**A yellow and red logo

Description automatically generated**

**Department of Information and Communication Technology**

**Faculty of Technology**

**University of Ruhuna**

**Database Management Systems Practicum**

**ICT 1222**

**Assignment 02 – Mini Project Group**

**Group Number 05**

|  |
| --- |
| TG/2021/1036 - P.N.N.K.Fernando |
| TG/2021/1041 - K.R.S.N.Bandara |
| TG/2021/1086 - D.G.P.T.Amamadu |
| TG/2021/1047 - G.P.K.Gurathna |

Submitted to: Mr.P.H.P. Nuwan Laksiri

Submitted by:

Contents

[Brief introduction about the problem/group project 3](#_Toc149495232)

[Brief introduction to the solution 3](#_Toc149495233)

[Proposed ER/EER diagram 4](#_Toc149495234)

[Proposed Relational mapping Diagram. 5](#_Toc149495235)

[Table structure of solution 6](#_Toc149495236)

[Tools and technologies that you have used. 9](#_Toc149495237)

[Security measures that you have taken to protect your Database. 9](#_Toc149495238)

[Brief description about DB Accounts/Users and the reasons for creating such Accounts/Users. 9](#_Toc149495239)

[Code snippets to support your work. 10](#_Toc149495240)

[Problems that you faced during the development of the solution. 11](#_Toc149495241)

[Solutions/how you have overcome the above identified problems. 11](#_Toc149495242)

[New database technologies/trends that you have used to develop the backend. 11](#_Toc149495243)

[If you are going to host your backend, where are you going to host it and reasons for the selection. 11](#_Toc149495244)

[If you are going to host your backend in a cloud environment what are the things/changes that you have to do in your backend. 11](#_Toc149495245)

[Individual contribution to the backend development. 12](#_Toc149495246)

[References 12](#_Toc149495247)

# Brief introduction about the problem/group project

The adoption of a Learning Management System (LMS) within the hospital setting has ushered in a new era of streamlined data management, addressing several critical issues previously associated with manual systems.

