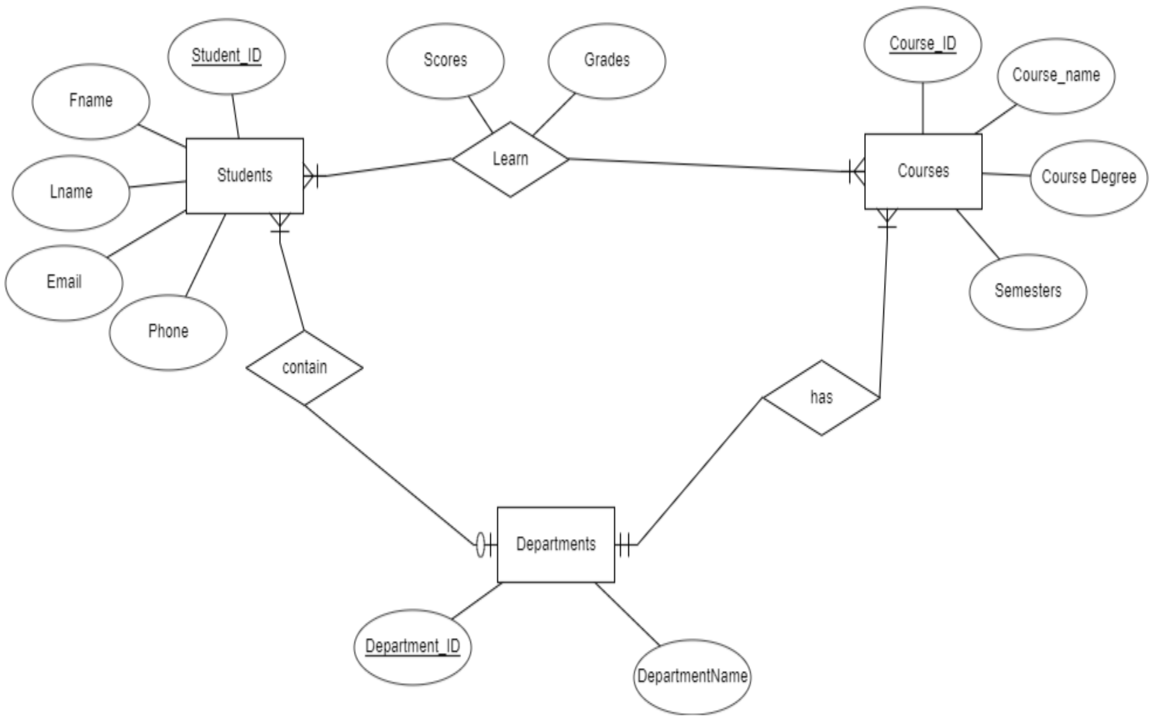
THE COLLEGE

By: Ahmed Mohamed Farid

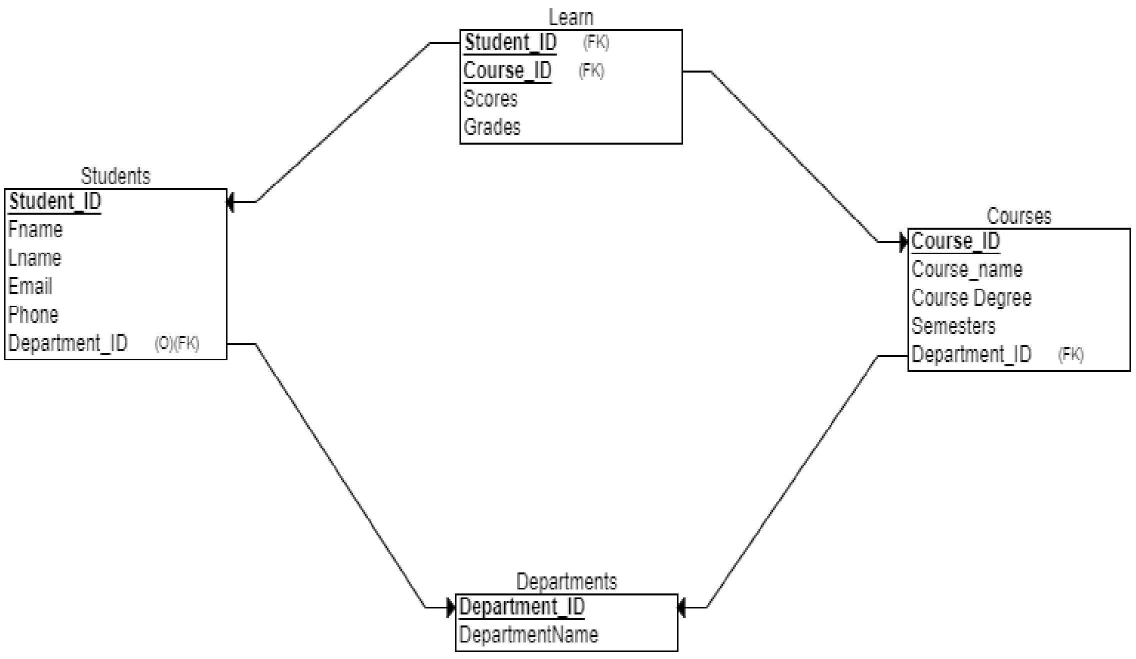
- Database Design :-

  - Each student must have a **unique Student ID**.
  - Students can have additional attributes such as **Fname**, **LName** , **Email**, and **Phone**.
  - Courses are identified by a **unique Course ID**.
  - Each course has associated attributes like **Course name**, **Course Degree** and **Semesters**.
  - A **many-to-many relationship** exists between students and courses, indicating that students can learn multiple courses, and each course can be learned by multiple students.
  - Consider storing **Scores** for each student in each course, which can further connect to **Grades**
  - Each department is identified by a **unique Department ID**.
  - Departments have an associated **Department Name** attribute.
  - A **many-to-one relationship** exists from students to departments, meaning each student is contained in only one department.
  - Similarly, a **many-to-one relationship** exists from courses to departments, indicating that each course is offered by only one department.

ERD:-



- Relational Schema:-



- Procedures:-
  - UpdateStudentInformation:- takes any parameters of the students and update the date for the student in the Students table.

```
CREATE OR REPLACE PROCEDURE UNIVERSITY.UpdateStudentInformation(
   p_Table_Name IN VARCHAR2,
   p_ID IN NUMBER,
   p_New_Fname IN VARCHAR2 DEFAULT NULL,
   p_New_Lname IN VARCHAR2 DEFAULT NULL,
   p_New_Department_ID IN NUMBER DEFAULT NULL,
   p_New_Course_Name IN VARCHAR2 DEFAULT NULL,
   p_New_Score IN NUMBER DEFAULT NULL,
   p_New_Course_ID IN NUMBER DEFAULT NULL
) AS
BEGIN
   CASE p_Table_Name
      WHEN 'Students' THEN
         UPDATE Students
         SET Fname = COALESCE(p_New_Fname, Fname),
            Lname = COALESCE(p_New_Lname, Lname),
            Department_ID = COALESCE(p_New_Department_ID, Department_ID)
         WHERE Student_ID = p_ID;

      WHEN 'Courses' THEN
         UPDATE Courses
         SET Course_Name = COALESCE(p_New_Course_Name, Course_Name),
            Department_ID = COALESCE(p_New_Department_ID, Department_ID)
         WHERE Course_ID = p_ID;

      WHEN 'Grades' THEN
         UPDATE Grades
         SET Scores= COALESCE(p_New_Score, Scores)
