Information
Technology
Institute

2024

UNIVERSITY CASE STUDY

Prepared by :
Ali Magdy

Contents

I. Database Design

Overview:

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

Requirements:

Department

- Each department is uniquely identified by Dep_ID.
- It has a name, represented by Dep_Name.

Program

- Departments offer one or more programs.
- Programs are characterized by Program_ID and Program_Name.
- Each program includes a list of courses.
- A course is exclusively offered by one program.

Student

- Students are enrolled in a specific program.
- Student details include first name, last name, gender, National ID, email, phone, and address (city, street).

Course

- Courses have a unique identifier and a name.
- Each course is associated with only one program.

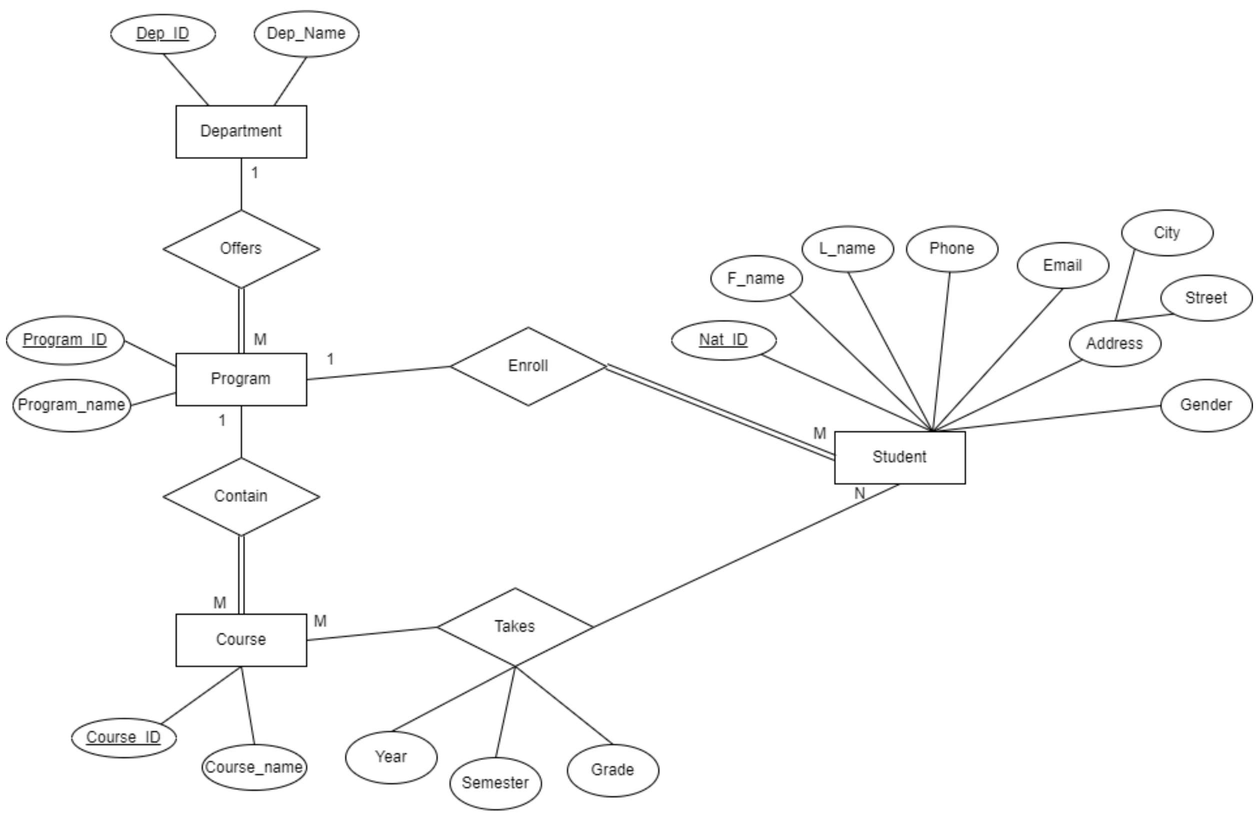
Grade

- Grade percentages (e.g., 60, 70) are recorded for each student attempting a course.
- The success percentage for each course is set at 60%.

Retaking Courses

- Students can retake a course in a subsequent semester (first or second) within a specified academic year (e.g., 2023/2024, 2022/2023).

The ER-Diagram:



Mapping and Normalization:

The Design already in the (3NF)

Department(<u>Dep_ID</u>, Dep_Name)

Student(<u>Nat_ID</u>, F_Name, L_Name,Gender, Phone, Email, City, Street,DOB, Prog_ID **FK**)

Program(<u>Prog_ID</u>, Prog_Name,Dep_ID **FK**)

Course(<u>Course_ID</u>, Course_Name,Prog_ID **FK**)

Student_Course(<u>Nat_ID **FK**,Course_ID **FK** ,Year , Semester</u> , Grade)

II. SQL Implementation

Creation of Tables and constraints:

```
CREATE TABLE UNIVERSITY.Department (
    Dep_ID NUMBER PRIMARY KEY,
    Dep_Name VARCHAR2(255) NOT NULL
);

-- Create Program table in UNIVERSITY schema
CREATE TABLE UNIVERSITY.Program (
    Prog_ID NUMBER PRIMARY KEY,
    Prog_Name VARCHAR2(255) NOT NULL,
    Dep_ID NUMBER,
    FOREIGN KEY (Dep_ID) REFERENCES UNIVERSITY.Department(Dep_ID) ON DELETE CASCADE
);

CREATE TABLE UNIVERSITY.Student (
    Nat_ID NUMBER PRIMARY KEY,
    F_Name VARCHAR2(255) NOT NULL,
    L_Name VARCHAR2(255) NOT NULL,
    Gender VARCHAR2(10) NOT NULL,
    Phone VARCHAR2(20),
    Email VARCHAR2(255),
    City VARCHAR2(255),
    Street VARCHAR2(255),
    Prog_ID NUMBER,
    FOREIGN KEY (Prog_ID) REFERENCES UNIVERSITY.Program(Prog_ID) ON DELETE SET NULL
);

CREATE TABLE UNIVERSITY.Course (
    Course_ID NUMBER PRIMARY KEY,
    Course_Name VARCHAR2(255) NOT NULL,
    Prog_ID NUMBER,
    FOREIGN KEY (Prog_ID) REFERENCES UNIVERSITY.Program(Prog_ID) ON DELETE CASCADE
);

-- Create Student_Course table in UNIVERSITY schema
CREATE TABLE UNIVERSITY.Student_Course (
    Nat_ID NUMBER,
    Course_ID NUMBER,
    Year VARCHAR2(20),
    Semester VARCHAR2(20),
    Grade NUMBER,
    PRIMARY KEY (Nat_ID, Course_ID, Year, Semester),
    FOREIGN KEY (Nat_ID) REFERENCES UNIVERSITY.Student(Nat_ID) ON DELETE CASCADE,
    FOREIGN KEY (Course_ID) REFERENCES UNIVERSITY.Course(Course_ID) ON DELETE CASCADE);
```

Populating Sample Data

Here is a snippet from our sample data:

```
--Departments
INSERT INTO UNIVERSITY.Department (Dep_ID, Dep_Name) VALUES (1, 'Computer Science');

-- Programs for Computer Science (Dep_ID = 1)
INSERT INTO UNIVERSITY.Program (Prog_ID, Prog_Name, Dep_ID) VALUES (100, 'Software
Engineering', 1);


--Courses for Software Engineering Program (Prog_ID =100)
INSERT INTO UNIVERSITY.Course (Course_ID, Course_Name, Prog_ID) VALUES (1000, 'Introduction to
Programming', 100);
INSERT INTO UNIVERSITY.Course (Course_ID, Course_Name, Prog_ID) VALUES (1001, 'Data Structures
and Algorithms', 100);
INSERT INTO UNIVERSITY.Course (Course_ID, Course_Name, Prog_ID) VALUES (1002, 'Object-Oriented
Programming', 100);
INSERT INTO UNIVERSITY.Course (Course_ID, Course_Name, Prog_ID) VALUES (1003, 'Database
Systems', 100);
INSERT INTO UNIVERSITY.Course (Course_ID, Course_Name, Prog_ID) VALUES (1004, 'Software
Engineering Principles', 100);

--Student Information
INSERT INTO UNIVERSITY.Student (Nat_ID, F_Name, L_Name, Gender, Phone, Email, City, Street,
Prog_ID) VALUES
(29501020100012, 'Ahmed', 'Mohamed', 'Male', '01001234567', 'ahmed.mohamed@example.com', 'Cairo',
'12 El-Tahrir Street', 100);
```

