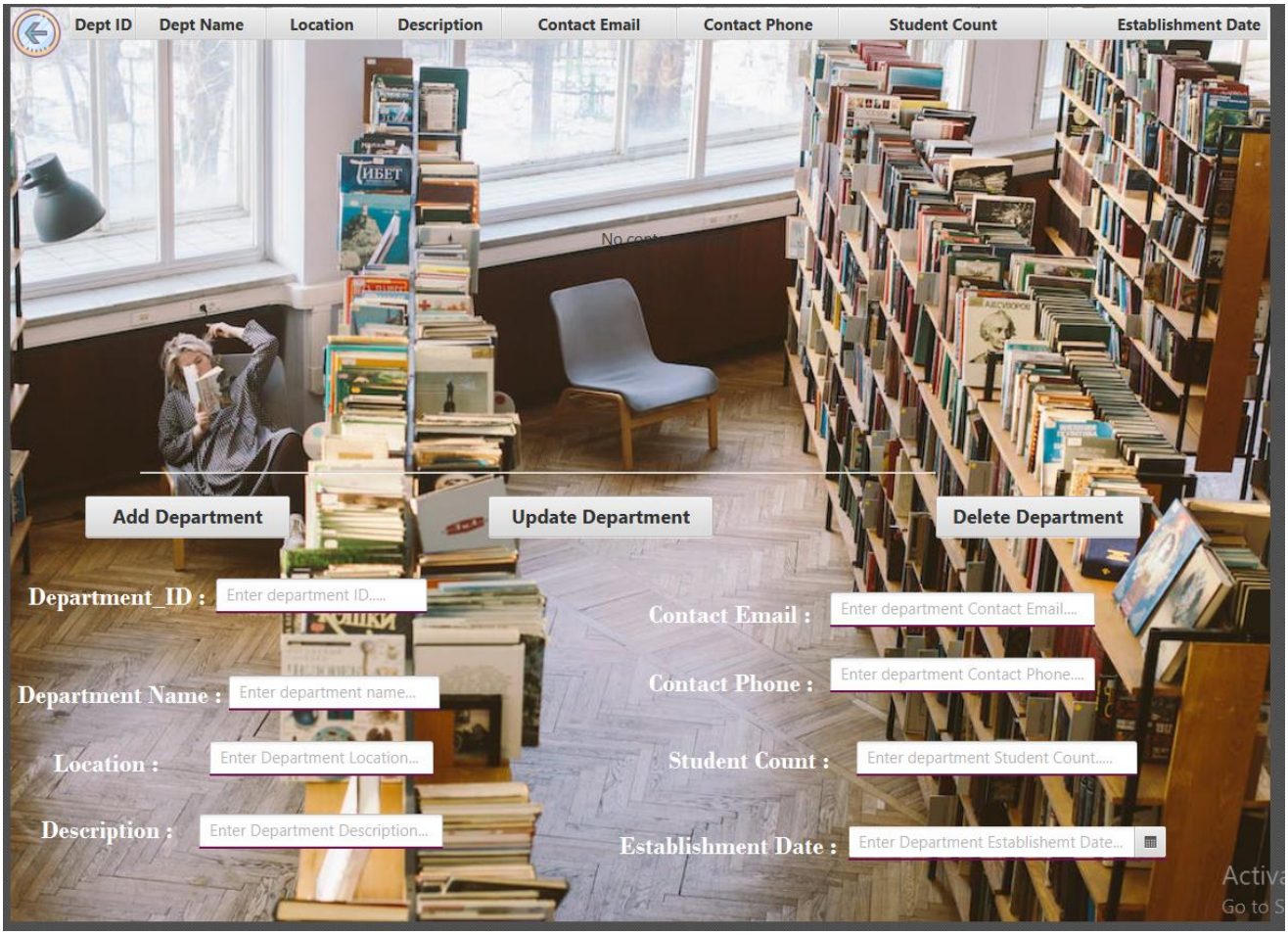
Home Scene:


Students
Departments
Courses
Grades
Enroll
Report
About us
Log out
Welcome to ElNwawy
University

