Course Performance Information System

DBSL mini project

**Team members:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Reg. no. | Roll no. | Section |
| Anika Jha | 210905366 | 54 | CSE, section C |
| Dhruthi K | 210905388 | 58 |

# Abstract:

This project involves design and implementation of a system to record course performance information. The system will enable teachers to input and update marks for each subject for a student. It will also enable them to modify, insert and delete student records, update the cut-off for subjects as well as view information about a given subject or student. The project has been implemented using Python, Tkinter (GUI) and Oracle SQL.

**Problem Statement:** Design and implementation of course performance management system for updation of cut-offs, grade display and modification of student records.

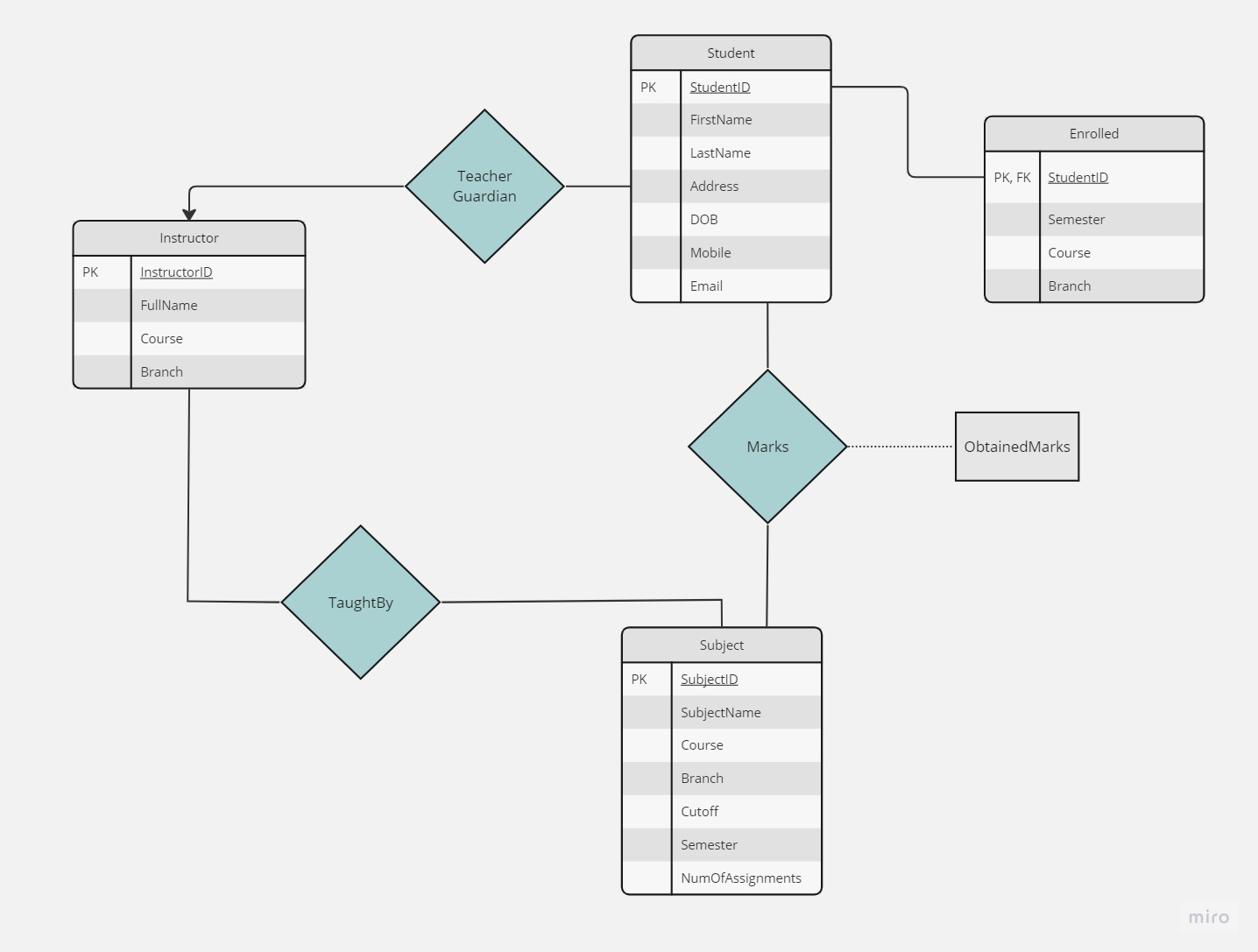
**Data Requirements:**

* Student Information: Name, ID, Contact details, Address, Enrolment details
* Course Information: Course name, Course code, Branch
* Instructor Information: Name, ID, Subjects taught
* Grade Information: Subject-wise cut-offs

**Functional Requirements:**

* Computation of sum of marks to get the total course marks
* Insertion, updation and deletion of student records
* Provision for the number of assignments/exams to not be predefined
* Appropriate grading for subjects
* Provision for cut-offs to be specified for various grades

# ER Diagram and Relational Schema



# DDL Commands to Create Tables

CREATE TABLE Student (

