****

**University Management System Documentation**

**Sherif Ashraf**

* **Database Design**
* **Tables:**

1. Students table:

* Student\_id: Primary key and a unique identifier for each student.
* First\_name: first name of the student.
* Last\_name: last name of the student.
* Email: email address of the student.
* Phone\_number: phone number of the student.
* DOB: date of birth of the student.
* Street: the street in which the student lives.
* City: the city in which the student resides.
* Semester: the semester of the student.
* Department\_id: Foreign key referencing the departments table.

1. Departments table

* Department\_id: Primary key and a unique identifier for each department.
* Department\_name: the name of the department.

1. Courses table:

* Course\_id: Primary key and a unique identifier for each course.
* Course\_name: name of the course.
* Description: a description about the course.
* Credit\_hours: Credit hours per course.
* Semester: the semester of the course.
* Department\_id: Foreign key referencing the departments table.

1. Enrolled table:

* Student\_id: Foreign key referencing the students’ table.
* Course\_id: Foreign key referencing the courses’ table.
* Grade: the grade of the student in the course.
* **Relationships:**

1. Students and departments:

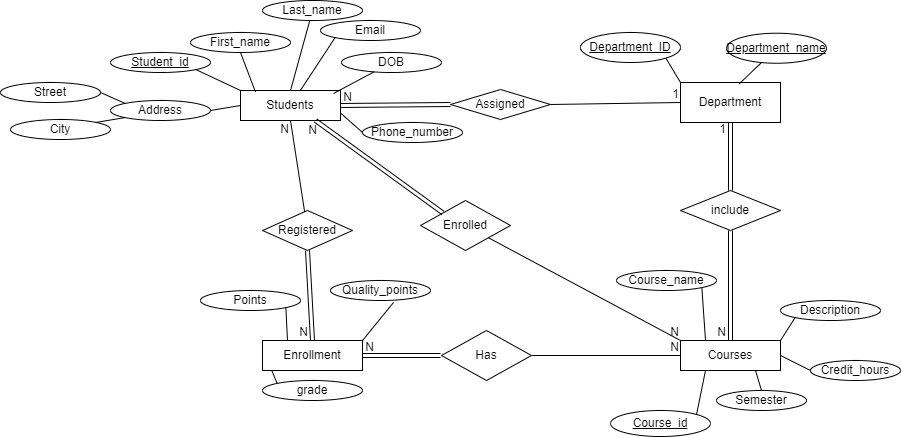
* Cardinality relationship: many-to-one (M:1)
  + Student can be assigned to only one department, but department can have more than one student assigned to it.
* Participation: each student must be assigned to at least one department, but department may have not any students assigned to it.

1. Department and courses:

* Cardinality relationship: one-to-many (1:M)
  + Course are assigned to only one department, but department can have more than one course assigned to it.
* Participation: each course must be assigned to at least one department, and each department must have at least one course assigned to it.

1. Students and courses

* Cardinality relationship: many-to-Many (M:N)
  + Student can be enrolled into more than one course, and each course can be enrolled by many students.
* Participation: each student must be enrolled into at least one course, but course may have not any students enrolled into it.
* **ERD Design:**



* **Mapping and Normalization:**

1. Students (student\_id(PK), first\_name, last\_name, email, phone\_number, DOB, Street, City, CGPA, semester, department\_id(FK))
2. Departments (Department\_id(PK), department\_name)
3. Courses (Course\_id(PK), course\_name, description, credit\_hours, semester, department\_id(FK))
4. Enrolled (student\_id(FK), course\_id(FK), grade, points, quality\_points)

