UNIVERSITY of GREENWICH

Alliance with FPT Education

ASSIGNMENT 2 FRONT SHEET

Qualification	**TEC Level 5 HND Diploma in Computing**		
Unit number and title	**Unit 04: Database Design & Development**		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Nguyen Quoc Anh	**Student ID**	GCH18888
Class	GCH0718	**Assessor name**	

Student declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P2	P3	P4	P5	M2	M3	M4	M5	D2	D3
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

❒ Summative Feedback:	❒ Resubmission Feedback:
2.1	

Grade:	Assessor Signature:	Date:
Signature & Date:		

UNIVERSITY of GREENWICH
Alliance with FPT Education

Contents

I. Implementation (P2)

1. Code snippets to create each table and insert some sample data for each table

Student Account

```
CREATE TABLE StudentAccount
(
    Student_Email nVARCHAR(50) PRIMARY KEY CHECK (Student_Email LIKE '%@fpt.edu.vn'),
    Student_Password varchar(50) NOT NULL
)
```

Teacher Account

```
CREATE TABLE TeacherAccount
(
    Teacher_Email nVARCHAR(50) PRIMARY KEY CHECK (Teacher_Email LIKE '%@fe.edu.vn'),
    Teacher_Password varchar(50) NOT NULL
)
```

Example data:

```
INSERT StudentAccount
VALUES
('namvhgch20002@fpt.edu.vn','123456789'),
('anhnqgch20001@fpt.edu.vn','123456789'),
('minhnngch20003@fpt.edu.vn','123456789'),
('phonglhgch20004@fpt.edu.vn','123456789'),
('hoanhgch20005@fpt.edu.vn','123456789'),
('namnpgch20006@fpt.edu.vn','123456789'),
('hungtmgbh20007@fpt.edu.vn','123456789'),
('trangnhgch20008@fpt.edu.vn','123456789'),
('anhttgch20009@fpt.edu.vn','123456789'),
('phinxgch20010@fpt.edu.vn','123456789'),
('ducnhgch20011@fpt.edu.vn','123456789'),
('duyetvagch20012@fpt.edu.vn','123456789'),
('chinhgch20013@fpt.edu.vn','123456789'),
('phungmpgch20014@fpt.edu.vn','123456789')

INSERT TeacherAccount
VALUES
('quandh@fe.edu.vn','123456789'),
('hienpm@fe.edu.vn','123456789'),
('oanhnv@fe.edu.vn','123456789'),
('anvd@fe.edu.vn','123456789'),
('datnq@fe.edu.vn','123456789'),
('yenlh@fe.edu.vn','123456789'),
('minhn@fe.edu.vn','123456789')
```

UNIVERSITY of GREENWICH

Alliance with FPT Education

Student

Create table

```
CREATE TABLE Student
(
    Student_ID varchar(50) PRIMARY KEY CHECK (Student_ID LIKE  ('G[B,C,D]H[0-9][0-9][0-9][0-9][0-9]')),
    Student_Lname   Nvarchar(50) Not null, CHECK (Student_Lname Like '%[^0-9]%'),
    Student_Fname   Nvarchar(50) Not null, CHECK (Student_Lname Like '%[^0-9]%'),
    Student_Email   Nvarchar(50)    Not null UNIQUE,
    Student_Phone   Char(11)    Not null unique, CHECK (Student_phone Not like '%[^0-9]%'),
    Student_Sex Varchar(50) Not null, CHECK (Student_Sex in ('Male','Female')),
    Student_Bdate   Date,
    Major   Nvarchar(50)    Not Null, CHECK (Major in ('GCH','GBH','GDH')),
    Image   Nvarchar(50),
    CONSTRAINT fk_student_account FOREIGN KEY (Student_Email) REFERENCES dbo.StudentAccount(Student_Email)


)
```

Example data

```
INSERT dbo.Student
(
  Student_ID,
    Student_Lname,
    Student_Fname,
    Student_Email,
    Student_Phone,
    Student_Sex,
    Student_Bdate,
    Major,
    Image
)
VALUES
(    'GCH20001',         -- Student_ID - varchar(50)
    N'Anh',        -- Student_Lname - nvarchar(50)
    N'Nguyễn',        -- Student_Fname - nvarchar(50)
    N'anhnqgch20001@fpt.edu.vn',        -- Student_Email - nvarchar(50)
    '0966141598',         -- Student_Phone - char(11)
    'Male',         -- Student_Sex - varchar(50)
   '1998-5-21', -- Student_Bdate - date
    N'GCH',        -- Major - nvarchar(50)
    N''         -- Image - nvarchar(50)
    )
```

```
INSERT student
VALUES
('GCH20002',N'Nam',N'Vũ Hải', 'namvhgch20002@fpt.edu.vn','0966143422', 'Male', '2000-4-24','GCH',''),
('GCH20003',N'Minh',N'Nguyễn Nhật', 'minhnngch20003@fpt.edu.vn','0945671521', 'Male', '2000-3-5','GCH',''),
('GCH20004',N'Phong',N'Lê Hồng', 'phonglhgch20004@fpt.edu.vn','0966121875', 'Male', '2000-7-12','GCH',''),
('GCH20005',N'Hòa',N'Nguyễn Huy', 'hoanhgch20005@fpt.edu.vn','0966141521', 'Male', '2000-11-7','GCH',''),
('GCH20006',N'Nam',N'Nguyễn Phương', 'namnpgch20006@fpt.edu.vn','0955232524', 'Male', '2000-6-18','GCH',''),
('GBH20007',N'Hưng',N'Trịnh Mạnh', 'hungtmgbh20007@fpt.edu.vn','0912859982', 'Male', '1998-12-25','GBH',''),
('GBH20008',N'Trang',N'Nguyễn Huyền', 'trangnhgch20008@fpt.edu.vn','0345239821', 'Female', '1998-1-1','GBH',''),
('GBH20009',N'Anh',N'Trịnh Thu', 'anhttgch20009@fpt.edu.vn','0339231234', 'Female', '1998-5-12','GBH',''),
('GBH20010',N'Phi',N'Ngô Xuân', 'phinxgch20010@fpt.edu.vn','0336989623', 'Male', '1998-4-17','GBH',''),
('GBH20011',N'Đức',N'Nguyễn Huy', 'ducnhgch20011@fpt.edu.vn','0912560942', 'Male', '1998-12-1','GBH',''),
('GBH20012',N'Duyệt',N'Vũ An', 'duyetvagch20012@fpt.edu.vn','0398761234', 'Male', '2000-4-22','GBH',''),
('GDH20013',N'Chi',N'Nguyễn Hà', 'chinhgch20013@fpt.edu.vn','0965721998', 'Female', '1998-11-13','GDH',''),
('GDH20014',N'Phương',N'Phùng Minh', 'phungmpgch20014@fpt.edu.vn','0944491923', 'Female', '1998-5-12','GDH','')
```

Class

Create table

```
CREATE TABLE Class
(
    Class_ID    Nvarchar(50)    Primary key, CHECK (Class_ID LIKE ('G[B,C,D]H[0-9][0-9][0-9][0-9]')),
    Class_Name  Varchar(20),
    Class_Date  date Not null
)
```

Example data

```
INSERT dbo.Class
(
      Class_ID,
      Class_Name,
      Class_Date
)
VALUES
(     'GCH0717',        -- Class_ID - nvarchar(50)
      '',           -- Class_Name - varchar(20)
      GETDATE()  -- Class_Date - date
      ),
(     'GCH0718',        -- Class_ID - nvarchar(50)
      '',           -- Class_Name - varchar(20)
      GETDATE()  -- Class_Date - date
      ),
(     'GCH0719',        -- Class_ID - nvarchar(50)
      '',           -- Class_Name - varchar(20)
      GETDATE()  -- Class_Date - date
      )
```

```
INSERT Class
VALUES
(   'GDH0724','',GETDATE()),
(   'GBH0725','',GETDATE()),
(   'GBH0727','',GETDATE())
```

Subject

Create table

```
CREATE TABLE Subject
(
    Subject_ID  Int Primary Key,
    Subject_Name    Varchar(50) Not null unique,
    Subject_Slot    Int Not null,
    Subject_Description Varchar(50)
)
```

Example data

UNIVERSITY of GREENWICH

Alliance with FPT Education

```
INSERT dbo.Subject
(
    Subject_ID,
    Subject_Name,
    Subject_Slot,
    Subject_Description
)
VALUES
(   1618,  -- Subject_ID - int
    'Programming', -- Subject_Name - varchar(50)
    40,  -- Subject_Slot - int
    ''  -- Subject_Description - varchar(50)
    )
```

```
INSERT subject
VALUES
(1620, 'Professional Practice', 40,''),
(1622, 'Database Design & Development', 40,''),
(1623, 'Security', 40,''),
(1625, 'Managing a Successful Computing Project', 80,''),
(1631, 'Software Development Life Cycle', 40,''),
(3513, 'Contextual Studies', 40,''),
(3515, 'Techniques & Processes', 40,''),
(0485, 'Business and the Business Environment', 40,''),
(0486, 'Marketing Essentials', 40,''),
(0487, 'Human Resource Management', 40,''),
(0491, 'Managing a sucessful Business Project', 40,'')
```

Teacher

Create table

```
CREATE TABLE Teacher
(
    Teacher_ID  Nvarchar(50)    Primary Key,
    Teacher_Lname   Nvarchar(50)    Not null, CHECK (Teacher_Lname Like '%[^0-9]%'),
    Teacher_Fname   Nvarchar(50)    Not null, CHECK (Teacher_Lname Like '%[^0-9]%'),
    Teacher_Phone   Char(11)    Not null unique, CHECK (Teacher_phone Not like '%[^0-9]%'),
    Teacher_Email   nvarchar(50) Not null UNIQUE,
    Teacher_Bdate DATE,
    CONSTRAINT fk_teacher_account FOREIGN KEY (Teacher_Email) REFERENCES dbo.TeacherAccount(Teacher_Email)

)
```

Example data

```
INSERT dbo.Teacher
(
    Teacher_ID,
    Teacher_Lname,
    Teacher_Fname,
    Teacher_Phone,
    Teacher_Email,
    Teacher_Bdate
)
VALUES
(   N'1',      -- Teacher_ID - nvarchar(50)
    N'Quân',      -- Teacher_Lname - nvarchar(50)
    N'Đỗ Hồng',      -- Teacher_Fname - nvarchar(50)
    '0912312384',       -- Teacher_Phone - char(11)
    N'quandh@fe.edu.vn',      -- Teacher_Email - nvarchar(50)
    '1982-5-21' -- Teacher_Bdate - date
    )
```

```
INSERT Teacher
VALUES
(N'2',N'Hiền',N'Phùng Minh','0338958191','hienpm@fe.edu.vn','1960-5-25'),
(N'3',N'Oanh',N'Nguyễn Văn','0912560941','oanhnv@fe.edu.vn','1959-6-20'),
(N'4',N'An',N'Vũ Du','0989589182','anvd@fe.edu.vn','1980-3-17'),
(N'5',N'Đạt',N'Nguyễn Quốc','0344469896','datnq@fe.edu.vn','1990-5-23'),
(N'6',N'Yến',N'Lê Hải','0345324216','yenlh@fe.edu.vn','1990-10-25'),
(N'7',N'Minh',N'Nguyễn','0984177862','minhn@fe.edu.vn','1990-1-22')
```

Course

Create Table

```
CREATE TABLE Teacher
(
    Teacher_ID  Nvarchar(50)    Primary Key,
    Teacher_Lname   Nvarchar(50)    Not null, CHECK (Teacher_Lname Like '%[^0-9]%'),
    Teacher_Fname   Nvarchar(50)    Not null, CHECK (Teacher_Lname Like '%[^0-9]%'),
    Teacher_Phone   Char(11)    Not null unique, CHECK (Teacher_phone Not like '%[^0-9]%'),
    Teacher_Email   nvarchar(50) Not null unique,
    Teacher_Bdate date

)
```

Example data

```
INSERT dbo.Course
(
    Class_ID,
    Teacher_ID,
    Subject_ID,
    start_date
)
VALUES
(   'GCH0717',       -- Class_ID - varchar(50)
    N'1',      -- Teacher_ID - nvarchar(50)
    1618,        -- Subject_ID - int
    GETDATE() -- start_date - date
    )

INSERT course
VALUES
('GDH0724','4',3515,GETDATE()),
('GCH0717','3',1622,GETDATE()),
('GCH0718','2',1623,GETDATE()),
('GCH0719','6',1625,GETDATE()),
('GDH0724','5',3513,GETDATE()),
('GBH0725','3',485,GETDATE()),
('GBH0727','2',487,GETDATE())
```

RegisterStudent

Create table

```
CREATE TABLE RegisterStudent
(
    Class_ID    Nvarchar(50),
    Teacher_ID  Nvarchar(50)    ,
    Subject_ID  Int,
    Student_ID varchar(50),
    FinalResult VARCHAR(50) CHECK (FinalResult IN ('Refer','Pass','Merit','Distinction'))

    CONSTRAINT pk_register primary key(Class_ID,Teacher_ID,Subject_ID,Student_ID),
    CONSTRAINT fk_register_course foreign key(Class_ID,Teacher_ID,Subject_ID) references
    Course(Class_ID,Teacher_ID,Subject_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_register_student foreign key (Student_ID) references Student(Student_ID)
)
```

Example data

```
INSERT dbo.RegisterStudent
VALUES
('GDH0724','4',3515,'GDH20013'),
('GDH0724','4',3515,'GDH20014'),
('GCH0717','3',1622,'GCH20001'),
('GCH0717','3',1622,'GCH20002'),
('GCH0717','3',1622,'GCH20003'),
('GCH0717','3',1622,'GCH20004'),
('GCH0718','2',1623,'GCH20005'),
('GCH0718','2',1623,'GCH20006'),
('GCH0718','2',1623,'GCH20001'),
('GCH0718','2',1623,'GCH20002'),
('GCH0719','5',1625,'GCH20003'),
('GBH0725','3',485,'GBH20011'),
('GBH0725','3',485,'GBH20012'),
('GBH0725','3',485,'GBH20008'),
('GBH0725','3',485,'GBH20007'),
('GBH0725','3',485,'GBH20009'),
('GBH0727','2',487,'GBH20008'),
('GBH0727','2',487,'GBH20007'),
('GBH0727','2',487,'GBH20011'),
('GBH0727','2',487,'GBH20010')
```

ASM

```
CREATE TABLE ASM
(
    ASM_ID int primary key identity(1,1),
    Class_ID    Nvarchar(50),
    Teacher_ID  Nvarchar(50)    ,
    Subject_ID  Int  ,
    Student_ID varchar(50) ,

    CONSTRAINT fk_asm_register foreign key(Class_ID,Teacher_ID,Subject_ID,Student_ID) references
    RegisterStudent(Class_ID,Teacher_ID,Subject_ID,Student_ID),
)
```

Example data

```
INSERT ASM
VALUES
('GDH0724','4',3515,'GDH20013'),
('GDH0724','4',3515,'GDH20014'),
('GCH0717','3',1622,'GCH20001'),
('GCH0717','3',1622,'GCH20002'),
('GCH0717','3',1622,'GCH20003'),
('GCH0717','3',1622,'GCH20004'),
('GCH0718','2',1623,'GCH20005'),
('GCH0718','2',1623,'GCH20006'),
('GCH0718','2',1623,'GCH20001'),
('GCH0718','2',1623,'GCH20002'),
('GCH0719','5',1625,'GCH20003'),
('GBH0725','3',485,'GBH20011'),
('GBH0725','3',485,'GBH20012'),
('GBH0725','3',485,'GBH20008'),
('GBH0725','3',485,'GBH20007'),
('GBH0725','3',485,'GBH20009'),
('GBH0727','2',487,'GBH20008'),
('GBH0727','2',487,'GBH20007'),
('GBH0727','2',487,'GBH20011'),
('GBH0727','2',487,'GBH20010')
```

P

```
CREATE TABLE P
(
    P_ID int primary key identity(1,1),
    P_Status bit Not null,
    ASM_ID int

    constraint pk_p_asm foreign key (ASM_ID) references ASM(ASM_ID)
)
```

Example data

M

```
CREATE TABLE M
(
    M_ID int primary key identity(1,1),
    M_Status bit Not null,
    ASM_ID int

    constraint pk_m_asm foreign key (ASM_ID) references ASM(ASM_ID)
)
```

Example data

D

```
CREATE TABLE D
(
    D_ID int primary key identity(1,1),
    D_Status bit Not null,
    ASM_ID int

    constraint pk_d_asm foreign key (ASM_ID) references ASM(ASM_ID)
)
```

Example data for P,M,D

```
INSERT P VALUES (1,2),(1,2),(1,2),(1,2),(1,2),(1,2)
INSERT M VALUES (1,2),(1,2),(1,2),(1,2),(1,2),(1,2)
INSERT D VALUES (1,2),(1,2),(1,2),(1,2),(1,2),(1,2)
INSERT P VALUES (1,3),(1,3),(1,3),(1,3),(1,3),(1,3)
INSERT M VALUES (0,3),(0,3),(0,3),(0,3)
INSERT D VALUES (0,3),(0,3),(0,3)
INSERT P VALUES (1,4),(1,4),(1,4),(1,4),(1,4),(1,4)
INSERT M VALUES (1,4),(1,4),(0,4),(0,4)
INSERT D VALUES (1,4),(0,4),(0,4)
INSERT P VALUES (1,5),(1,5),(1,5),(1,5)
INSERT M VALUES (1,5),(1,5),(1,5),(1,5)
INSERT D VALUES (1,5),(1,5),(0,5)
INSERT P VALUES (1,6),(1,6),(1,6),(1,6),(1,6)
INSERT M VALUES (0,6),(0,6),(0,6),(0,6)
INSERT D VALUES (0,6),(0,6),(0,6)
INSERT P VALUES (1,7),(0,7),(0,7),(1,7),(1,7)
INSERT M VALUES (0,7),(1,7),(0,7),(0,7)
INSERT D VALUES (0,7),(0,7),(0,7)
INSERT P VALUES (1,8),(0,8),(0,8),(1,8),(1,8)
INSERT M VALUES (0,8),(1,8),(0,8),(0,8)
INSERT D VALUES (0,8),(0,8),(1,8)
INSERT P VALUES (1,9),(1,9),(0,9),(1,9),(1,9)
INSERT M VALUES (1,9),(1,9),(0,9),(0,9)
INSERT D VALUES (1,9),(1,9),(1,9)
```

