UNIVERSITY SYSTEM

Yousef Saber Abdelkarim

AGENDA

introduction

Database design

SQL implementation

PLSQL

Automation script

Java application

Reporting

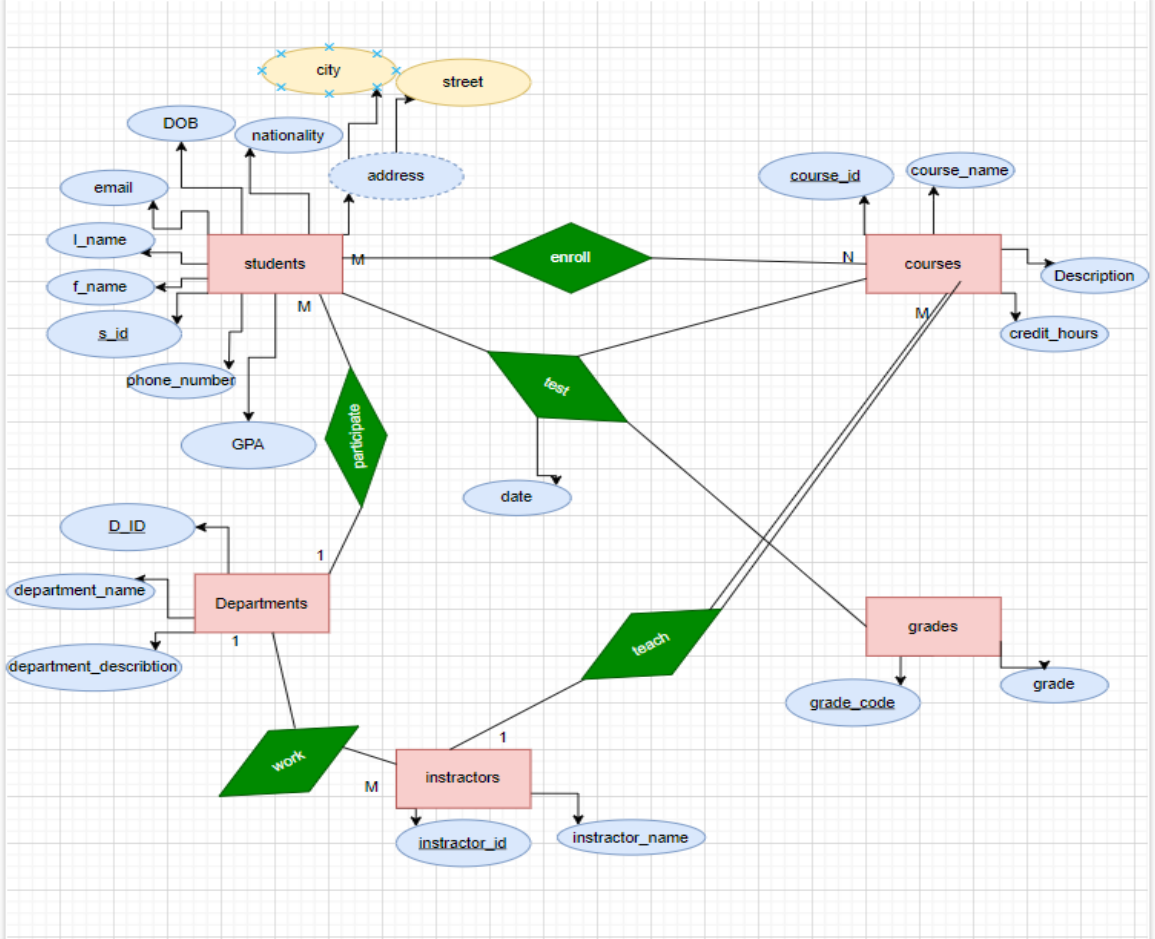
INTRODUCTION

This detailed guide provides a thorough overview of a University System, including its design, execution, and features. It encompasses database structuring, SQL incorporation, PL/SQL procedures, an automated script, and a Java program. Each segment is intricately explained to ensure a comprehensive grasp of the system's framework, aiding in its smooth development, upkeep, and issue resolution.