* **SQL Implementation:**
* **Students Table:**
* 'STUDENT\_ID': (Primary Key).
* 'FIRST\_NAME': not null.
* 'LAST\_NAME': not null.
* 'EMAIL': not null and unique.
* 'PHONE\_NUMBER': nullable.
* 'DOB': not null.
* 'STREET': not null.
* 'CITY': not null.
* 'GPA': nullable.
* 'DEPARTMENT\_ID': Foreign key for Departments table, not null.
* 'SEMESTER': nullable.
* **Courses Table:**
* 'COURSE\_ID': (Primary key).
* 'COURSE\_NAME': not null.
* 'DESCRIPTION': nullable.
* 'CREDIT\_HOURS': not null.
* 'DEPARTMENT\_ID': Foreign key for Departments table, not null.
* 'SEMESTER': not null.
* **Departments Table:**
* 'DEPARTMENT\_ID': (Primary key).
* 'DEPARTMENT\_NAME': not null.
* **Enrollments Table:**
* 'STUDENT\_ID': Foreign key for Students table, not null.
* 'COURSE\_ID' Foreign key for Courses table, not null.
* 'GRADE': nullable.
* 'POINTS': nullable.
* 'QUALITY\_POINTS': nullable.
* **PL/SQL Implementation:**

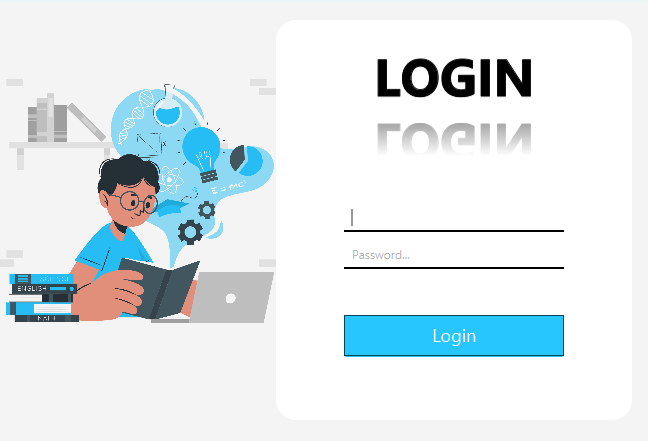
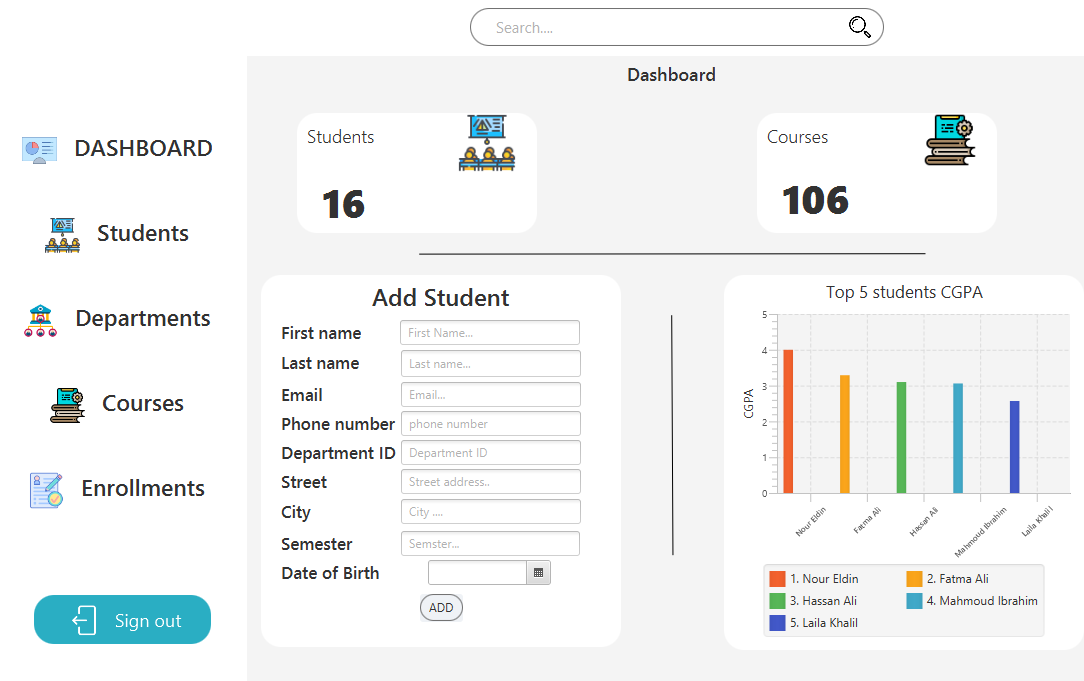
1. **Procedures:**