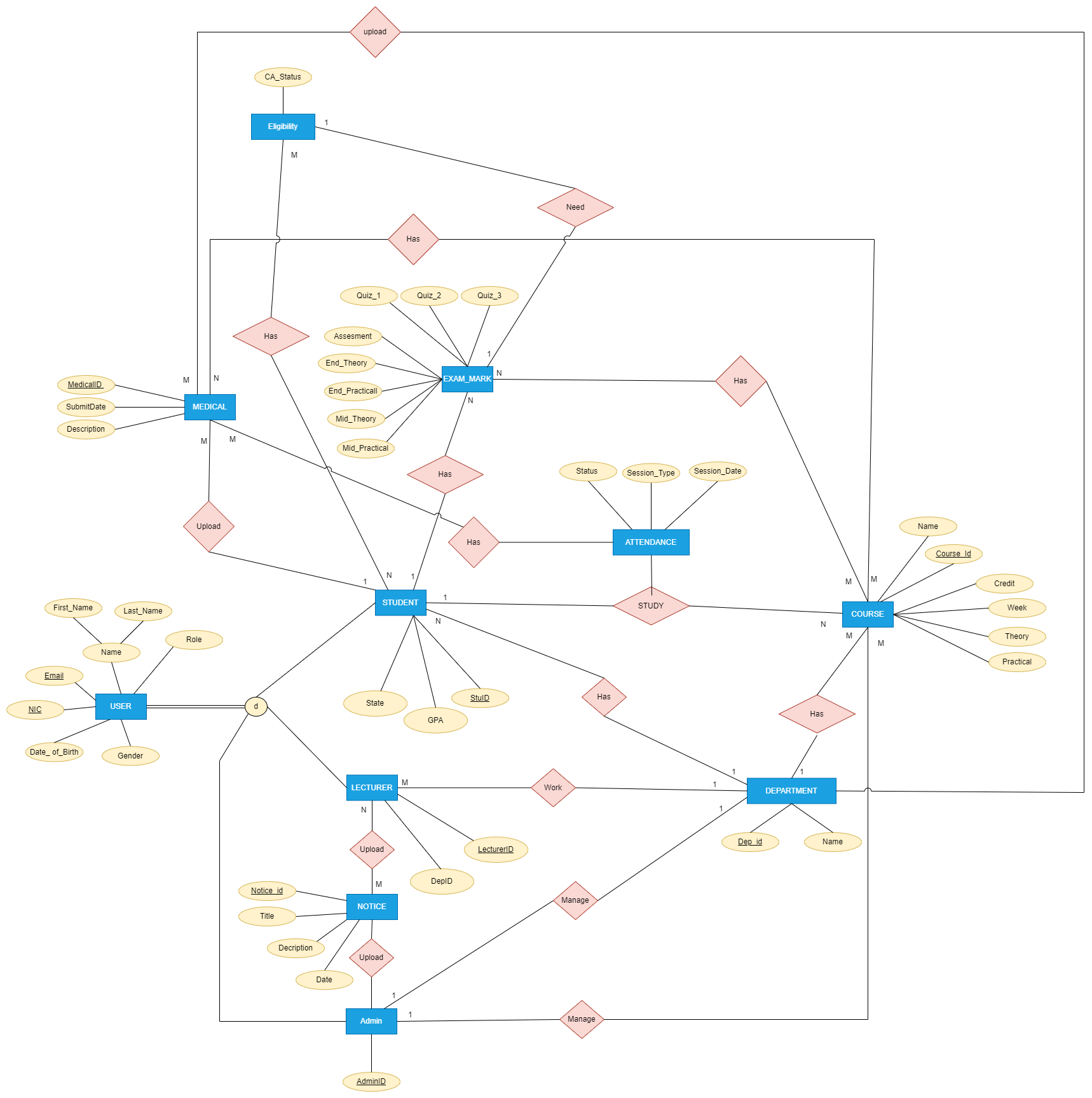
* **Efficient Data Entry and Updates:** With the LMS, the need for direct human involvement in data entry and updates has been significantly reduced.
* **Redundancy Elimination:** The LMS effectively identifies and eliminates data redundancies, reducing the chances of data duplication and ensuring data integrity.
* **Rapid Data Access:** Emergency data retrieval is now swift and efficient. The LMS allows for quick access to necessary information, eliminating the need for manual file searches.
* **Space Optimization:** The vast physical space previously allocated for data file storage has been freed up. The digital nature of the LMS minimizes physical storage requirements.
* **Data Security and Backups:** Data security has been significantly enhanced. The LMS provides robust security measures, including access controls.