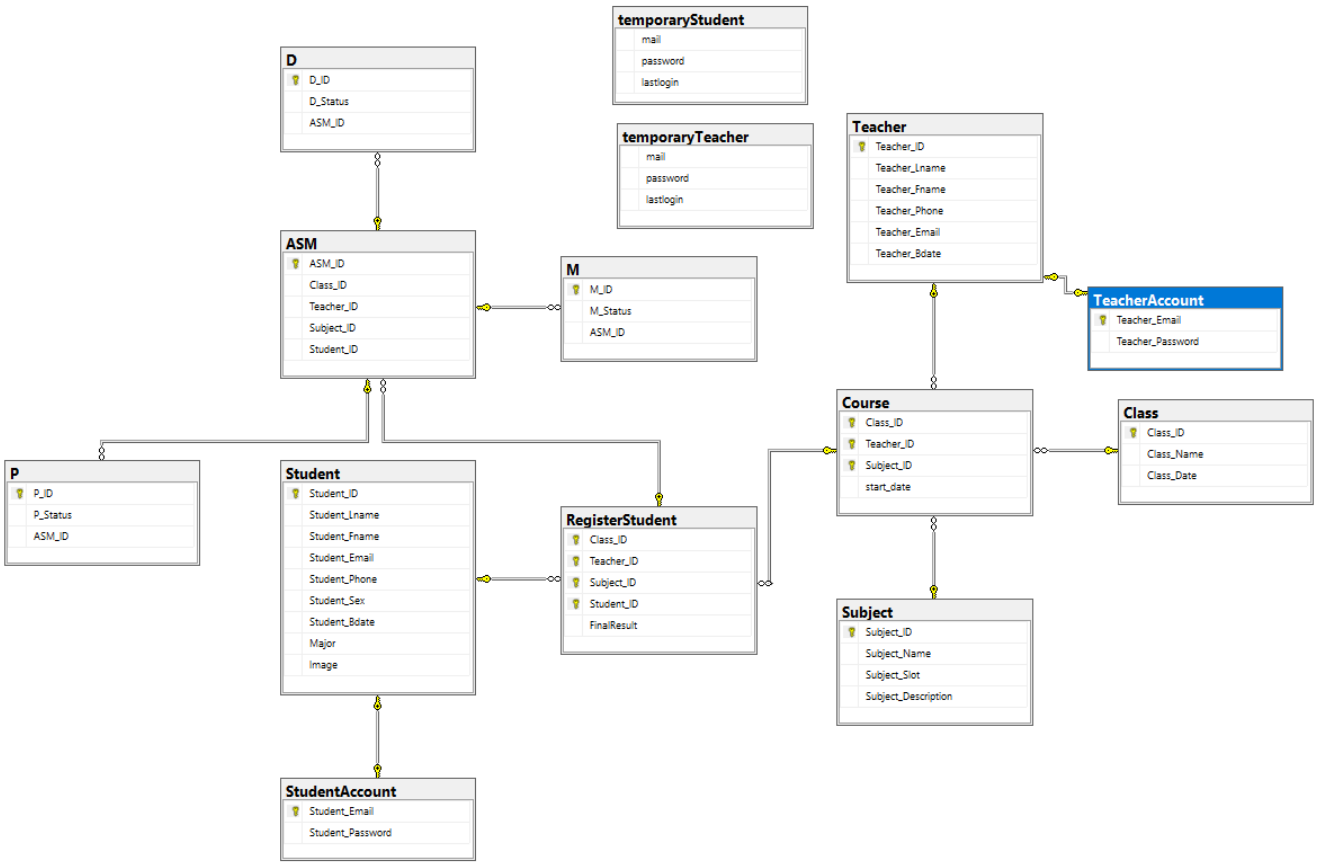
Temporary for login system

```
CREATE TABLE temporaryStudent
(mail NVARCHAR(50), password VARCHAR(50),  lastlogin date)

CREATE TABLE temporaryTeacher
(mail NVARCHAR(50), password VARCHAR(50) , lastlogin date)
```

UNIVERSITY of GREENWICH
Alliance with FPT Education

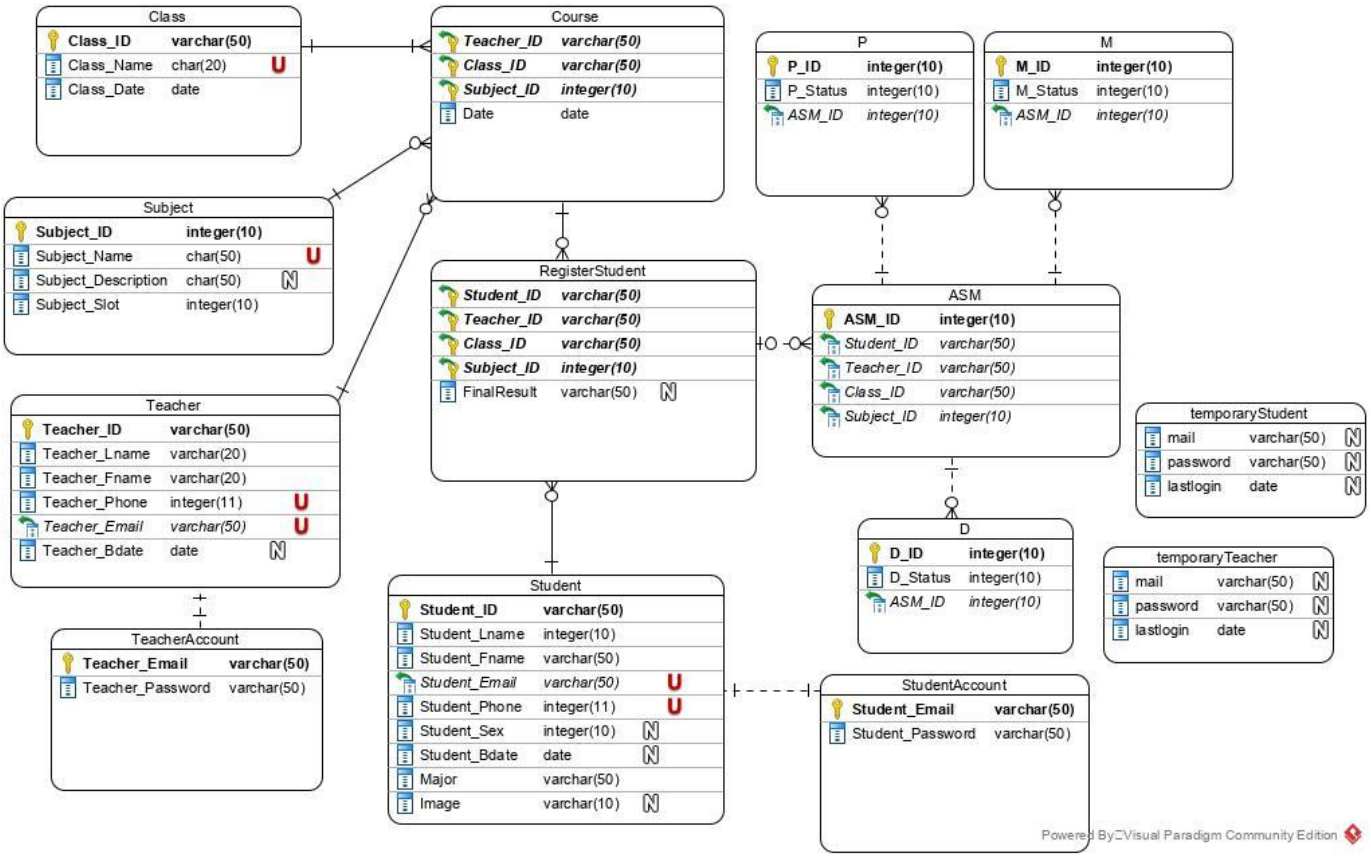
1. Generated Database Diagram of your Implementation



2. Explanations about any changes comparing to your design (if there is)

I have changed something comparing to my design. I will explain detail below:

1. ERD



2. Physical Design

- Student

Column Name	Data type	Constraint
Student_ID	Nvarchar(50)	Primary Key, CHECK (student_ID LIKE (G'[B,C,D]H[0-9] [0-9] [0-9] [0-9] [0-9])
Student_Lname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Student_Fname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Student_Email	Nvarchar(50)	Not null, unique, Foreign Key
Student_Phone	Char(11)	Not null,unique, CHECK (Not like ('%[^0-9]%')
Student_Sex	Varchar(50)	Not null, CHECK (in (Male,Female)
Student_Bdate	Date	
Major	Nvarchar(50)	Not Null, CHECK (in GCH,GBH,GDH)
Image	Nvarchar(50)	

- Class

Column Name	Data type	Constraint
Class_ID	Nvarchar(50)	Primary key, CHECK Like (G'[B,C,D]H[0-9] [0-9] [0-9] [0-9])
Class_Name	Varchar(20)	Not null, unique
Class_Date	date	Not null

- Subject

Column Name	Data type	Constraint
Subject_ID	Int	Primary Key, IDENTITY(1,1)
Subject_Name	Varchar(50)	Not null, unique
Subject_Slot	Int	Not null
Subject_Description	Varchar(50)	

- Teacher

Column Name	Data type	Constraint
Teacher_ID	nvarchar(50)	Primary Key,
Teacher_Lname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Teacher_Fname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Teacher_Phone	Varchar(50)	Not null, unique, CHECK Like ('%[^0-9]%')
Teacher_Email	nvarchar(50)	Not null, unique, Foreign Key
Teacher_Bdate	Date	

- Course

Column Name	Data type	Constraint
Teacher_ID	Nvarchar(50)	Primary key ,Foreign key
Class_ID	Nvarchar(50)	Primary key ,Foreign key
Subject_ID	int	Primary key ,Foreign key
Date	Date	Not null

- RegisterStudent

Column Name	Data type	Constraint
Student_ID	Nvarchar(50)	Primary key ,Foreign Key
Teacher_ID	Nvarchar(50)	Primary key ,Foreign key
Class_ID	Nvarchar(50)	Primary key ,Foreign key
Subject_ID	int	Primary key ,Foreign key
FinalResult	Varchar(50)	CHECK (FinalResult in ('Refer' , 'Pass' , 'Merit', 'Distinction')

- ASM (Assignment)

Column Name	Data type	Constraint
ASM_ID	Int	Primary Key, IDENTITY(1,1)
Student_ID	Nvarchar(50)	Foreign Key
Teacher_ID	Nvarchar(50)	Foreign key
Class_ID	Nvarchar(50)	Foreign key
Subject_ID	int	Foreign key

- P

Column Name	Data type	Constraint
P_ID	Int	Primary Key, IDENTITY(1,1)
P_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- M

Column Name	Data type	Constraint
M_ID	Int	Primary Key, IDENTITY(1,1)
M_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- D

Column Name	Data type	Constraint
D_ID	Int	Primary Key, IDENTITY(1,1)
D_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- TeacherAccount

Column Name	Data type	Constraint
Teacher_Email	Nvarchar(50)	Primarykey, Check (Teacher_Email LIKE '%@fe.edu.vn')
Teacher_Password	Varchar(50)	

- StudentAccount

Column Name	Data type	Constraint
Student_Email	Nvarchar(50)	Primarykey, Check (Student _Email LIKE '%@fpt.edu.vn')
Student _Password	Varchar(50)	

- temporaryStudent

Column Name	Data type	Constraint
Mail	Nvarchar(50)	
Password	Varchar(50)	
Lastlogin	date	

- temporaryTeacher

Column Name	Data type	Constraint
Mail	Nvarchar(50)	
Password	Varchar(50)	
Lastlogin	Date	

3. Table

- Because, the system will be used for different users and access levels.So we need account system to classify them. Therefore, i have add 2 more table are TeacherAccount and StudentAccount. We have other way to which no need to create them. But i think, the system will have many improvement in future so it's not excessive. For support the login system, i also add 2 more table for storage temporary email and password for checking process.

4. Remove many validation : Because they also have constraint for foreign key reference to other primary key. So we don't need to set any validation for them.Beside, Subjectid should be enter by admin instead of auto increment. Specifically:

- Remove identity(1,1) of Subjectid in Subject table, Course table and RegisterStudent table. From now, user can enter it by themself.
- Remove Check Constraint for Email data in Student and Teacher. Because i have add 2 new table Teacher Account And Student Account with the primary key are email in each table. So we also have constraint for primary key – foreign key.
- Remove Check Constraint for StudentID and ClassID in two table Course and RegisterStudent. The reason i mentioned above.
- Add FinalResult to RegisterStudent to determine the final outcome of student
- Add temporaryStudent/Teacher table for login function.

5. Change data-type

I have change data type of few attribute for more rational.

6. Add new constraint

New CHECK constraint on student_email and teacher_email. The entered email must have @fpt.edu.vn for student and @fe.edu.vn for teacher.

7. **New business rule**
   - Student can only relearn a subject more than 3 times.

II. User Interfaces & Querries (P3)

(M2) Implement a fully functional database system which includes system security and database maintenance. Your querries need to include:

1. **Procedures**

a. Student table

```
CREATE PROC insertupdatedeleteStudent
@id VARCHAR(50), @lname nvarchar(50), @fname nvarchar(50), @email varchar(50),
@phone char(10), @sex varchar(50), @date date, @major varchar(50), @image varchar(50),@statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT dbo.Student
            VALUES(@id,@lname,@fname,@email,@phone,@sex,@date,@major,@image)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.Student
            SET
                Student_Lname = @lname,
                Student_Fname =@fname,
                Student_Email = @email,
                Student_Phone = @phone,
                Student_Sex = @sex,
                Student_Bdate = @date,
                Image = @image
            WHERE Student_ID = @id
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.Student
            WHERE Student_ID = @id
        END
```

b. Teacher table

```
CREATE PROC insertupdatedeleteTeacher
@id VARCHAR(50), @lname NVARCHAR(50),@fname NVARCHAR(50), @phone CHAR(10),
@email VARCHAR(50), @date DATE, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT Teacher
            VALUES(@id,@lname,@fname,@phone,@email,@date)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.Teacher
            SET Teacher_Lname = @lname,
                Teacher_Fname = @fname,
                Teacher_Phone = @phone,
                Teacher_Email = @email,
                Teacher_Bdate = @date
            WHERE
                Teacher_ID = @id
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.Teacher
            WHERE Teacher_ID = @id
        END
```

c. Insert,Update,Delete on Subject table

```
CREATE PROC insertupdatedeleteSubject
@id VARCHAR(50), @name VARCHAR(50), @slot INT , @description VARCHAR(50), @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT Subject
            VALUES(@id,@name,@slot,@description)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.Subject
            SET
                Subject_Name = @name,
                Subject_Slot = @slot,
                Subject_Description = @description
            WHERE Subject_ID = @id
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM  dbo.Subject
            WHERE Subject_ID = @id
        END
```

d. Insert,Update,Delete on Class table

```
CREATE PROC insertupdatedeleteClass
@id VARCHAR(50), @name VARCHAR(50), @date DATE, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT Class
            VALUES(@id,@name,@date)
        END

    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.Class
            SET
                Class_Name = @name,
                Class_Date = @date
            WHERE
                Class_ID = @id
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.Class
            WHERE Class_ID = @id
        END
```

e. Insert,Update,Delete on Course table

```
CREATE PROC insertupdatedeleteCourse
@classid VARCHAR(50), @teacherid VARCHAR(50),  @subjectid INT, @date DATE ,
@newclass VARCHAR(50), @newteacher VARCHAR(50), @newsubject INT, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT course
            VALUES(@newclass,@newteacher,@newsubject,@date)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.Course
            SET
                Class_ID = @newclass,
                Teacher_ID = @newteacher,
                Subject_ID = @newsubject,
                start_date = @date
            WHERE
                Class_ID = @classid AND
                Teacher_ID = @teacherid AND
                Subject_ID = @subjectid
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.Course
            WHERE Teacher_ID = @teacherid AND
                  Subject_ID = @subjectid AND
                  Class_ID = @classid
        END
```

f. Insert,Update,Delete on Register table

```
CREATE PROC insertupdatedeleteRegister
@classid VARCHAR(50), @teacherid VARCHAR(50),@studentid VARCHAR(50),  @subjectid INT,
@newclass VARCHAR(50), @newteacher VARCHAR(50),@newstudent VARCHAR(50), @newsubject INT, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT RegisterStudent
            VALUES(@newclass,@newteacher,@newsubject,@newstudent)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.RegisterStudent
            SET
                Class_ID = @newclass,
                Teacher_ID = @newteacher,
                Subject_ID = @newsubject,
                Student_ID = @newstudent
            WHERE
                Class_ID = @classid AND
                Teacher_ID = @teacherid AND
                Subject_ID = @subjectid AND
                Student_ID = @studentid
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.RegisterStudent
            WHERE Teacher_ID = @teacherid AND
                  Subject_ID = @subjectid AND
                  Class_ID = @classid AND
                  Student_ID = @studentid
        END
```

g. Insert,Update,Delete on ASM

```
CREATE PROC insertupdatedeleteASM
@newclass VARCHAR(50), @newteacher VARCHAR(50), @newsubject INT, @newstudent VARCHAR(50),
@class VARCHAR(50), @teacher VARCHAR(50), @subject INT, @student VARCHAR(50), @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT dbo.ASM
            VALUES (@newclass,@newteacher,@newsubject,@newstudent)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.ASM
            SET Class_ID = @newclass,
                Teacher_ID = @newteacher,
                Subject_ID = @newsubject,
                Student_ID = @newstudent

            WHERE
                  Class_ID = @class AND
                  Teacher_ID = @teacher AND
                  Subject_ID = @subject AND
                  Student_ID = @student
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.ASM
            WHERE
                 Class_ID = @class AND
                 Teacher_ID = @teacher AND
                 Subject_ID = @subject  AND
                 Student_ID = @student
        END
```

h. Insert,Update,Delete on P,M,D table

```
CREATE PROC insertupdatedeleteP
@id INT , @status BIT, @asmid INT, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT P
            VALUES(@status, @asmid)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.P
            SET P_Status = @status
            WHERE P_ID = @id AND ASM_ID = @asmid
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.P
            WHERE P_ID = @id AND ASM_ID = @asmid
        END
```

```
CREATE PROC insertupdatedeleteM
@id INT , @status BIT, @asmid INT, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT M
            VALUES(@status, @asmid)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.M
            SET M_Status = @status
            WHERE M_ID = @id AND ASM_ID = @asmid
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.M
            WHERE M_ID = @id AND ASM_ID = @asmid
        END
```

```
CREATE PROC insertupdatedeleteD
@id INT , @status BIT, @asmid INT, @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT D
            VALUES(@status, @asmid)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE dbo.D
            SET D_Status = @status
            WHERE D_ID = @id AND ASM_ID = @asmid
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM dbo.D
            WHERE D_ID = @id AND ASM_ID = @asmid
        END
```

i. Insert,Update,Delete Account on TeacherAccount and StudentAccount

```
CREATE PROC account_student
@mail NVARCHAR(50), @password VARCHAR(50),@newemail NVARCHAR(50), @newpassword VARCHAR(50),@statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT StudentAccount
            VALUES(@mail,@password)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE StudentAccount
            SET Student_Password = @newpassword,
            Student_Email = @newemail
            WHERE Student_Email = @mail
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM StudentAccount
            WHERE Student_Email = @mail
        END
CREATE PROC account_teacher
@mail NVARCHAR(50),@password VARCHAR(50),@newemail NVARCHAR(50),@newpassword VARCHAR(50), @statement VARCHAR(50)
AS
    IF @statement = 'Insert'
        BEGIN
            INSERT TeacherAccount
            VALUES(@mail,@password)
        END
    IF @statement = 'Update'
        BEGIN
            UPDATE TeacherAccount
            SET Teacher_Password = @newpassword,
            Teacher_Email = @newemail
            WHERE Teacher_Email = @mail
        END
    IF @statement = 'Delete'
        BEGIN
            DELETE FROM TeacherAccount
            WHERE Teacher_Email = @mail
        END
```