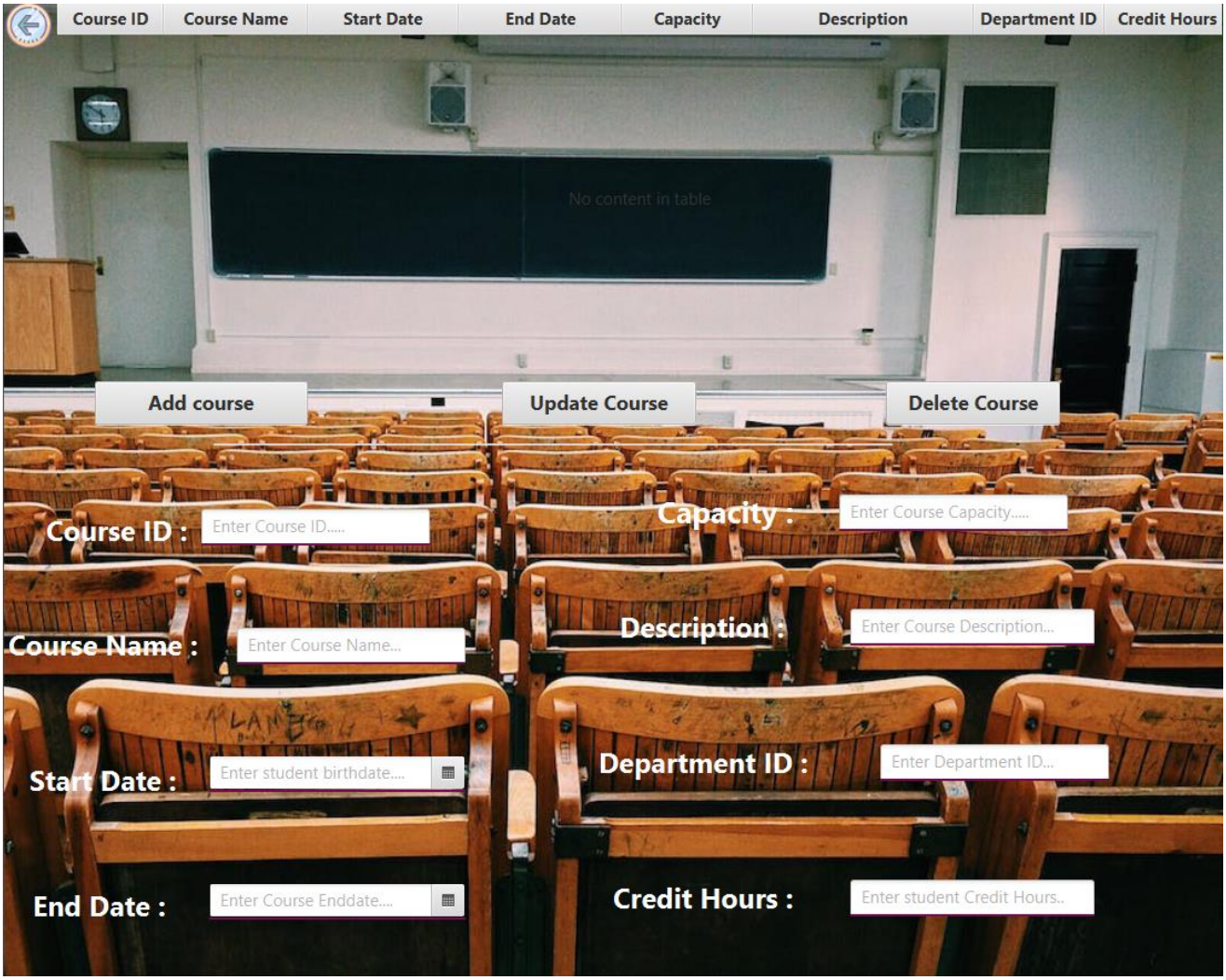
Students Scene:



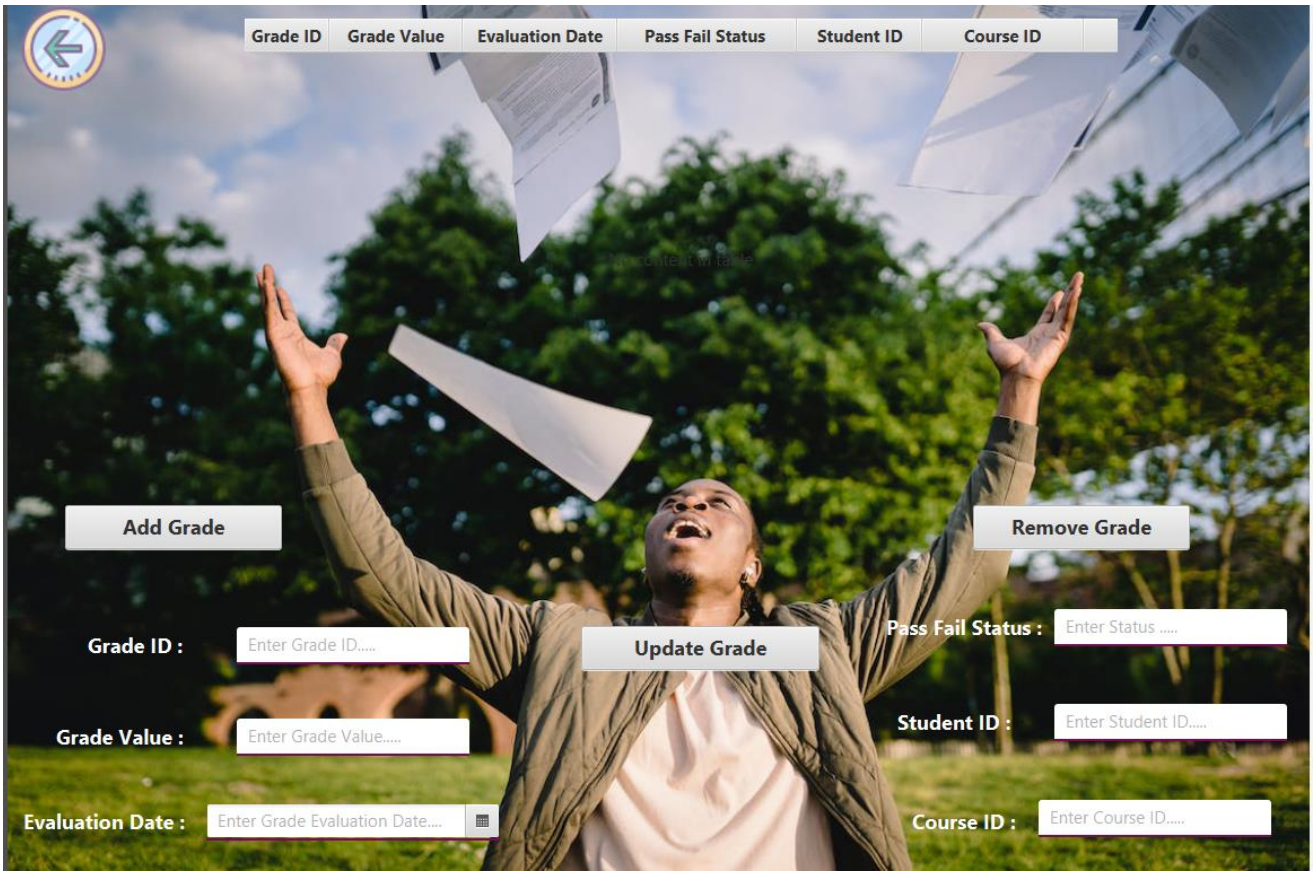
Departments Scene:



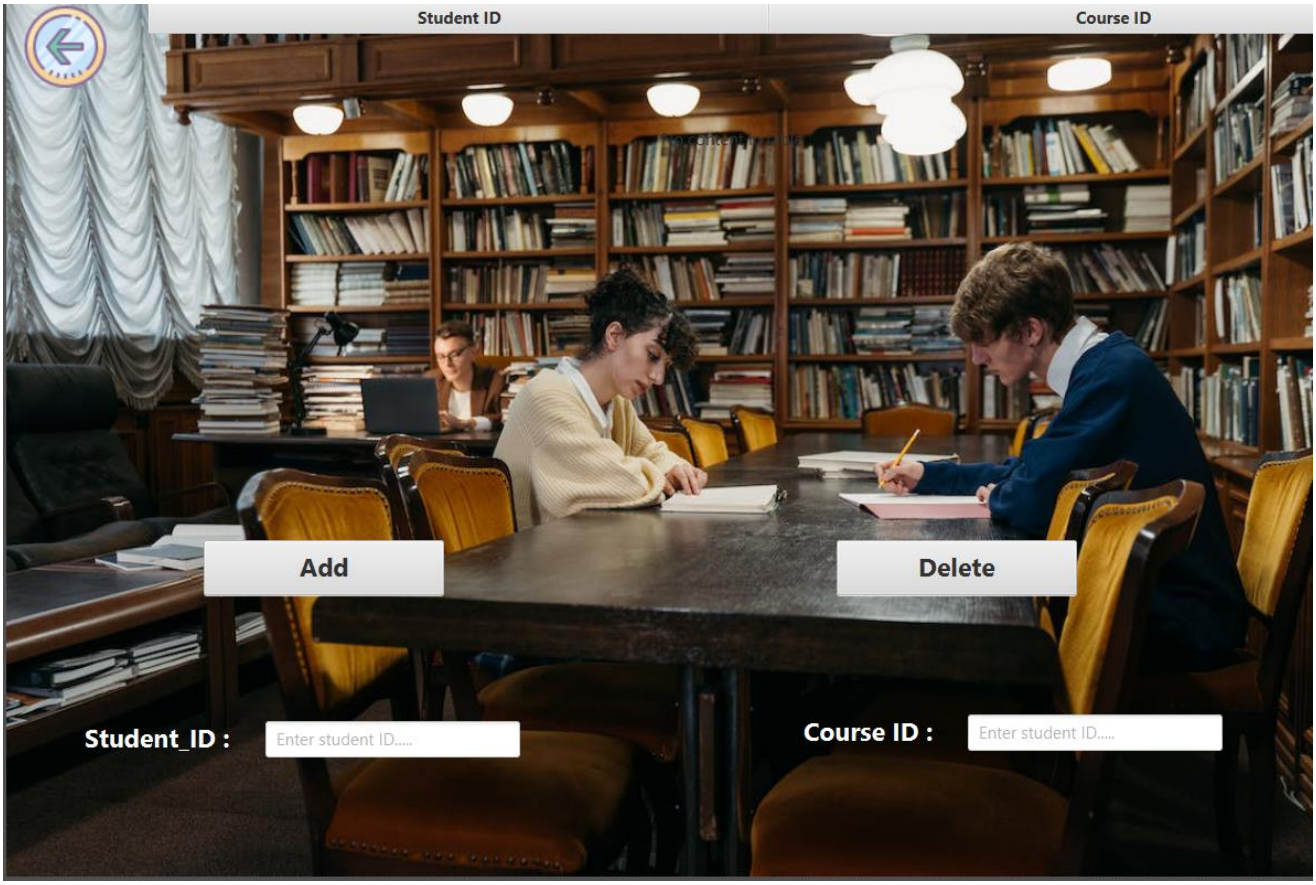
Courses Scene:



Grades Scene:

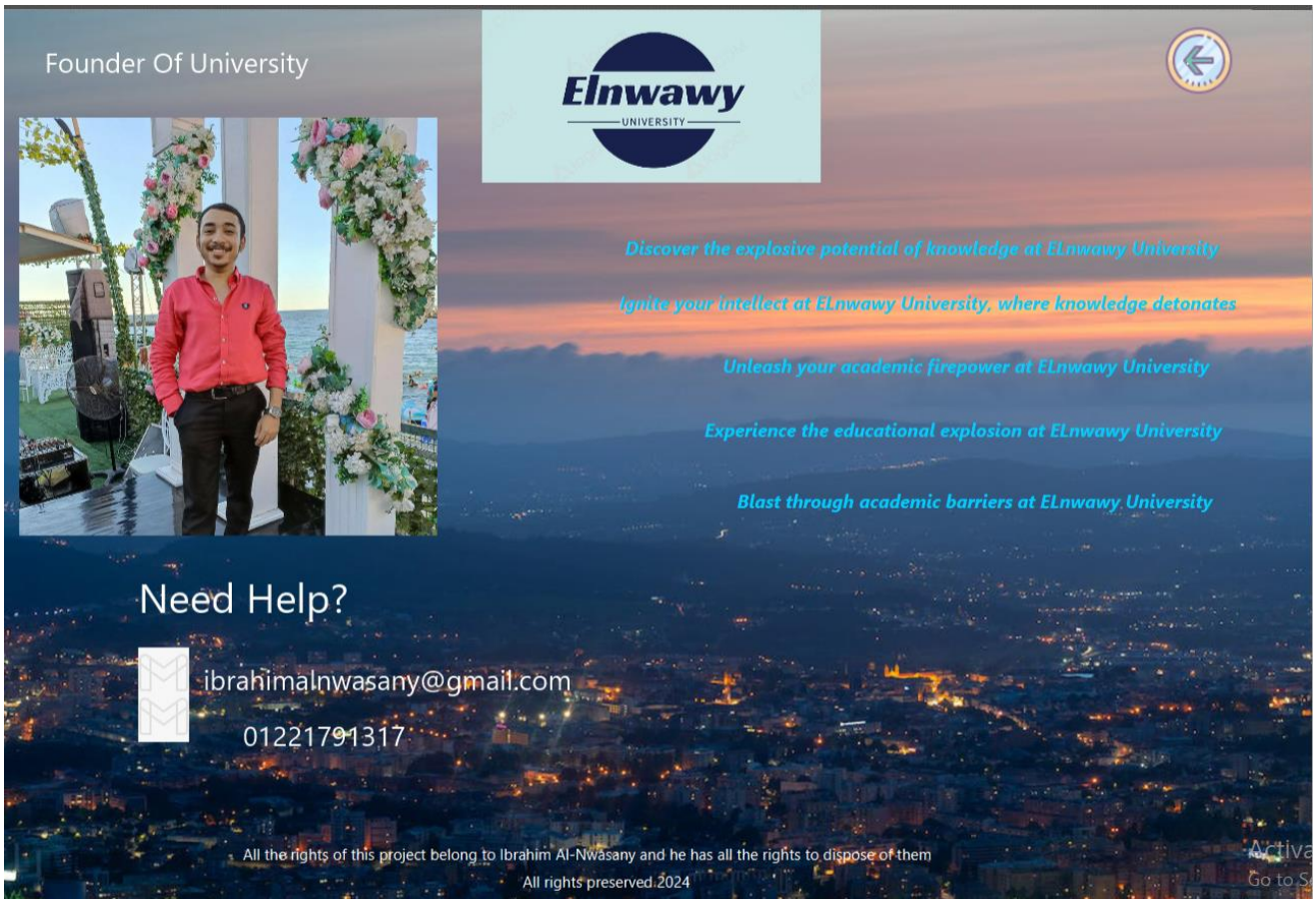


Enroll Scene:



Report Scene:

About Scene:


Founder Of University
Elnwawy
UNIVERSITY
Discover the explosive potential of knowledge at ELnwawy University
Ignite your intellect at ELnwawy University, where knowledge detonates
Unleash your academic firepower at ELnwawy University
Experience the educational explosion at ELnwawy University
Blast through academic barriers at ELnwawy University
Need Help?
ibrahimalnwasany@gmail.com
01221791317
All the rights of this project belong to Ibrahim Al-Nwasany and he has all the rights to dispose of them
All rights preserved.2024

The Application Controllers (All codes in java file code) :

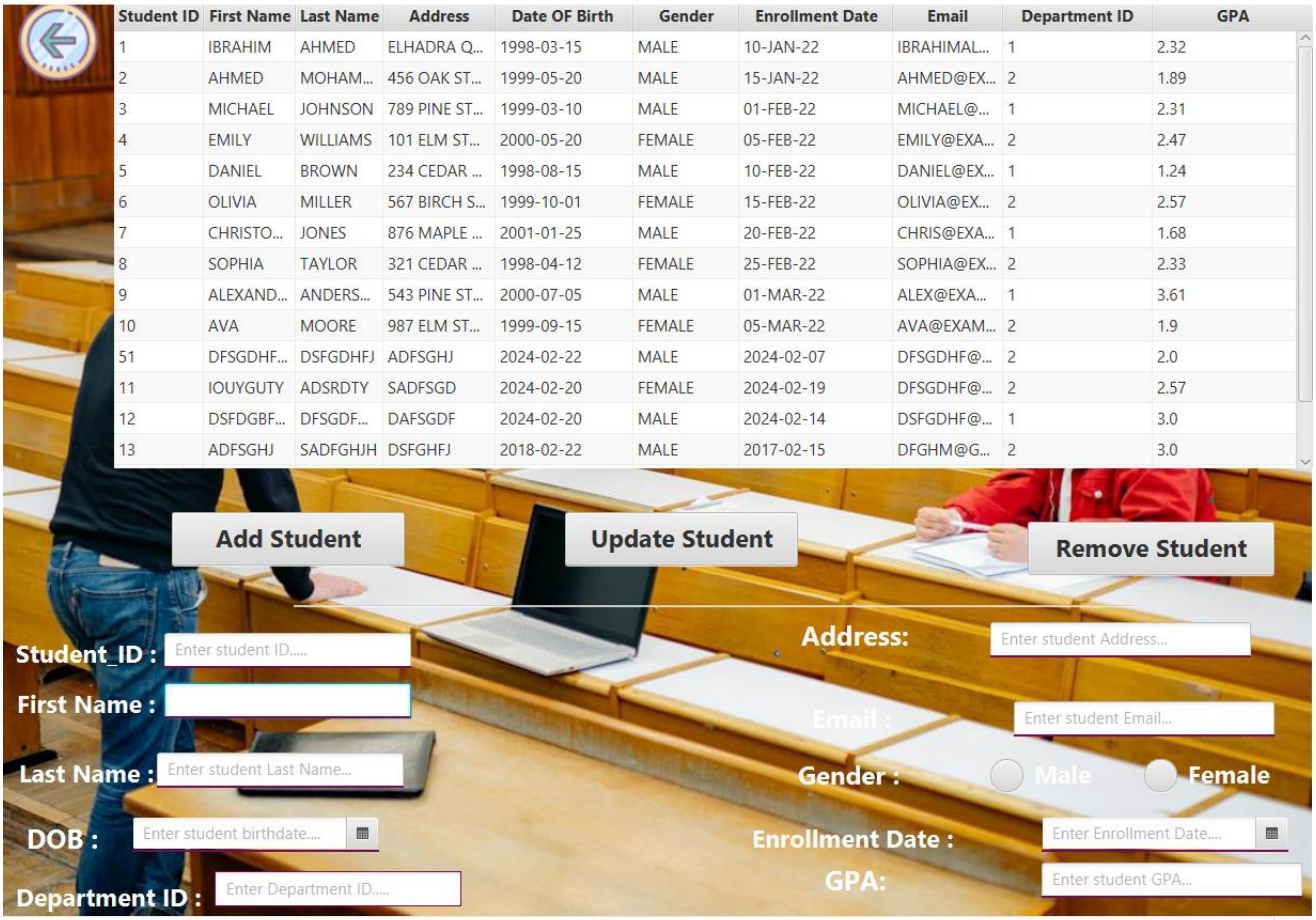
Login controller:

The controller class manages the login functionality for the JavaFX application. It initializes GUI elements defined in the FXML file, such as labels, text fields, and buttons. The **loggin()** method handles user authentication by comparing entered credentials with predefined admin credentials. If authentication is successful, it switches to the home scene using the **switchToHomeScene()** method. This method loads the home scene FXML file, sets it as the active scene, and displays it on the stage. If authentication fails, it can display an error message or handle the situation as needed. Overall, the controller orchestrates the login process and scene navigation based on user input.

Home controller:

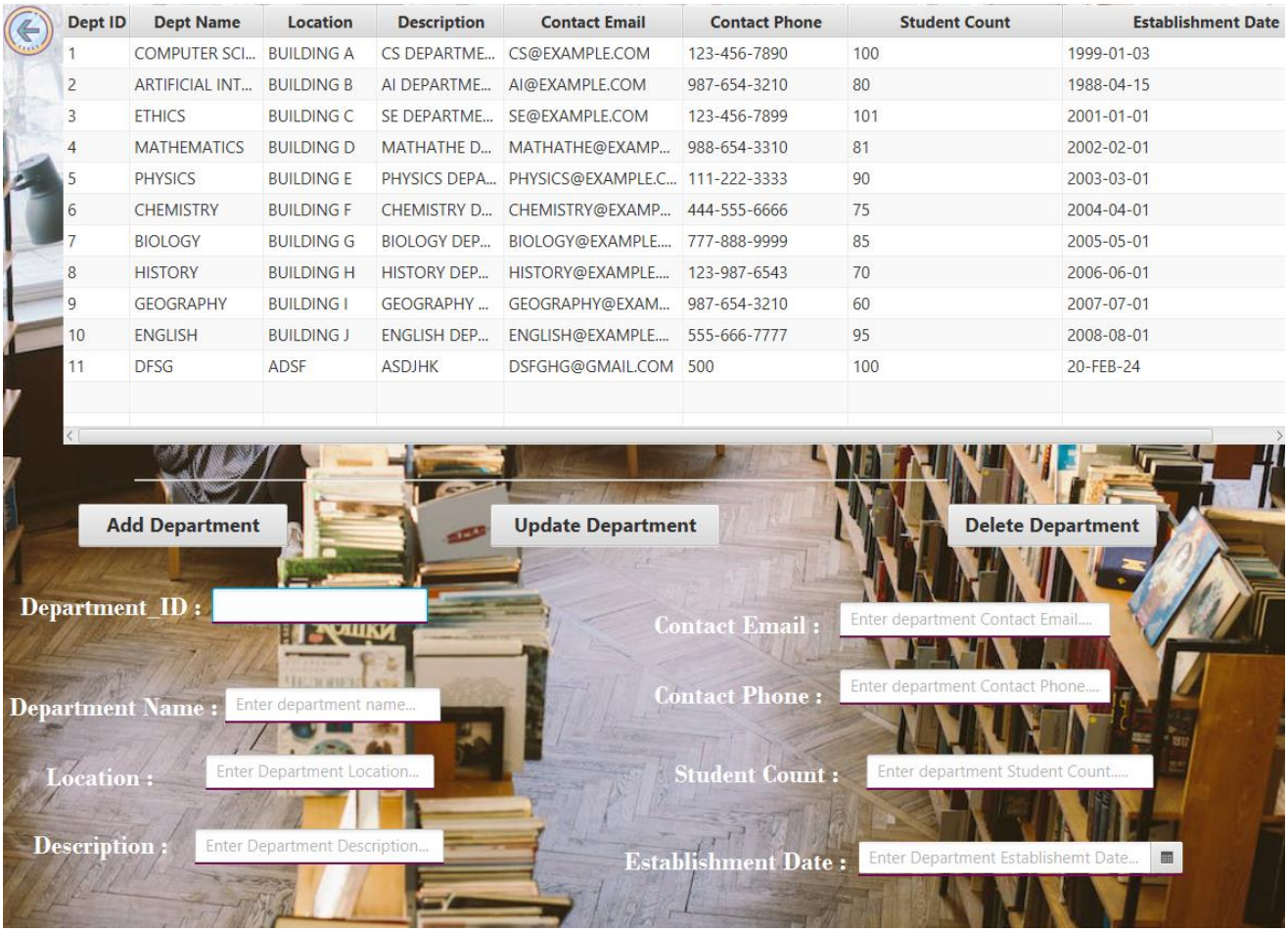
The HomeController class manages the navigation and functionality of the home screen in the JavaFX application. It initializes GUI elements defined in the FXML file, such as buttons, labels, and image views. The controller includes methods triggered by button clicks, enabling users to access different functionalities like viewing students, departments, courses, reports, grades, enrollment, logging out, and accessing information about the application. Each button click invokes the loadFXML() method, which loads the corresponding FXML file into the replacePane AnchorPane, effectively replacing the current content with the new scene. Overall, the HomeController facilitates seamless navigation and interaction within the application's home screen.