III. PLSQL Implementation

PlSQL Procedure for Updating Student Info:

```
CREATE OR REPLACE PROCEDURE UNIVERSITY.update_student_info(
    p_nat_id NUMBER,
    p_new_f_name VARCHAR2,
    p_new_l_name VARCHAR2,
    p_new_gender VARCHAR2,
    p_new_phone VARCHAR2,
    p_new_email VARCHAR2,
    p_new_city VARCHAR2,
    p_new_street VARCHAR2,
    p_new_prog_id NUMBER
)
IS
BEGIN

    UPDATE UNIVERSITY.Student
    SET F_Name = p_new_f_name,
        L_Name = p_new_l_name,
        Gender = p_new_gender,
        Phone = p_new_phone,
        Email = p_new_email,
        City = p_new_city,
        Street = p_new_street,
        Prog_ID = p_new_prog_id
    WHERE Nat_ID = p_nat_id;
END;
```

PlSQL Function For Calculating Student GPA:

```
CREATE OR REPLACE FUNCTION calculate_gpa(p_nat_id NUMBER  )
RETURN VARCHAR2
IS
    v_total_percentage NUMBER;
    v_gpa Number(10,2);
BEGIN
    -- Calculate the total percentage for the student's courses
    SELECT AVG(max(grade))
    INTO v_total_percentage
    FROM UNIVERSITY.Student_Course
    WHERE Nat_ID = p_nat_id
    GROUP BY Course_ID;
    -- Apply case statement for different GPA segments [1]
    CASE
        WHEN v_total_percentage >= 95 THEN v_gpa := 4;
        WHEN v_total_percentage >= 90 AND v_total_percentage < 95 THEN v_gpa := 3.67;
        WHEN v_total_percentage >= 85 AND v_total_percentage < 90 THEN v_gpa := 3.33;
        WHEN v_total_percentage >= 80 AND v_total_percentage < 85 THEN v_gpa := 3;
        WHEN v_total_percentage >= 75 AND v_total_percentage < 80 THEN v_gpa := 2.67;
        WHEN v_total_percentage >= 70 AND v_total_percentage < 75 THEN v_gpa := 2.33;
        WHEN v_total_percentage >= 65 AND v_total_percentage < 70 THEN v_gpa := 2;
        WHEN v_total_percentage >= 60 AND v_total_percentage < 65 THEN v_gpa := 1.67;
        WHEN v_total_percentage >= 0 AND v_total_percentage < 60 THEN v_gpa := 0;
        ELSE v_gpa := null ;
    END CASE;

    RETURN v_gpa;
END;
```

[1] Classification of Grades based on actual data of Future University which apply to Benha University too.
https://www.universitiesegypt.com/calculate-gpa-egypt#:~:text=A%20value%20(called%20points)%20is,each%20course%20you've%20taken

PlSQL Function For Calculating Course GPA:

```
CREATE OR REPLACE FUNCTION UNIVERSITY.calculate_course_gpa(p_course_id NUMBER  )
RETURN VARCHAR2
IS
    v_total_percentage NUMBER;
    v_gpa Number(10,2);
BEGIN
    -- Calculate the total percentage for the course
    SELECT max(grade)
    INTO v_total_percentage
    FROM UNIVERSITY.Student_Course
    WHERE Course_ID = p_course_id;
    -- Apply case statement for different GPA segments
    CASE
        WHEN v_total_percentage >= 95 THEN v_gpa := 4;
        WHEN v_total_percentage >= 90 AND v_total_percentage < 95 THEN v_gpa := 3.67;
        WHEN v_total_percentage >= 85 AND v_total_percentage < 90 THEN v_gpa := 3.33;
        WHEN v_total_percentage >= 80 AND v_total_percentage < 85 THEN v_gpa := 3;
        WHEN v_total_percentage >= 75 AND v_total_percentage < 80 THEN v_gpa := 2.67;
        WHEN v_total_percentage >= 70 AND v_total_percentage < 75 THEN v_gpa := 2.33;
        WHEN v_total_percentage >= 65 AND v_total_percentage < 70 THEN v_gpa := 2;
        WHEN v_total_percentage >= 60 AND v_total_percentage < 65 THEN v_gpa := 1.67;
        WHEN v_total_percentage >= 0 AND v_total_percentage < 60 THEN v_gpa := 0;
        ELSE v_gpa := null ;
    END CASE;

    RETURN v_gpa;
END;
```

PlSQL Trigger For Inserting Grade Records for each Course in The Program Whose the Student enrolled in :

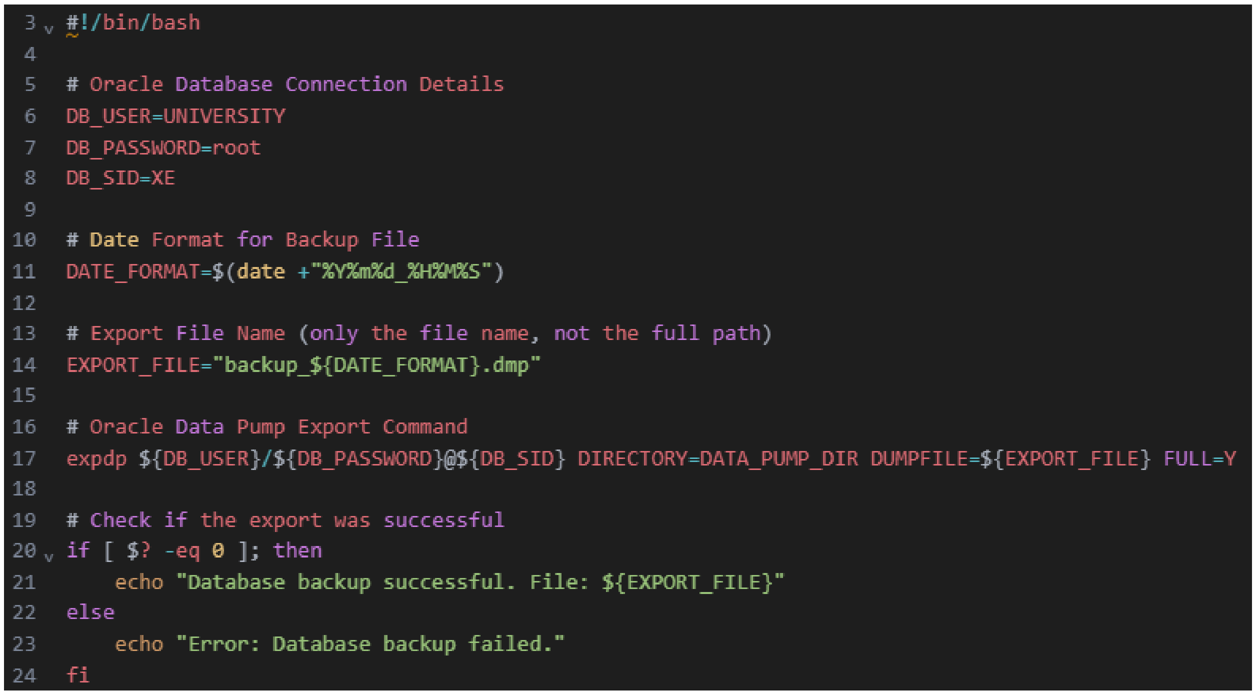
```
CREATE OR REPLACE TRIGGER add_student_trigger
AFTER INSERT ON student
FOR EACH ROW
DECLARE
    v_prog_id  NUMBER;
    v_nat_id   NUMBER;
    v_count    NUMBER;
    v_course_id NUMBER;

BEGIN
    v_prog_id := :NEW.prog_id;
    v_nat_id := :NEW.nat_id;

    -- Get the count of courses for the given prog_id
    SELECT COUNT(course_id) INTO v_count FROM course WHERE prog_id = v_prog_id;

    -- Loop through the courses and insert into student_course
    FOR i IN 1..v_count
    LOOP
        -- Retrieve the course_id using row_number
        SELECT course_id INTO v_course_id
        FROM (
            SELECT course_id, ROW_NUMBER() OVER (ORDER BY prog_id) num
            FROM course
            WHERE prog_id = v_prog_id
            ORDER BY prog_id
        )
        WHERE num = i;

        -- Insert into student_course
        INSERT INTO student_course(nat_id, course_id) VALUES (v_nat_id, v_course_id);

    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle exceptions
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END add_student_trigger;
```