j. Change password for teacher and student

```
CREATE PROC password_student
@email NVARCHAR(50), @newpassword VARCHAR(50)
AS
    UPDATE dbo.StudentAccount
    SET Student_Password = @newpassword
    WHERE Student_Email = @email

go
CREATE PROC password_teacher
@email NVARCHAR(50), @newpassword VARCHAR(50)
AS
    UPDATE dbo.StudentAccount
    SET Student_Password = @newpassword
    WHERE Student_Email = @email
```

k. Get Student's Grade

```
CREATE FUNCTION UF_Result(@num BIT)
RETURNs varchar(20)
AS
BEGIN
    IF (@num = 1)
        RETURN 'Pass'
    ELSE
        RETURN 'Fail'

        RETURN 'None'
END



--GET GRADE from Table
GO
CREATE PROC getGrade
@class VARCHAR(50), @teacher VARCHAR(50), @subject INT, @student VARCHAR(50)
AS
    SELECT ROW_NUMBER() OVER (ORDER BY P_ID) AS P, dbo.UF_Result(P_Status) FROM P WHERE ASM_ID =
(
    SELECT ASM_ID FROM ASM
    WHERE Class_ID = @class AND Teacher_ID = @teacher AND subject_ID = @subject AND Student_ID = @student
)
    SELECT ROW_NUMBER() OVER (ORDER BY M_ID) AS M, dbo.UF_Result(M_Status) FROM M WHERE ASM_ID =
(
    SELECT ASM_ID FROM ASM
    WHERE Class_ID = @class AND Teacher_ID = @teacher AND subject_ID = @subject AND Student_ID = @student
)

    SELECT ROW_NUMBER() OVER (ORDER BY D_ID) AS D, dbo.UF_Result(D_Status) FROM D WHERE ASM_ID =
(
    SELECT ASM_ID FROM ASM
    WHERE Class_ID = @class AND Teacher_ID = @teacher AND subject_ID = @subject AND Student_ID = @student
)
```

l. Procedure for get list of student in course

```
CREATE PROC getListOfStudent
@class VARCHAR(50), @teacher VARCHAR(50), @subject INT
AS
    Begin
        SELECT S.Student_ID, S.Student_Fname + ' ' + S.Student_Lname AS  NAME, S.Student_Email, S.Student_Phone,
        S.Student_Sex, YEAR(GETDATE())-YEAR(S.Student_Bdate) AS Age, S.Major FROM dbo.Student AS S, dbo.RegisterStudent
        WHERE S.Student_ID =  RegisterStudent.Student_ID AND
        dbo.RegisterStudent.Class_ID = @class AND
        dbo.RegisterStudent.Subject_ID = @subject AND
        dbo.RegisterStudent.Teacher_ID = @teacher
    END
```

m. Get Course information

```
CREATE PROC getCourseInformation
AS
SELECT c.Class_ID,t.Teacher_Fname + ' ' + t.Teacher_Lname AS Teacher, c.Subject_ID FROM Course AS c, dbo.Teacher AS t
WHERE c.Teacher_ID = t.Teacher_ID
```

n. Login

```
CREATE PROC login_student_check(@mail NVARCHAR(50), @password VARCHAR(50))
AS
    INSERT temporaryStudent
    VALUES(@mail,@password,GETDATE())


GO
CREATE PROC login_teacher_check(@mail NVARCHAR(50), @password VARCHAR(50))
AS
    INSERT temporaryTeacher
    VALUES(@mail,@password,GETDATE())
```

o. Get Sex and Major

```
CREATE PROC getMajor
AS
SELECT Major FROM dbo.Student
GROUP BY major
GO

CREATE PROC getSex
AS
SELECT Student_Sex FROM dbo.Student
GROUP BY Student_Sex
```

p. Get Course Final Result

Get ASM_ID

```
CREATE FUNCTION getASM (@class VARCHAR(50), @teacher VARCHAR(50), @subject INT, @student VARCHAR(50))
RETURNs INT
AS
    BEGIN
        DECLARE @asmid INT
        SELECT @asmid = ASM_ID FROM ASM
        WHERE Subject_ID = @subject AND Class_ID = @class AND Teacher_ID = @teacher AND Student_ID = @student
        RETURN @asmid
    END
GO
```

Get Final Result Of Assignment

```
CREATE FUNCTION getALl(@asm INT)
RETURNS VARCHAR(50)
AS
    BEGIN
        DECLARE @result VARCHAR(50)
        DECLARE @P_count INT,@M_count INT,@D_count INT
        SELECT @P_count = COUNT(P_Status)   FROM P
        WHERE ASM_ID = @asm AND P_Status = 0
        SELECT @M_count = COUNT(M_status)   FROM dbo.M
        WHERE ASM_ID = @asm AND m_Status = 0
        SELECT @d_count = COUNT(D_Status)  FROM dbo.D
        WHERE ASM_ID = @asm AND D_Status = 0

        IF @P_count > 0
            BEGIN
                SET @result = 'Fail'
            END
        ELSE
            BEGIN
                IF @M_count = 0
                    BEGIN
                         set @result = 'Merit'
                    END
                ELSE
                    BEGIN
                        SET @result = 'Pass'
                    END
                IF @D_count = 0
                    BEGIN
                        SET @result = 'Distinction'
                    END
            END
            RETURN @result
    END
```

Get Grade Of Student In Course

```
CREATE PROC getCourseGrade
@class VARCHAR(50), @teacher VARCHAR(50), @subject INT
as
SELECT S.Student_ID,S.Student_Fname + ' ' + S.Student_Lname, A.Class_ID, A.Subject_ID,
A.Teacher_ID, dbo.getALl(dbo.getASM(A.Class_ID,A.Teacher_ID,A.Subject_ID,A.Student_ID))
FROM asm AS A,dbo.Student AS S
WHERE   A.Student_ID = S.Student_ID AND
        A.Class_ID = @class AND
        A.Teacher_ID = @teacherAND AND
        A.Subject_ID = @subject
```

q. Get Course Result for Student

```
CREATE PROC getCourseGradeStudent
@student VARCHAR(50)
as
SELECT S.Student_ID,S.Student_Fname + ' ' + S.Student_Lname AS Name, A.Class_ID, A.Subject_ID,
A.Teacher_ID, dbo.getALl(dbo.getASM(A.Class_ID,A.Teacher_ID,A.Subject_ID,A.Student_ID)) AS FinalResult
FROM asm AS A,dbo.Student AS S
WHERE   A.Student_ID = S.Student_ID AND
        A.Student_ID = 'GCH20001'

        EXEC dbo.getCourseGradeStudent @student = 'GCH20001' -- varchar(50)
```

2. Triggers

a) Trigger for student login

```
CREATE TRIGGER login_check_student ON temporaryStudent
AFTER INSERT
AS
    BEGIN
        IF EXISTS (SELECT COUNT(*) FROM Inserted  AS b , StudentAccount AS sa WHERE b.mail = sa.Student_Email
        AND b.password = sa.Student_password
        GROUP BY b.password HAVING COUNT(*) = 1)
        BEGIN
            PRINT 'Login successful'
        END
        ELSE
            BEGIN
                PRINT 'Login Failed'
                ROLLBACK
            END
    END
```

b) Trigger for teacher login

```
CREATE TRIGGER login_check_teacher ON temporaryTeacher
AFTER INSERT
AS
    BEGIN
        IF EXISTS (SELECT COUNT(*) FROM Inserted  AS b , dbo.TeacherAccount AS TA WHERE b.mail = Ta.Teacher_Email
        AND b.password = Ta.Teacher_Password
        GROUP BY b.password HAVING COUNT(*) = 1)
        BEGIN
            PRINT 'Login successful'
        END
        ELSE
            BEGIN
                PRINT 'Login Failed'
                ROLLBACK
            END
    END

GO
```

c) Trigger for maximum Teacher in Course

```
CREATE TRIGGER maximumTeacher ON dbo.Course
AFTER INSERT, UPDATE
AS
    IF EXISTS(SELECT COUNT(*) FROM Inserted AS i,dbo.Course AS c
    WHERE i.Class_ID = c.Class_ID AND i.Subject_ID = c.Subject_ID
    GROUP BY c.Class_ID,c.Subject_ID HAVING COUNT(*) = 2)
    BEGIN
        PRINT 'Dont allow more than one teacher in course'
        ROLLBACK
    END
    ELSE
        PRINT 'Successful'
```

d) Trigger for maximum registration of student in a subject

```
CREATE TRIGGER maximumRegister ON dbo.RegisterStudent
AFTER INSERT,UPDATE
AS
    IF EXISTS(SELECT COUNT(*) FROM Inserted AS i, dbo.RegisterStudent AS c
    WHERE
             i.Student_ID = c.Student_ID AND
             i.Subject_ID = c.Subject_ID
    GROUP BY i.Subject_ID, i.Student_ID
    HAVING COUNT(*) > 3)
    BEGIN
        PRINT 'Student cant relearn  one subject more than 3 times'
        ROLLBACK
    END
    ELSE PRINT 'Successful'
```

3. Security

```
CREATE ROLE Staff

CREATE ROLE student

CREATE ROLE teacher
```

Staff is highest authority so they can use almost function or interact with the system

```
GRANT EXECUTE ON dbo.insertupdatedeleteASM TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteClass TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteCourse TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteD TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteM TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteP TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteRegister TO Staff
GRANT EXECUTE ON dbo.insertupdatedeleteStudent TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteTeacher TO staff
GRANT EXECUTE ON dbo.insertupdatedeleteSubject TO staff
GRANT EXECUTE ON dbo.getall TO staff
GRANT EXECUTE ON dbo.getASM TO staff
GRANT EXECUTE ON dbo.getCourseGrade TO staff
GRANT EXECUTE ON dbo.getCourseInformation TO Staff
GRANT EXECUTE ON dbo.getListOfStudent TO staff
GRANT EXECUTE ON dbo.getCourseGradeStudent TO staff
```

Teacher

Teacher can only grading student assignment. They can't enter student or class or anything to system,

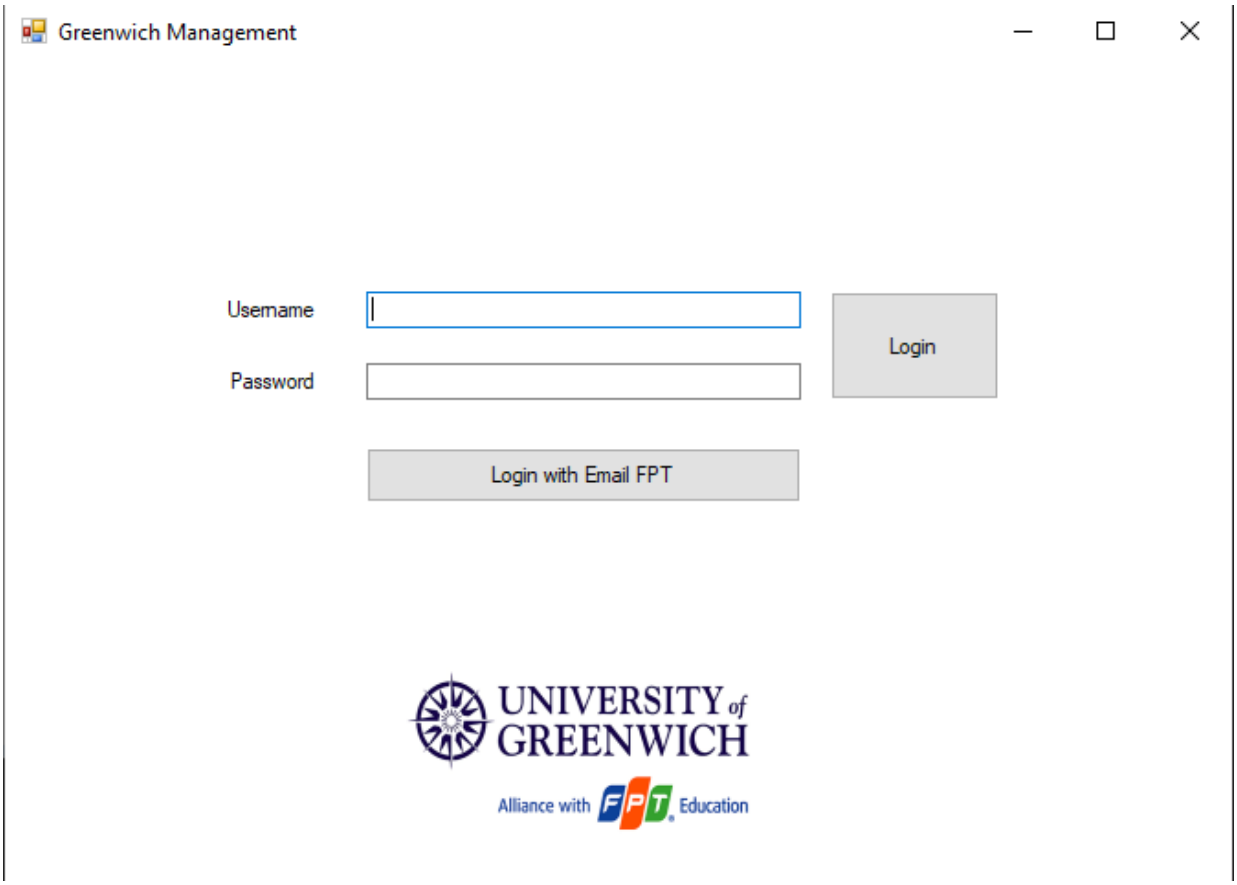
```
GRANT EXECUTE ON dbo.insertupdatedeleteD TO teacher
GRANT EXECUTE ON dbo.insertupdatedeleteM TO teacher
GRANT EXECUTE ON dbo.insertupdatedeleteP TO teacher
GRANT EXECUTE ON dbo.getASM TO teacher
GRANT EXECUTE ON dbo.getCourseGrade TO teacher
GRANT EXECUTE ON dbo.getCourseInformation TO teacher
GRANT EXECUTE ON dbo.getListOfStudent TO teacher
GRANT EXECUTE ON dbo.getTranscript TO teacher
```

Student

```
GRANT EXECUTE ON dbo.getListOfStudent TO student
GRANT EXECUTE ON dbo.getGrade TO student
GRANT EXECUTE ON dbo.getCourseInformation TO student
GRANT EXECUTE ON dbo.getCourseGradeStudent TO student
```

Student can see their grade and information about course.

4. **With each user interface**
   Login




```
EXEC dbo.login_student_check @mail = N'anhnqgch20001@fpt.edu.vn',    -- nvarchar(50)
                             @password = '123456789' -- varchar(50)
```

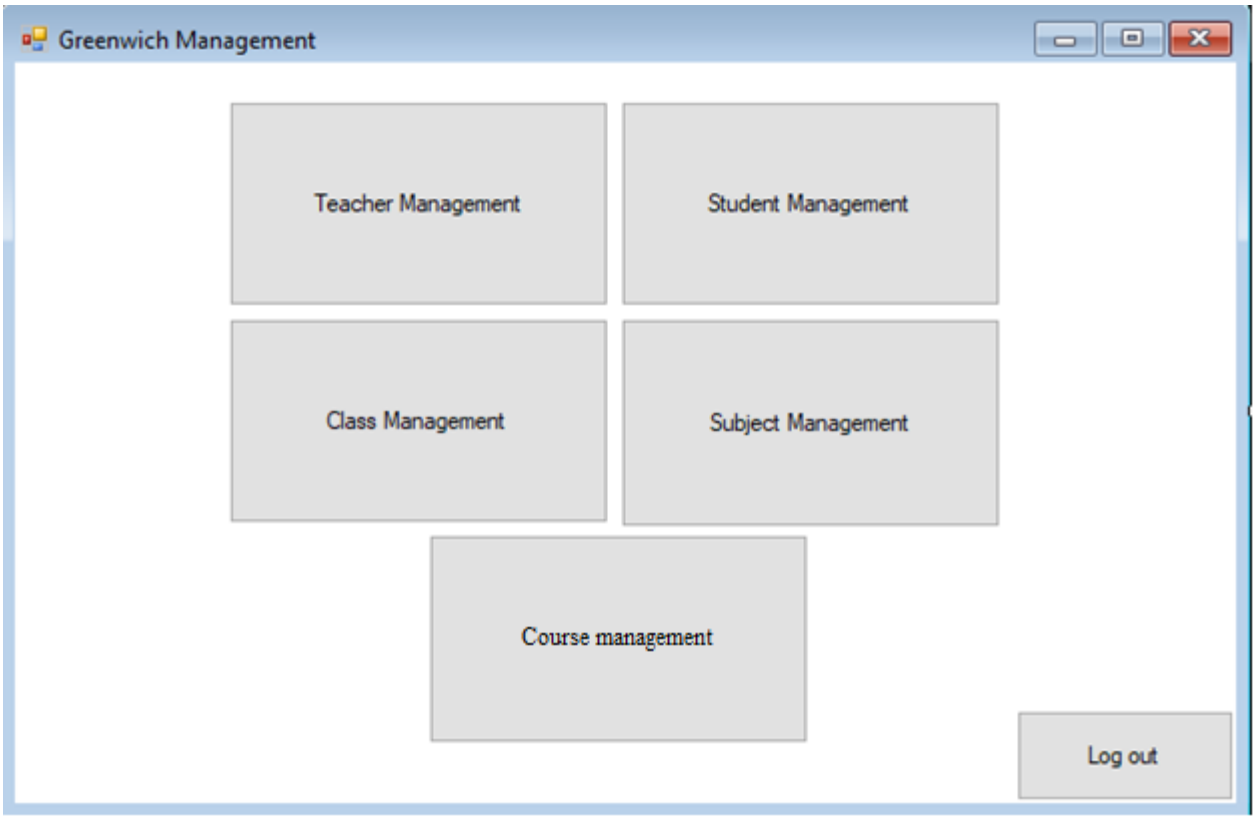
After login successful, program will record login log of this user.