   StudentID VARCHAR2(10) PRIMARY KEY,

   FirstName VARCHAR2(30),

   LastName VARCHAR2(30),

   Address VARCHAR2(100),

   DOB DATE,

   Mobile VARCHAR2(20),

   Email VARCHAR2(50)

);

CREATE TABLE Enrolled (

   StudentID VARCHAR2(10) PRIMARY KEY,

   Semester VARCHAR2(10),

   Course VARCHAR2(20),

   Branch VARCHAR2(30),

   CONSTRAINT fk\_enrolled

        FOREIGN KEY (StudentID)

        REFERENCES Student(StudentID)

        ON DELETE CASCADE

);

CREATE TABLE Instructor (

    InstructorID VARCHAR2(10) PRIMARY KEY,

    FullName VARCHAR2(50),

    Course VARCHAR2(20),

    Branch VARCHAR2(30)

);

CREATE TABLE TaughtBy (

    SubjectID VARCHAR2(30),

    InstructorID VARCHAR2(10),

    CONSTRAINT fk\_taughtby

        FOREIGN KEY (SubjectID)

        REFERENCES Subject(SubjectID)

        ON DELETE CASCADE

);

CREATE TABLE TeacherGuardian (

    StudentID VARCHAR2(10),

    InstructorID VARCHAR2(10),

    CONSTRAINT fk\_teachergrd

        FOREIGN KEY (StudentID)

        REFERENCES Student(StudentID)

        ON DELETE CASCADE

);

CREATE TABLE Subject (

    SubjectID VARCHAR2(10) PRIMARY KEY,

    SubjectName VARCHAR2(30),

    Course VARCHAR2(20),

    Branch VARCHAR2(30),

    Cutoff NUMBER,

    Semester NUMBER,

    NumOfAssignments NUMBER

);

CREATE TABLE Marks (

    StudentID VARCHAR2(10),

    SubjectID VARCHAR2(10),

    ObtainedMarks NUMBER(10),

    CONSTRAINT fk\_marks

        FOREIGN KEY (StudentID)

        REFERENCES Student(StudentID)