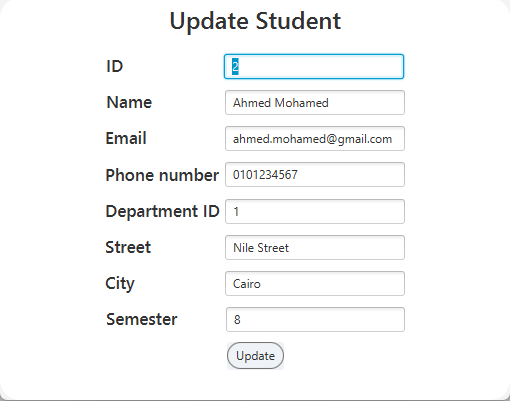
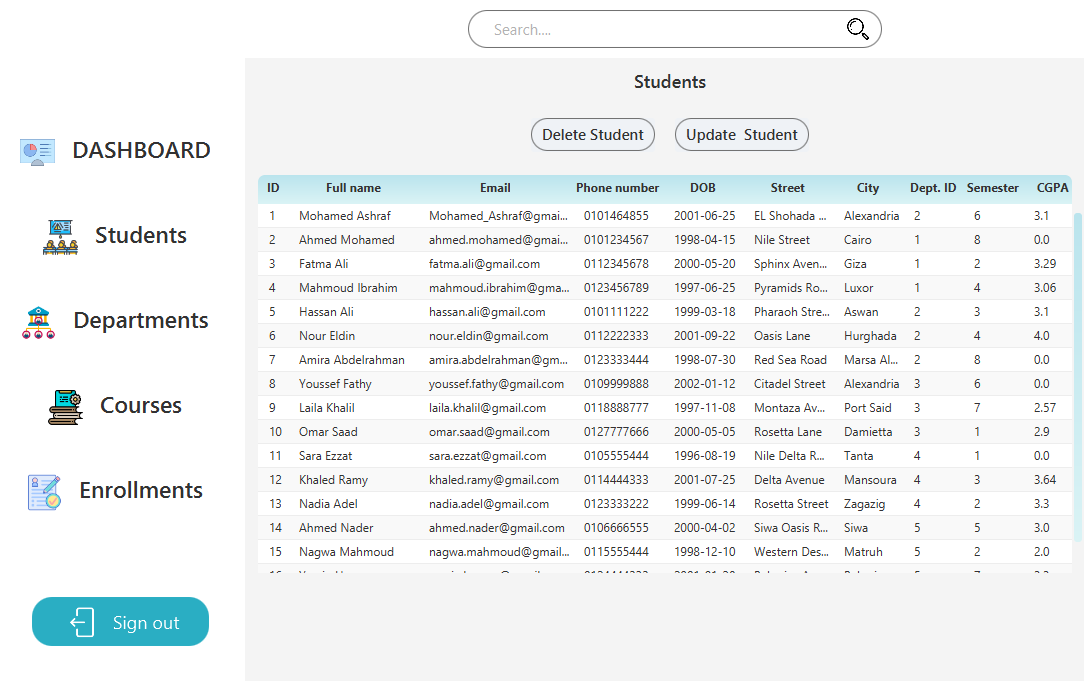
* **Stored Procedure: DROP\_SEQ\_PROCEDURE**
* Procedure that deletes any sequences related to university schema in order to create a custom sequence later on.
* **Stored Procedure: CREATE\_SEQ**
* Procedure that takes primary key column name, and table name as parameters, and it create new sequence based on the maximum number in the primary key column plus one.
* **Stored Procedure: CALC\_CGPA**- Procedure that takes Student\_ID as a parameter and make calculations, based on these calculations, it updates the cumulative GPA for the student with the Student\_ID it provided with.