3	anhnqgch20001@fpt.edu.vn	123456789	2020-03-15

```
EXEC dbo.login_teacher_check @mail = N'oanhnv@fe.edu.vn',    -- nvarchar(50)
                             @password = '123456789' -- varchar(50)
```

3	quandh@fe.edu.vn	123456789	2020-03-15

Home interface



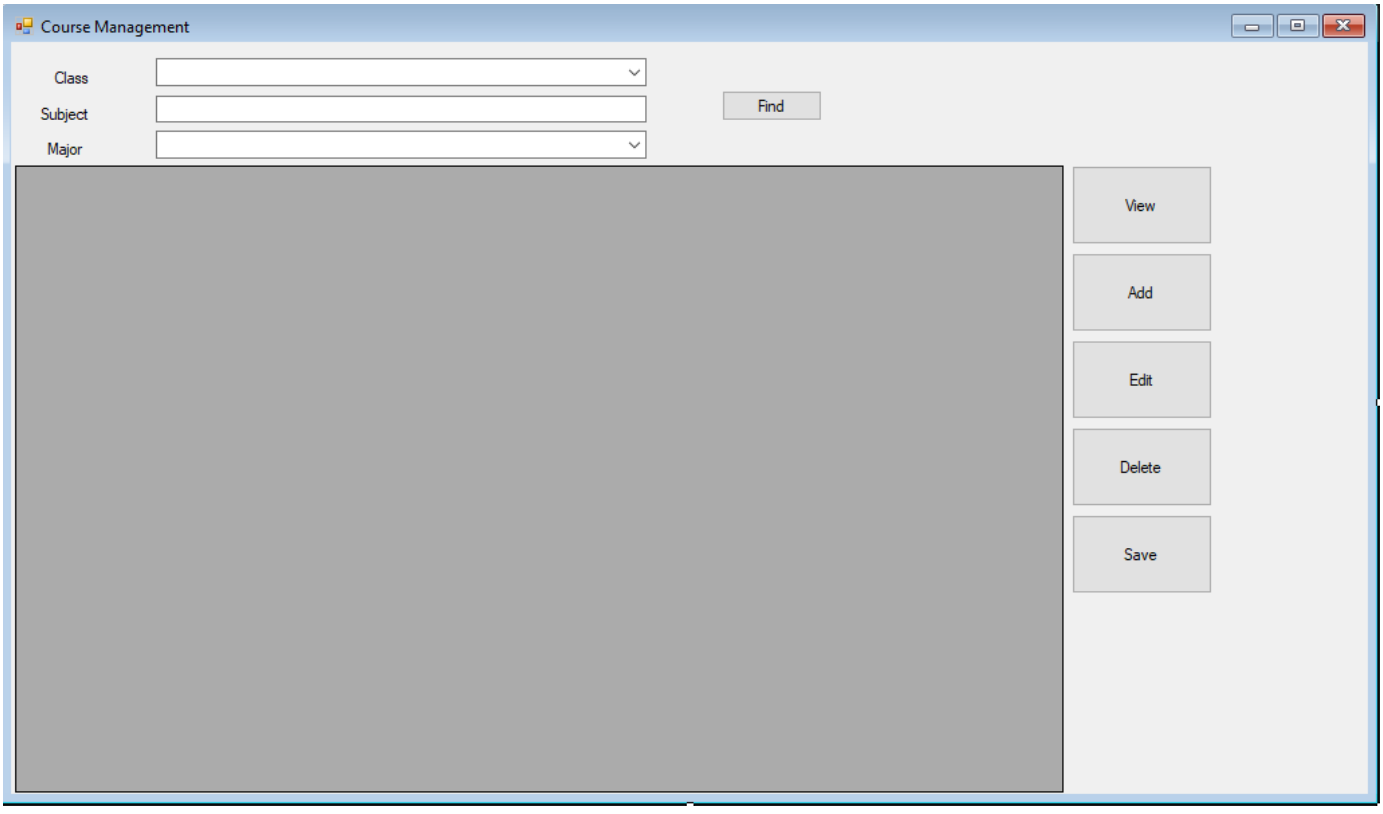
This display is only for teacher or higher level staff. In this display, we have 5 option for user to be choose.

1. Course management

Data will be filled to datagridview from database. In this display, program will show list course will information field are ClassID , Teacher full name and Subject.

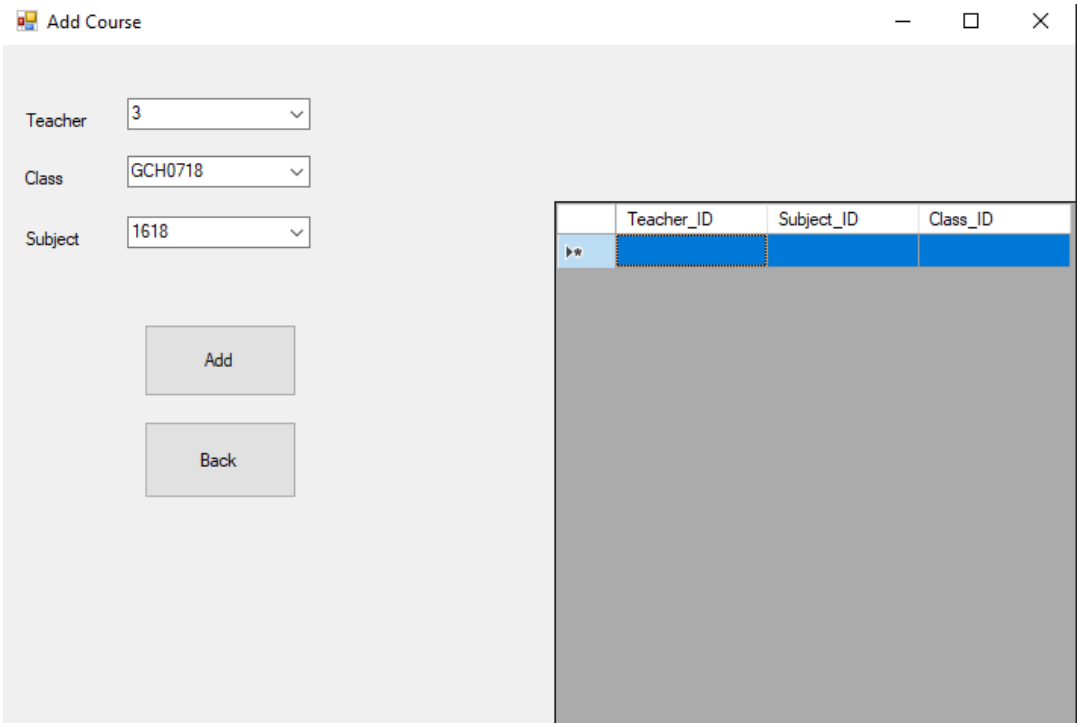
If user is teacher, they only can view information about course

Below is screen for staff. They can use add,edit ,delete function.



```
EXEC dbo.getCourseInformation
```

2. Add Course



After click to add button in Course Management, New form will be opened. In this form, we will 3 combobox to select class,teacher,subject for one course before create. After finish selection, we can click to Add and view the result in datagridview. After that we can click Back button for back to Course management screen or if we can stay for add more Course.

Query for combobox

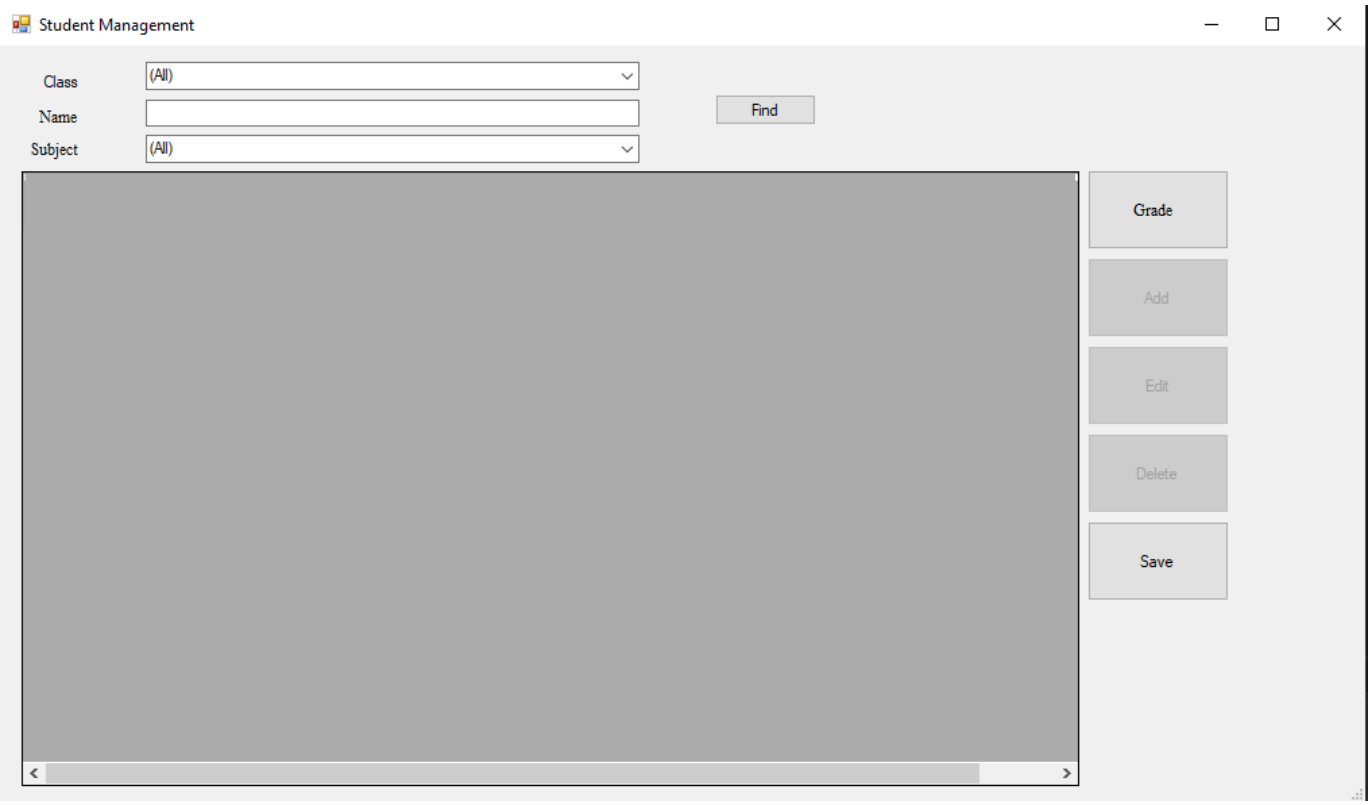
```
SELECT Class_ID FROM dbo.Class
SELECT Teacher_ID FROM dbo.Teacher
SELECT Subject_ID FROM dbo.Subject
```

In case, A course has one teacher already, data wont allow to add a course which have the same subjectid and classid to system.

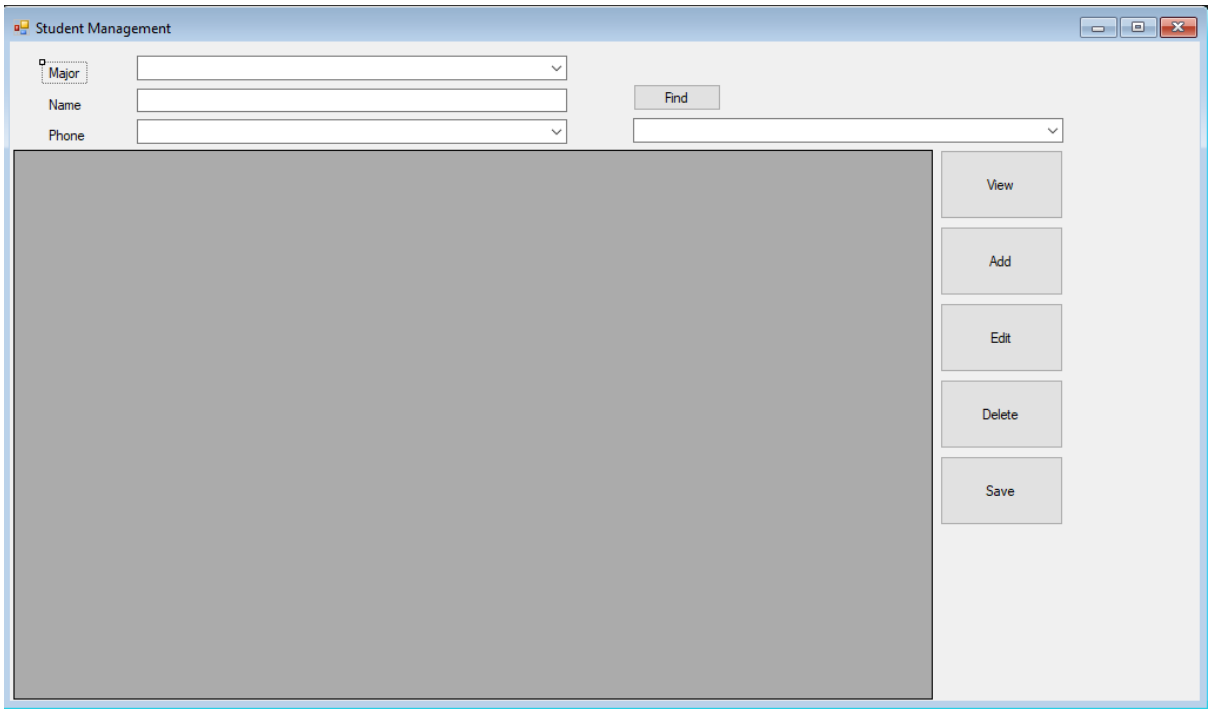
```
CREATE TRIGGER maximumTeacher ON dbo.Course
AFTER INSERT, UPDATE
AS
    IF EXISTS(SELECT COUNT(*) FROM Inserted AS i,dbo.Course AS c
    WHERE i.Class_ID = c.Class_ID AND i.Subject_ID = c.Subject_ID
    GROUP BY c.Class_ID,c.Subject_ID HAVING COUNT(*) = 2)
    BEGIN
        PRINT 'Dont allow more than one teacher in course'
        ROLLBACK
    END
    ELSE
        PRINT 'Successful'
```

3. Student management

Similar for course, teacher only can’t use other function exclude Grade



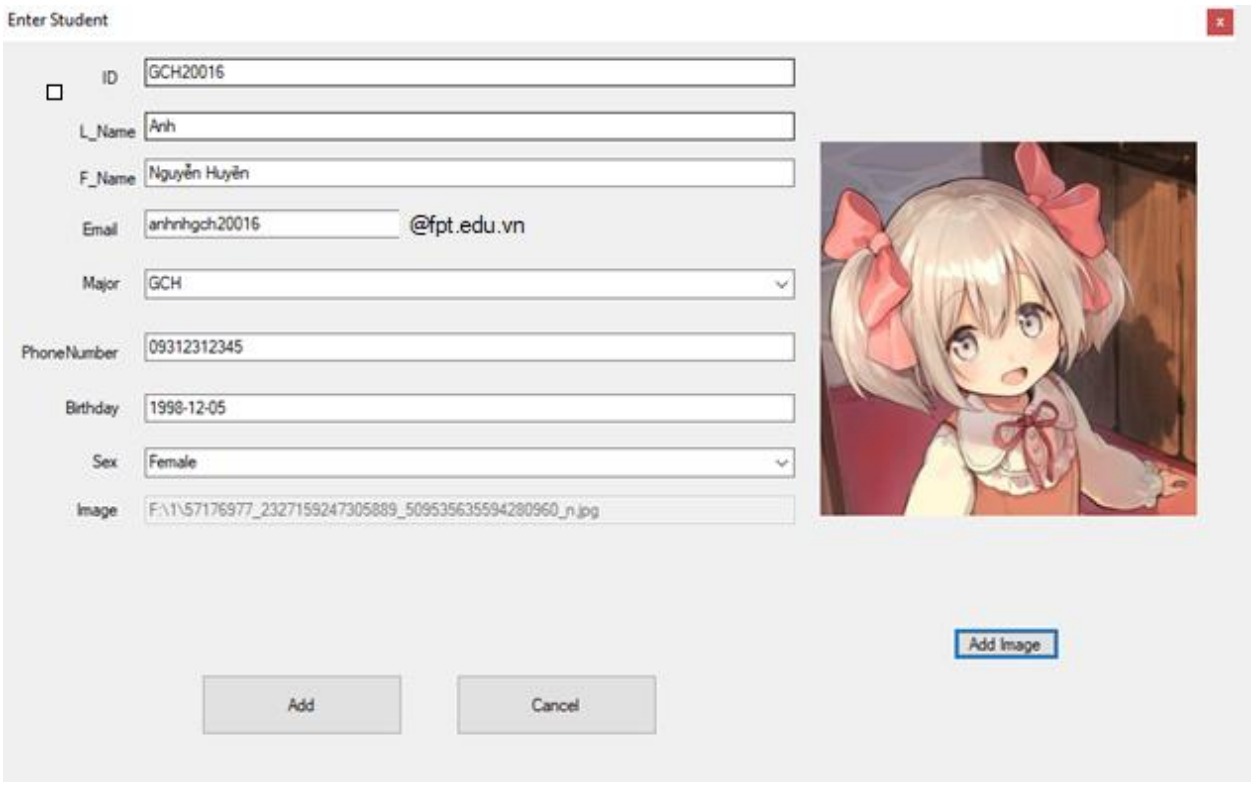
Teacher can find a course by Class and Subject. They can only see course that they are teaching.




```
EXEC dbo.getListOfStudent @class = 'GBH0725',    -- varchar(50)
                          @teacher = '3', -- varchar(50)
                          @subject = 485    -- int
```

4. Add student to list

After choose “Add” option in Student management screen. Program will appear one form which allow user to enter Student information. After completing all required fields, we can enter to input them to system or cancel to close form. In this form, we can enter information with 5 textbox. We also have 2 combobox for Sex and Major to make sure the user doesn't enter wrong one. Beside, user can enter student image with add image button. Moreover email domain will be define and user only need to enter mailbox name. In database, image is allow null field, so we can enter it or not.



We have query with getMajor and getSex procedure for fill the major combobox. When loading form program will exec there two procedure.