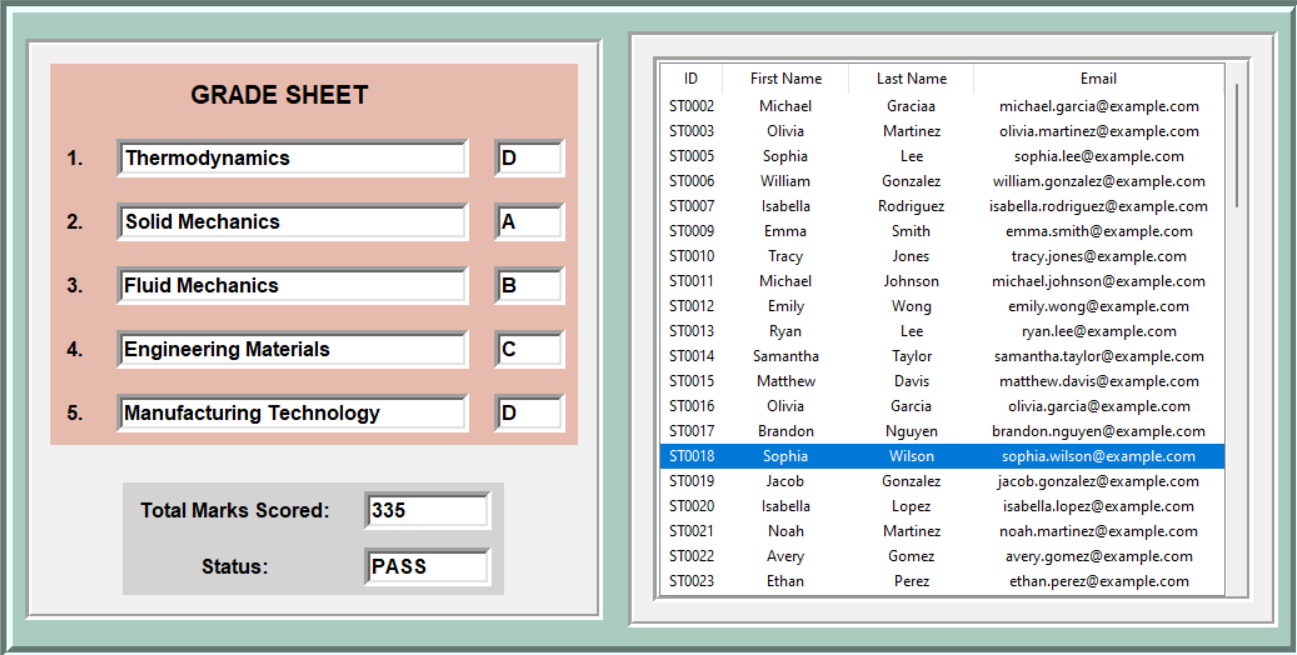
        ON DELETE CASCADE

);