# Brief introduction to the solution

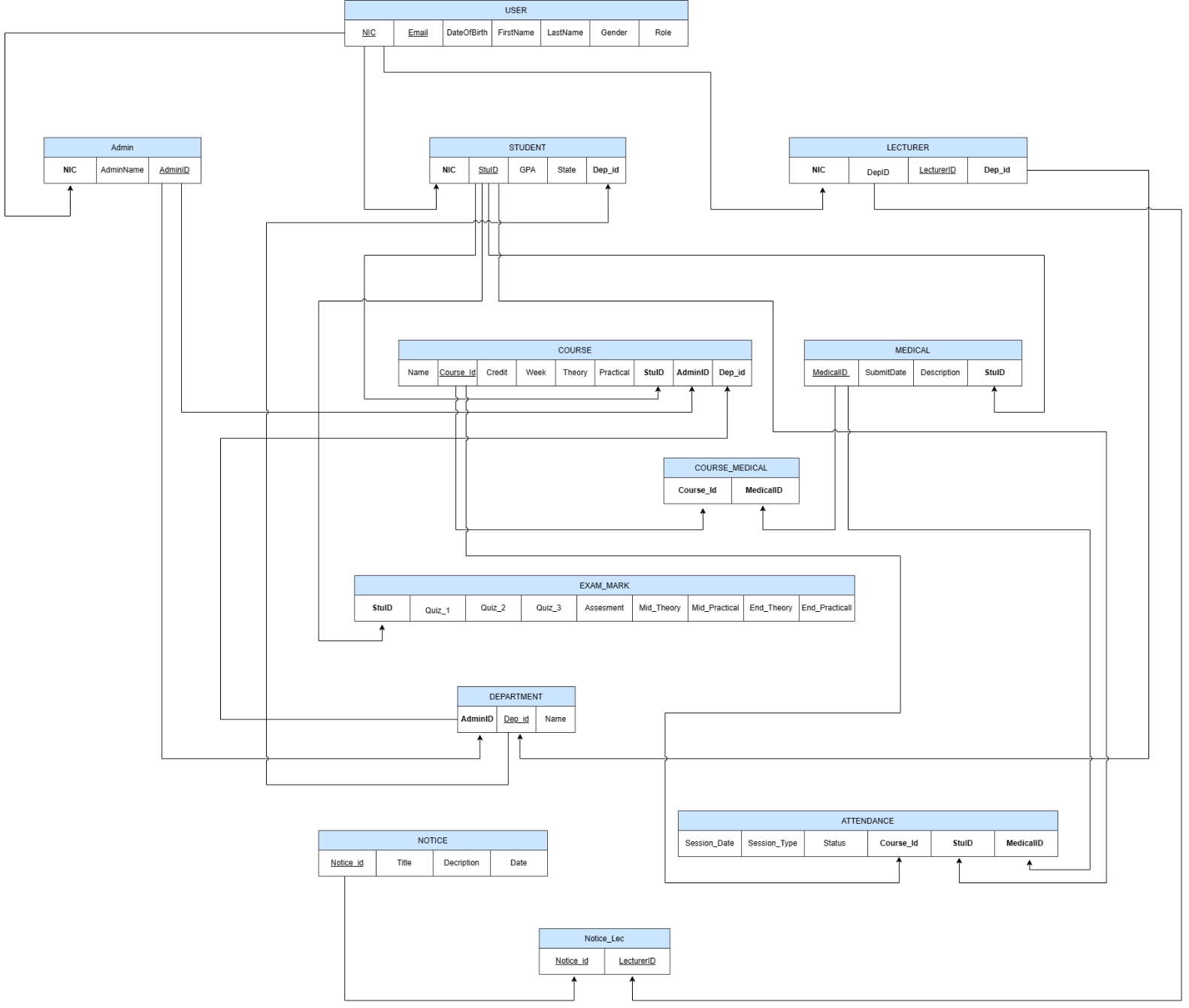
To avoid these problems, we developed and introduced an automated database management system. In here data entering and updating and retrieving is very easy when comparing these Manual and these new systems. In that system we use primary keys to avoid data redundancies. No data duplications and no space wasting happen.

When they want emergency data they can search and find by using key words. It is very quickly. No need to wait so long. To store all kind of data there should be only computer system with enough storage. No need more physical space to store like a manual system. If any data is lost easily can recover it because of backup facilities. In that data base management system does not allow to enter invalid, irrelevant data or incorrect data. System can identify those anomalies. If there is any essential data system not allowed to miss out it. All the data processing is happed in automatable way. There is no need to involve humans for that. Data accuracy is high. Automated systems can limit access authority. Only authorized people can access and operate the system. So, in this kind of systems security is high. The risk of data destruction is minimal. Few people can handle this system very easily. Lower operating cost and labor is low.