PlSQL Trigger For Inserting Grade Records(in student_course table) for each Course in The Program That Was Updated(in student table) :

```
CREATE OR REPLACE TRIGGER Update_StudentProgram_Trigger
AFTER Update OF prog_id ON student
FOR EACH ROW
DECLARE
    v_prog_id  NUMBER;
    v_nat_id   NUMBER;
    v_count    NUMBER;
    v_course_id NUMBER; -- Declare v_course_id

BEGIN
    v_prog_id := :NEW.prog_id;
    v_nat_id := :NEW.nat_id;

    -- Get the count of courses for the given prog_id and delete the before saved record
    SELECT COUNT(course_id) INTO v_count FROM course WHERE prog_id = v_prog_id;
    Delete from student_course where nat_id = v_nat_id;
    -- Loop through the courses and insert into student_course
    FOR i IN 1..v_count
    LOOP
        -- Retrieve the course_id using row_number
        SELECT course_id INTO v_course_id
        FROM (
            SELECT course_id, ROW_NUMBER() OVER (ORDER BY prog_id) num
            FROM course
            WHERE prog_id = v_prog_id
            ORDER BY prog_id
        )
        WHERE num = i;

        -- Insert into student_course
        INSERT INTO student_course(nat_id, course_id) VALUES (v_nat_id, v_course_id);

    END LOOP;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle exceptions
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END ;
/
```

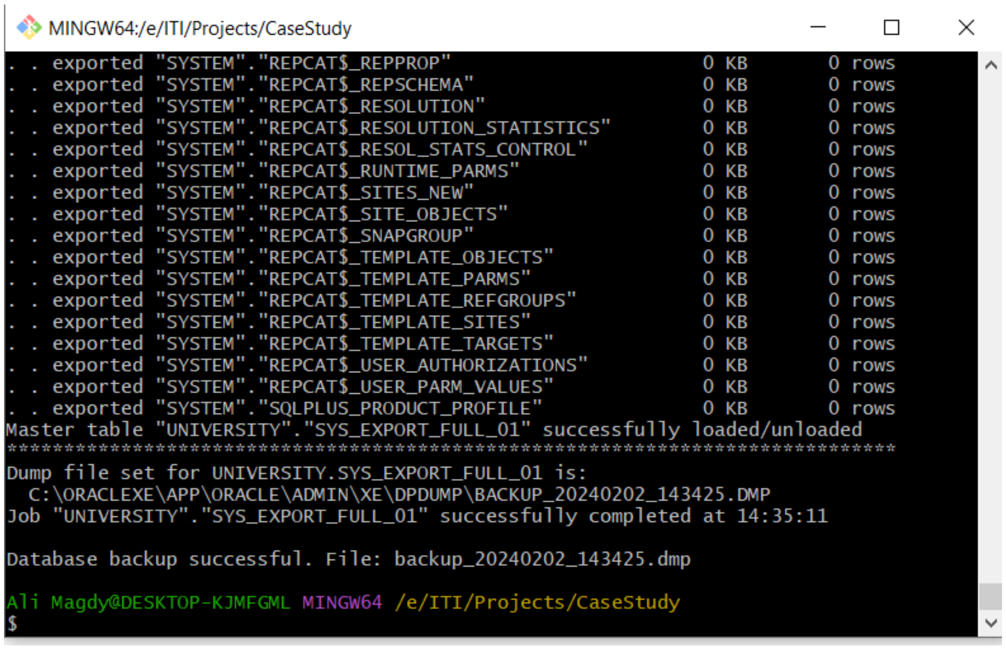
IV. Automation Scripts

Bash script for database backup.



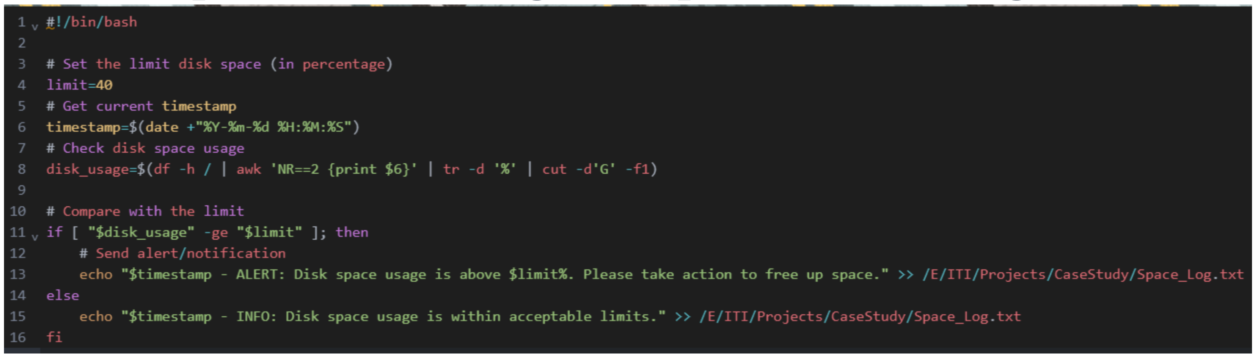
```
#!/bin/bash

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

# 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
```