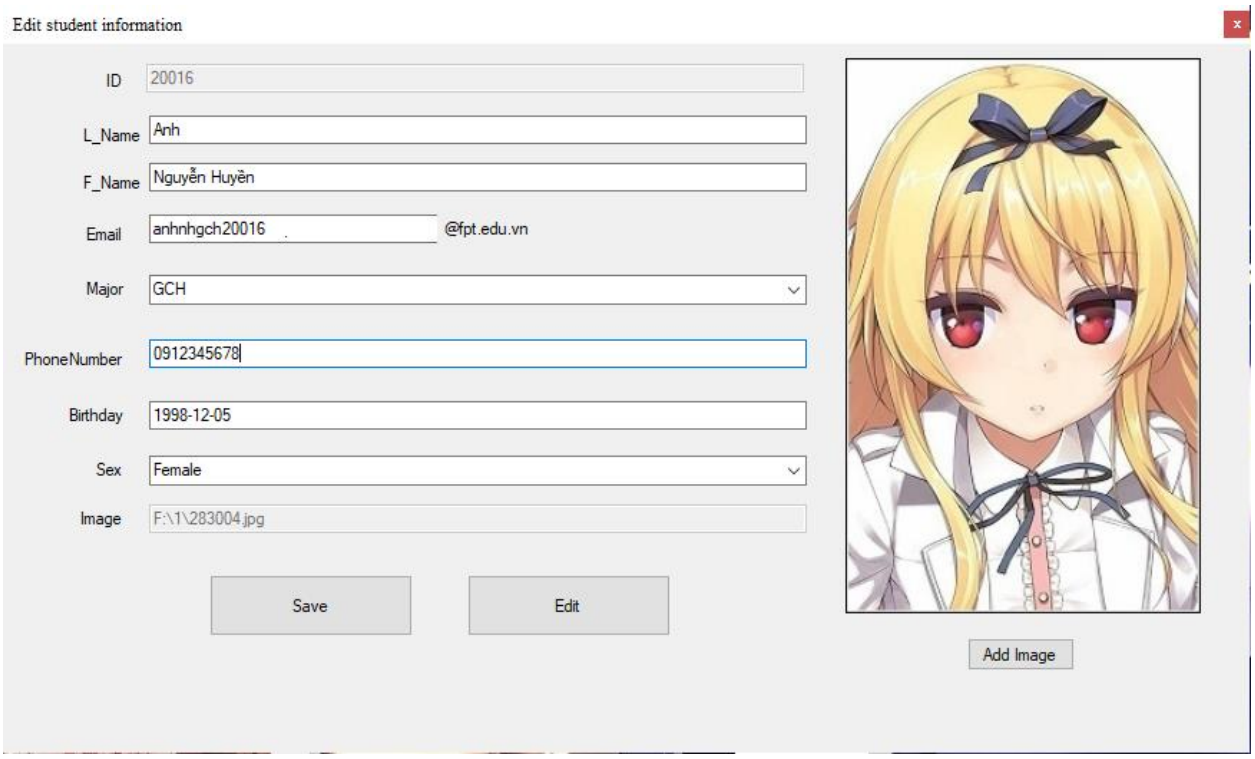
```
EXEC dbo.getMajor
```

```
EXEC dbo.getSex
```

And InsertupdatedeleteStudent Procecdure for input student information to database.

```
EXEC dbo.insertupdatedeleteStudent @id = 'GCH20016',            -- varchar(50)
                                   @lname = N'Anh',         -- nvarchar(50)
                                   @fname = N'Nguyễn Huyền',         -- nvarchar(50)
                                   @email = 'anhnhgch20016@fpt.edu.vn',          -- varchar(50)
                                   @phone = '0931231234',          -- char(10)
                                   @sex = 'Female',            -- varchar(50)
                                   @date = '1998-12-05', -- date
                                   @major = 'GCH',           -- varchar(50)
                                   @image = '',          -- varchar(50)
                                   @statement = 'Insert'       -- varchar(50)
```

5. Edit student information



If we click to edit button, Edit student form will appear. This form is very similar to the "Add student form". But we will Save button instead Add. All information of choosen student will be fill in the form exclude Studentid, because studentid is not accepted to change. After we edit it, we can click to save button to save the change or click exit.

```
EXEC dbo.insertupdatedeleteStudent @id = 'GCH20016',             -- varchar(50)
                                   @lname = N'Anh',          -- nvarchar(50)
                                   @fname = N'Nguyễn Huyền',          -- nvarchar(50)
                                   @email = 'anhnhgch20016@fpt.edu.vn',          -- varchar(50)
       Change this =>>>            @phone = '0912345678',           -- char(10)
                                   @sex = 'Female',             -- varchar(50)
                                   @date = '1998-12-05', -- date
                                   @major = 'GCH',           -- varchar(50)
                                   @image = '',           -- varchar(50)
                                   @statement = 'Update'        -- varchar(50)
```

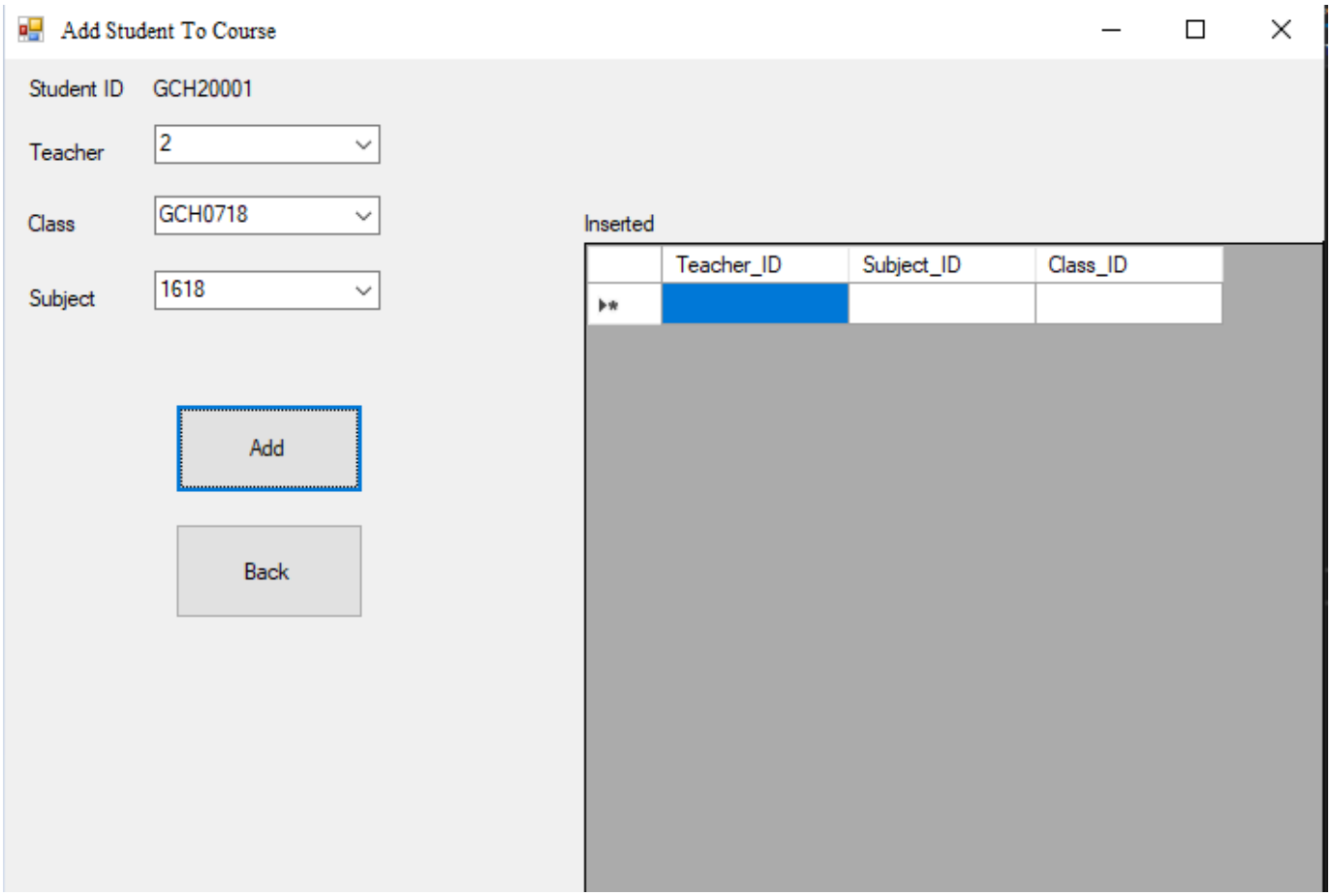
6. Delete Student in list

We can select one student in datagridview (student list) then click to delete button. All information about this student will be remove from list.

```
EXEC dbo.insertupdatedeleteStudent @id = 'GCH20016',            -- varchar(50)
                                   @lname = N'',          -- nvarchar(50)
                                   @fname = N'',          -- nvarchar(50)
                                   @email = '',           -- varchar(50)
                                   @phone = '',           -- char(10)
                                   @sex = '',             -- varchar(50)
                                   @date = '2020-03-11', -- date
                                   @major = '',           -- varchar(50)
                                   @image = '',           -- varchar(50)
                                   @statement = 'Delete'       -- varchar(50)
```

7. Insert student to course

To open this form, we need click one row (a student) then click view button student management. In this form, we are adding new record for RegisterStudent as Registration of Student who is clicked in student management form. In this form, we can choose course information with 3 data field in combobox or we can select the available course in the table Inserted.



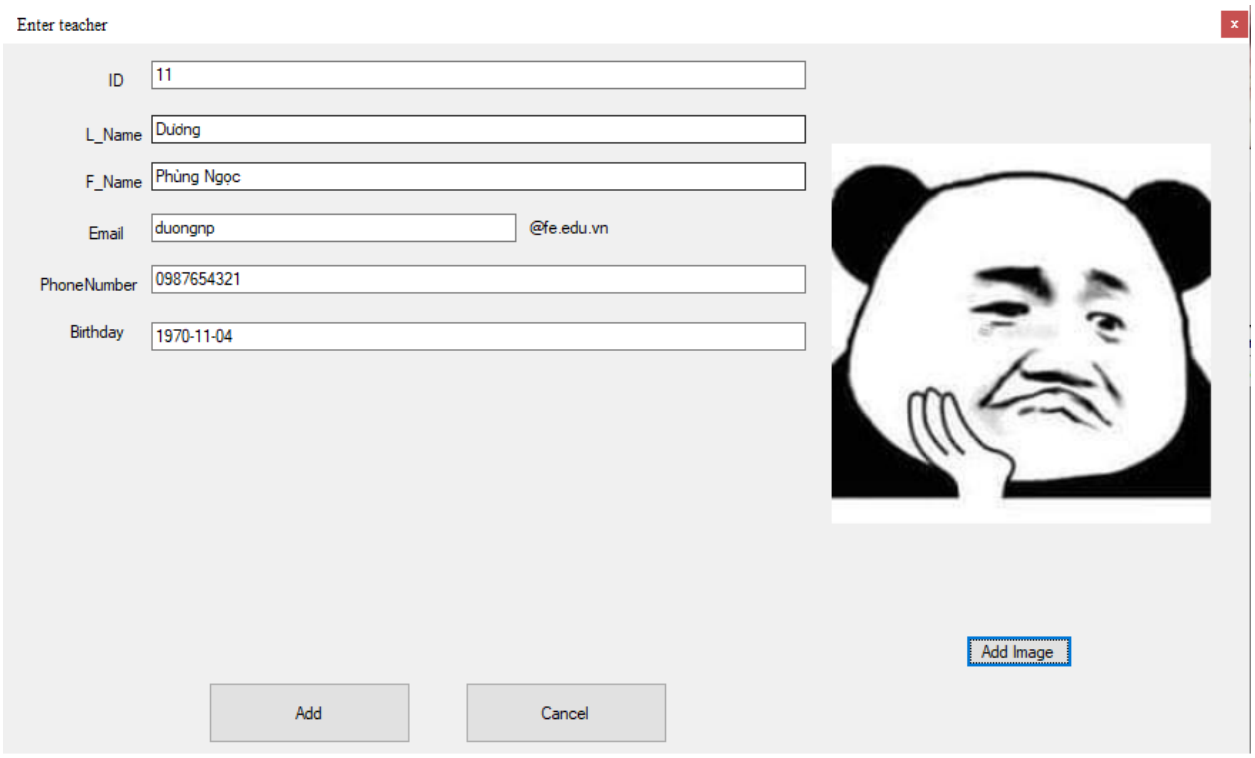
```
EXEC dbo.insertupdatedeleteRegister @classid = '',     -- varchar(50)
                                    @teacherid = '',   -- varchar(50)
                                    @studentid = '',   -- varchar(50)
                                    @subjectid = 0,    -- int
                                    @newclass = 'GCH0718',    -- varchar(50)
                                    @newteacher = '2', -- varchar(50)
                                    @newstudent = 'GCH20001', -- varchar(50)
                                    @newsubject = 1618,  -- int
                                    @statement = 'Insert'    -- varchar(50)
```

In case a student has relearn a subject more than 3 times. Program wont allow to add him to any course that have these subject.

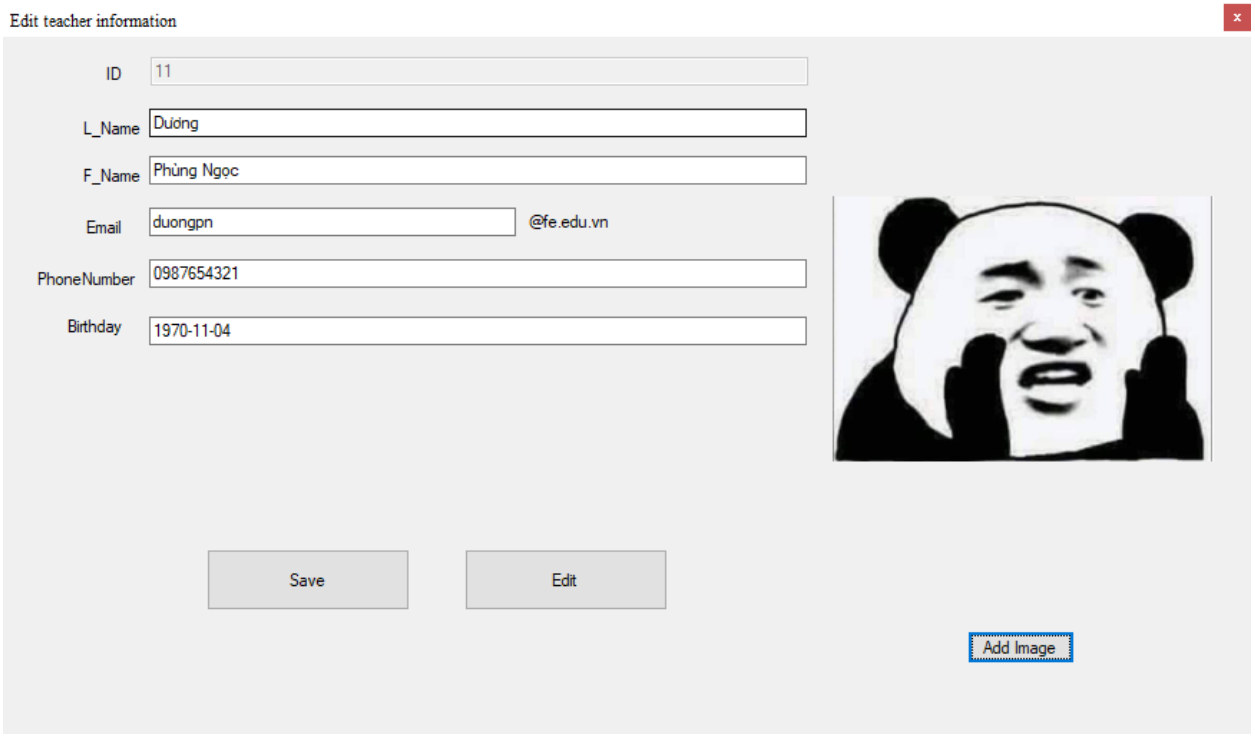
```
CREATE TRIGGER maximumRegister ON dbo.RegisterStudent
AFTER INSERT,UPDATE
AS
    IF EXISTS(SELECT COUNT(*) FROM Inserted AS i, dbo.RegisterStudent AS c
    WHERE
             i.Student_ID = c.Student_ID AND
             i.Subject_ID = c.Subject_ID
    GROUP BY i.Subject_ID, i.Student_ID
    HAVING COUNT(*) > 3)
    BEGIN
        PRINT 'Student cant relearn  one subject more than 3 times'
        ROLLBACK
    END
    ELSE PRINT 'Successful'
```

8. Add teacher

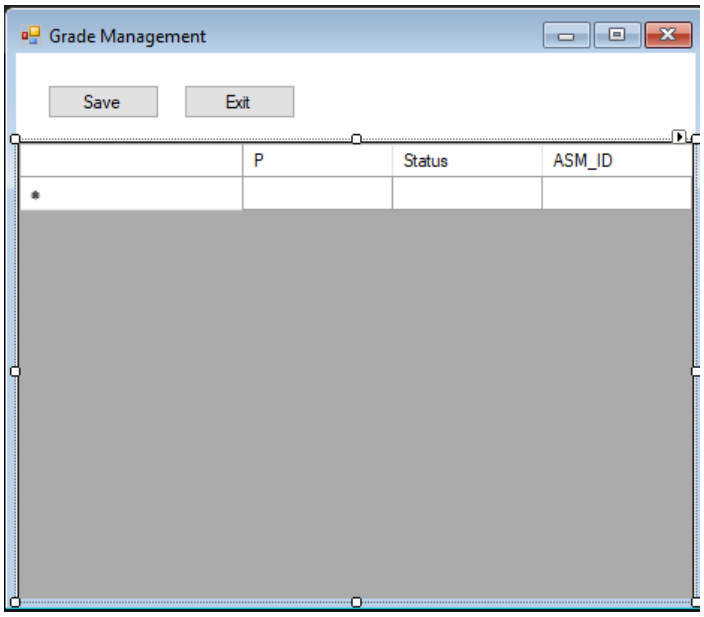
"Teacher add form" has a similar mechanism of operation for students. It only different in domain name of email.