Subtitle

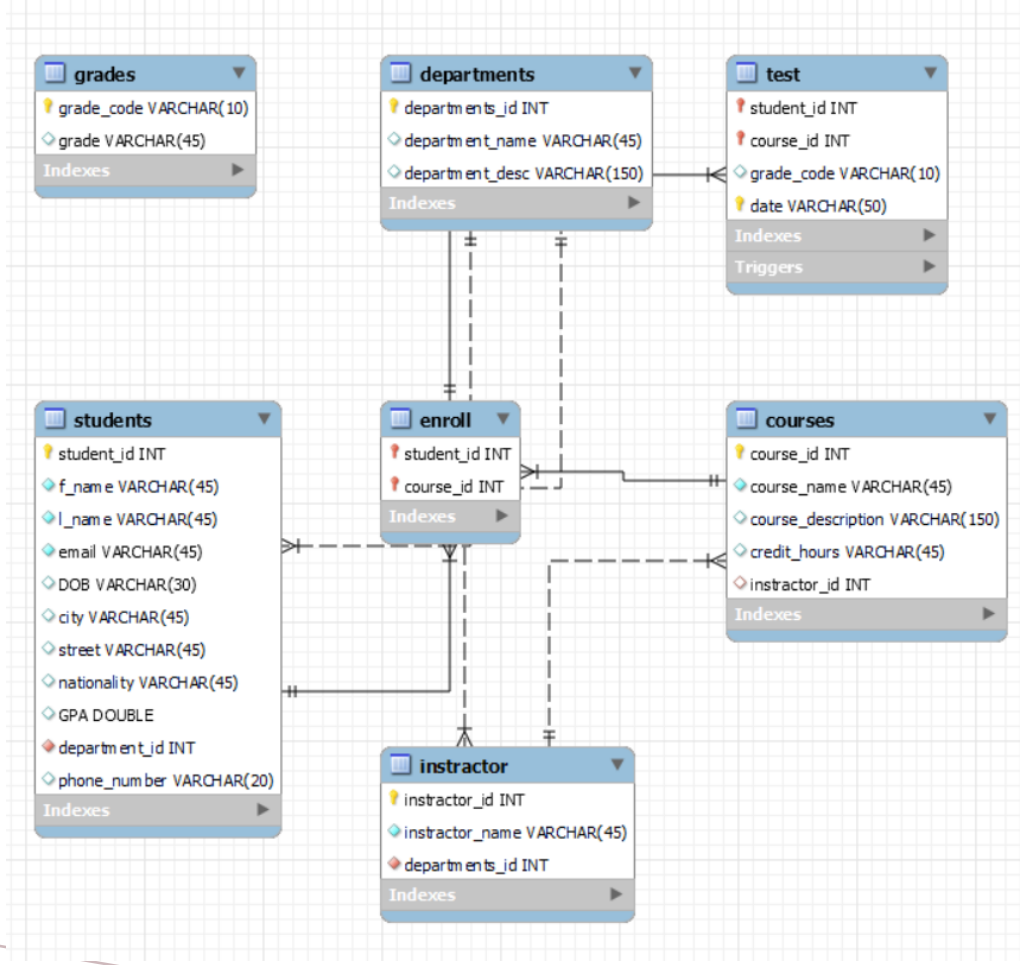
DATABASE DESIGN

ERD



THE ERD OFFERS A DETAILED INSIGHT INTO THE RELATIONAL DATABASE SCHEMA DESIGNED TO MANAGE STUDENT, INSTRUCTORS ,COURSE, DEPARTMENT, AND GRADE DATA EFFICIENTLY.