On Running the Script



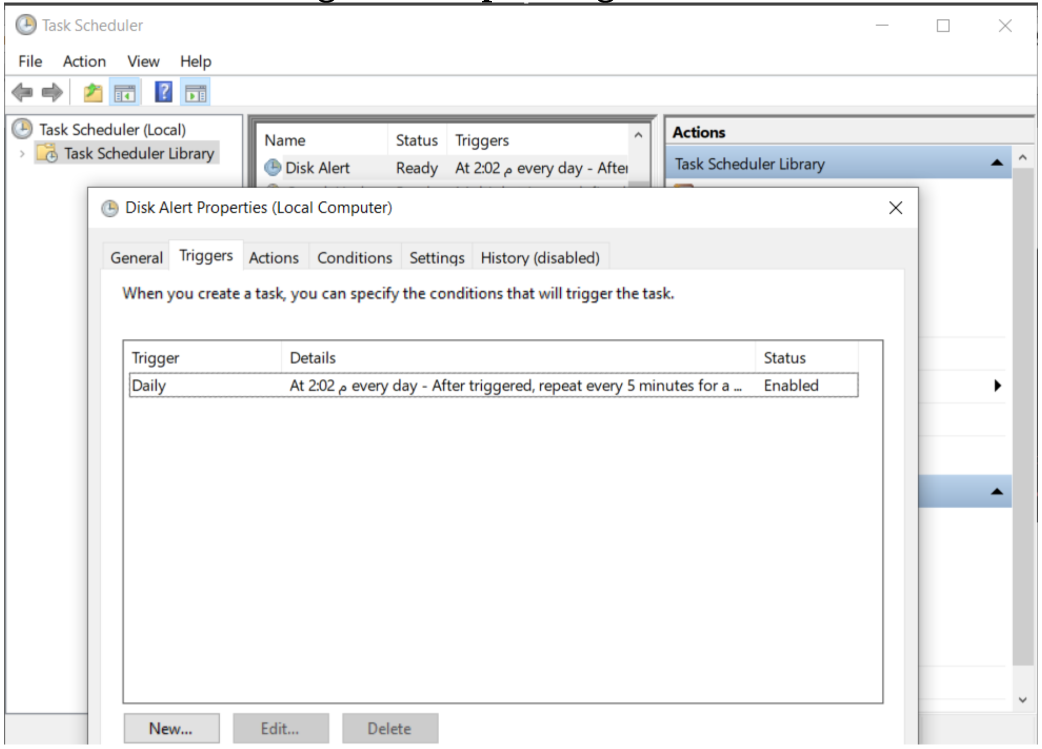
```
. . exported "SYSTEM"."REPCAT$_REPPROP"                      0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_REPSCHEMA"                    0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_RESOLUTION"                   0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_RESOLUTION_STATISTICS"        0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_RESOL_STATS_CONTROL"          0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_RUNTIME_PARMS"                0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_SITES_NEW"                    0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_SITE_OBJECTS"                 0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_SNAPGROUP"                    0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_TEMPLATE_OBJECTS"             0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_TEMPLATE_PARMS"               0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_TEMPLATE_REFGROUPS"           0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_TEMPLATE_SITES"               0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_TEMPLATE_TARGETS"             0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_USER_AUTHORIZATIONS"          0 KB       0 rows
. . exported "SYSTEM"."REPCAT$_USER_PARM_VALUES"             0 KB       0 rows
. . exported "SYSTEM"."SQLPLUS_PRODUCT_PROFILE"              0 KB       0 rows
Master table "UNIVERSITY"."SYS_EXPORT_FULL_01" successfully loaded/unloaded
******************************************************************************
Dump file set for UNIVERSITY.SYS_EXPORT_FULL_01 is:
  C:\ORACLEXE\APP\ORACLE\ADMIN\XE\DPDUMP\BACKUP_20240202_143425.DMP
Job "UNIVERSITY"."SYS_EXPORT_FULL_01" successfully completed at 14:35:11

Database backup successful. File: backup_20240202_143425.dmp

Ali Magdy@DESKTOP-KJMFGML MINGW64 /e/ITI/Projects/CaseStudy
$
```

Bash script for monitoring disk space and sending alerts.



```bash
#!/bin/bash

# Set the limit disk space (in percentage)
limit=40
# Get current timestamp
timestamp=$(date +"%Y-%m-%d %H:%M:%S")
# Check disk space usage
disk_usage=$(df -h / | awk 'NR==2 {print $6}' | tr -d '%' | cut -d'G' -f1)

# Compare with the limit
if [ "$disk_usage" -ge "$limit" ]; then
    # Send alert/notification
    echo "$timestamp - ALERT: Disk space usage is above $limit%. Please take action to free up space." >> /E/ITI/Projects/CaseStudy/Space_Log.txt
else
    echo "$timestamp - INFO: Disk space usage is within acceptable limits." >> /E/ITI/Projects/CaseStudy/Space_Log.txt
fi
```

Schedule of Running The Script Using Task Scheduler

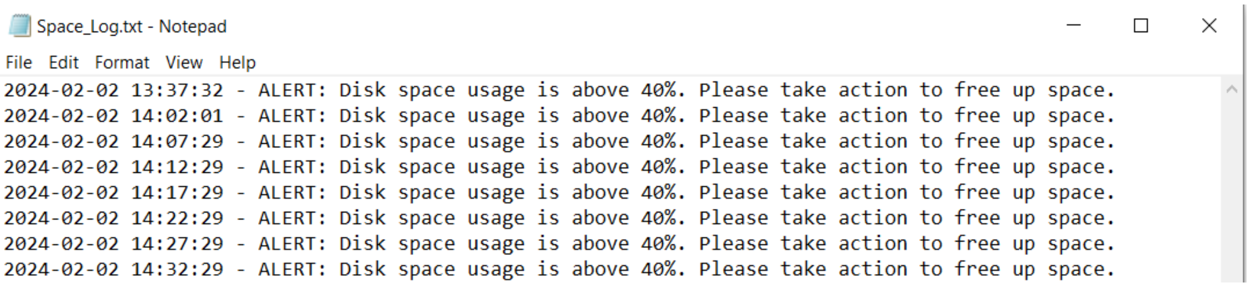


Name	Status	Triggers
Disk Alert	Ready	At 2:02 م every day - After

When you create a task, you can specify the conditions that will trigger the task.

Trigger	Details	Status
Daily	At 2:02 م every day - After triggered, repeat every 5 minutes for a ...	Enabled

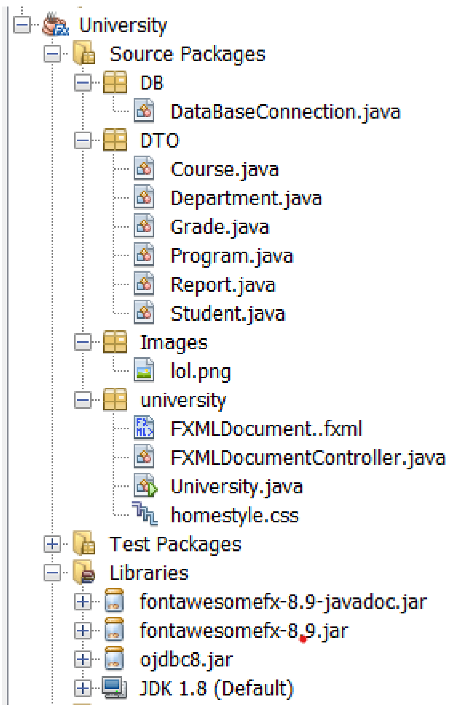
Sending Alert into Log files each 5 minutes as scheduled



```
2024-02-02 13:37:32 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:02:01 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:07:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:12:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:17:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:22:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:27:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
2024-02-02 14:32:29 - ALERT: Disk space usage is above 40%. Please take action to free up space.
```

V. Java Application Development

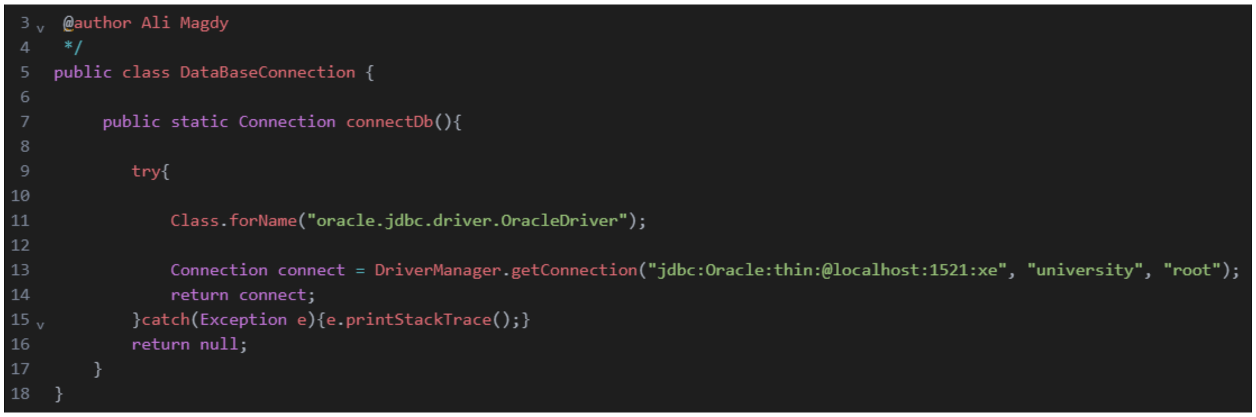
The Application Architecture:



```
University
  Source Packages
    DB
      DataBaseConnection.java
    DTO
      Course.java
      Department.java
      Grade.java
      Program.java
      Report.java
      Student.java
    Images
      lol.png
    university
      FXMLDocument..fxml
      FXMLDocumentController.java
      University.java
      homestyle.css
  Test Packages
  Libraries
    fontawesomefx-8.9-javadoc.jar
    fontawesomefx-8.9.jar
    ojdbc8.jar
    JDK 1.8 (Default)
```

DB Package:

- Contain the java class that establish the connection with oracle database.
- ConnectDb() method is being called in each method that need a database connection.