9. Edit teacher information



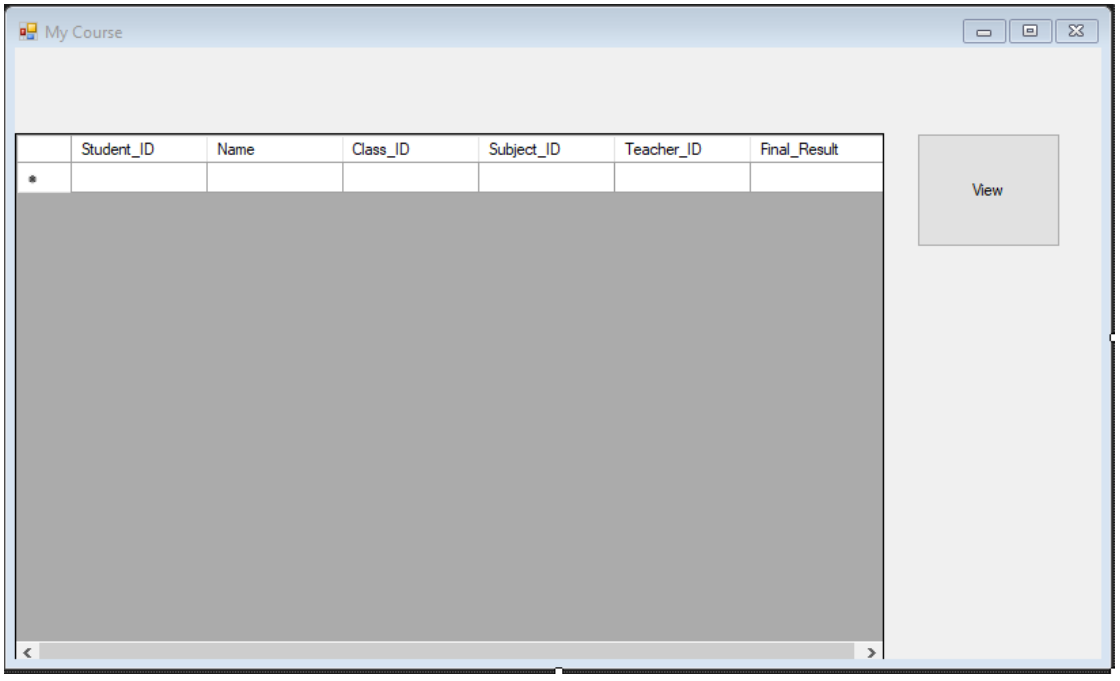
10. Add Grade for StudentRegister



In this form, user can enter grade for each criterion of student. 0 is false and 1 is pass. After that user must enter asm_id to finish one criterion.

```
EXEC dbo.insertupdatedeleteP @id = 1,         -- int
                             @status = 0, -- bit
                             @asmid = 1,      -- int
                             @statement = 'Insert' -- varchar(50)
```

c) Student

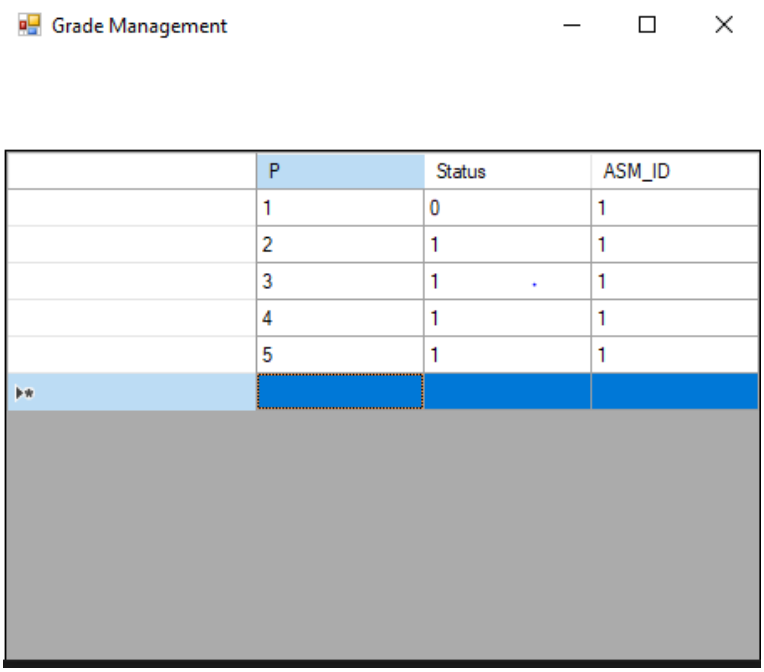


In this form, student can see list of all their course and final result. It will show NULL if student is still studying or not grading.

Query

```
EXEC dbo.getCourseGradeStudent @student = 'GCH20001'
```

They can click to one course in datagridview and click button View to see detail about criteria.



Query

```
EXEC dbo.getGrade @class = 'GBH0725',    -- varchar(50)
                  @teacher = '3', -- varchar(50)
                  @subject = 485,  -- int
                  @student = 'GBH20011'  -- varchar(50)
```

III. Development Evaluation (M3, D2)

(M3)Explain how your queries fit with the system requirements which support business decisions (such as statistical reports: e.g Numbers of orders based on day/month/years; Numbers of top products,etc…)

In the previous report, we have we recorded the following user requests.

(D2): Evaluate critically whether your solutions can fit with the requirements, and explain if the solution can help to improve the business better (more effective, save time, ….)

a) Requirement of teacher and staff

Requirement	Query	Explain	Status	Evaluate

Input function: enter student,teacher, class,course, subject, registerstudent information with many data fields	Student
	`EXEC dbo.insertupdatedeleteStudent @id = 'GCH20015', -- varchar(50)`
	`@lname = N'Minh', -- nvarchar(50)`
	`@fname = N'Phạm Anh', -- nvarchar(50)`
	`@email = '20015@fpt.edu.vn', -- varchar(50`
	`@phone = '0987654321', -- char(10)`
	`@sex = 'Male', -- varchar(50)`
	`@date = '1997-03-10', -- date`
	`@major = 'GCH', -- varchar(50)`
	`@image = '', -- varchar(50)`
	`@statement = 'Insert' -- varchar(50)`
		This query can enter student information as required.	Achived	Database has been build for storage many information field for manage student grade convenient and accurate

	Class
	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50)`
	`@name = 'Data structure and algorithm', -- varchar(50`
	`@slot = 40, -- int`
	`@description = '', -- varchar(50)`
	`@statement = 'Insert' -- varchar(50)`
		This query can enter class information as required.	Achived	

	Course
	`EXEC dbo.insertupdatedeleteCourse @classid = '', -- varchar(50)`
	`@teacherid = '', -- varchar(50)`
	`@subjectid = 0, -- int`
	`@date = '2020-03-10', -- date`
	`@newclass = 'GCH0718', -- varchar(`
	`@newteacher = '3', -- varchar(50)`
	`@newsubject = 1623, -- int`
	`@statement = 'Insert' -- varchar(5`
		This query can enter course information as required.	Achived	

	Subject
	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50)`
	`@name = 'Data structure and algorithm', -- varchar(50`
	`@slot = 40, -- int`
	`@description = '', -- varchar(50)`
	`@statement = 'Insert' -- varchar(50)`
		This query can enter subject information as required.	Achived	

	Teacher
	`EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varchar(50`
	`@name = '', -- varchar(50)`
	`@date = '2020-03-10', -- date`
	`@statement = 'Insert' -- varchar(50`
		This query can enter teacher information as required.	Achived	

	RegisterStudent
	`EXEC dbo.insertupdatedeleteRegister @classid = '', -- varchar(50)`
	`@teacherid = '', -- varchar(50)`
	`@studentid = '', -- varchar(50)`
	`@subjectid = 0, -- int`
	`@newclass = 'GBH0725', -- varchar(50)`
	`@newteacher = '3', -- varchar(50)`
	`@newstudent = 'GBH20010', -- varchar(50)`
	`@newsubject = 485, -- int`
	`@statement = 'Insert' -- varchar(50)`
		This query can enter RegisterStudent information as required.	Achived	

Grading function: We need to enter exactly grade for student in each subject and in case they are study a subject more than one time
 	
Beside, The school's grading system is also quite special with many criteria and many assignment for one subject
	Insert for RegisterStudent
	`EXEC dbo.insertupdatedeleteRegister @classid = '', -- varchar(50)`
	`@teacherid = '', -- varchar(50)`
	`@studentid = '', -- varchar(50)`
	`@subjectid = 0, -- int`
	`@newclass = 'GBH0725', -- varchar(50)`
	`@newteacher = '3', -- varchar(50)`
	`@newstudent = 'GBH20010', -- varchar(50)`
	`@newsubject = 485, -- int`
	`@statement = 'Insert' -- varchar(50)`
		Because, one student can learn a subject in many course.	Achived	The design and relationship of entities in database are appropriate to determine the grades of different student registrations.
				So we can classify students' scores through learning sessions

	Insert ASM
	`EXEC dbo.insertupdatedeleteASM @newclass = 'GCH0718', -- varchar(50)`
	`@newteacher = '2', -- varchar(50)`
	`@newsubject = 1623, -- int`
	`@newstudent = 'GCH20007', -- varchar(50)`
	`@class = '', -- varchar(50)`
	`@teacher = '', -- varchar(50)`
	`@subject = 0, -- int`
	`@student = '', -- varchar(50)`
	`@statement = 'Insert' -- varchar(50)`
		ASM_ID is set for identity Because a subject can have many asm. To define who is the owner of asm. We will have RegisterStudent key.	Achived	

	Insert P,M,D
	`EXEC dbo.insertupdatedeleteP @id = 1, -- int`
	`@status = 1, -- bit`
	`@asmid = 2, -- int`
	`@statement = 'Insert' -- varchar(50)`
	`EXEC dbo.insertupdatedeleteM @id = 1, -- int`
	`@status = 0, -- bit`
	`@asmid = 2, -- int`
	`@statement = 'Insert' -- varchar(50)`
	`EXEC dbo.insertupdatedeleteD @id = 1, -- int`
	`@status = 0, -- bit`
	`@asmid = 2, -- int`
	`@statement = 'Insert' -- varchar(50)`
		P,M,D id are also set identity because one asm has many criterion. ASM_Id define asm owner of P,M,D	Achived	

Edit function: The system must have a function for edit all	Update Student	This query is allows users to update student information	Achived	In the development process, many procedures has been created included procedure for edit

information fields exclude some unique fields like id	EXEC dbo.insertupdatedeleteStudent @id = 'GBH20015', -- varchar(50) @lname = N'Minh', -- nvarchar(50) @fname = N'Phạm Anh', -- nvarchar(50) @email = '20015@fpt.edu.vn', -- varcha @phone = '0987654321', -- char(10) @sex = 'Female', -- varchar(50) @date = '2997-03-10', -- date @major = 'GBH', -- varchar(50) @image = '', -- varchar(50) @statement = 'Update' -- varchar(50)			function. Procedure allow user can edit information of any table in database with their id or primary key. Beside, I have add more option allow user can also change id and primary key of few table like course and register student.
	Update Class EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varc @name = 'updated', -- varc @date = '2020-03-10', -- date @statement = 'Update' -- varch	This query is allows users to update class information	Achived	
	Update Teacher EXEC dbo.insertupdatedeleteTeacher @id = '8', -- varchar(50) @lname = N'Vượng', -- nvarchar(50) @fname = N'Nguyễn Nhật', -- nvarchar(50) @phone = '2132189898', -- char(10) @email = 'update_email@fe.edu.vn', -- v @date = '2020-03-10', -- date @statement = 'Update' -- varchar(50)	This query is allows users to update teacher information	Achived	
	Update Subject EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50) @name = 'Data structure and algorithm', -- v @slot = 40, -- int @description = 'Update description', -- varchar(50) @statement = 'Update' -- varchar(50)	This query is allows users to update subject information	Achived	
	Update Course EXEC dbo.insertupdatedeleteCourse @classid = 'GCH0718', -- var @teacherid = '3', -- varchar(5 @subjectid = 1620, -- int @date = '2020-03-10', -- date @newclass = 'GCH0717', -- var @newteacher = '2', -- varchar(5 @newsubject = 1625, -- int @statement = 'Update' -- varc	This query is allows users to update course information	Achived	
	Update RegisterStudent EXEC dbo.insertupdatedeleteRegister @classid = 'GBH0725', -- varc @teacherid = '3', -- varchar(50 @studentid = 'GBH20010', -- var @subjectid = 485, -- int @newclass = 'GBH0727', -- varc @newteacher = '2', -- varchar(50 @newstudent = 'GBH20009', -- var @newsubject = 487, -- int @statement = 'Update' -- varch	This query is allows users to update a registration information	Achived	
	Update P (Similar for M,D) EXEC dbo.insertupdatedeleteP @id = 1, -- int @status = 0, -- bit @asmid = 1, -- int @statement = 'Update' -- varcha	This query is allows users to update status of criteria	Achived	
Delete function: Many information field are expired or	Detele Student EXEC dbo.insertupdatedeleteStudent @id = 'GCH20015', -- varchar(50 @lname = N'', -- nvarchar(50) @fname = N'', -- nvarchar(50) @email = '', -- varchar(50) @phone = '', -- char(10) @sex = '', -- varchar(50) @date = '2020-03-10', -- date @major = '', -- varchar(50) @image = '', -- varchar(50) @statement = 'Delete' -- varchar(50)	This query can remove a student from database	Achived	Similar edit function, we have some procedure for delete any record. Beside, constraint foreign key of many table

wrong and don't need to storage anymore. They can delete it.	Delete Class
	`EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varc`
	`@name = '', -- varchar(50)`
	`@date = '2020-03-10', -- date`
	`@statement = 'Delete' -- varch`
		This query can remove a class from database	Achived	has been set update and delete cascade. It help user can edit or remove in parent table but information in child table will be updated instantly.

	Delete Teacher
	`EXEC dbo.insertupdatedeleteTeacher @id = '8', -- varchar(`
	`@lname = N'', -- nvarchar(`
	`@fname = N'', -- nvarchar(`
	`@phone = '', -- char(10)`
	`@email = '', -- varchar(5`
	`@date = '2020-03-10', -- date`
	`@statement = 'Delete' -- varc`
		This query can remove a teacher from database	Achived	

	Delete Subject
	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varcha`
	`@name = '', -- varchar(5`
	`@slot = 0, -- int`
	`@description = '', -- varchar(5`
	`@statement = 'Delete' -- varc`
		This query can remove a subject from database	Achived	

	Delete Course
	`EXEC dbo.insertupdatedeleteCourse @classid = 'GCH0717', -- var`
	`@teacherid = '2', -- varchar(`
	`@subjectid = 1625, -- int`
	`@date = '2020-03-10', -- date`
	`@newclass = '', -- varchar(5`
	`@newteacher = '', -- varchar(5`
	`@newsubject = 0, -- int`
	`@statement = 'Delete' -- varc`
		This query can remove a course from database	Achived	

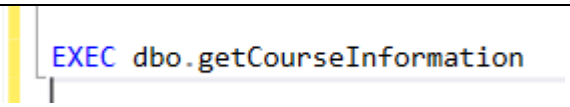
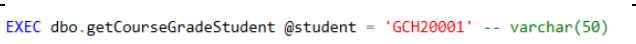
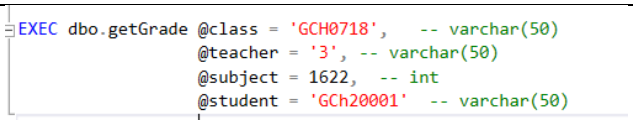
	Delete RegisterStudent
	`EXEC dbo.insertupdatedeleteRegister @classid = 'GBH0727', -- varc`
	`@teacherid = '2', -- varchar(5`
	`@studentid = 'GBH20009', -- var`
	`@subjectid = 487, -- int`
	`@newclass = '', -- varchar(50`
	`@newteacher = '', -- varchar(50`
	`@newstudent = '', -- varchar(50`
	`@newsubject = 0,-- int`
	`@statement = 'Delete' -- varch`
		This query can remove a registration of student from database	Achived	

	Delete P (Similar for M,D)
	`EXEC dbo.insertupdatedeleteP @id = 108, -- int`
	`@status = 0, -- bit`
	`@asmid = 1, -- int`
	`@statement = 'Delete' -- varch`
		This query can remove a criteria from database	Achived	

Get grade of student in course after	`EXEC dbo.getCourseGrade @class = 'GCH0718', -- varchar(50)`
	`@teacher = '2', -- varchar(50)`
	`@subject = 1623 -- int`
		This query can get final result of student in a course base on p,m,d table	Achived	I have created the a function for find the asm_id and a function for calculate the final result of student then used it in procedure for find the final result of student in a course. I help teacher and staff can calculate the final result of student are refer or pass,merit,distinctio

				n base on their criterion from P,M,D table.

b) Requirement Of Student

1	View their course		This query can show list of course for student	Achived	I have create a procedure to allow student can see the list of course with class_id and teacher name, subjectid,… It help solve the requirement of student and help them can distinguish between the courses as well as easier to manage
2	View their Final Resul (Can't see other result)		This query can show all course of a student and their result	Achived	I have created a function and a procedure to finish this query. It can help student see all their result in all course but they can't see other result.
3	View detail about result		This query can show detail about P,M,D criterion.	Achived	Beside the final result. Student can see detail about their asm and result. In case they got refer, they can know how and why?

IV. Testing (P4, M4)

1. Test log

UNIVERSITY of GREENWICH

Alliance with FPT Education

Test	Title	Input/query	Expected Result	Result	Status

1	Insert Student's information	`EXEC dbo.insertupdatedeleteStudent @id = 'GCH20015', -- varchar(50)`
		`@lname = N'Minh', -- nvarchar(50)`
		`@fname = N'Phạm Anh', -- nvarchar(50)`
		`@email = '20015@fpt.edu.vn', -- varchar(50)`
		`@phone = '0987654321', -- char(10)`
		`@sex = 'Male', -- varchar(50)`
		`@date = '1997-03-10', -- date`
		`@major = 'GCH', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update Student's information	`EXEC dbo.insertupdatedeleteStudent @id = 'GBH20015', -- varchar(50)`
		`@lname = N'Minh', -- nvarchar(50)`
		`@fname = N'Phạm Anh', -- nvarchar(50)`
		`@email = '20015@fpt.edu.vn', -- varchar(50)`
		`@phone = '0987654321', -- char(10)`
		`@sex = 'Female', -- varchar(50)`
		`@date = '2997-03-10', -- date`
		`@major = 'GBH', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete Student's information	`EXEC dbo.insertupdatedeleteStudent @id = 'GCH20015', -- varchar(50)`
		`@lname = N'', -- nvarchar(50)`
		`@fname = N'', -- nvarchar(50)`
		`@email = '', -- varchar(50)`
		`@phone = '', -- char(10)`
		`@sex = '', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@major = '', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

2	Insert Class's information	`EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varchar(50)`
		`@name = '', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update Class's information	`EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varchar(50)`
		`@name = 'updated', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete Class's information	`EXEC dbo.insertupdatedeleteClass @id = 'GCH0729', -- varchar(50)`
		`@name = '', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

3	Insert Teacher's information	`EXEC dbo.insertupdatedeleteTeacher @id = '8', -- varchar(50)`
		`@lname = N'Vượng', -- nvarchar(50)`
		`@fname = N'Nguyễn Nhật', -- nvarchar(50)`
		`@phone = '0998989898', -- char(10)`
		`@email = 'vuongnn@fe.edu.vn', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update Teacher's information	`EXEC dbo.insertupdatedeleteTeacher @id = '8', -- varchar(50)`
		`@lname = N'Vượng', -- nvarchar(50)`
		`@fname = N'Nguyễn Nhật', -- nvarchar(50)`
		`@phone = '2132189898', -- char(10)`
		`@email = 'update_email@fe.edu.vn', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete Teacher's information	`EXEC dbo.insertupdatedeleteTeacher @id = '8', -- varchar(50)`
		`@lname = N'', -- nvarchar(50)`
		`@fname = N'', -- nvarchar(50)`
		`@phone = '', -- char(10)`
		`@email = '', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

4	Insert Subject's information	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50)`
		`@name = 'Data structure and algorithm', -- varchar(50)`
		`@slot = 40, -- int`
		`@description = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update Subject's information	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50)`
		`@name = 'Data structure and algorithm', -- varchar(50)`
		`@slot = 40, -- int`
		`@description = 'Update description', -- varchar(50)`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete Subject's information	`EXEC dbo.insertupdatedeleteSubject @id = '1649', -- varchar(50)`
		`@name = '', -- varchar(50)`
		`@slot = 0, -- int`
		`@description = '', -- varchar(50)`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