         WHERE Student_ID = p_ID AND Course_ID = COALESCE(p_New_Course_ID, Course_ID); --

      ELSE
         DBMS_OUTPUT.PUT_LINE('Invalid table name');
   END CASE;
END;
```

- Triggers:-
  - Insert_student_grade:- after adding a new student into the system the trigger adds this student with his department's courses into Grades table.

```
CREATE OR REPLACE TRIGGER UNIVERSITY.insert_student_grades
AFTER INSERT ON UNIVERSITY.STUDENTS FOR EACH ROW
DECLARE
BEGIN
   -- Insert the student into the GRADES table with department courses
   FOR course_rec IN (SELECT course_id FROM courses WHERE department_id =
:NEW.department_id)
   LOOP
      INSERT INTO grades (student_id, course_id, scores)
      VALUES (:NEW.student_id, course_rec.course_id, NULL);
   END LOOP;
END;
```

- Delete_student_grades:- if a student is deleted from the Students table the trigger will delete his department's courses from the Grades table.

```
CREATE OR REPLACE TRIGGER UNIVERSITY.delete_student_grades
AFTER DELETE ON UNIVERSITY.STUDENTS FOR EACH ROW
BEGIN
   DELETE FROM UNIVERSITY.GRADES
   WHERE STUDENT_ID = :OLD.STUDENT_ID;
END;
```

- UpdateGrade:- when the student's scores stored into the Grades table the trigger calculate his Grade.

```
CREATE OR REPLACE TRIGGER UNIVERSITY.UpdateGrade
BEFORE INSERT OR UPDATE ON UNIVERSITY.GRADES FOR EACH ROW
BEGIN

   :new.Grades := CASE
                 WHEN :new.Scores > 0.75 * 150 THEN 'A'
                 WHEN :new.Scores >= 0.65 * 150 AND :new.Scores < 0.75 * 150 THEN 'B'
                 ELSE 'C'
               END;
END;
```

- **Functions:-**
  - Calculate Student's GPA :-

```
CREATE OR REPLACE FUNCTION UNIVERSITY.CalculateGPA(p_Student_ID IN NUMBER) RETURN
NUMBER IS
   v_TotalGPA NUMBER := 0;
   v_TotalCourses NUMBER := 0;
   v_CoursePercentage NUMBER;
BEGIN
   FOR course_rec IN (
      SELECT G.Scores, C.Course_Degree
      FROM Grades G
      JOIN Courses C ON G.Course_ID = C.Course_ID
      WHERE G.Student_ID = p_Student_ID
   ) LOOP

      v_CoursePercentage := (course_rec.Scores / course_rec.Course_Degree) * 100;

      v_TotalGPA := v_TotalGPA + (v_CoursePercentage / 25);
      v_TotalCourses := v_TotalCourses + 1;
   END LOOP;


   IF v_TotalCourses > 0 THEN
      RETURN ROUND(v_TotalGPA / v_TotalCourses, 2);
   ELSE
      RETURN NULL;
   END IF;
```