# Proposed ER/EER diagram



# Proposed Relational mapping Diagram.



# Table structure of solution

A black screen with white text

Description automatically generated

|  |  |
| --- | --- |
| User | |
| NIC | INT |
| Email | VARCHAR(50) |
| FirstName | VARCHAR(20) |
| LastName | VARCHAR(20) |
| Date Of Birth | DATE |
| Roles | VARCHAR(20) |
| Gender | CHAR(1) |

|  |  |
| --- | --- |
| Student | |
| stuID | VARCHAR(20), |
| States | VARCHAR(20), |
| NIC | INT |

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

|  |  |
| --- | --- |
| Admin | |
| adminID | VARCHAR(20) |
| NIC | INT |

|  |  |
| --- | --- |
| Department | |
| DepID | VARCHAR(20) |
| name | VARCHAR(20) |

A black screen with white text

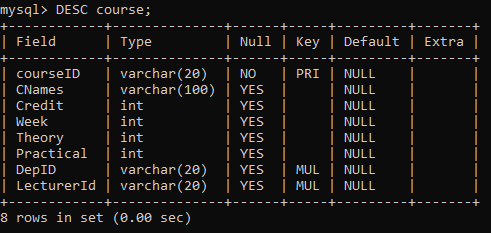
Description automatically generated

|  |  |
| --- | --- |
| Lecturer | |
| LecturerID | VARCHAR(20) |
| NIC | INT |
| DepID | VARCHAR(20) |

A screenshot of a computer screen

Description automatically generated

|  |  |
| --- | --- |
| Course | |
| CourseID | VARCHAR(20) |
| Cnames | VARCHAR(100) |
| Credit | INT |
| Week | INT |
| Theory | INT |
| Practical | INT |
| DepID | VARCHAR(20) |
| LECTURERID | VARCHAR(20) |



|  |  |
| --- | --- |
| Exammarks | |
| Quiz1 | INT |
| Quiz2 | VARCHAR(100) |
| Quiz3 | INT |
| MidPractical | INT |
| MidTheory | INT |
| EndPractical | INT |
| EndTheory | VARCHAR(20) |
| Assesment | INT |
| StuID | VARCHAR(20) |
| CourseID | VARCHAR(20) |
| DepID | VARCJAR(20) |

A screen shot of a computer

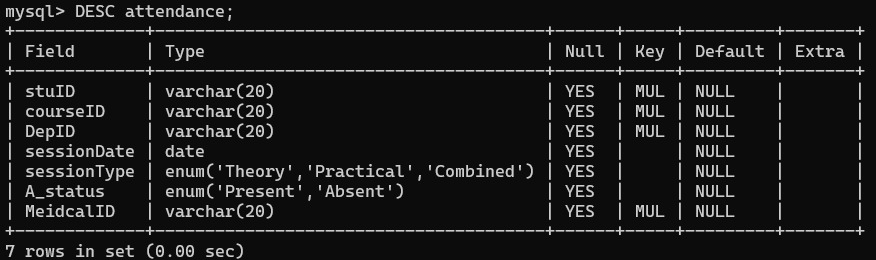
Description automatically generated

A black screen with white text

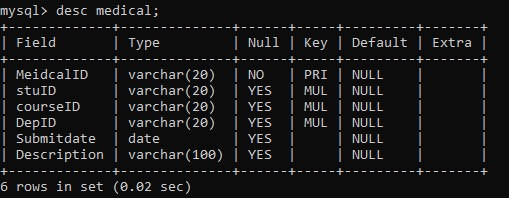
Description automatically generated

|  |  |
| --- | --- |
| Notice | |
| NoticeID | VARCHAR(20) |
| Title | VARCHAR(20) |
| Description | VARCHAR(100) |
| Date | Date |
| LecturerID | VARCHAR(20) |

|  |  |
| --- | --- |
| Attendance | |
| StuID | VARCHAR(20) |
| CourseID | VARCHAR(20) |
| DepID | VARCHAR(20) |
| SessionDate | Date |
| SessionType | ENUM(‘Theory’,’Practical’,’Combined’) |
| A\_status | ENUM(‘Present’,’Absent’) |
| MedicalID | VARCHAR(20) |



|  |  |
| --- | --- |
| Medical | |
| MedicalID | VARCHAR(20) |
| StuID | VARCHAR(20) |
| CourseID | VARCHAR(20) |
| DepID | VARCHAR(20) |
| Submitdate | Date |
| Description | VARCHAR(100) |



|  |  |
| --- | --- |
| Eligibility | |
| CAStatus | VARCHAR(20) |
| CourseID | VARCHAR(20) |
| StuID | VARCHAR(20) |