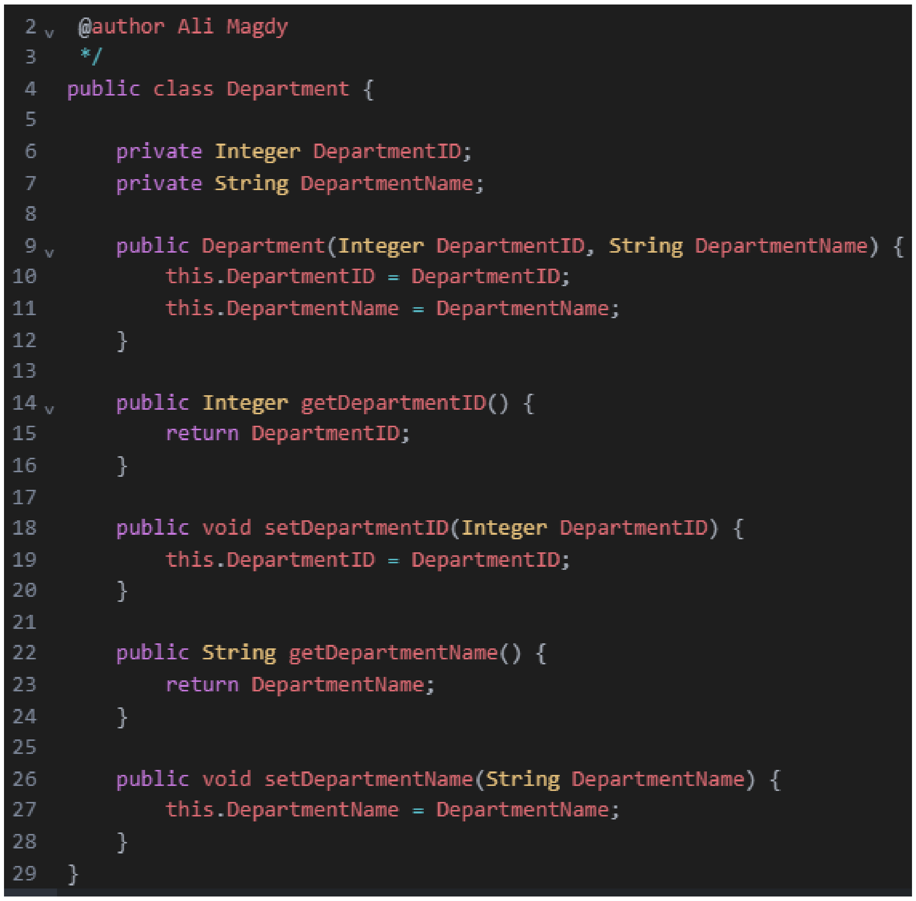


```java
 @author Ali Magdy
 */
public class DataBaseConnection {

     public static Connection connectDb(){

        try{

            Class.forName("oracle.jdbc.driver.OracleDriver");

            Connection connect = DriverManager.getConnection("jdbc:Oracle:thin:@localhost:1521:xe", "university", "root");
            return connect;
        }catch(Exception e){e.printStackTrace();}
        return null;
    }
}
```

DTO 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: Department DTO

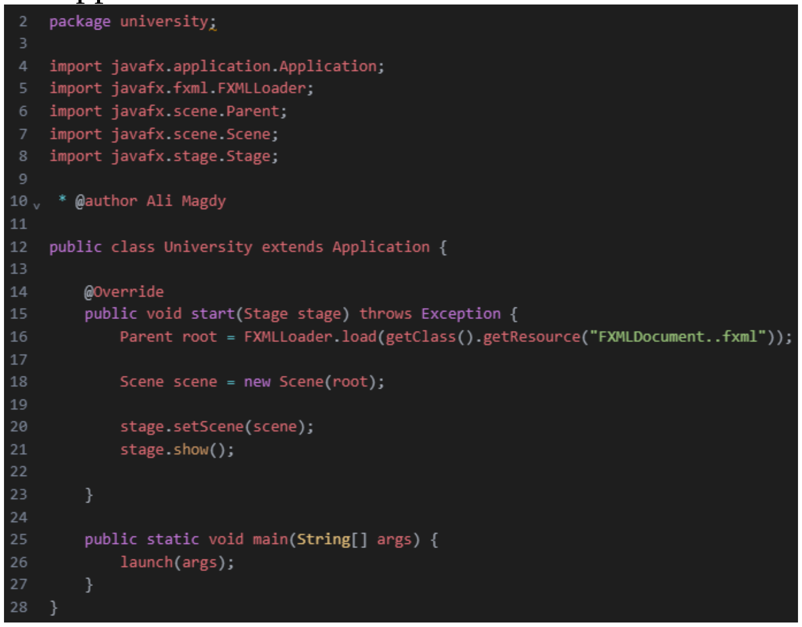


```
 @author Ali Magdy
 */
public class Department {

    private Integer DepartmentID;
    private String DepartmentName;

    public Department(Integer DepartmentID, String DepartmentName) {
        this.DepartmentID = DepartmentID;
        this.DepartmentName = DepartmentName;
    }

    public Integer getDepartmentID() {
        return DepartmentID;
    }

    public void setDepartmentID(Integer DepartmentID) {
        this.DepartmentID = DepartmentID;
    }

    public String getDepartmentName() {
        return DepartmentName;
    }

    public void setDepartmentName(String DepartmentName) {
        this.DepartmentName = DepartmentName;
    }
}
```

University Package

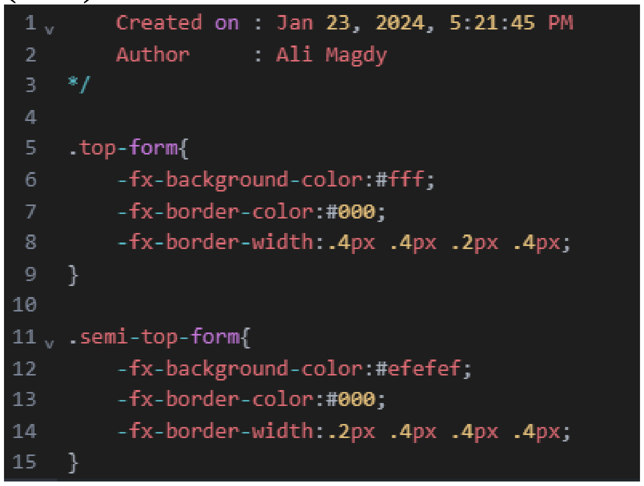
Components:

University.java class responsible for initializing and configuring the JavaFX application, loading the user interface layout from an FXML file, and displaying the application window.



```java
package university;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

 * @author Ali Magdy

public class University extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument..fxml"));

        Scene scene = new Scene(root);

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

    }

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

Homestyle.**css** (CSS) file containing styling rules for a graphical user interface (GUI).

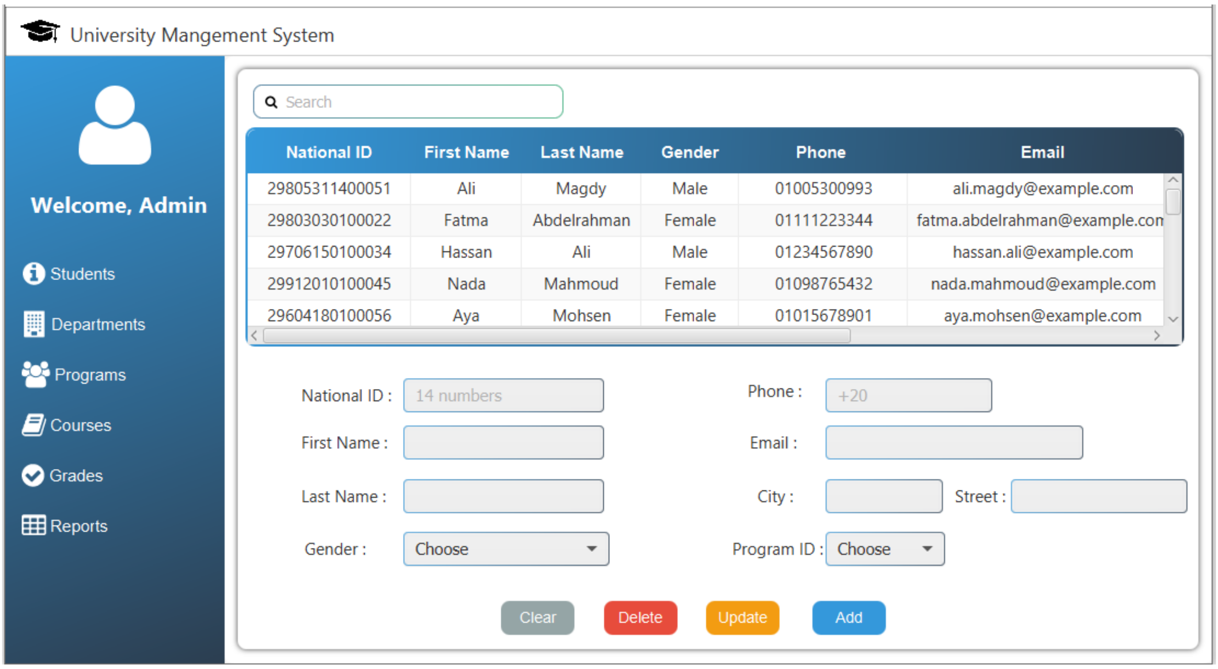


```css
    Created on : Jan 23, 2024, 5:21:45 PM
    Author     : Ali Magdy
*/

.top-form{
    -fx-background-color:#fff;
    -fx-border-color:#000;
    -fx-border-width:.4px .4px .2px .4px;
}