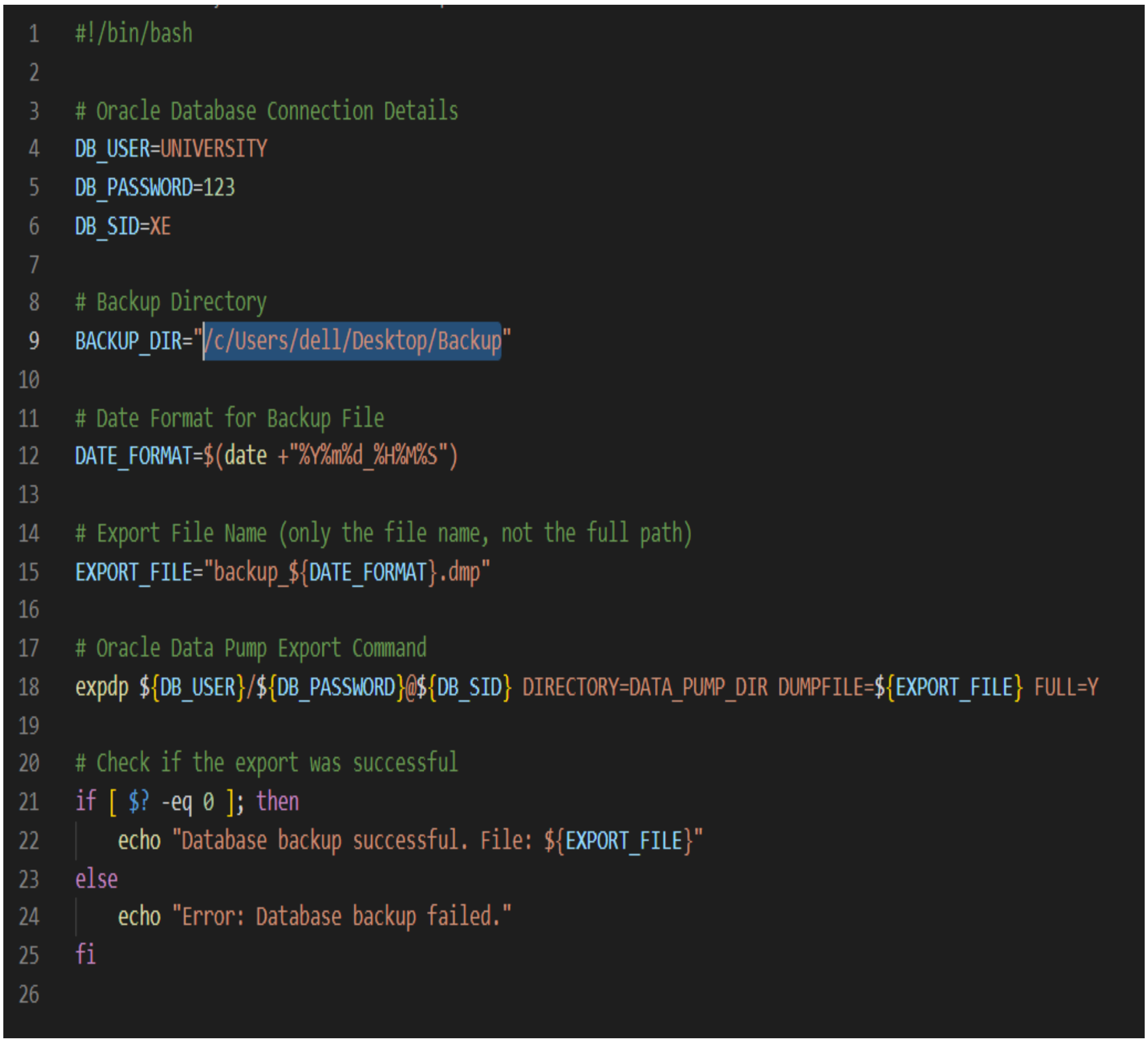
```
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Student with ID ' || p_Student_ID || ' not found.');
        RETURN NULL;
    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('An error occurred.');
        RETURN NULL;
END;
```

- Calculate Avg Course's GPA :-

```
CREATE OR REPLACE FUNCTION UNIVERSITY.CalculateCourseGPA(p_Course_ID IN NUMBER)
RETURN NUMBER IS

    v_TotalGPA NUMBER := 0;
    v_TotalStudents NUMBER := 0;
    v_CoursePercentage NUMBER;
    v_MaxCourseDegree NUMBER := 150;

BEGIN
    FOR course_rec IN (
        SELECT G.Scores, C.Course_Degree
        FROM Grades G
        JOIN Courses C ON G.Course_ID = C.Course_ID
        WHERE G.Course_ID = p_Course_ID
    ) LOOP




        v_CoursePercentage := (course_rec.Scores / v_MaxCourseDegree) *100;


        v_TotalGPA := v_TotalGPA + (v_CoursePercentage /25);
        v_TotalStudents := v_TotalStudents + 1;
    END LOOP;


    IF v_TotalStudents > 0 THEN
        RETURN ROUND(v_TotalGPA / v_TotalStudents, 2);
    ELSE
        RETURN NULL;
    END IF;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Course with ID ' || p_Course_ID || ' not found.');
        RETURN NULL;
    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('An error occurred.');
       RETURN NULL;
END;
```