A black screen with white text

Description automatically generated

|  |  |
| --- | --- |
| Course | |
| CourseID | VARCHAR(20) |
| CNames | VARCHAR(100) |
| Credit | INT |
| Week | INT |
| Theory | INT |
| Practical | INT |
| DepID | VARCHAR(20) |
| LecturerID | VARCHAR(20) |

A black screen with white text

Description automatically generated

# Tools and technologies that you have used.

Draw.io:

* Used to draw ER diagram, relational schema, and table structure.

MySQL SERVER, VS code, Notepad:

* Used to create database and maintain.

GitHub and GitHub Desktop:

* Version Control

# Security measures that you have taken to protect your Database.

* Admin - With All privileges with Grant Option for all the tables in the database
* Dean - With All privileges without Grant for all the tables in the database
* Lecturer – All privileges without Grant and user creation for all the tables in the database
* Technical Officer - Read, write, and update permissions for attendance related tables/views.
* Student - Read permission for final attendance and final marks/Grades tables/views.

# Brief description about DB Accounts/Users and the reasons for creating such Accounts/Users.

Our Learning Management System contain below user accounts:

* Admin –
  + Admin can access full system.
  + Maintain database.
* Dean –
  + Can Access any tables and review.
  + Update tables and data.
* Lecturer –
  + Access course table and exammarks table and maintain.
  + Can Update table.
* Technical Officer –
  + maintain Read, write and update permissions for attendance related tables/views.
* Student –
  + Access course tables and can only view marks.

# Code snippets to support your work.

-- Create the StudentQuiz\_Marks view

CREATE VIEW StudentQuiz\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

        ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 AS `Five\_Percent\_Average\_Quiz\_Score`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentAssessment\_Marks view with a 5% adjusted assessment mark

CREATE VIEW StudentAssessment\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

    (e.Assesments \* 0.05) AS `Adjusted\_Assessment\_Mark`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentMidterm\_Marks view with 20% weighted MidPractical and MidTheory scores

CREATE VIEW StudentMidterm\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

    ((e.MidTheory + e.MidPractical) \* 0.20) AS `MidMark`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentCA\_Marks view  level 1

CREATE VIEW StudentCA\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

             ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 AS `Quiz\_Score` ,

             (e.Assesments \* 0.05) AS `Assessment\_Mark` ,

             ((e.MidTheory + e.MidPractical) \* 0.20) AS `MidMark`

     FROM exammarks e

     INNER JOIN student s ON e.stuid = s.stuid

     WHERE s.states != 'suspended';

-- Create the StudentCA\_Marks view  level 2

SELECT e.stuid, e.courseid, e.depid,

       ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 AS `Quiz\_Score`,

       (e.Assesments \* 0.05) AS `Assessment\_Mark`,

       ((e.MidTheory + e.MidPractical) \* 0.20) AS `MidMark`,

       ((

           ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 +

           (e.Assesments \* 0.05) +

           ((e.MidTheory + e.MidPractical) \* 0.20)

       )) as `CA\_Marks`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentCA\_Marks view

CREATE VIEW StudentCA\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

    ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 AS `Quiz\_Score`,

    (e.Assesments \* 0.05) AS `Assessment\_Mark`,

    ((e.MidTheory + e.MidPractical) \* 0.20) AS `MidMark`,

    (

        ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 +

        (e.Assesments \* 0.05) +

        ((e.MidTheory + e.MidPractical) \* 0.20)