SQL IMPLEMENTATION



Design Logical Map

PLSQL

Procedure 1 : to update students

Procedure 2 : to update courses

To calculate GPA for student

Trigger

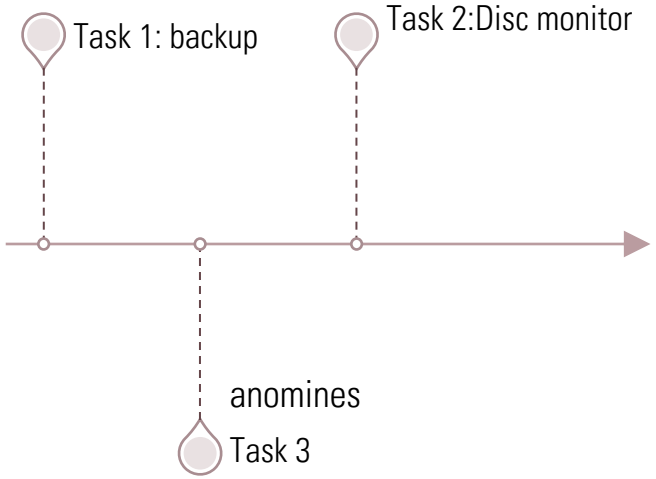
```mysql
DELIMITER //

CREATE PROCEDURE update_course(
    IN p_course_id INT,
    IN p_course_name VARCHAR(255))
BEGIN
    UPDATE courses
    SET course_name = p_course_name
    WHERE course_id = p_course_id;
END;
//

DELIMITER ;

CALL update_course(1, 'it');
```

```mysql
DELIMITER //

CREATE PROCEDURE update_students(
    IN p_student_id INT,
    IN p_f_name VARCHAR(255),
    IN p_l_name VARCHAR(255),
    IN p_email VARCHAR(255),
    IN p_city VARCHAR(255),
    IN p_phone_number VARCHAR(15)
)
BEGIN
    UPDATE students
    SET f_name = p_f_name,
        l_name = p_l_name,
        email = p_email,
        city = p_city,
        phone_number = p_phone_number
    WHERE student_id = p_student_id;
END;
//
```

```mysql
DELIMITER //

CREATE TRIGGER calculate_gpa
AFTER INSERT ON Test
FOR EACH ROW
BEGIN
    DECLARE v_total_points DECIMAL(8, 2);
    DECLARE v_total_hours INT;
    DECLARE v_gpa DECIMAL(8, 2);

    -- Calculate total points for all courses taken by the student
    SELECT COALESCE(SUM(Grades.grade * Courses.credit_hours), 0)
    INTO v_total_points
    FROM Test
    INNER JOIN Grades ON Test.grade_code = Grades.grade_code
    INNER JOIN Courses ON Test.course_id = Courses.course_id
    WHERE Test.student_id = NEW.student_id;

    -- Calculate total hours for all courses taken by the student
    SELECT COALESCE(SUM(Courses.credit_hours), 0)
    INTO v_total_hours
    FROM Test
    INNER JOIN Courses ON Test.course_id = Courses.course_id
    WHERE Test.student_id = NEW.student_id;

    -- Calculate GPA
    IF v_total_hours > 0 THEN
        SET v_gpa = v_total_points / v_total_hours;
    ELSE
        SET v_gpa = 0;
    END IF;

    -- Update GPA in the students table for the same student
    UPDATE students
    SET GPA = v_gpa
    WHERE student_id = NEW.student_id;
END;
//
```

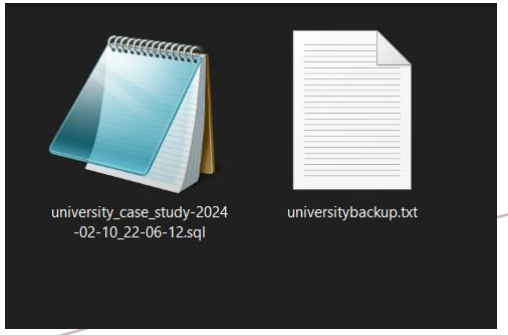
AUTOMATION SCRIPT

Task 1: backup




```bash
#!/bin/bash

# Database connection details
DB_HOST="localhost"
DB_PORT="3306"
DB_USER="root"
DB_PASS="yousefsaber_1999"
DB_NAME="university_case_study"

# Backup directory
BACKUP_DIR="D:/New Folder/backup"

# Date format for backup file
DATE=$(date +"%Y-%m-%d_%H-%M-%S")

# Backup file name
BACKUP_FILE="$BACKUP_DIR/$DB_NAME-$DATE.sql"

# Check if the backup directory exists, otherwise create it
if [ ! -d "$BACKUP_DIR" ]; then
    mkdir -p "$BACKUP_DIR"
    if [ $? -ne 0 ]; then
        echo "Failed to create backup directory: $BACKUP_DIR"
        exit 1
    fi
fi

# Create a backup using mysqldump
mysqldump -h "$DB_HOST" -P "$DB_PORT" -u "$DB_USER" -p"$DB_PASS" "$DB_NAME" > "$BACKUP_FILE"

# Check if the backup was successful
if [ $? -eq 0 ]; then
    echo "Backup completed successfully: $BACKUP_FILE"
else
    echo "Backup failed"
fi
```

```
yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ vim universitybackup.txt

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ chmod +x universitybackup.txt

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ ./universitybackup.txt
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Backup completed successfully: D:/New Folder/backup/university_case_study-2024-02-11_20-19-53.sql

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$
```