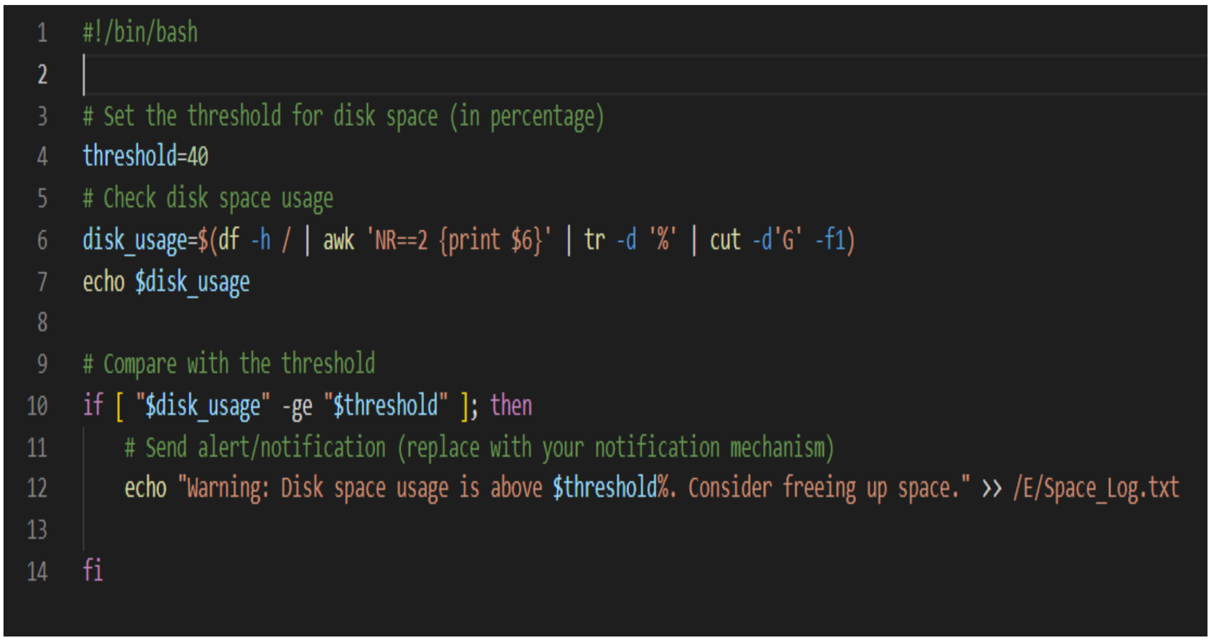
- Sequences :-
  - STUDENTS_SEQ
  - COURSES_SEQ
  - DEPARTMENTS_SEQ
  - GRADES_SEQ
- Automation Scripts :-

- Backup for database



```bash
#!/bin/bash

# Oracle Database Connection Details
DB_USER=UNIVERSITY
DB_PASSWORD=123
DB_SID=XE

# Backup Directory
BACKUP_DIR="/c/Users/dell/Desktop/Backup"

# Date Format for Backup File
DATE_FORMAT=$(date +"%Y%m%d_%H%M%S")

# Export File Name (only the file name, not the full path)
EXPORT_FILE="backup_${DATE_FORMAT}.dmp"

# Oracle Data Pump Export Command
expdp ${DB_USER}/${DB_PASSWORD}@${DB_SID} DIRECTORY=DATA_PUMP_DIR DUMPFILE=${EXPORT_FILE} FULL=Y

# Check if the export was successful
if [ $? -eq 0 ]; then
    echo "Database backup successful. File: ${EXPORT_FILE}"
else
    echo "Error: Database backup failed."
fi

```

- Disk Space Monitoring :-



```bash
#!/bin/bash

# Set the threshold for disk space (in percentage)
threshold=40
# Check disk space usage
disk_usage=$(df -h / | awk 'NR==2 {print $6}' | tr -d '%' | cut -d'G' -f1)
echo $disk_usage

# Compare with the threshold
if [ "$disk_usage" -ge "$threshold" ]; then
    # Send alert/notification (replace with your notification mechanism)
    echo "Warning: Disk space usage is above $threshold%. Consider freeing up space." >> /E/Space_Log.txt

fi
```

- Java Application :-

- In the Java application we have a db folder containing the classes of student, department and course.
- We have a db folder containing the data access layer class which has the methods that are connected with the database.
- Also we have an images folder containing images for each scene.