    ) as `CA\_Marks`,

    CASE

        WHEN (

            ((e.quiz1 + e.quiz2 + e.quiz3 - LEAST(e.quiz1, LEAST(e.quiz2, e.quiz3))) / 2) \* 0.05 +

            (e.Assesments \* 0.05) +

            ((e.MidTheory + e.MidPractical) \* 0.20)

        ) >= 15 THEN 'Eligible'

        ELSE 'Not\_Eligible'

    END AS `Eligibility\_Status`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentEnd\_Marks view with a 70% adjusted assessment mark

CREATE VIEW StudentEnd\_Marks AS

SELECT e.stuid, e.courseid, e.depid,

    ((e.EndPractical  + e.EndTheory ) \* 0.70) AS `EndMark`

FROM exammarks e

INNER JOIN student s ON e.stuid = s.stuid

WHERE s.states != 'suspended';

-- Create the StudentFinal\_Marks view

create view StudentFinal\_Marks AS

select c.stuid, c.courseid, c.depid, (c.CA\_Marks + e.EndMark) as FinalMark

from studentca\_marks c, StudentEnd\_Marks e , attendance\_eligibility a

where c.Eligibility\_Status = 'Eligible' and a.Eligiblity = 'Eligible' and

c.stuid = e.stuid and

c.courseid = e.courseid and

c.depid = e.depid and

a.stuid = e.stuid and

a.stuid = c.stuid and

a.courseid = e.courseid and

a.courseid = c.courseid ;

-- create the StudentGradePoint view

create view StudentGradePoint AS

select  stuid, courseid, depid ,

   case

      when(FinalMark>= 85 and FinalMark<= 100) then 'A+'

      when(FinalMark>= 80 and FinalMark<= 84.999999) then 'A'

      when(FinalMark>= 75 and FinalMark<= 79.999999) then 'A-'

      when(FinalMark>= 70 and FinalMark<= 74.999999) then 'B+'

      when(FinalMark>= 65 and FinalMark<= 69.999999) then 'B'

      when(FinalMark>= 60 and FinalMark<= 64.999999) then 'B-'

      when(FinalMark>= 55 and FinalMark<= 59.999999) then 'C+'

      when(FinalMark>= 50 and FinalMark<= 54.999999) then 'C'

      when(FinalMark>= 45 and FinalMark<= 49.999999) then 'C-'

      when(FinalMark>= 40 and FinalMark<= 44.999999) then 'D+'

      when(FinalMark>= 35 and FinalMark<= 39.999999) then 'D'

      when(FinalMark>= 0 and FinalMark<= 34.999999) then 'F'

      ELSE 'Not\_Valid'

   end   AS 'Grade' ,

  case

      when(FinalMark>= 85 and FinalMark<= 100) then 4.0

      when(FinalMark>= 80 and FinalMark<= 84.999999) then 4.0

      when(FinalMark>= 75 and FinalMark<= 79.999999) then 3.7

      when(FinalMark>= 70 and FinalMark<= 74.999999) then 3.3

      when(FinalMark>= 65 and FinalMark<= 69.999999) then 3.0

      when(FinalMark>= 60 and FinalMark<= 64.999999) then 2.7

      when(FinalMark>= 55 and FinalMark<= 59.999999) then 2.3

      when(FinalMark>= 50 and FinalMark<= 54.999999) then 2.0

      when(FinalMark>= 45 and FinalMark<= 49.999999) then 1.7

      when(FinalMark>= 40 and FinalMark<= 44.999999) then 1.3

      when(FinalMark>= 35 and FinalMark<= 39.999999) then 1.0

      when(FinalMark>= 0 and FinalMark<= 34.999999) then 0

      ELSE 'Not\_Valid'

   end   AS 'Grade\_Point'

from studentfinal\_marks;

-- create the StudentGradePoint view with repeater marks reduced

create view StudentGradePoint AS

SELECT

    stuid,

    courseid,

    depid,

    CASE

        WHEN stuid LIKE 'TG/2021/%' THEN

            CASE

                WHEN FinalMark >= 85 AND FinalMark <= 100 THEN 'A+'