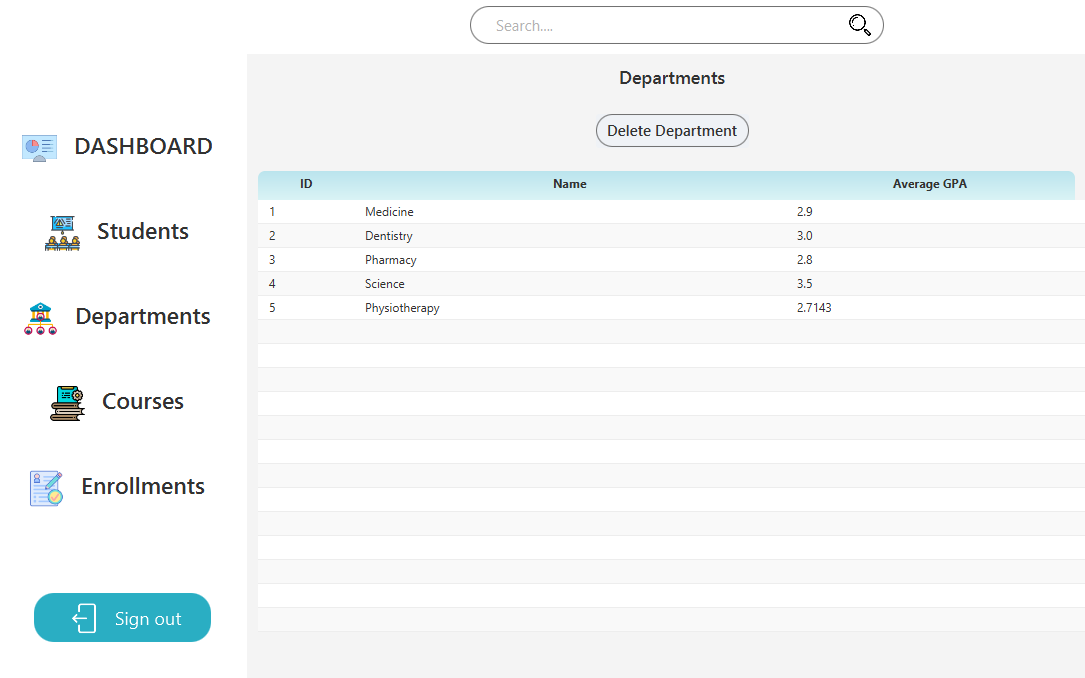
1. **Functions:**

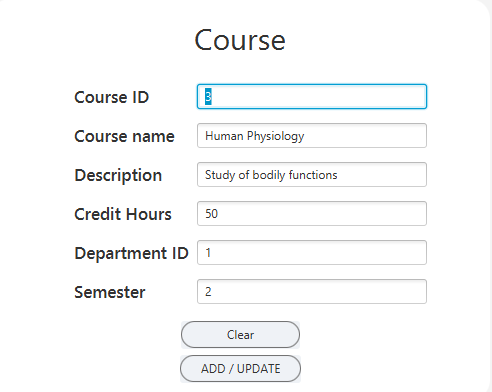
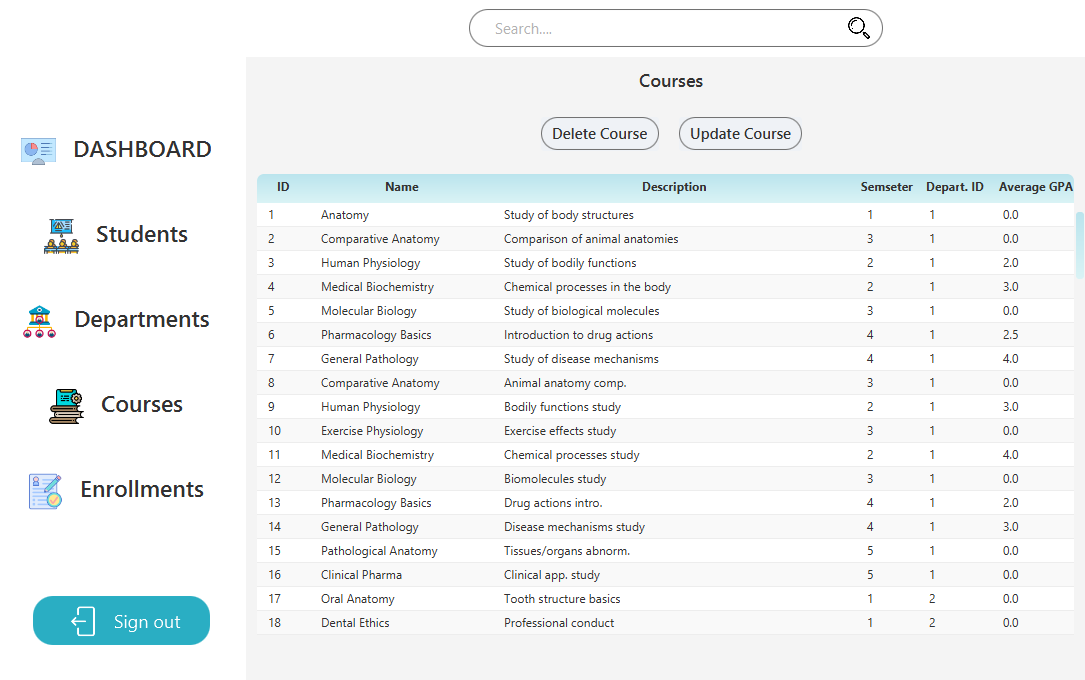
* **Function: CHECK\_PK\_NO**
* This function takes the schema name, and table name to check for the number and datatype of the primary key columns for the provided table name, if the table has no primary keys, composite primary key, or non-numeric primary key it will be discarded, otherwise it will return the number of primary key columns for the table and the name of the primary key column to be used for creating the sequence and trigger.
* **Function: calc\_avg\_DEPT\_gpa**
* This function makes calculations based on department\_ID it takes as parameter and return the average GPA for all courses registered in this department.
* **Function: calc\_avg\_gpa**
* This function makes calculations based on Course\_ID it takes as parameter and return the average GPA for all enrollments related to this course.