.semi-top-form{
    -fx-background-color:#efefef;
    -fx-border-color:#000;
    -fx-border-width:.2px .4px .4px .4px;
}
```

FXMLDocumentcontroller.java The main class of our project that contain the whole methods used in the applications and the handling of each GUI component.

The Application Scenes:

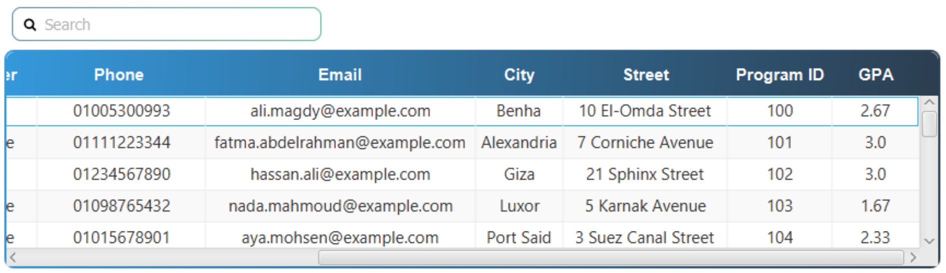
Student Information Scene



National ID	First Name	Last Name	Gender	Phone	Email
29805311400051	Ali	Magdy	Male	01005300993	ali.magdy@example.com
29803030100022	Fatma	Abdelrahman	Female	01111223344	fatma.abdelrahman@example.con
29706150100034	Hassan	Ali	Male	01234567890	hassan.ali@example.com
29912010100045	Nada	Mahmoud	Female	01098765432	nada.mahmoud@example.com
29604180100056	Aya	Mohsen	Female	01015678901	aya.mohsen@example.com

National ID : 14 numbers
Phone : +20
First Name :
Email :
Last Name :
City :
Street :
Gender : Choose
Program ID : Choose

The rest of the stored information

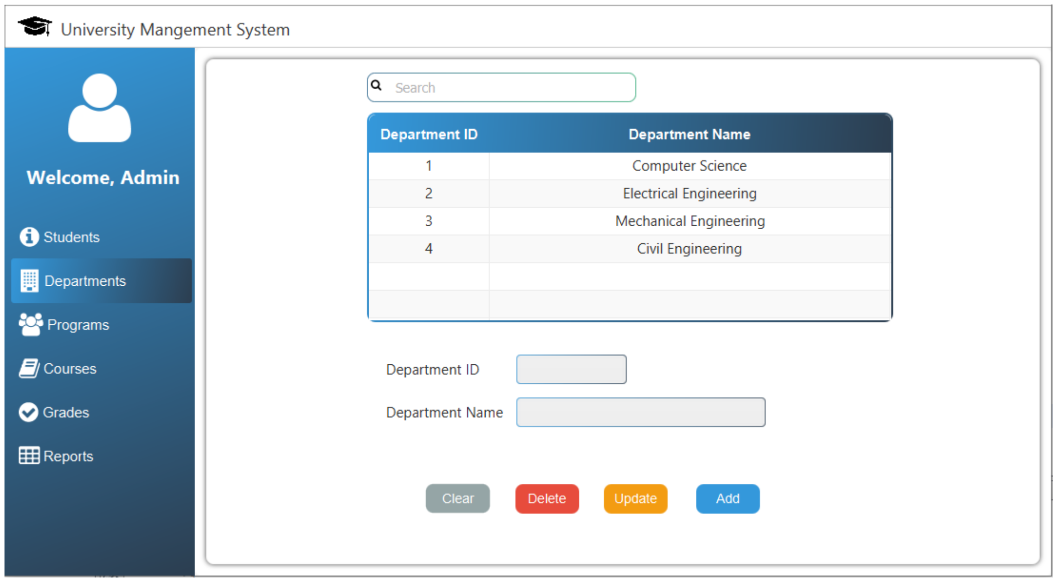


er	Phone	Email	City	Street	Program ID	GPA
	01005300993	ali.magdy@example.com	Benha	10 El-Omda Street	100	2.67
e	01111223344	fatma.abdelrahman@example.com	Alexandria	7 Corniche Avenue	101	3.0
	01234567890	hassan.ali@example.com	Giza	21 Sphinx Street	102	3.0
e	01098765432	nada.mahmoud@example.com	Luxor	5 Karnak Avenue	103	1.67
e	01015678901	aya.mohsen@example.com	Port Said	3 Suez Canal Street	104	2.33

This scene displays student information including GPA -calculated from PLSQL function declared before- in a table view.

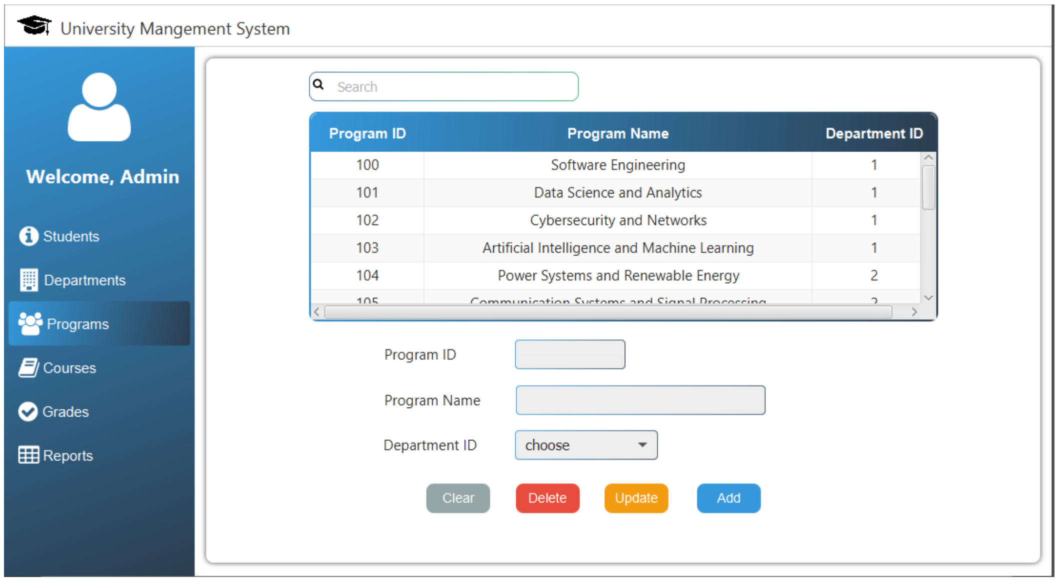
Include CRUD operations of students and utilizing PLSQL procedure used for updating student information.

Departments and Programs Scenes:



Department ID	Department Name
1	Computer Science
2	Electrical Engineering
3	Mechanical Engineering
4	Civil Engineering

Department ID	
Department Name	

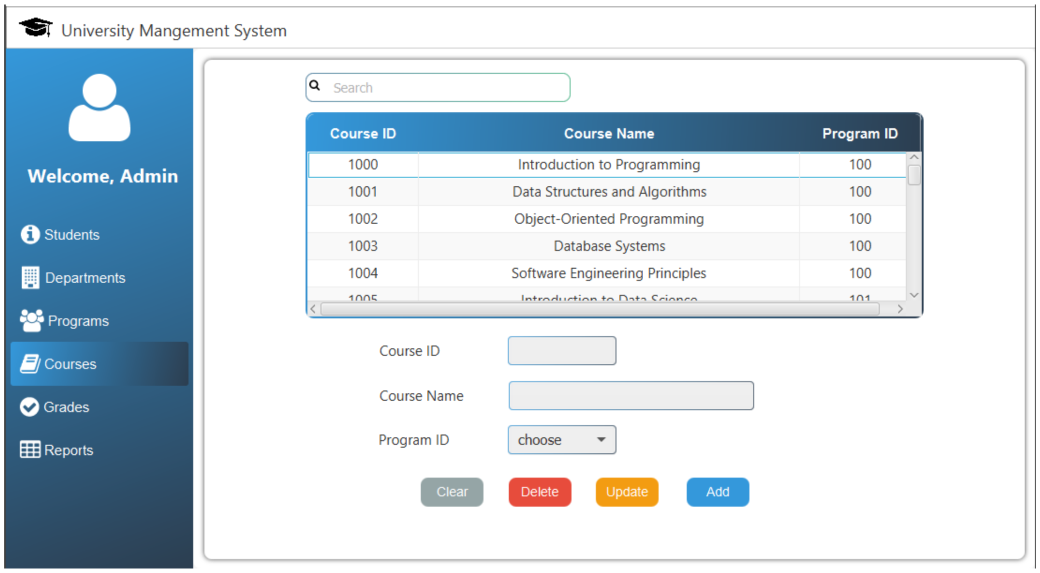


Program ID	Program Name	Department ID
100	Software Engineering	1
101	Data Science and Analytics	1
102	Cybersecurity and Networks	1
103	Artificial Intelligence and Machine Learning	1
104	Power Systems and Renewable Energy	2
105	Communication Systems and Signal Processing	2

Program ID	
Program Name	
Department ID	choose

These scenes display Department and Programs information in table view and also the CRUD operations needed

Courses and Grades Scenes:


University Mangement System

Welcome, Admin

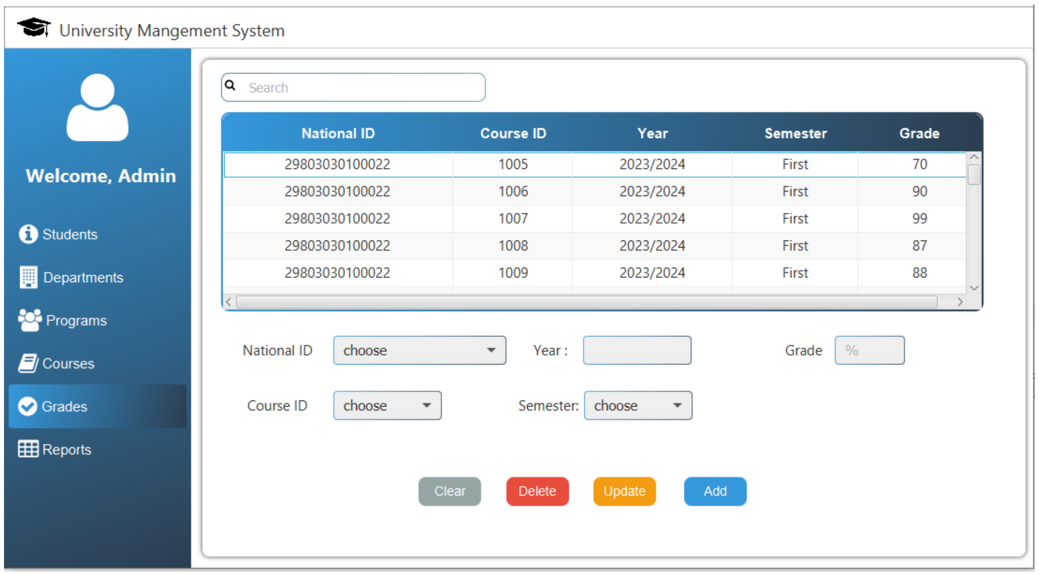
Students
Departments
Programs
Courses
Grades
Reports

Search

Course ID	Course Name	Program ID
1000	Introduction to Programming	100
1001	Data Structures and Algorithms	100
1002	Object-Oriented Programming	100
1003	Database Systems	100
1004	Software Engineering Principles	100
1005	Introduction to Data Science	101

Course ID
Course Name
Program ID: choose

Clear Delete Update Add


University Mangement System

Welcome, Admin

Students
Departments
Programs
Courses
Grades
Reports

Search

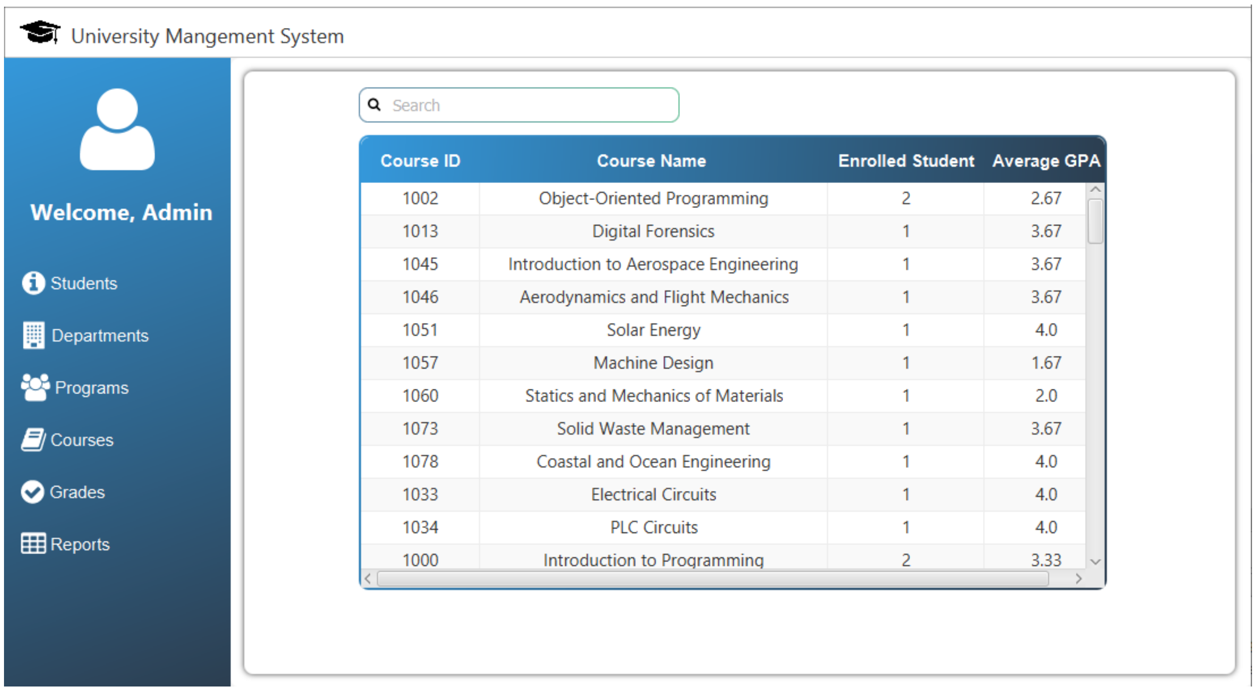
National ID	Course ID	Year	Semester	Grade
29803030100022	1005	2023/2024	First	70
29803030100022	1006	2023/2024	First	90