                WHEN FinalMark >= 80 AND FinalMark <= 84.999999 THEN 'A'

                WHEN FinalMark >= 75 AND FinalMark <= 79.999999 THEN 'A-'

                WHEN FinalMark >= 70 AND FinalMark <= 74.999999 THEN 'B+'

                WHEN FinalMark >= 65 AND FinalMark <= 69.999999 THEN 'B'

                WHEN FinalMark >= 60 AND FinalMark <= 64.999999 THEN 'B-'

                WHEN FinalMark >= 55 AND FinalMark <= 59.999999 THEN 'C+'

                WHEN FinalMark >= 50 AND FinalMark <= 54.999999 THEN 'C'

                WHEN FinalMark >= 45 AND FinalMark <= 49.999999 THEN 'C-'

                WHEN FinalMark >= 40 AND FinalMark <= 44.999999 THEN 'D+'

                WHEN FinalMark >= 35 AND FinalMark <= 39.999999 THEN 'D'

                WHEN FinalMark >= 0 AND FinalMark <= 34.999999 THEN 'F'

                ELSE 'Not\_Valid'

            END

        ELSE

            CASE

                WHEN FinalMark >= 85 AND FinalMark <= 100 THEN 'C'

                WHEN FinalMark >= 80 AND FinalMark <= 84.999999 THEN 'C'

                WHEN FinalMark >= 75 AND FinalMark <= 79.999999 THEN 'C'

                WHEN FinalMark >= 70 AND FinalMark <= 74.999999 THEN 'C'

                WHEN FinalMark >= 65 AND FinalMark <= 69.999999 THEN 'C'

                WHEN FinalMark >= 60 AND FinalMark <= 64.999999 THEN 'C'

                WHEN FinalMark >= 55 AND FinalMark <= 59.999999 THEN 'C'

                WHEN FinalMark >= 50 AND FinalMark <= 54.999999 THEN 'C'

                WHEN FinalMark >= 45 AND FinalMark <= 49.999999 THEN 'C-'

                WHEN FinalMark >= 40 AND FinalMark <= 44.999999 THEN 'D+'

                WHEN FinalMark >= 35 AND FinalMark <= 39.999999 THEN 'D'

                WHEN FinalMark >= 0 AND FinalMark <= 34.999999 THEN 'F'

                ELSE 'Not\_Valid'

            END

    END AS 'Grade',

    CASE

        WHEN stuid LIKE 'TG/2021/%' THEN

            CASE

                WHEN FinalMark >= 85 AND FinalMark <= 100 THEN 4.0

                WHEN FinalMark >= 80 AND FinalMark <= 84.999999 THEN 4.0

                WHEN FinalMark >= 75 AND FinalMark <= 79.999999 THEN 3.7

                WHEN FinalMark >= 70 AND FinalMark <= 74.999999 THEN 3.3

                WHEN FinalMark >= 65 AND FinalMark <= 69.999999 THEN 3.0

                WHEN FinalMark >= 60 AND FinalMark <= 64.999999 THEN 2.7

                WHEN FinalMark >= 55 AND FinalMark <= 59.999999 THEN 2.3

                WHEN FinalMark >= 50 AND FinalMark <= 54.999999 THEN 2.0

                WHEN FinalMark >= 45 AND FinalMark <= 49.999999 THEN 1.7

                WHEN FinalMark >= 40 AND FinalMark <= 44.999999 THEN 1.3

                WHEN FinalMark >= 35 AND FinalMark <= 39.999999 THEN 1.0

                WHEN FinalMark >= 0 AND FinalMark <= 34.999999 THEN 0

                ELSE 'Not\_Valid'