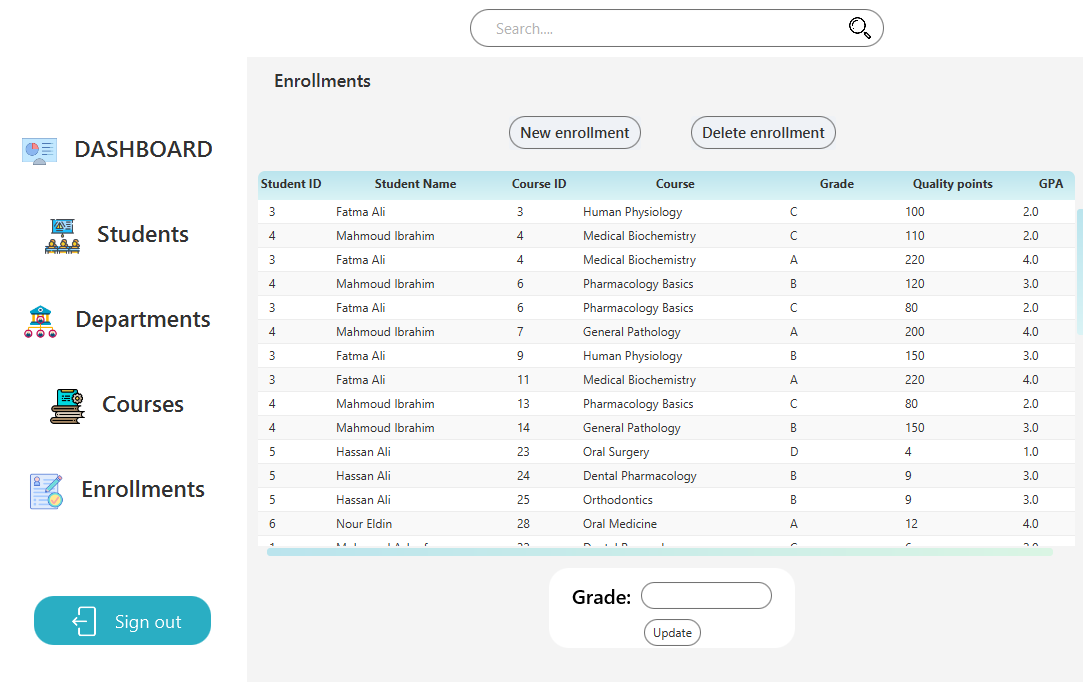
1. **Triggers:**

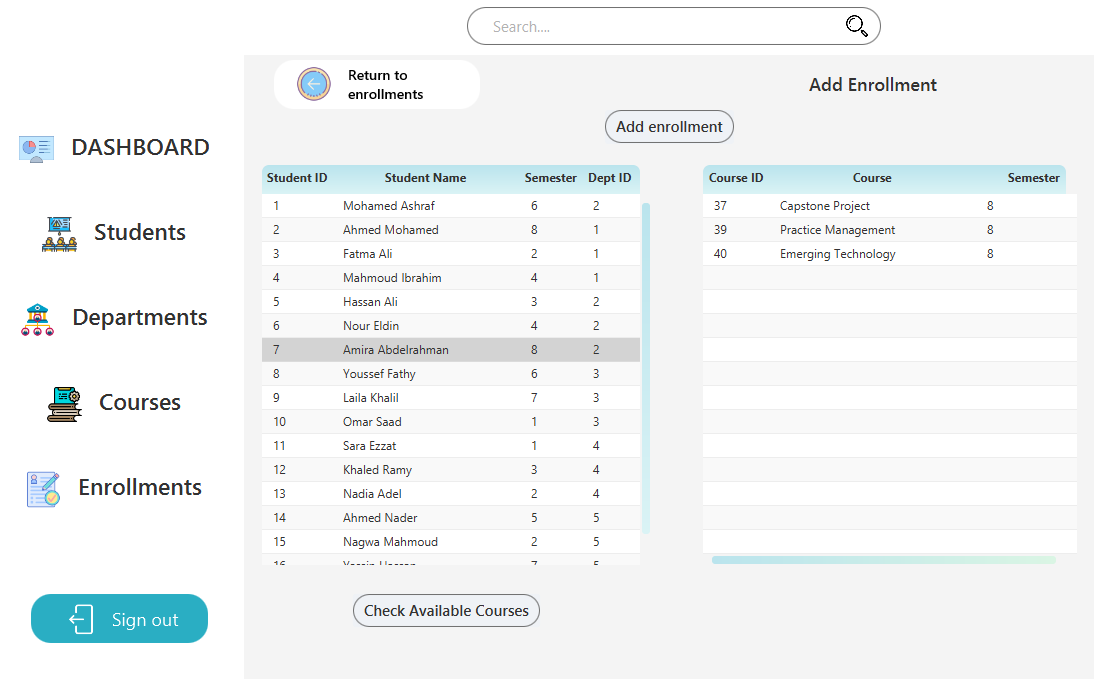
* **Triggers: COURSES\_TRG, DEPARTMENTS\_TRG and STUDENTS\_TRG**
* Triggers made with a PL/SQL script using DROP\_SEQ\_PROCEDURE, CREATE\_SEQ PROCEDURE, CHECK\_PK\_NO FUNCTION and dynamic SQL to create trigger for each table has a single numeric primary key column.
* The action of those triggers is that when inserting new record, they place the primary key column with the next value in the sequence automatically which makes insertion a lot easier and faster.
* **Trigger: POINTS\_TRIGGER**
* This trigger acts on Enrollments table when updating the grade of any record to (A, B, C or D), the trigger act accordingly and update the points of the updated record to the defined points for each grade.
* **Trigger: CGPA\_TRIGGER**
* Compound trigger acts on enrollments table, it is collecting the student\_ids for the updated enrollments records in a collection, then makes calculations and updates students table using CALC\_CGPA Procedure to update cumulative GPA for the student his/her enrollment has been updated with a grade.
* **Automation Scripts (Bash Script)**
* **Purpose**
* A script for automated backup for the database at regular intervals.
* Checking for remaining disk space.
* Gives warning in a specified log file for any errors.
* **Functionality**
* Backup for Oracle DBMS on UNIVERSITY user every day at 17:00 after regular work hours with backup date and time in backup directory.
* Checking the remaining disk space available on the backup partition, and gives a warning if the usage exceeds 75% of the total partition space.
* Logging the success or failure with warning for any backup or disk check procedure in file called University\_log.log in logs directory.
* **Java Application Development:**
* Application Developed using Java using OOP and MCV principles, Connected with University database using Oracle JDC with multiple functionalities and reports on Students, courses, departments and enrollments records.
* **Features:**
* **Students Management**:
* Viewing: viewing all the students registered in the university with their name, email, phone number, date of birth, address, semester and cumulative GPA.
* Inserting: Registering a new student to the university.