Task 2:Disc monitor

```bash
#!/bin/bash

# Email address to receive alerts
recipient_email="yosefsaber390@gmail.com"
log_file="D:/New Folder/backup.log"
# Function to monitor disk usage
monitor_disk_usage() {
    echo "Disk Usage:"
    df -h | awk 'NR==2 {print "Used: " $5}'
    disk_usage=$(df -h | awk 'NR==2 {print $5}' | sed 's/%//')
    if [ $disk_usage -gt 70 ]; then
        echo "Disk usage is above 70%!"
        # Send email alert
        echo "Disk usage is above 70% on $(hostname)" | mail -s "Disk Alert" "$recipient_email"
        echo "Disk usage is above 70% on $(hostname)" >> "${log_file}"

    fi
}

# Main function to run all monitoring functions
main() {
    monitor_disk_usage
}
```

```
yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ vim monitor

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ chmod +w monitor

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ chmod +x monitor

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ ./monitor
Disk Usage:
Used: 98%
Disk usage is above 70%!
./monitor: line 14: mail: command not found

yousef saber@DESKTOP-331H94E MINGW64 /d/New Folder/backup
$ |
```

backup.log - Notepad

File Edit Format View Help

```
Disk usage is above 70% on DESKTOP-331H94E
Disk usage is above 70% on DESKTOP-331H94E
Disk usage is above 70% on DESKTOP-331H94E
Disk usage is above 70% on DESKTOP-331H94E
Disk usage is above 70% on DESKTOP-331H94E
```

JAVA APPLICATION

The Application contain 7 main scenes:

- Login scene.
- Students' scene.
- Courses scene.
- Departments scene.
- Instructor scene.
- Reporting (GPA) scene.
- Enroll scene.
- Test scene.

Code Structure:

The code is structured into various methods and functions, each responsible for specific tasks. The main sections include:

- Scenes: Manages all scenes related functionalities.
- Main Controller: Controls the main functionality and GUI switching.
- Database Access Layer: Provides database connectivity and SQL operations.
- Data Transfer: Object (DTO) For Each Entity Showed In the app.

Scenario:

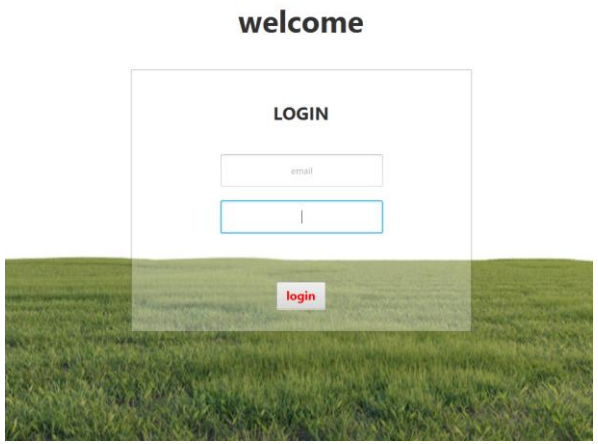
1-Run the from file ((file name). java)

```java
package university_case_study;

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

/**
 *
 * @author yousef saber
 */
public class University_case_study extends Application {

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

        Scene scene = new Scene(root);

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


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

2- It will access the login controller and send you to the login fxml

Login controller some details:

```java
private Parent root;
private Stage stage;
private Scene scene;