            END

        ELSE

            CASE

                WHEN FinalMark >= 85 AND FinalMark <= 100 THEN 2.0

                WHEN FinalMark >= 80 AND FinalMark <= 84.999999 THEN 2.0

                WHEN FinalMark >= 75 AND FinalMark <= 79.999999 THEN 2.0

                WHEN FinalMark >= 70 AND FinalMark <= 74.999999 THEN 2.0

                WHEN FinalMark >= 65 AND FinalMark <= 69.999999 THEN 2.0

                WHEN FinalMark >= 60 AND FinalMark <= 64.999999 THEN 2.0

                WHEN FinalMark >= 55 AND FinalMark <= 59.999999 THEN 2.0

                WHEN FinalMark >= 50 AND FinalMark <= 54.999999 THEN 2.0

                WHEN FinalMark >= 45 AND FinalMark <= 49.999999 THEN 1.7

                WHEN FinalMark >= 40 AND FinalMark <= 44.999999 THEN 1.3

                WHEN FinalMark >= 35 AND FinalMark <= 39.999999 THEN 1.0

                WHEN FinalMark >= 0 AND FinalMark <= 34.999999 THEN 0

                ELSE 'Not\_Valid'

            END

    END AS 'Grade\_Point'

FROM studentfinal\_marks;

-- check attendance eligibility

CREATE VIEW attendance\_eligibility AS

SELECT stuID, courseID,sessionType, COUNT(A\_status)/15 \* 100 AS "80% Percentage" , IF((COUNT(A\_status)/15 \* 100)>=80,"Eligible","Not Eligible") As "Eligiblity"

FROM attendance

WHERE A\_status = "Present"

GROUP BY stuID, courseID,sessionType;

-- Create View retrive all notice

CREATE VIEW notices AS

SELECT \* FROM notice ORDER BY Date DESC;

CREATE PROCEDURE SubjectAttendance(

    IN p\_stuID VARCHAR(20),

    IN p\_courseID VARCHAR(20)

)

BEGIN

    SELECT stuID, courseID, DepID, sessionDate, sessionType, A\_status, MeidcalID

    FROM attendance

    WHERE stuID = p\_stuID AND courseID = p\_courseID;

END;//

-- call PROCEDURE

CALL SubjectAttendance('TG/2021/1011', 'ENG1222');

-- Attendance Percentage PROCEDURE

CREATE PROCEDURE CalculateAttendancePercentage(

    IN p\_stuID VARCHAR(20),

    IN p\_courseID VARCHAR(20),

    IN p\_DepID VARCHAR(20)

)

BEGIN

    SELECT

        stuID,

        courseID,

        A\_status,

        (COUNT(A\_status) / 15 \* 100) AS Percentage

    FROM

        attendance

    WHERE

        stuID = p\_stuID

        AND courseID = p\_courseID

        AND DepID = p\_DepID

    GROUP BY stuID, courseID, A\_status;

END;

-- call PROCEDURE

CALL CalculateAttendancePercentage('TG/2021/1011', 'ENG1222','DpICT');

# Problems that you faced during the development of the solution.

* Difficult to creation of attendance table, exam mark table, hard to find suitable foreign keys.
* Maintain attendance table data and marks table data.
* We want to change ER diagram and Relational map.
* When we add data to tables there are some errors of data types.

# Solutions/how you have overcome the above identified problems.

* Attendance table data set is separate to weekday files.
* Attendance data set type repetitive tying.

# New database technologies/trends that you have used to develop the backend.

* MySQL
* MySQL Workbench
* Visual Studio Code
* Notepad ++
* GitHub

# If you are going to host your backend, where are you going to host it and reasons for the selection.

* + Anyone can access the database in any place.
  + Real time accessing.
  + Easy to maintain database.

# If you are going to host your backend in a cloud environment what are the things/changes that you have to do in your backend.

* + We want to buy a database server and creating data base into the online server.
  + Change some data types.
  + We have a lots of data in our database.

# Individual contribution to the backend development.

# References

Lecture Notes

W3Schools.com