5	Insert Course's information	`EXEC dbo.insertupdatedeleteCourse @classid = '', -- varchar(50)`
		`@teacherid = '', -- varchar(50)`
		`@subjectid = 0, -- int`
		`@date = '2020-03-10', -- date`
		`@newclass = 'GCH0718', -- varchar(50)`
		`@newteacher = '3', -- varchar(50)`
		`@newsubject = 1623, -- int`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update Course's information	`EXEC dbo.insertupdatedeleteCourse @classid = 'GCH0718', -- varchar(50)`
		`@teacherid = '3', -- varchar(50)`
		`@subjectid = 1620, -- int`
		`@date = '2020-03-10', -- date`
		`@newclass = 'GCH0717', -- varchar(50)`
		`@newteacher = '2', -- varchar(50)`
		`@newsubject = 1625, -- int`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete Course's information	`EXEC dbo.insertupdatedeleteCourse @classid = 'GCH0717', -- varchar(50)`
		`@teacherid = '2', -- varchar(50)`
		`@subjectid = 1625, -- int`
		`@date = '2020-03-10', -- date`
		`@newclass = '', -- varchar(50)`
		`@newteacher = '', -- varchar(50)`
		`@newsubject = 0, -- int`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

6	Insert RegisterStudent's information	`EXEC dbo.insertupdatedeleteRegister @classid = '', -- varchar(50)`
		`@teacherid = '', -- varchar(50)`
		`@studentid = '', -- varchar(50)`
		`@subjectid = 0, -- int`
		`@newclass = 'GBH0725', -- varchar(50)`
		`@newteacher = '3', -- varchar(50)`
		`@newstudent = 'GBH20010', -- varchar(50)`
		`@newsubject = 485, -- int`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update RegisterStudent's information	`EXEC dbo.insertupdatedeleteRegister @classid = 'GBH0725', -- varchar(50)`
		`@teacherid = '3', -- varchar(50)`
		`@studentid = 'GBH20010', -- varchar(50)`
		`@subjectid = 485, -- int`
		`@newclass = 'GBH0727', -- varchar(50)`
		`@newteacher = '2', -- varchar(50)`
		`@newstudent = 'GBH20009', -- varchar(50)`
		`@newsubject = 487, -- int`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete RegisterStudent's information	`EXEC dbo.insertupdatedeleteRegister @classid = 'GBH0727', -- varchar(50)`
		`@teacherid = '2', -- varchar(50)`
		`@studentid = 'GBH20009', -- varchar(50)`
		`@subjectid = 487, -- int`
		`@newclass = '', -- varchar(50)`
		`@newteacher = '', -- varchar(50)`
		`@newstudent = '', -- varchar(50)`
		`@newsubject = 0,-- int`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

7	Insert RegisterStudent's information	`EXEC dbo.inserupdatedeleteASM @class = 'GBH0725', -- varchar(50)`
		`@teacher = '3', -- varchar(50)`
		`@subject = 485, -- int`
		`@student = 'GBH20007', -- varchar(50)`
		`@asm_Id = 0, -- int`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update RegisterStudent's information	`EXEC dbo.inserupdatedeleteASM @class = 'GBH0727', -- varchar(50)`
		`@teacher = '2', -- varchar(50)`
		`@subject = 487, -- int`
		`@student = 'GBH20008', -- varchar(50)`
		`@asm_Id = 21, -- int`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete RegisterStudent's information	`EXEC dbo.inserupdatedeleteASM @class = '', -- varchar(50)`
		`@teacher = '', -- varchar(50)`
		`@subject = 0, -- int`
		`@student = '', -- varchar(50)`
		`@asm_Id = 21, -- int`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

8	Insert P Grade	`EXEC dbo.insertupdatedeleteP @id = 0, -- int`
		`@status = 1, -- bit`
		`@asmid = 1, -- int`
		`@statement = 'Insert' -- varchar(50)`
			Added	(1 row affected)	Pass

	Update P Grade	`EXEC dbo.insertupdatedeleteP @id = 1, -- int`
		`@status = 0, -- bit`
		`@asmid = 1, -- int`
		`@statement = 'Update' -- varchar(50)`
			Updated	(1 row affected)	Pass

	Delete P Grade	`EXEC dbo.insertupdatedeleteP @id = 108, -- int`
		`@status = 0, -- bit`
		`@asmid = 1, -- int`
		`@statement = 'Delete' -- varchar(50)`
			Deleted	(1 row affected)	Pass

9	Insert validate Student' information	Insert validate data for Student_ID
		`EXEC dbo.insertupdatedeleteStudent @id = 'GGG12345', -- varchar(50)`
		`@lname = N'Anh', -- nvarchar(50)`
		`@fname = N'Nguyễn', -- nvarchar(50)`
		`@email = '12312@fpt.edu.vn', -- varchar(50)`
		`@phone = '1234567890', -- char(10)`
		`@sex = 'Male', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@major = 'GCH', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Student", column 'Student_ID'.	Pass

		Insert validate data for Student_Email
		`EXEC dbo.insertupdatedeleteStudent @id = 'GCH12345', -- varchar(50)`
		`@lname = N'Anh', -- nvarchar(50)`
		`@fname = N'Nguyễn', -- nvarchar(50)`
		`@email = '12312@fff.edu.vn', -- varchar(50)`
		`@phone = '1234567890', -- char(10)`
		`@sex = 'Male', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@major = 'GCH', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Student", column 'Student_Email'	Pass

		Insert validate data for Major
		`EXEC dbo.insertupdatedeleteStudent @id = 'GCH12345', -- varchar(50)`
		`@lname = N'Anh', -- nvarchar(50)`
		`@fname = N'Nguyễn', -- nvarchar(50)`
		`@email = '12312@fpt.edu.vn', -- varchar(50)`
		`@phone = '1234567892', -- char(10)`
		`@sex = 'Male', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@major = 'Dasuo', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Student", column 'Major'.	Pass

		Insert validate data for Student_Phone
		`EXEC dbo.insertupdatedeleteStudent @id = 'GCH12345', -- varchar(50)`
		`@lname = N'Anh', -- nvarchar(50)`
		`@fname = N'Nguyễn', -- nvarchar(50)`
		`@email = '12312@fpt.edu.vn', -- varchar(50)`
		`@phone = '123456h1asd7892', -- char(10)`
		`@sex = 'Male', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@major = 'GCH', -- varchar(50)`
		`@image = '', -- varchar(50)`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Student", column 'Student_Phone'	Pass

10	Insert validate Teacher' information	Insert validate data for Teacher_Phone
		`EXEC dbo.insertupdatedeleteTeacher @id = '9', -- varchar(50)`
		`@lname = N'Nguyễn', -- nvarchar(50)`
		`@fname = N'Anh', -- nvarchar(50)`
		`@phone = '1231231jk213', -- char(10)`
		`@email = 'nguyenanh@fe.edu.vn', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Teacher", column 'Teacher_Phone'.	Pass

		Insert validate data for Teacher_email
		`EXEC dbo.insertupdatedeleteTeacher @id = '9', -- varchar(50)`
		`@lname = N'Nguyễn', -- nvarchar(50)`
		`@fname = N'Anh', -- nvarchar(50)`
		`@phone = '1231231123', -- char(10)`
		`@email = 'nguyenanh@fap.edu.vn', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Teacher", column 'Teacher_Email'.	Pass

10	Insert validate Teacher' information	Insert validate data for Class_ID
		`EXEC dbo.insertupdatedeleteClass @id = 'BBB12312', -- varchar(50)`
		`@name = '', -- varchar(50)`
		`@date = '2020-03-10', -- date`
		`@statement = 'Insert' -- varchar(50)`
			Error	The conflict occurred in database "ASM2_1622", table "dbo.Class", column 'Class_ID'.	Pass

11	Login as Student (correct email and password	`EXEC dbo.login_student_check @mail = N'anhnqgch20001@fpt.edu.vn', -- nvarchar(50)` `@password = '123456789' -- varchar(50)`	Login successful	Login successful	Pass
	Login as Teacher (correct email and password	`EXEC dbo.login_teacher_check @mail = N'oanhnv@fe.edu.vn', -- nvarchar(50)` `@password = '123456789' -- varchar(50)`	Login successful	Login successful	Pass
12	Login as Student (incorrect email and password	`EXEC dbo.login_student_check @mail = N'anhnqgch20001@fpt.edu.vn', -- nvarchar(50)` `@password = 'a123456789' -- varchar(50)`	Login Failed	Login Failed	Pass
	Login as Teacher (incorrect email and password	`EXEC dbo.login_teacher_check @mail = N'oanhnv@fe.edu.vn', -- nvarchar(50)` `@password = 'b123456789' -- varchar(50)`	Login Failed	Login Failed	Pass
13	Add more than one	`INSERT dbo.Course` `VALUES('GCH0718',N'2',485,GETDATE() )`	Failed	Dont allow more than one teacher in course Msg 3609, Level 16, State 1, Line 652	Pass

	teacher for a course				
14	Register for student more than 3 time for a subject	`INSERT dbo.RegisterStudent` `VALUES('GCH0719', N'3', 485, 'GBH20007')`	Failed	Student cant relearn one subject more than 3 times Msg 3609, Level 16, State 1, Line 665	Pass

M4: You will need to critically review the actual outcomes of the tests against your expected outcomes; were your tests adequate and appropriate; did they cover all user requirements?

Most of the cases in which data were interacted were tested.

Besides the normal scenario, validate cases have also been tested. The results obtained were as expected.

Delete, Input, Update function have no error, all they have worked correct base on constraints. In Case, Input or update validate data, they won't allow to enter them into database obtain as expected.

Beside the regular constraint, trigger constraints are active and prevent data changes because of the predefined constraints.

Extremes of data still can't test in this time, so i willl prepare to test it in near time.

In conclusion, Almost test have covered all the user requirement and not have any error for now. Beside, User get any error or problem, i will fix it as soon as possible

V. Evaluation (M5, D3)

M5: Explains why, and how, validation/verification were used, or could have been used, in your final solution (from the screens designed and the supporting queries)

D3: Assess any future improvements that may be required to ensure the continued effectiveness of the database system.

VI. Technical Document

Author: Nguyễn Quốc Anh

Update History

Date	Version	Part	Detail	Implementer
10/2/2020	1	Create		Nguyễn Quốc Anh
17/2/2020	1.1	Edit	Define user requirement and business rule. Complete the design phase as required by the user include ERD, Physical design.	Nguyễn Quốc Anh
9/3/2020	2 (Current Version)	Edit	Edit the design for to optimize Change some constraint in relative of entity Add new entity to erd Complete the database: Example data has been inputed Complete the user interface. Test plan has been performed in normal case and validate case.	Nguyễn Quốc Anh

I. Introduction

1. History

I'm an employed as a Database Developer in FPT Software. The company has been approached by FPT university which is expanding due to the growth of the number of students. FPT is currently facing difficulties in dealing with managing the university. They have more than 60 class with 20-30 students in each class in only Hanoi base. Their current system is showing their weakness like slow loading. It's take so much time if someone want to setrieve student information. Beside, the old system is messy, so the management become more difficulty when the amount of students are increase so quickly.Firsly, student will submit their assignments in CMS website. Each class will have other course for each subject, student must to submit on exactly course which they are studying. After submitted, student can only view their submission and can't see other submission. can read all student's submission then grade them, give some comment about assignment and notify the result for student. Student can see the final result is refer,pass,merit,distinction. also attach a pdf file which details of criteria achieved or failed in assignment so student can see it. But it's not the end of grading process. After finish grading on cms. They need to enter them into the excel file which they got from staff. Lastly, staff will get excel file from teacher then enter student's grade to storage system. These process has many problem, must enter grade to excel file and staff enter it again to storage system. It is very confusing and time consuming. Moreover, an old system don't has many function which help can directly enter student grade. In conclusion, they need more effective solution for manage students.

2. Scope

We hope, with this new database system. We can solve the outstanding problems of the old system. Besides, we can implement some other features to improve the experience of teachers as well as help in managing students and grades. From now, teacher can input student's grade directly instead of staff.

3. Objectives

- Student management.
- Student grading system.
- Teacher and subject registration.

4. SRS (System Requirement Specification).

All atributes are nessesary about Student, Teacher,Class , Subject and the most important is grade.

a) User requirements:

This system can be use by teacher, student any other staff.

- Student
- Allow student can see their result and information but deny to see other result and information.
- Student will get notification when their submission were grade.
- Student can search information about course. They can see course name with subject and class but they need to enter password (or not) to join for see content inside.
- Student can register to relearn if they have failed in any subject. But they still need to wait if there are no class this time.
- Teacher and staff
- Input function: Department of studies need enter student information with many data fields are mentioned above. Beside, they also need to add information about subject, teacher, class. Teacher can directly enter grade of student instead of sending them to education department.
- Grading function: The most important is grading mechanism. We need to enter exactly grade for student in each subject and in case they are study a subject more than one time. The system need to storage different result for each time.
- Edit function: Mistakes are inevitable in process of entering information. The system must have a function for edit all information fields exclude some unique fields like id.
- Search function:
- Delete function: Sometime, they have many information field are expired or wrong and don't need to storage anymore. They can delete it.
  Display function

b) System Requirement

- This datatabase is used for grade management system and student management system of the University of greenwich.
- This database can be used for multi-platform. Staff and teacher can use input data to it through Management application or website. Student can see their result in website.
- Decentralize user: Staff can see and interact with all information in database included grade,student information, teacher, course, subject. Teacher can only manage grade in courses which they manage. Student can only see student list and information about self.

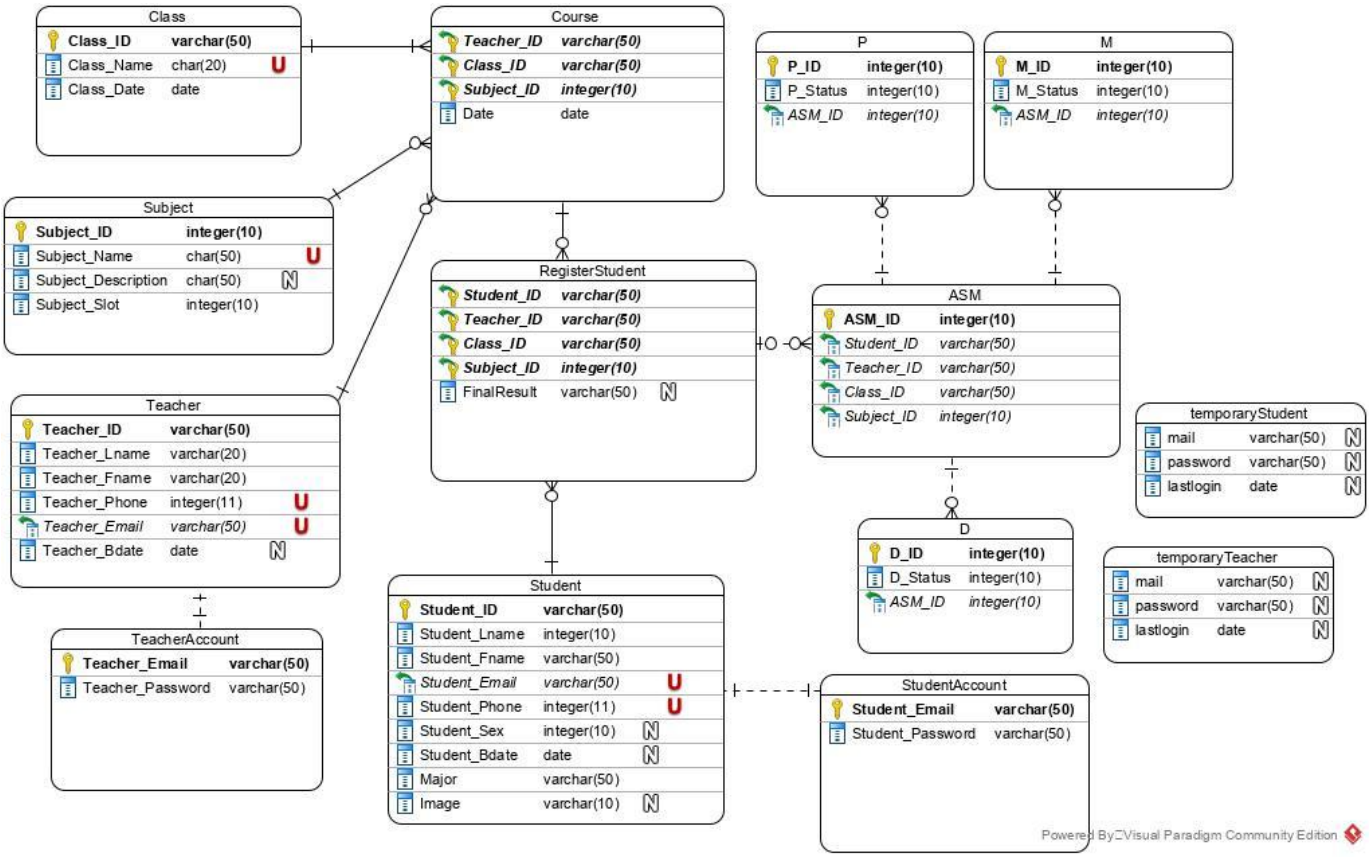
c) Businesss Rule

- The grade system are including many subject. We will have more than one assignment in one subject. Beside, we will have many criteria for each assignment. There are pass (P), merit (M) and distinction

(D). There are different number of each criteria in one assignment. For example: 1st assignment of networking subject has 7 pass, 4 merit, 3 distinction criteria. But in programming, we will have 2 asm with total 8 pass, 5 merit and 4 distinction criteria. They are divided to each assignment.

- A course is define with class and subject. Beside one course is only teach by one teacher.
- With each criteria, we will have status are pass or not pass. If you have one criteria is not pass. You will fail the assignment.
- The learning process of a class will take place continuously according to the sequence of subjects. But a student can study different subject in different class.
- Teacher can teach many subjects in many different class.
- Because, Students need to relearn a subject if they are failed in that subject before. So one student can have many different result for each subjects.

5. ERD



a) Entity

Entity	Detail
Student	It has all information which use in this grade management. Beside student_ID, is has two unique attribute to define a student with other are email and phone number
Class	To storage information about one class with class_id or class_name (if has) and date start
Subject	To storage information about a subject with subject_id, subject_name, description of subject and how long is it with slot attribute.
Teacher	Similar the student, it use to storage information of teacher.
Course	In this design by user requirement, course is define by a class with a subject and is teach by a teacher.
RegisterStudent	It can mean the result of a student in a course, So it has all attribute to define a course and student_id to distinguish between students
ASM	It use to storage one or more information about asm of course
P,M,D	It use to storage result of each criterion of student assignment. To Distinguish between asm, we add ASM_ID attribute to define that asm's criterion
Teacher/ Student account	It use to storage teacher and student account and password. In this system, login_id is also email
Temporary	It use save the login log of teacher and student. It's also help in trigger for login system

b) Relationship

Because, one student and teacher only have one email as their account and one account in account table is also use for one student or teacher so their relationship are one to one.