* Updating: updating the email, phone number or address of the record of a registered student.
* Deleting students: Deleting specific student from database with warning of deleting all of his enrollments.
* **Courses Management**:
* Viewing: viewing all the courses available in the university with their department and semester.
* Inserting: inserting a new course with its name, description, department id and semester into the system.
* Updating: Update the course information name, description.
* Deleting courses: Deleting a course with a warning before deletion that it will delete the course with all records related to it in the enrollments.
* Viewing reports: allows viewing the average GPA for each course.
* **Departments Management**:
* Viewing: viewing all the departments in the university with their name and ID.
* Inserting: register a new department and
* Viewing reports related to each department.
* **Enrollments**:
* Viewing: viewing all enrollments with the student id, course id, grade, points, credit hours and GPA.
* Inserting: registering a new enrollment for the student with available courses in the same department and semester.
* Updating: updating an enrollment by adding grade to it according to the student marks.
* Deleting: deleting enrollments for each student and course.
* **Reports:**
* Count of the students and courses.
* Average GPA for each department and course.
* Cumulative GPA for each student.
* **Structure:**
* **DTO (Data transfer objects) Package**: includes DTO class for each table using OOP concepts.
* **DAO (Data access objects) Package**: includes DAO class for each table that handles connection and query execution with Oracle JDC on University user.
* **Admin dashboard package**: includes the FXML files for each scene with specified controller for each FXML file, each scene inside a sub-package (FXML and Controller). Applying MCV concept.
* **Resources folder**: for the CSS file and resources used in the GUI designing.
* **Project:**
* **Login Screen:**
* To login and connect to the database.
* **Dashboard view:**
* View reports regarding the students and courses.
* Registering new student.
* Buttons to switch between different views or sign out.
* Students view:
* Showing all students in a table with all of their information.
* Allows deletion or updating students.