@Override
public void initialize(URL url, ResourceBundle rb) {

}

@FXML
private void login(ActionEvent event) throws IOException {
    if ("admin".equals(email.getText().toLowerCase()) && "123456789".equals(password.getText()))
        // Load the server_pane.fxml scene for the root user
        root = FXMLLoader.load(getClass().getResource("student.fxml"));
        stage = (Stage) login_btn.getScene().getWindow();
        scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    } else {
        wronglogin.setText("Wrong Email or Password");
    }
}
```



3-From login controller if the email and password is write will transport you to the student fxml scene which have the seven scene we mentioned up:

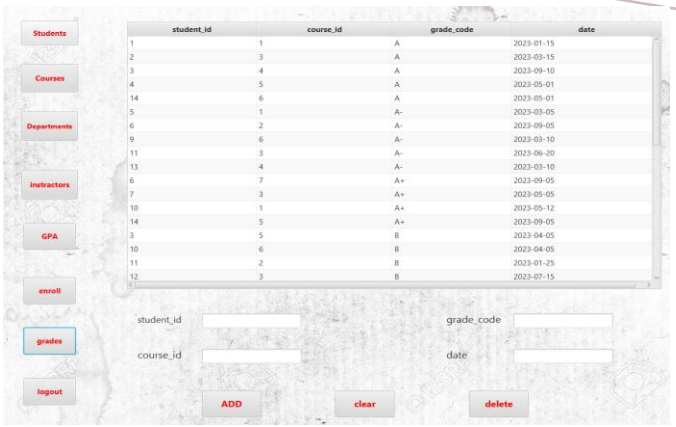
We will describe that button

A.(students-courses-departments-instructors-enroll-grades) have the same thing

1-tables which show the data

2-text filed which we can through it (add-update-delete-clear)

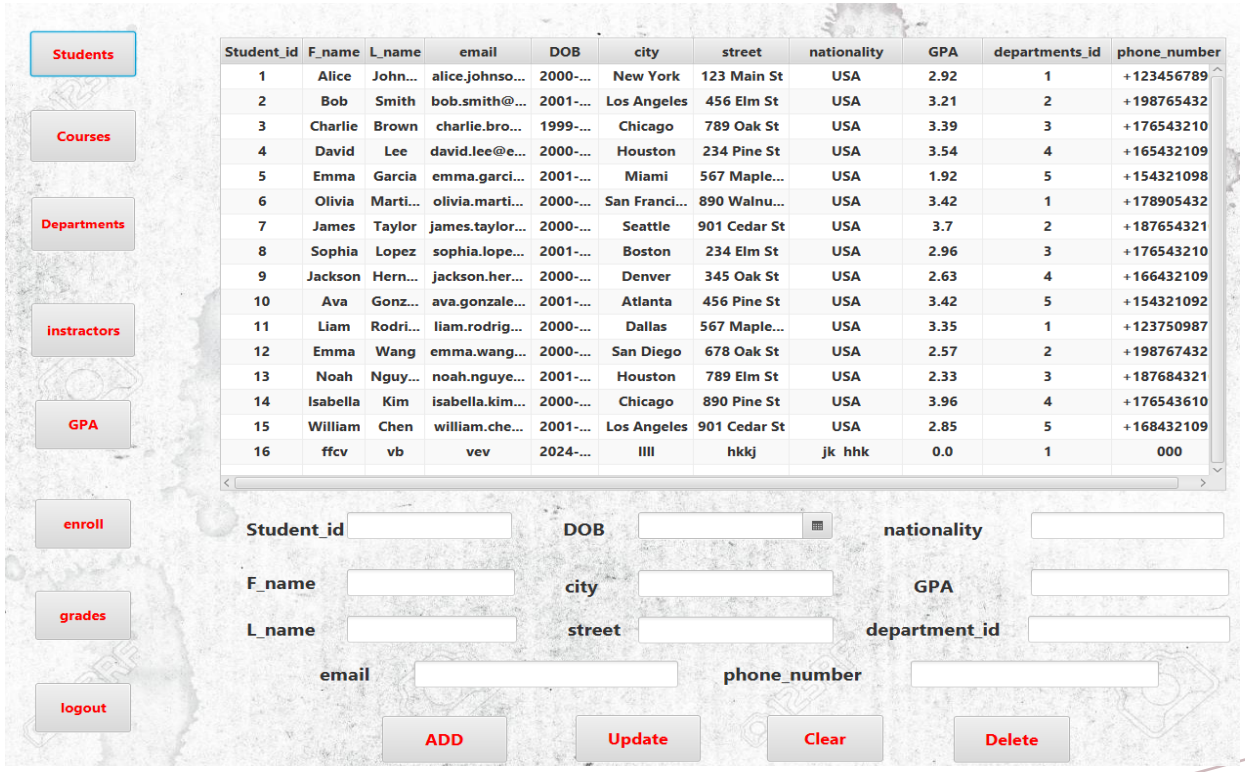
The data and putted in the table.