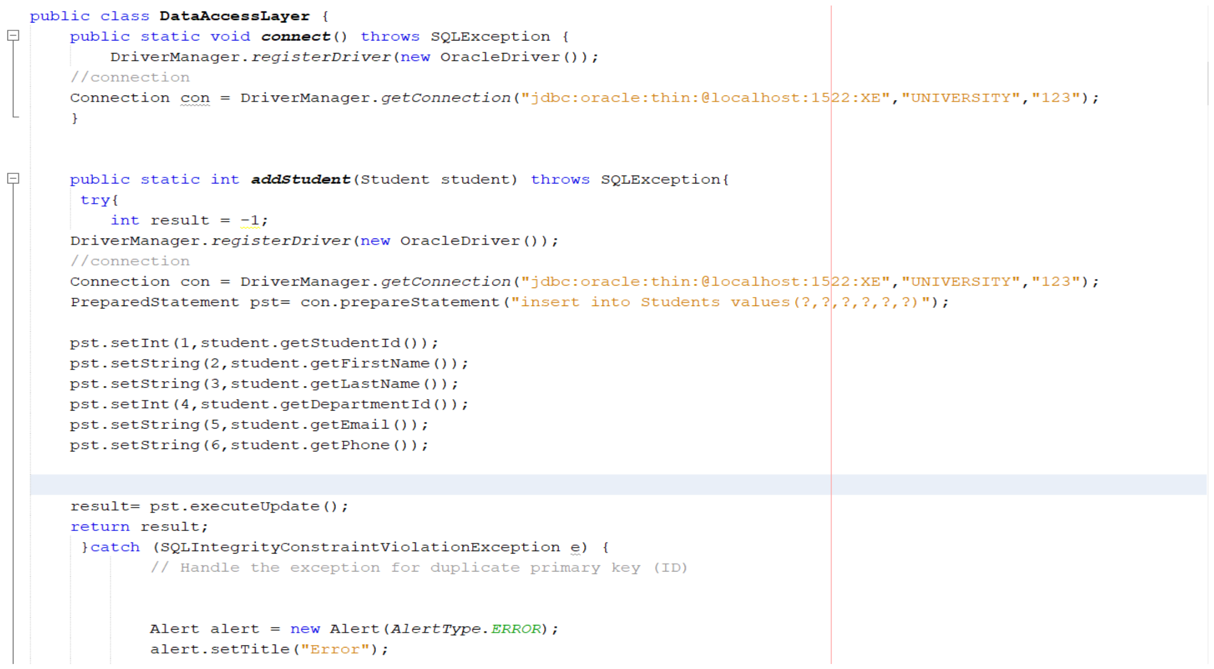
- And finally we have 3 scenes in our application:-

1) Log In scene.

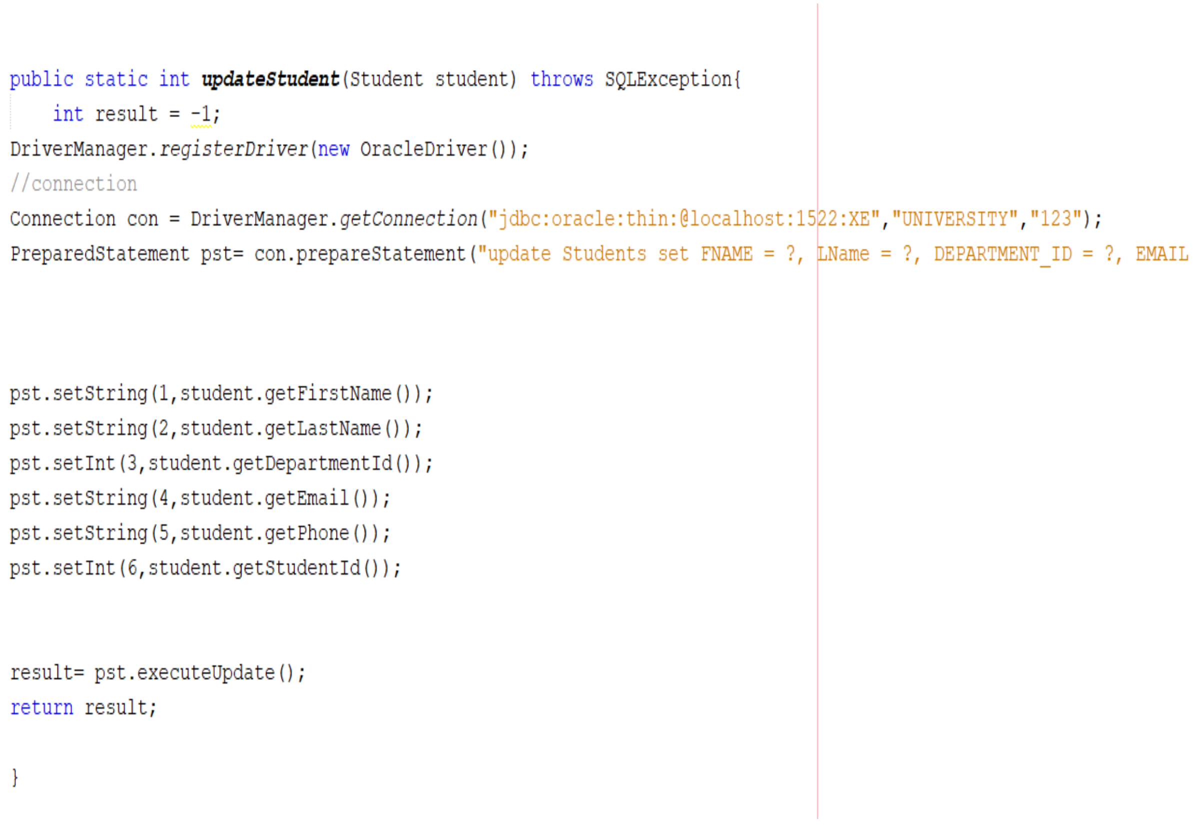
2) Student’s scene.