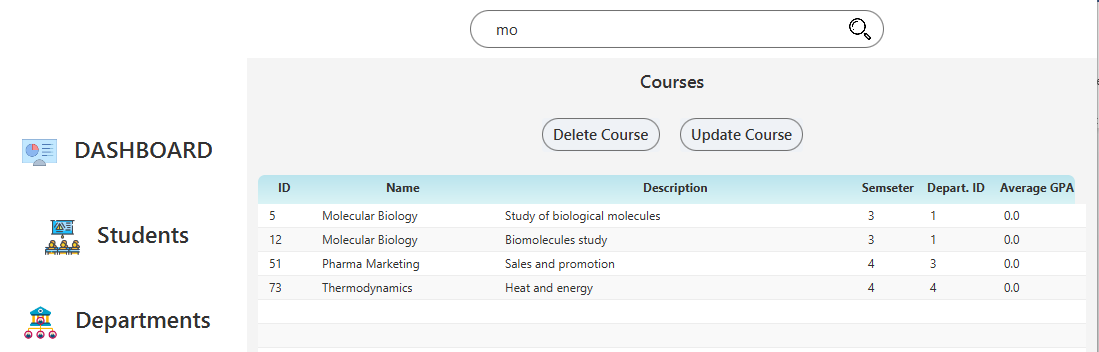


* **Departments’ view:**
* show the department id, name and average GPA for each department.
* **Courses’ view:**
* View course id, name, description, department id the course related to, semester that the student should register to this course in, and average GPA for this course.
* It allows deletion of the course with a warning for deletion of all of its enrollments.
* It allows updating course information using update button.
* Updating screen can be used also for adding a new course by clearing all the fields and enter the new course information.



* **Enrollment view:**
* View all enrollment with student id, course id, student name, course name, grade, quality points and GPA for each enrollment.
* Allow updating grade for each enrollment and add points and GPA regarding to the grade.
* Allow adding new enrollment by opening new scene and view the available courses for each student selected.



* **Search:**
* Search box allows searching while in the students, departments, courses or enrollments scenes.