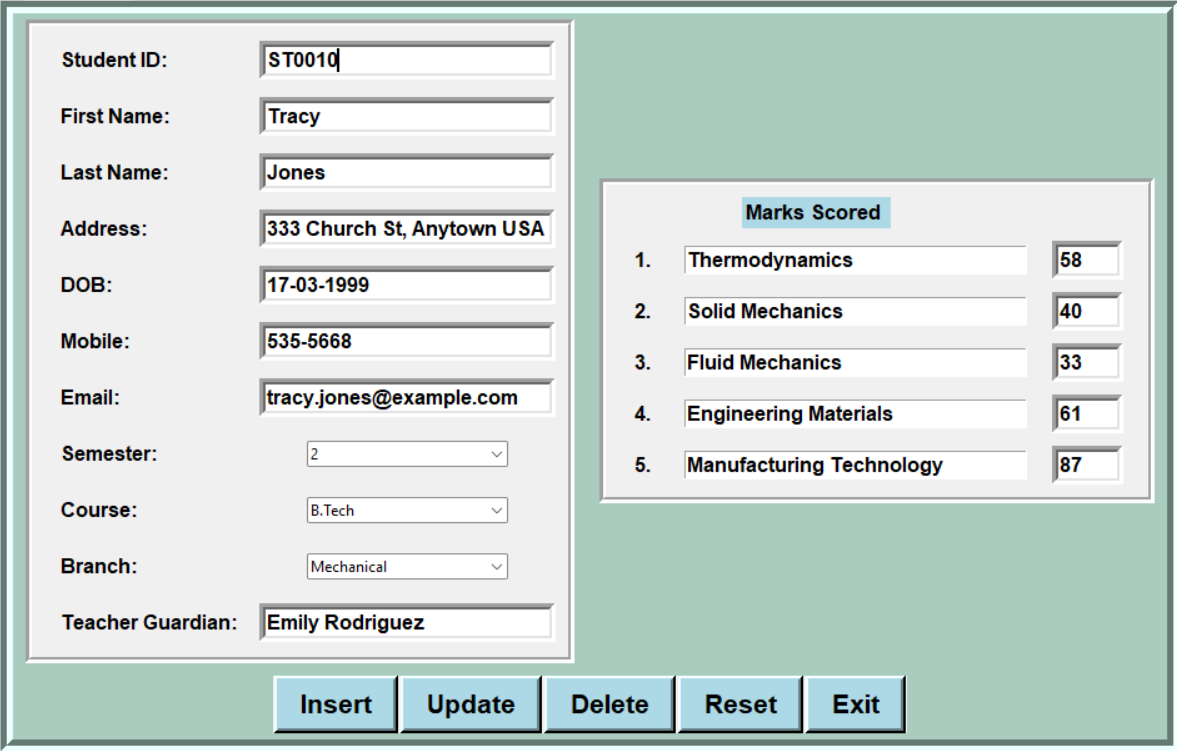
# SQL Queries

# UI Design

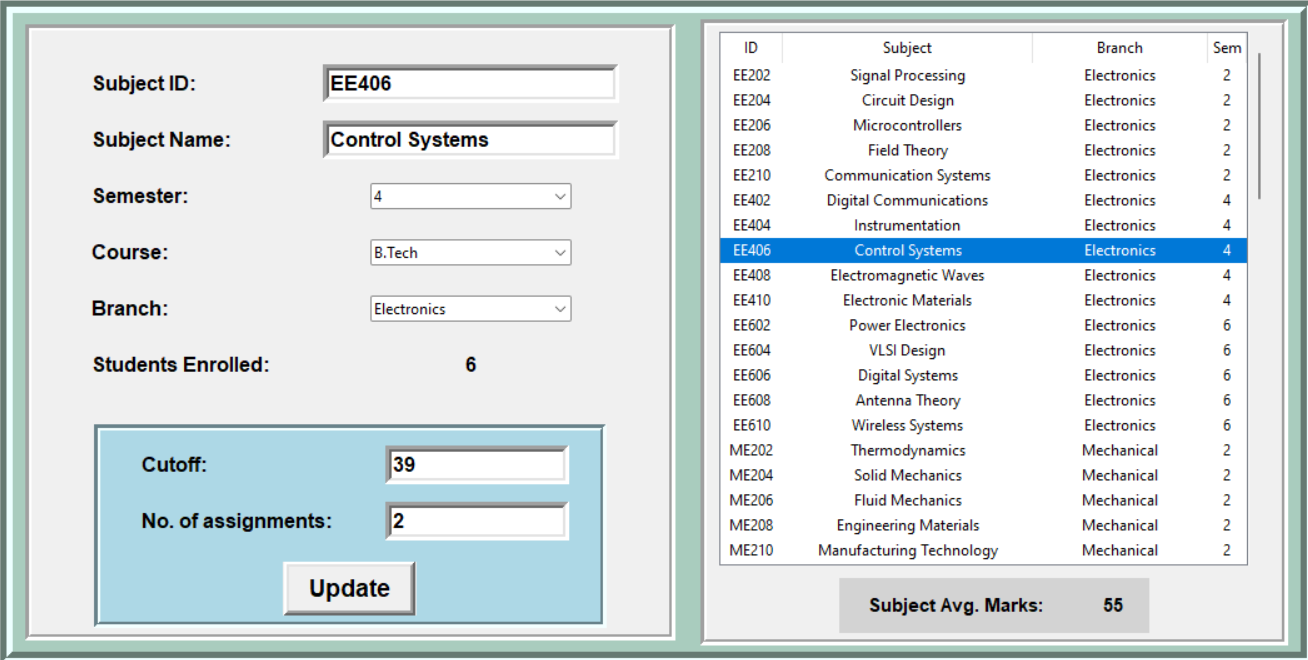
Screen 1: Grade Sheet display and Student selection