3) Departments and courses scene.

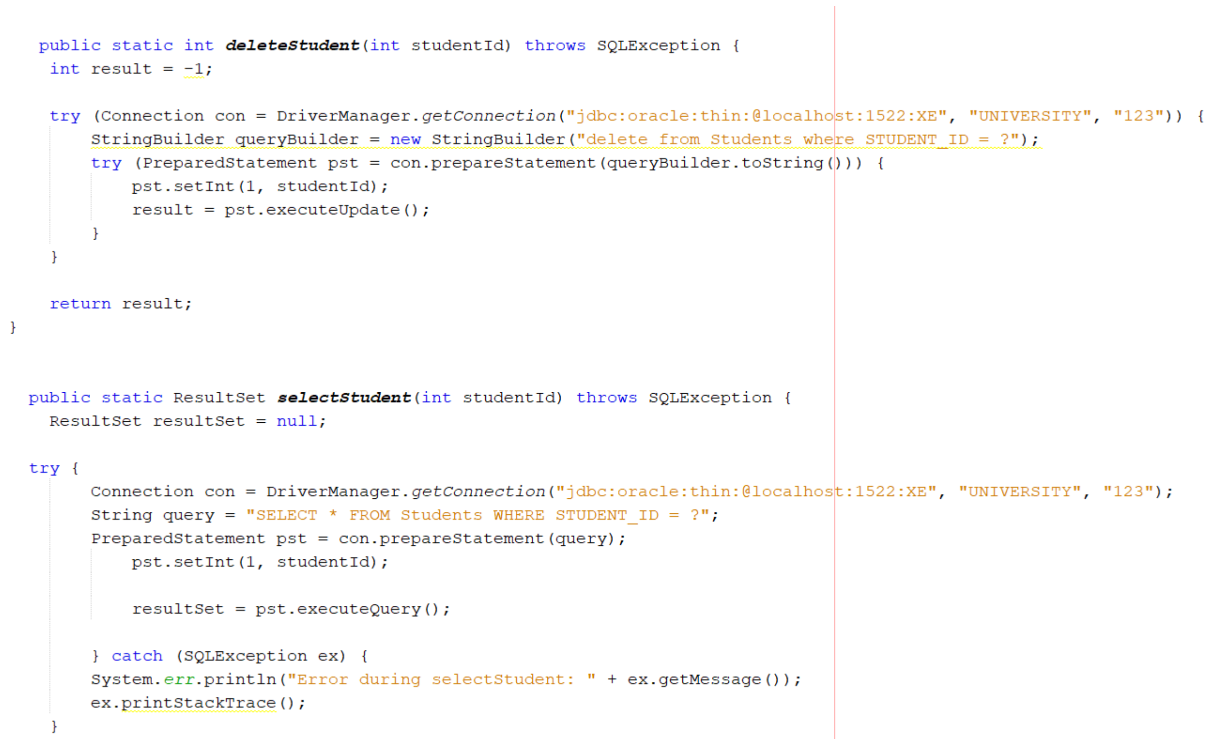
- DataAccessLayer :-



```
public class DataAccessLayer {
    public static void connect() throws SQLException {
        DriverManager.registerDriver(new OracleDriver());
    //connection
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:XE","UNIVERSITY","123");
    }


    public static int addStudent(Student student) throws SQLException{
     try{
        int result = -1;
    DriverManager.registerDriver(new OracleDriver());
    //connection
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:XE","UNIVERSITY","123");
    PreparedStatement pst= con.prepareStatement("insert into Students values(?,?,?,?,?,?)");

    pst.setInt(1,student.getStudentId());
    pst.setString(2,student.getFirstName());
    pst.setString(3,student.getLastName());
    pst.setInt(4,student.getDepartmentId());
    pst.setString(5,student.getEmail());
    pst.setString(6,student.getPhone());


    result= pst.executeUpdate();
    return result;
     }catch (SQLIntegrityConstraintViolationException e) {
            // Handle the exception for duplicate primary key (ID)


            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error");
```