Students Controller:



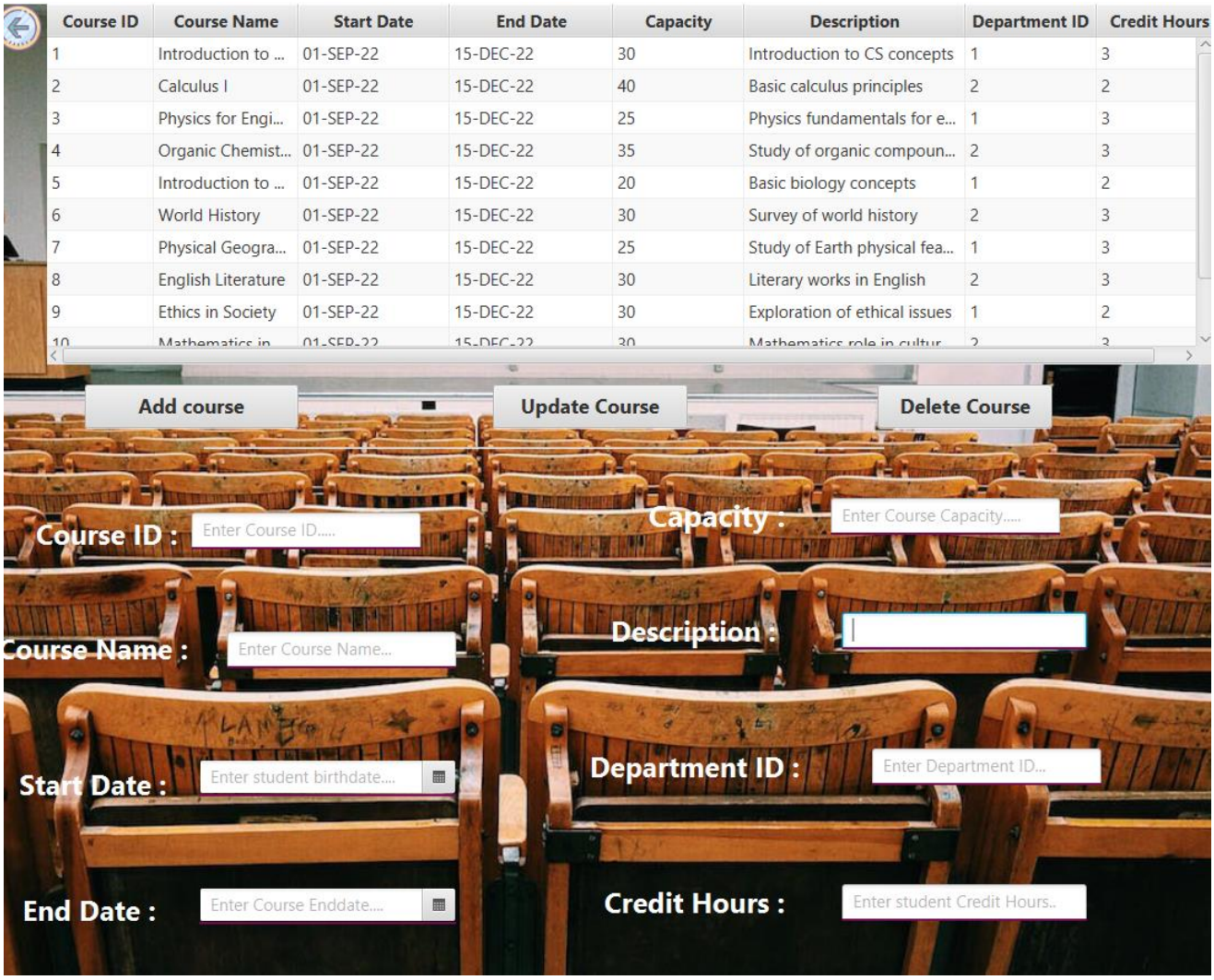
The StudentsController class manages the functionality of the students' management screen in the JavaFX application. It initializes GUI elements defined in the FXML file, such as text fields, labels, buttons, and a table view. The controller includes methods triggered by user actions, such as adding, updating, and removing students from the database. It establishes a connection to the database and loads student data into the TableView upon initialization. The controller also validates user input and displays appropriate alerts for error handling. Additionally, it includes a method to navigate back to the home screen when the corresponding button is clicked. Overall, the StudentsController facilitates effective management of student data within the application.

Departments Controller:



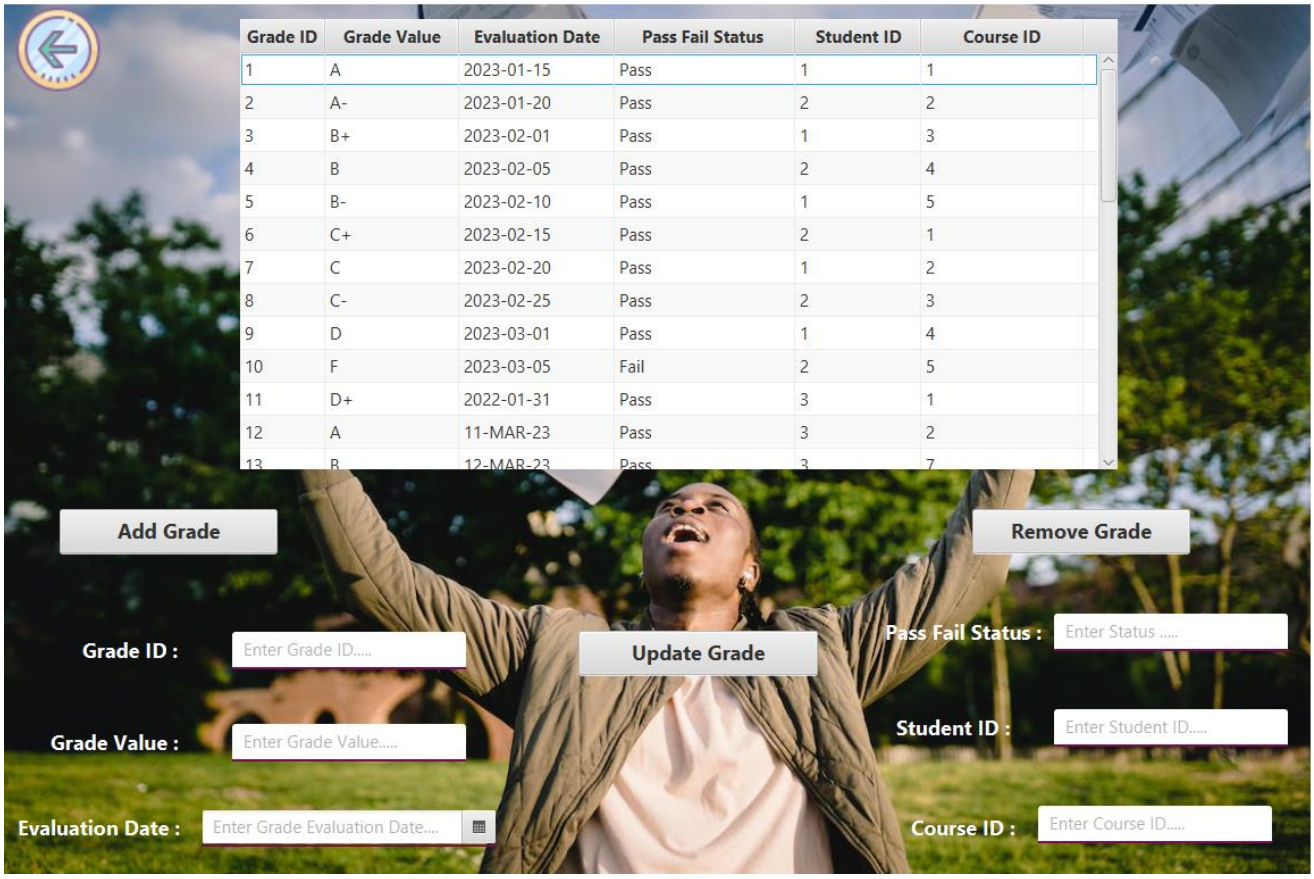
The **DepartmentsController** facilitates CRUD operations for department records in a JavaFX application. It establishes a connection to an Oracle database and initializes a TableView to display department details. Users can add, update, and delete department records, with input field validation ensuring data integrity. Navigation functionality allows users to move between screens, including returning to the home screen. Overall, the controller provides comprehensive management of department data within the application interface.

Course Controller:



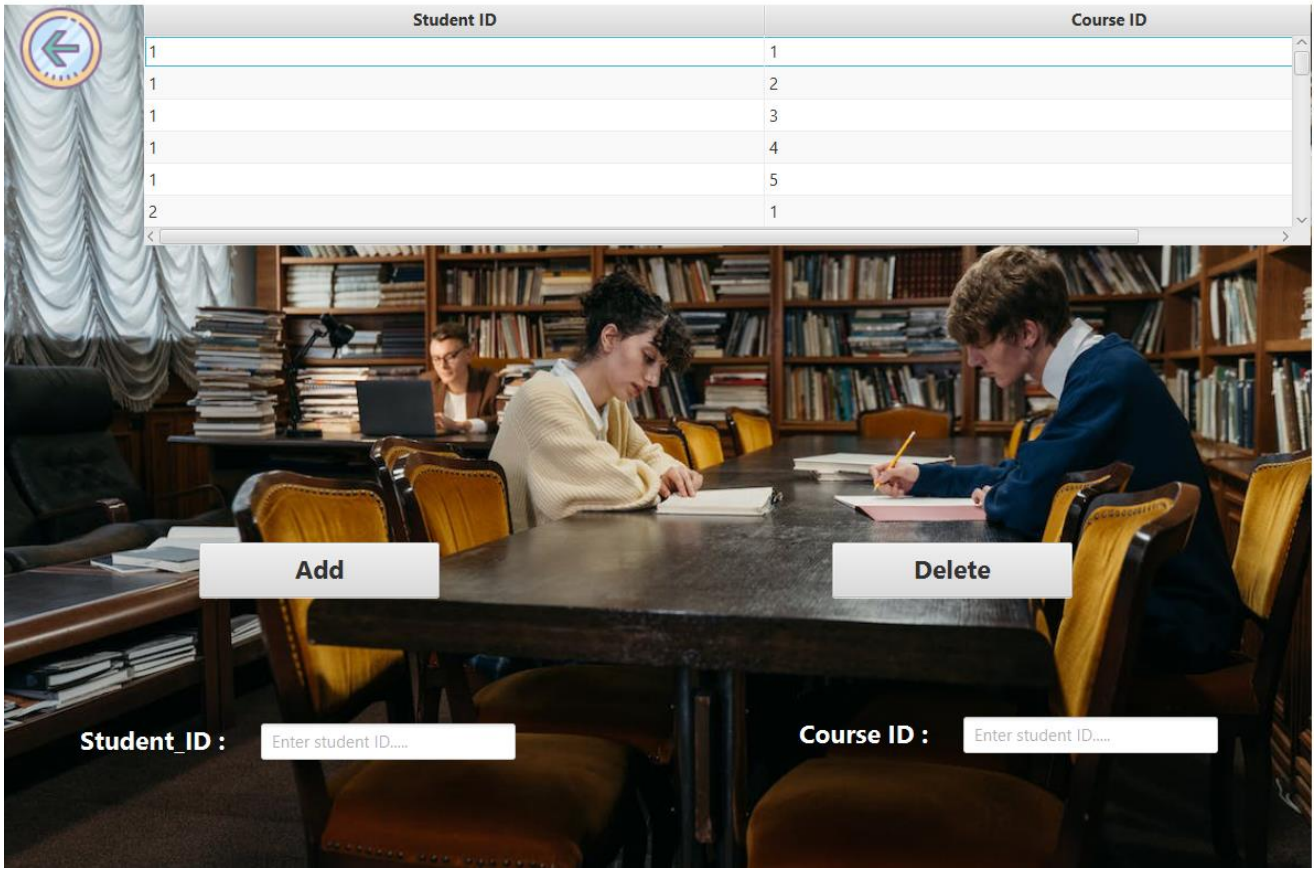
The **CourseController** manages course-related functionalities in a JavaFX application. It handles CRUD operations for courses stored in an Oracle database. The controller establishes a connection to the database, initializes a TableView to display course details, and allows users to add, update, and delete course records. Input field validation ensures data integrity, and navigation functionality permits users to return to the home screen. Overall, the controller provides comprehensive management of course data within the application interface.