b. Logout button : make you back again to login scene

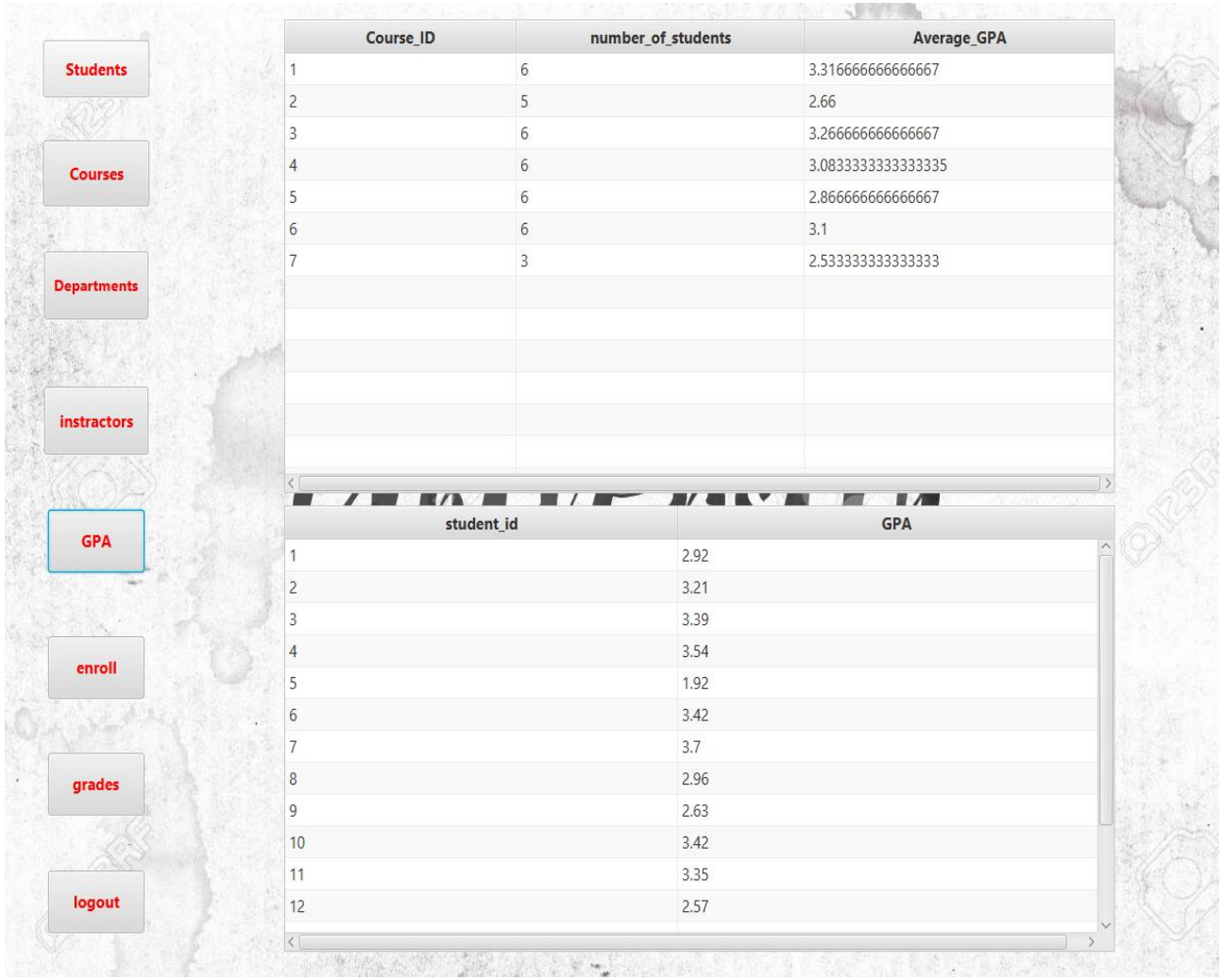
c .GPA button (have two tables) describe what is in the photo and this is the report that we do.

---and all that handle in the (student controller) which have around 1500 lines of code to have all things(action , alert, exceptions) in the scene and (DTOS) that I created.



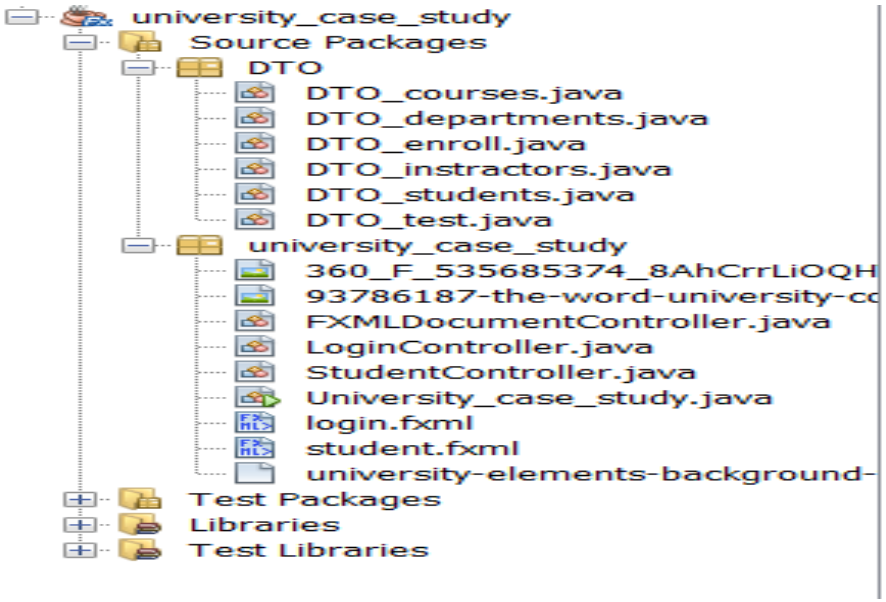
Student_id	F_name	L_name	email	DOB	city	street	nationality	GPA	departments_id	phone_number
1	Alice	John...	alice.johnso...	2000-...	New York	123 Main St	USA	2.92	1	+123456789
2	Bob	Smith	bob.smith@...	2001-...	Los Angeles	456 Elm St	USA	3.21	2	+198765432
3	Charlie	Brown	charlie.bro...	1999-...	Chicago	789 Oak St	USA	3.39	3	+176543210
4	David	Lee	david.lee@e...	2000-...	Houston	234 Pine St	USA	3.54	4	+165432109
5	Emma	Garcia	emma.garci...	2001-...	Miami	567 Maple...	USA	1.92	5	+154321098
6	Olivia	Marti...	olivia.marti...	2000-...	San Franci...	890 Walnu...	USA	3.42	1	+178905432
7	James	Taylor	james.taylor...	2000-...	Seattle	901 Cedar St	USA	3.7	2	+187654321