29803030100022	1007	2023/2024	First	99
29803030100022	1008	2023/2024	First	87
29803030100022	1009	2023/2024	First	88

National ID: choose
Year :
Grade: %
Course ID: choose
Semester: choose

Clear Delete Update Add

These scenes display Courses and Grades information in table view and also the CRUD operations needed.

Report Scene:

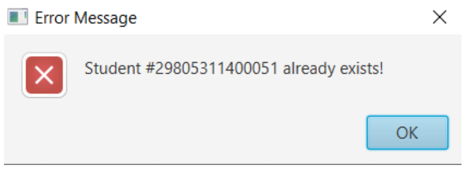


Course ID	Course Name	Enrolled Student	Average GPA
1002	Object-Oriented Programming	2	2.67
1013	Digital Forensics	1	3.67
1045	Introduction to Aerospace Engineering	1	3.67
1046	Aerodynamics and Flight Mechanics	1	3.67
1051	Solar Energy	1	4.0
1057	Machine Design	1	1.67
1060	Statics and Mechanics of Materials	1	2.0
1073	Solid Waste Management	1	3.67
1078	Coastal and Ocean Engineering	1	4.0
1033	Electrical Circuits	1	4.0
1034	PLC Circuits	1	4.0
1000	Introduction to Programming	2	3.33

This scene displays Course Report including average GPA for each course -calculated from PLSQL function declared before- in a table view.

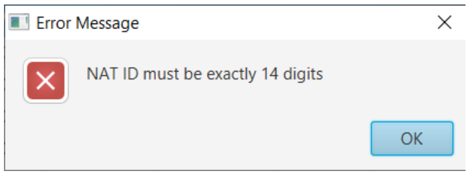
Anomalies Checks

Any Primary Key can’t be repeated



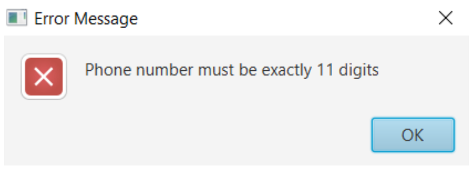
Student #29805311400051 already exists!

National ID shall be 14 digits



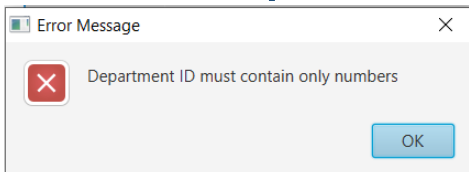
NAT ID must be exactly 14 digits

Phone number shall be 11 characters



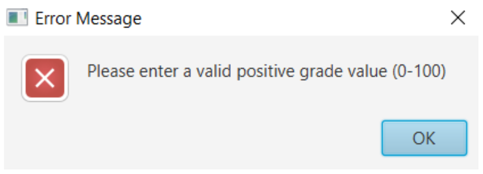
Phone number must be exactly 11 digits

ID shall be only Numbers



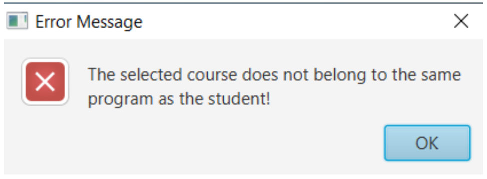
Department ID must contain only numbers

Grades shall be only positive number between 0 and 100



Please enter a valid positive grade value (0-100)

Student Cant Enroll in a course from another department

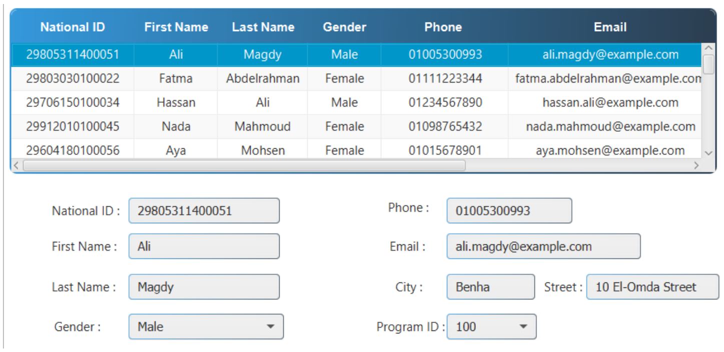


The selected course does not belong to the same program as the student!

GUI Features

Fields is filled on table view selection:

Used for easy data manipulations

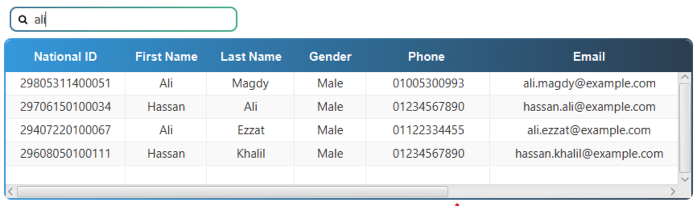


National ID	First Name	Last Name	Gender	Phone	Email
29805311400051	Ali	Magdy	Male	01005300993	ali.magdy@example.com
29803030100022	Fatma	Abdelrahman	Female	01111223344	fatma.abdelrahman@example.con
29706150100034	Hassan	Ali	Male	01234567890	hassan.ali@example.com
29912010100045	Nada	Mahmoud	Female	01098765432	nada.mahmoud@example.com
29604180100056	Aya	Mohsen	Female	01015678901	aya.mohsen@example.com

National ID :	29805311400051	Phone :	01005300993
First Name :	Ali	Email :	ali.magdy@example.com
Last Name :	Magdy	City :	Benha
		Street :	10 El-Omda Street
Gender :	Male	Program ID :	100

Search Text Field:

Easiness of finding the desired data

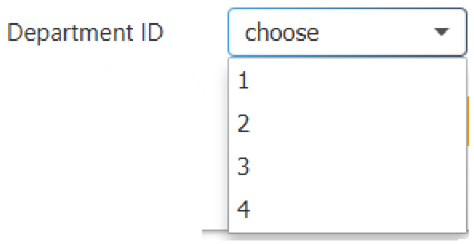


ali

National ID	First Name	Last Name	Gender	Phone	Email
29805311400051	Ali	Magdy	Male	01005300993	ali.magdy@example.com
29706150100034	Hassan	Ali	Male	01234567890	hassan.ali@example.com
29407220100067	Ali	Ezzat	Male	01122334455	ali.ezzat@example.com
29608050100111	Hassan	Khalil	Male	01234567890	hassan.khalil@example.com

Combo box that retrieve data from the database:

Enforce the user to insert only the right data to ensure data base integrity.


Department ID
choose
1
2
3
4