Grades Controller:



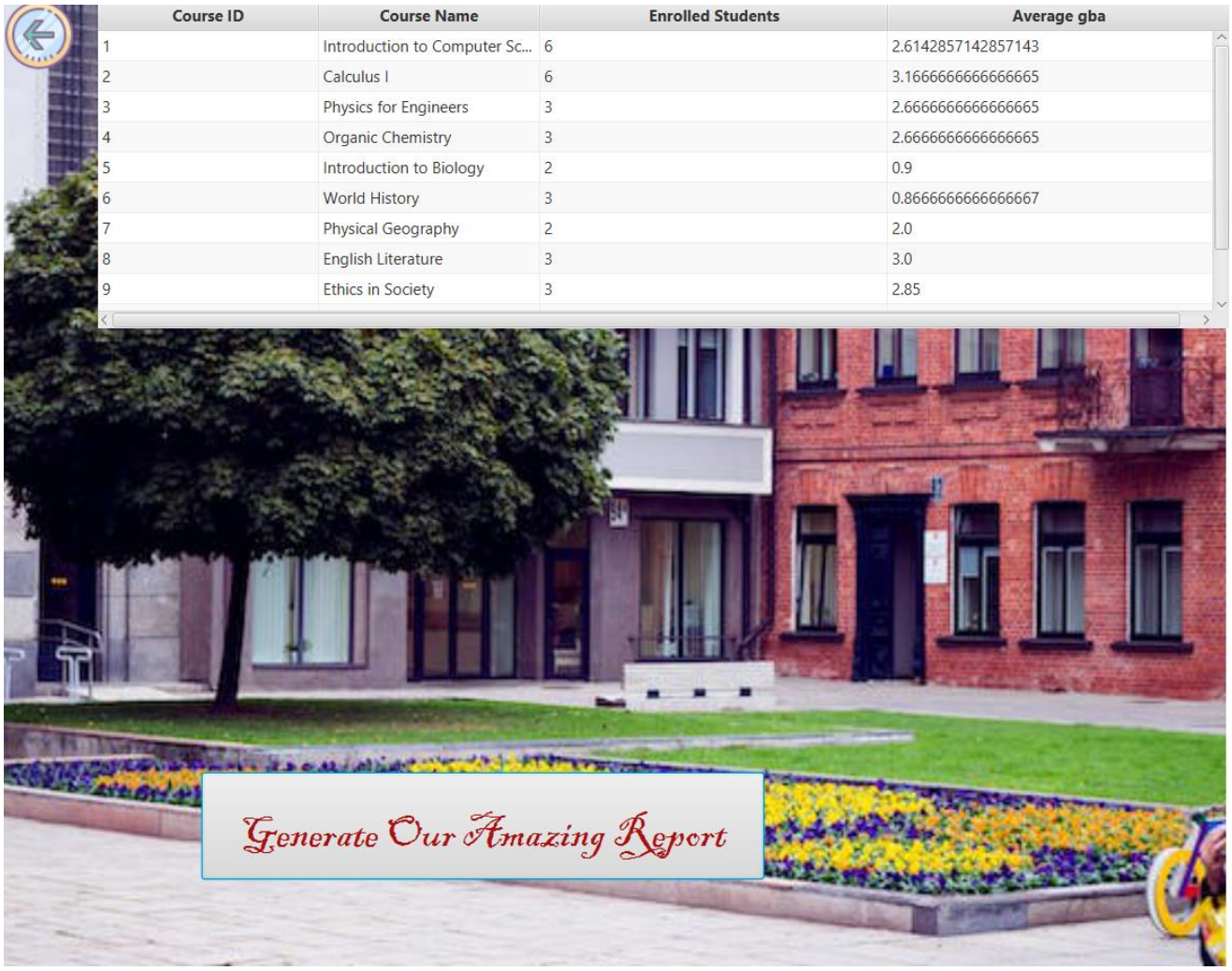
GradesController manages a grade management system. It connects to a database and initializes a table view to display grades. It allows users to add, update, and remove grades via corresponding methods. Input fields are validated for correctness, and alerts are shown for feedback. Additionally, it includes navigation functionality to return to the home screen.

Enroll Controller:



EnrollController facilitates student enrollment in courses. It connects to a database, initializes a TableView to display enrolled courses, and allows users to add or delete enrollments. Input fields are validated for correctness, with alerts displayed for feedback. Additionally, it includes navigation functionality to return to the home screen.

Report Controller:



ReportsController is responsible for generating reports on enrolled students and their average grade point averages (GBA) for each course. It connects to a database, initializes a Table View to display the reports, and allows users to generate reports by clicking a button. The **generatebutton** method executes an SQL query to fetch data on enrolled students and their average GBA for each course, then populates the Table View with the retrieved data. Additionally, it includes navigation functionality to return to the home screen.