Screen 2: Student details viewing, updation, insertion and deletion



Screen 3: Subject details viewing, updation of cut-off



# PL/SQL

## Triggers

*Trigger on Student Table:*

CREATE OR REPLACE TRIGGER logStudent

BEFORE INSERT OR UPDATE OR DELETE ON Student

FOR EACH ROW

BEGIN

CASE

WHEN INSERTING THEN

    INSERT INTO LogStudentChange

    VALUES(SYSDATE,:NEW.StudentID, :NEW.FirstName, :NEW.LastName, :NEW.Address, :NEW.DOB, :NEW.Mobile, :NEW.Email);

WHEN UPDATING OR DELETING THEN

    INSERT INTO LogStudentChange

    VALUES(SYSDATE,:OLD.StudentID, :OLD.FirstName, :OLD.LastName, :OLD.Address, :OLD.DOB, :OLD.Mobile, :OLD.Email);

END CASE;

END;

/

*Trigger on Subject Table:*

CREATE OR REPLACE TRIGGER logSubject

BEFORE UPDATE OF Cutoff, NumOfAssignments ON Subject

FOR EACH ROW

BEGIN

CASE

WHEN UPDATING THEN

    INSERT INTO LogSubjectChange

    VALUES(SYSDATE,:OLD.SubjectID, :OLD.Cutoff, :OLD.NumOfAssignments);

END CASE;

END;

/

## Functions

*Function to calculate total marks:*

CREATE OR REPLACE FUNCTION calcTotMarks(p\_student\_id IN VARCHAR2) RETURN NUMBER IS

  v\_marks NUMBER(10);

  v\_tot\_marks NUMBER(10) := 0;

  CURSOR c\_marks IS

    SELECT ObtainedMarks FROM Marks WHERE StudentID = p\_student\_id;

BEGIN

  FOR c IN c\_marks LOOP

    v\_marks := c.ObtainedMarks;

    v\_tot\_marks := v\_tot\_marks + v\_marks;

  END LOOP;

  RETURN v\_tot\_marks;

END;

/

*Function to calculate pass or fail:*

CREATE OR REPLACE FUNCTION calcStatus(p\_student\_id IN VARCHAR2) RETURN NUMBER IS

    v\_marks NUMBER(10);

    v\_cutoff NUMBER(10) := 0;

    v\_sub\_id VARCHAR2(10);

    res NUMBER := 1;

    CURSOR c\_marks IS

    SELECT \* FROM Marks WHERE StudentID = p\_student\_id;

BEGIN

    FOR c IN c\_marks LOOP

        v\_marks := c.ObtainedMarks;

        v\_sub\_id := c.SubjectID;

        SELECT Cutoff INTO v\_cutoff FROM Subject WHERE SubjectID = v\_sub\_id;

        IF v\_marks < v\_cutoff THEN

            res := 0;

        END IF;

    END LOOP;

    RETURN res;

END;

/

# DB connectivity

*Database connection through cx\_Oracle:*

import cx\_Oracle

cx\_Oracle.init\_oracle\_client(lib\_dir=r"path\to\oracle\21c\binaries")

dsn\_tns = cx\_Oracle.makedsn('localhost', '1521', service\_name='xe')

conn = cx\_Oracle.connect(user=r'system', password='<redacted>', dsn=dsn\_tns)

cursor = conn.cursor()

cursor.execute(<query statement>)

*PL/SQL calls through cx\_Oracle:*

var = cursor.callfunc("function\_name", return-type, [“param1”, “param2”])

*Data access through cx\_Oracle:*

cursor.execute(f"SELECT col1, col2 FROM table WHERE col3='{value}'")

rows=cursor.fetchall()

# References

<https://cx-oracle.readthedocs.io/en/latest/>

<https://docs.python.org/3/library/tk.html>