8	Sophia	Lopez	sophia.lope...	2001-...	Boston	234 Elm St	USA	2.96	3	+176543210
9	Jackson	Hern...	jackson.her...	2000-...	Denver	345 Oak St	USA	2.63	4	+166432109
10	Ava	Gonz...	ava.gonzale...	2001-...	Atlanta	456 Pine St	USA	3.42	5	+154321092
11	Liam	Rodri...	liam.rodrig...	2000-...	Dallas	567 Maple...	USA	3.35	1	+123750987
12	Emma	Wang	emma.wang...	2000-...	San Diego	678 Oak St	USA	2.57	2	+198767432
13	Noah	Nguy...	noah.nguye...	2001-...	Houston	789 Elm St	USA	2.33	3	+187684321
14	Isabella	Kim	isabella.kim...	2000-...	Chicago	890 Pine St	USA	3.96	4	+176543610
15	William	Chen	william.che...	2001-...	Los Angeles	901 Cedar St	USA	2.85	5	+168432109
16	ffcv	vb	vev	2024-...	llll	hkkj	jk hhk	0.0	1	000

Reporting



Course_ID	number_of_students	Average_GPA
1	6	3.316666666666667
2	5	2.66
3	6	3.266666666666667
4	6	3.0833333333333335
5	6	2.866666666666667
6	6	3.1
7	3	2.533333333333333

student_id	GPA
1	2.92
2	3.21
3	3.39
4	3.54
5	1.92
6	3.42
7	3.7
8	2.96
9	2.63
10	3.42
11	3.35
12	2.57



THANK YOU