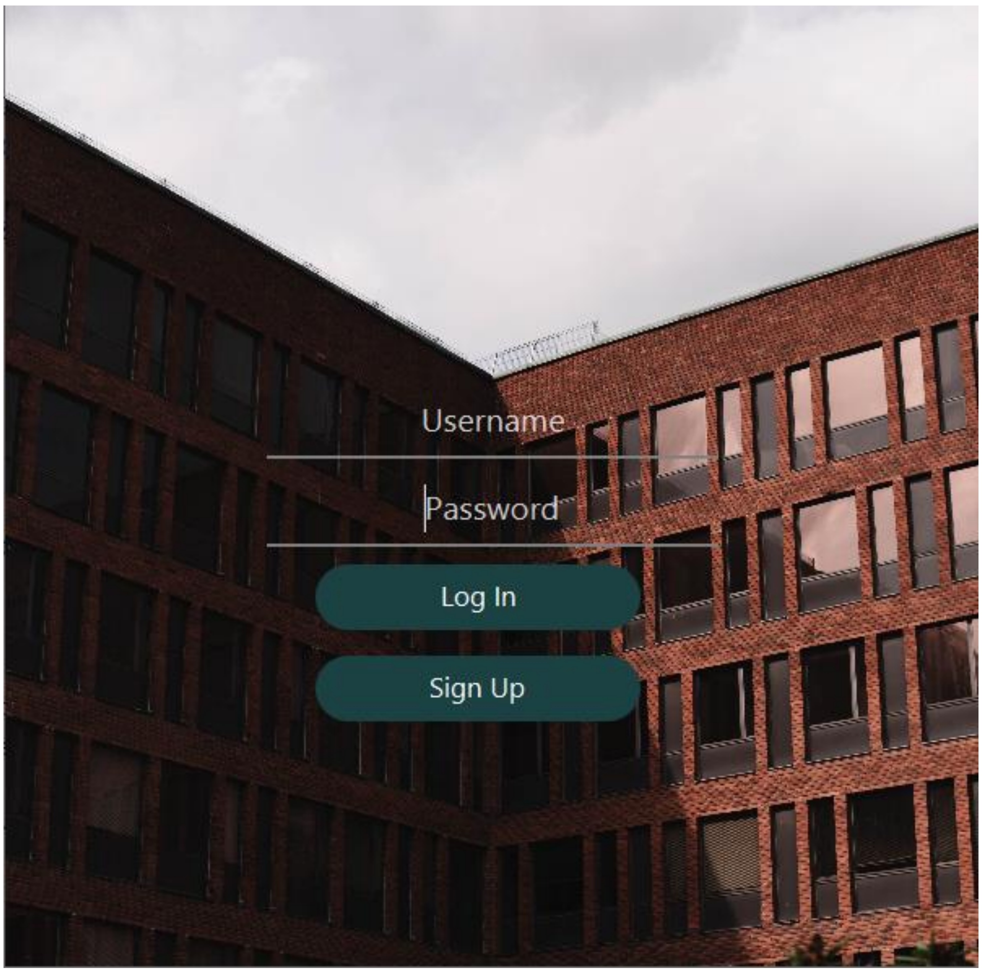


```
public static int updateStudent(Student student) throws SQLException{
    int result = -1;
DriverManager.registerDriver(new OracleDriver());
//connection
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:XE","UNIVERSITY","123");
PreparedStatement pst= con.prepareStatement("update Students set FNAME = ?, LName = ?, DEPARTMENT_ID = ?, EMAIL



pst.setString(1,student.getFirstName());
pst.setString(2,student.getLastName());
pst.setInt(3,student.getDepartmentId());
pst.setString(4,student.getEmail());
pst.setString(5,student.getPhone());
pst.setInt(6,student.getStudentId());


result= pst.executeUpdate();
return result;

}
```

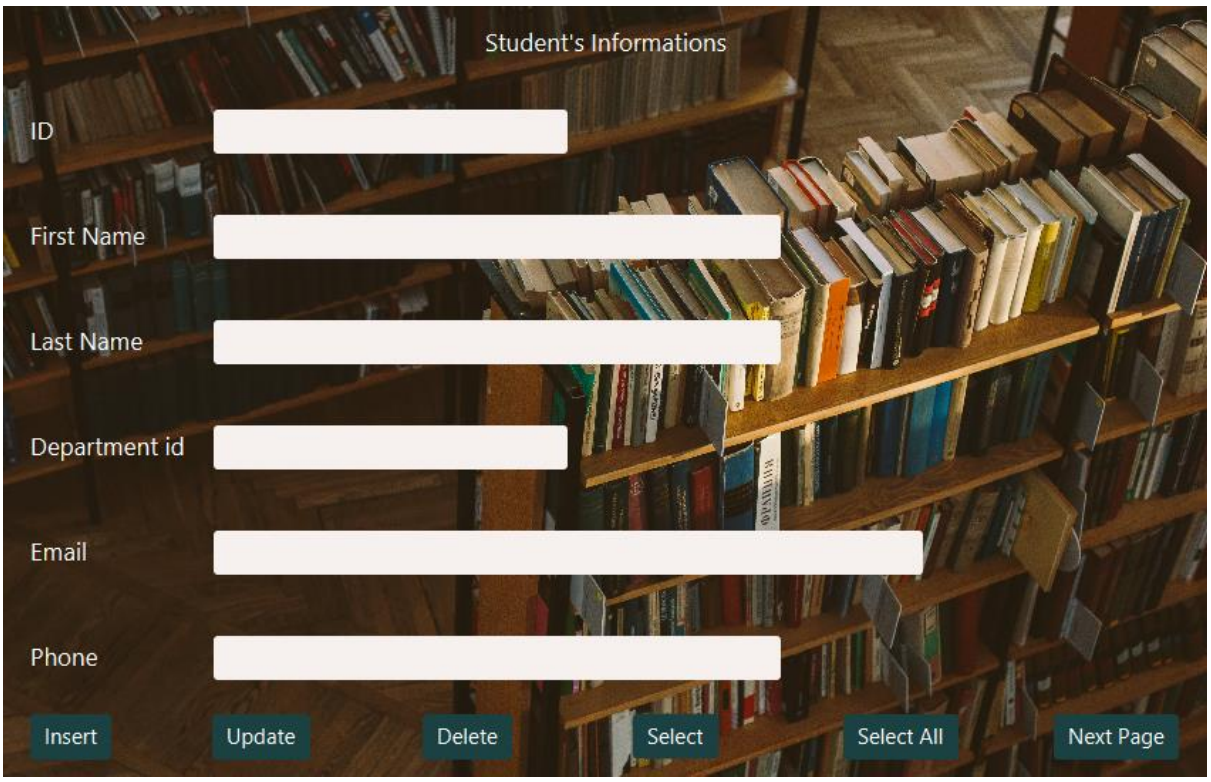


```java
    public static int deleteStudent(int studentId) throws SQLException {
     int result = -1;

     try (Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:XE", "UNIVERSITY", "123")) {
         StringBuilder queryBuilder = new StringBuilder("delete from Students where STUDENT_ID = ?");
         try (PreparedStatement pst = con.prepareStatement(queryBuilder.toString())) {
             pst.setInt(1, studentId);
             result = pst.executeUpdate();
         }
     }

     return result;
}


   public static ResultSet selectStudent(int studentId) throws SQLException {
     ResultSet resultSet = null;

   try {
         Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1522:XE", "UNIVERSITY", "123");
         String query = "SELECT * FROM Students WHERE STUDENT_ID = ?";
         PreparedStatement pst = con.prepareStatement(query);
             pst.setInt(1, studentId);

             resultSet = pst.executeQuery();

         } catch (SQLException ex) {
         System.err.println("Error during selectStudent: " + ex.getMessage());
         ex.printStackTrace();
     }
```

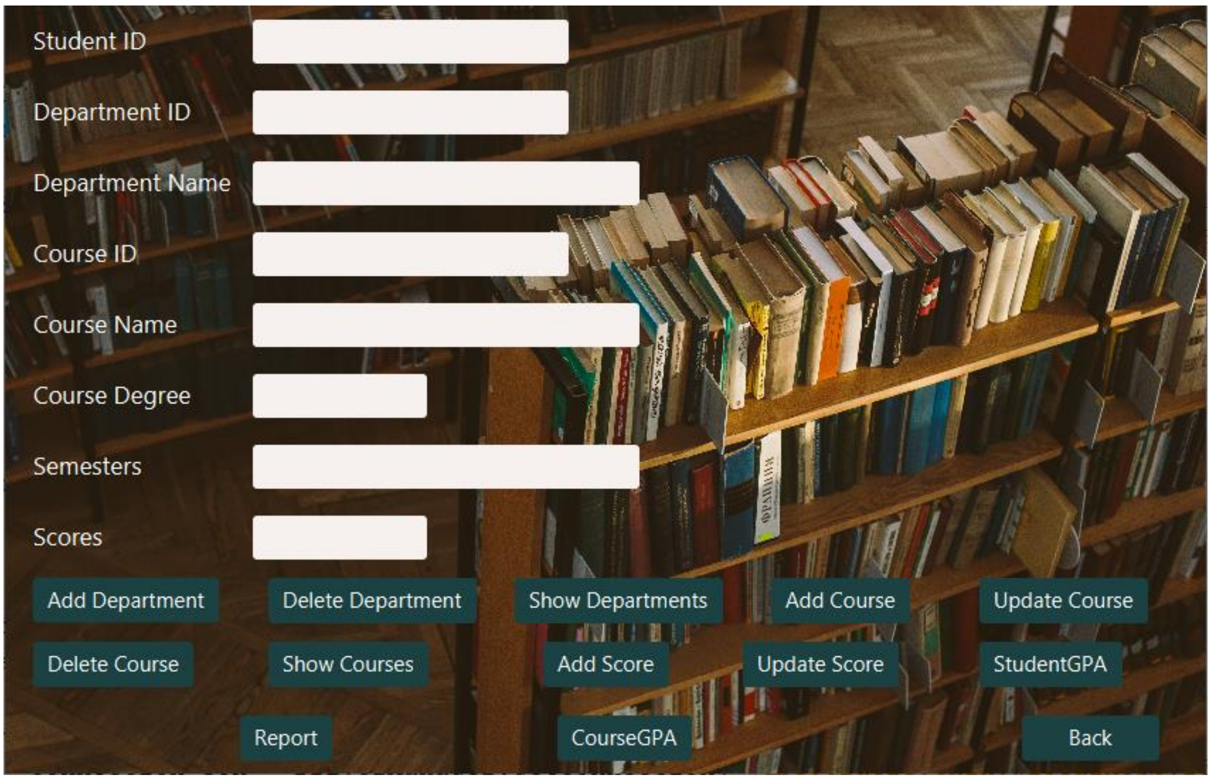
- Log In Scene :-



- Student’s Information Scene :-



- Departments and Courses Scene :-


Student ID
Department ID
Department Name
Course ID
Course Name
Course Degree
Semesters
Scores
Add Department
Delete Department
Show Departments
Add Course
Update Course
Delete Course
Show Courses
Add Score
Update Score
StudentGPA
Report
CourseGPA
Back