Teacher,Subject,Class can in many course but a course only have one of each them, so their relationship is one and many. (One Class to Many Course, One Teacher to Many Course, One Subject to Many Course),

Course can help many student join, So it can have many result/registration of student. But a result/registration is only for one course. So they relationship are one to many (one course to many registerStudent)

Student can study in many course and have many result, but a result/registration is only for one student

So their relationship are one to many (one student to many RegisterStudent)

Final result is calculated by many asm but one record in asm is only for one result, so their relationship are one to many (one RegisterStudent to many ASM)

One ASM have many P,M,D criteria but one criteria is only for one ASM. So we have one to many relationship.

6. Physical Design

- Student

Column Name	Data type	Constraint
Student_ID	Nvarchar(50)	Primary Key, CHECK (student_ID LIKE (G'[B,C,D]H[0-9] [0-9] [0-9] [0-9] [0-9])
Student_Lname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Student_Fname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Student_Email	Nvarchar(50)	Not null, unique, Foreign Key
Student_Phone	Char(11)	Not null,unique, CHECK (Not like ('%[^0-9]%')
Student_Sex	Varchar(50)	Not null, CHECK (in (Male,Female)
Student_Bdate	Date	
Major	Nvarchar(50)	Not Null, CHECK (in GCH,GBH,GDH)
Image	Nvarchar(50)	

- Class

Column Name	Data type	Constraint
Class_ID	varchar(50)	Primary key, CHECK Like (G'[B,C,D]H[0-9] [0-9] [0-9] [0-9])
Class_Name	Varchar(20)	Not null, unique
Class_Date	date	Not null

- Subject

Column Name	Data type	Constraint
Subject_ID	Int	Primary Key, IDENTITY(1,1)
Subject_Name	Varchar(50)	Not null, unique
Subject_Slot	Int	Not null
Subject_Description	Varchar(50)	

- Teacher

Column Name	Data type	Constraint
Teacher_ID	nvarchar(50)	Primary Key,
Teacher_Lname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Teacher_Fname	Nvarchar(50)	Not null, CHECK Like ('%[^0-9]%')
Teacher_Phone	Varchar(50)	Not null, unique, CHECK Like ('%[^0-9]%')
Teacher_Email	nvarchar(50)	Not null, unique, Foreign Key
Teacher_Bdate	Date	

- Course

Column Name	Data type	Constraint
Teacher_ID	Nvarchar(50)	Primary key ,Foreign key
Class_ID	varchar(50)	Primary key ,Foreign key
Subject_ID	int	Primary key ,Foreign key
Date	Date	Not null

- RegisterStudent

Column Name	Data type	Constraint
Student_ID	Nvarchar(50)	Primary key ,Foreign Key
Teacher_ID	Nvarchar(50)	Primary key ,Foreign key
Class_ID	varchar(50)	Primary key ,Foreign key
Subject_ID	int	Primary key ,Foreign key

- ASM (Assignment)

Column Name	Data type	Constraint
ASM_ID	Int	Primary Key, IDENTITY(1,1)
Student_ID	Nvarchar(50)	Foreign Key
Teacher_ID	Nvarchar(50)	Foreign key
Class_ID	varchar(50)	Foreign key
Subject_ID	int	Foreign key

- P

Column Name	Data type	Constraint
P_ID	Int	Primary Key, IDENTITY(1,1)
P_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- M

Column Name	Data type	Constraint
M_ID	Int	Primary Key, IDENTITY(1,1)
M_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- D

Column Name	Data type	Constraint
D_ID	Int	Primary Key, IDENTITY(1,1)

D_Status	Bool	NOT NULL
ASM_ID	Int	Foreign Key, NOT NULL

- TeacherAccount

Column Name	Data type	Constraint
Teacher_Email	Nvarchar(50)	Primarykey, Check (Teacher_Email LIKE '%@fe.edu.vn')
Teacher_Password	Varchar(50)	

- StudentAccount

Column Name	Data type	Constraint
Student_Email	Nvarchar(50)	Primarykey, Check (Student _Email LIKE '%@fpt.edu.vn')
Student _Password	Varchar(50)	

- temporaryStudent

Column Name	Data type	Constraint
Mail	Nvarchar(50)	
Password	Varchar(50)	
Lastlogin	Date	

- temporaryTeacher

Column Name	Data type	Constraint
Mail	Nvarchar(50)	
Password	Varchar(50)	
Lastlogin	Date	

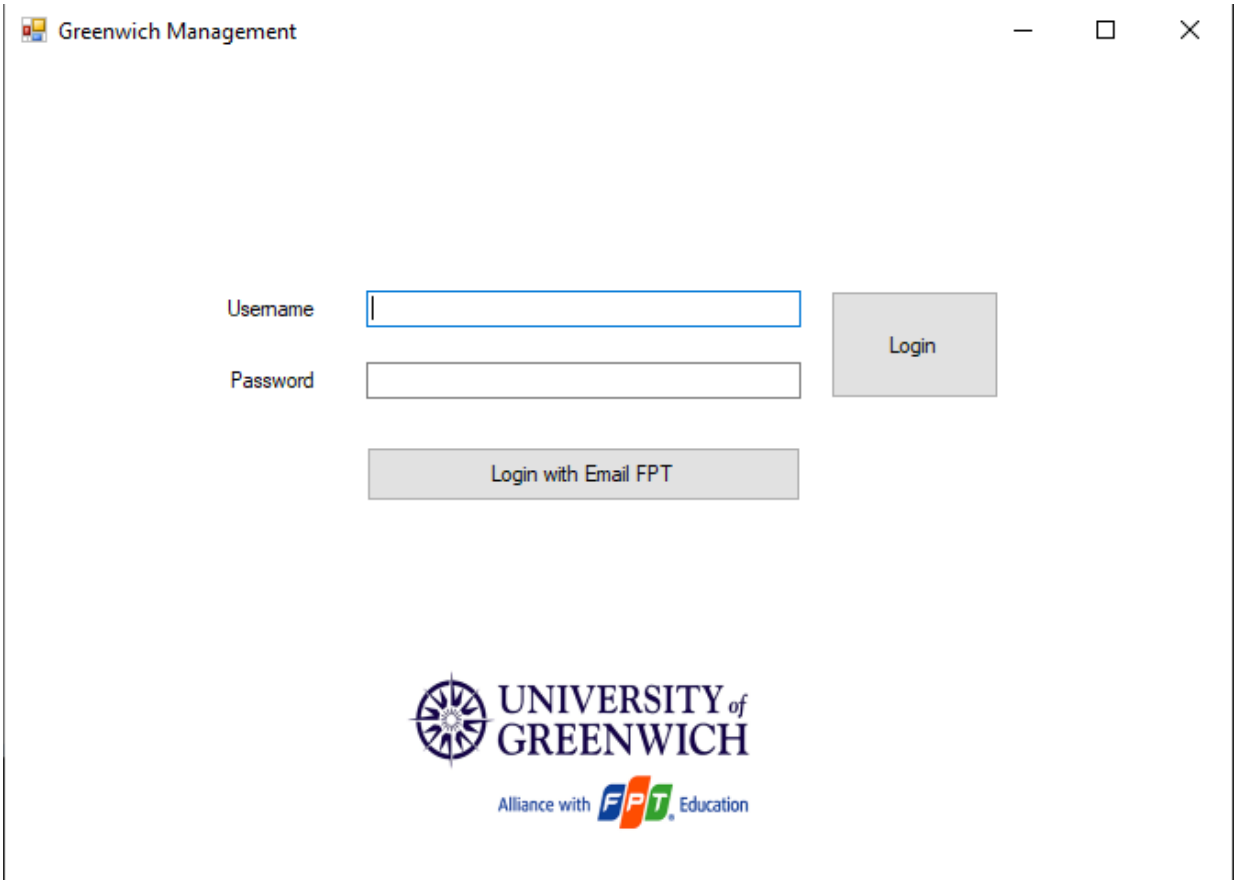
II.Possible and realistic improvements

Pros	Cons	Improvements
The design has met the requirements that greenwich FPT university have requested to be able to get new database to resolve their problem when the number of students increased so fast	Some tables are redundant and the use value is not really high.	There is a need for more in-depth study of user requirements to find problems that users may encounter but have not been addressed yet.
Additional constraint to validate data have been added to ensure that data is not corrupted during use	Haven't tested on extremely large amounts of data yet	More large-scale tests are needed to detect if there is a problem.
The winform interface for the management program has been implemented and the user object is divided.	The interface is still not nice and easy to see.	Research more on user interface and user experience to improve program interface
System has many procedure. It help program improve working performance and shorten run time.	Some combined procedure are trouble and hard for user to use. Because user must enter many fields so they can be confused.	I will omitted few combined procedure or divided to small part that more dedicated to each issue.
The trigger to closely check the validity of the data. Thereby, we can resolve the requirement of user.		I will continue to collect more user requirement incurred in using process to development and fix.
Some solution for security has been applied like access permission are different and login log for check last login if someone logged to their account.		Need apply more solution for security in future like restrict the process database, set a strong admin password, audit DB login.
Sample data has been imported successfully.	Haven't tested when inputting an extremely large amount of data	Need do more test to try the database capabilities. Beside, we need a mechanism for backup data when database got problem.

VII. User Document

- The user interfaces

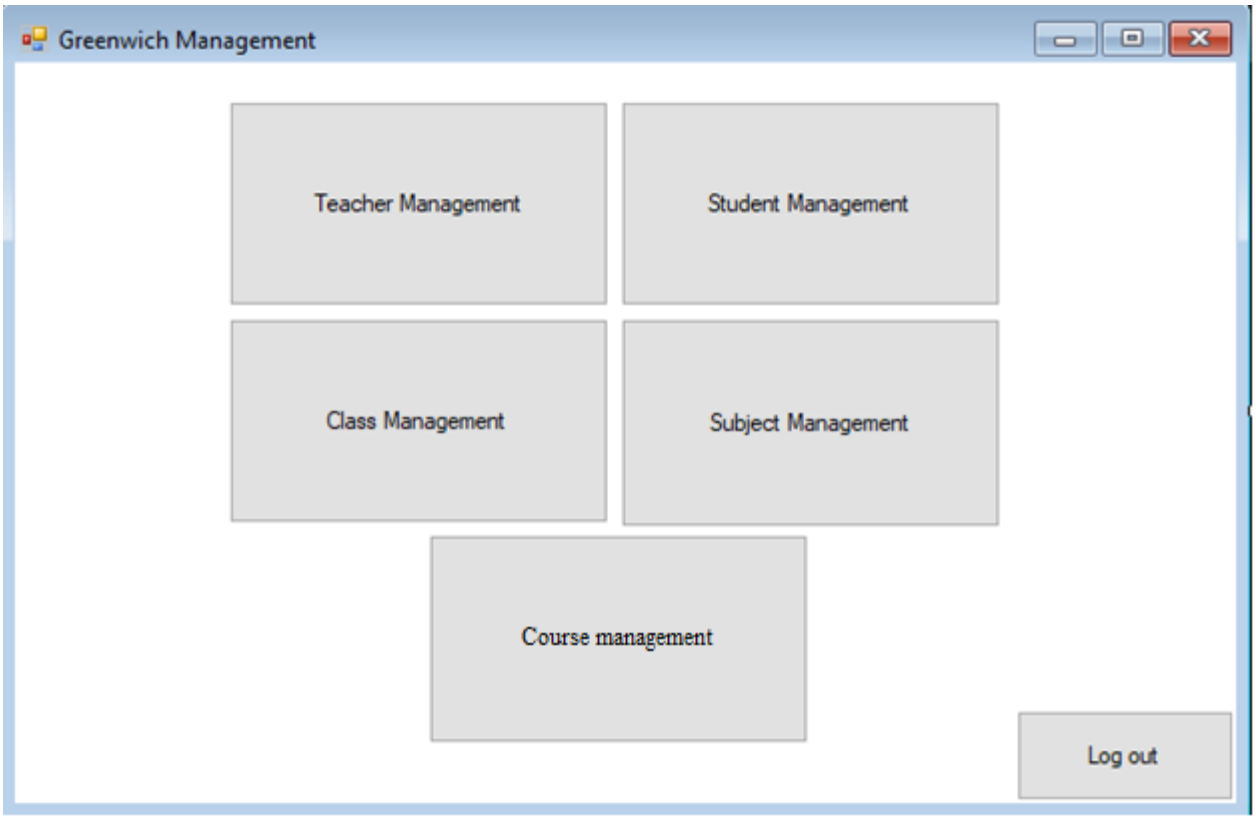
Login



After login successful, program will record login log of this user.

3	quandh@fe.edu.vn	123456789	2020-03-15

Home interface



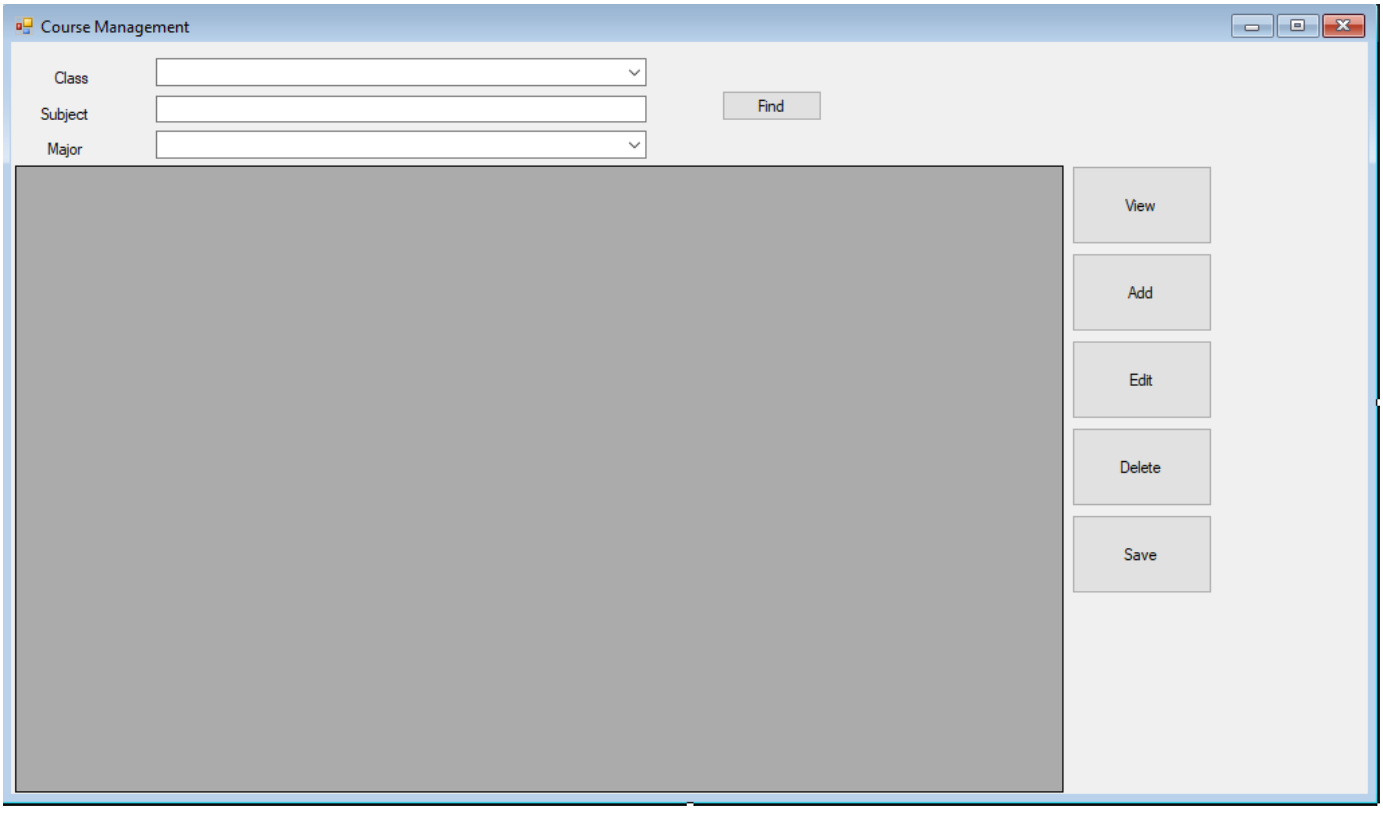
This display is only for teacher or higher level staff. In this display, we have 5 option for user to be choose.

11. Course management

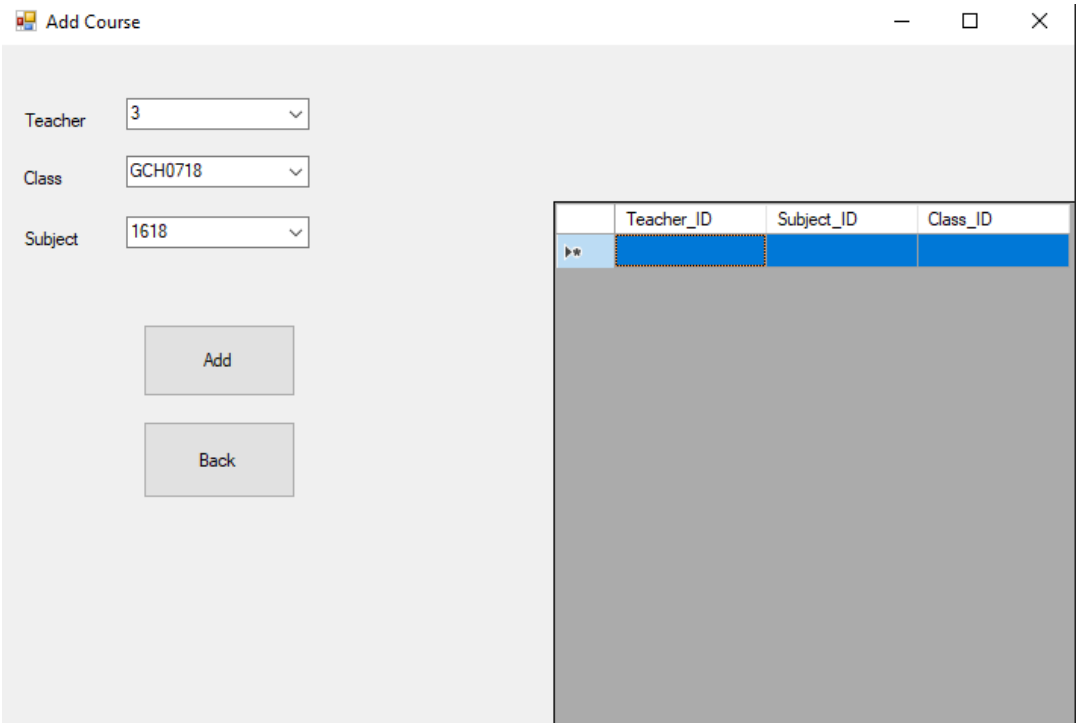
Data will be filled to datagridview from database. In this display, program will show list course will information field are ClassID , Teacher full name and Subject.

If user is teacher, they only can view information about course

Below is screen for staff. They can use add,edit ,delete function.

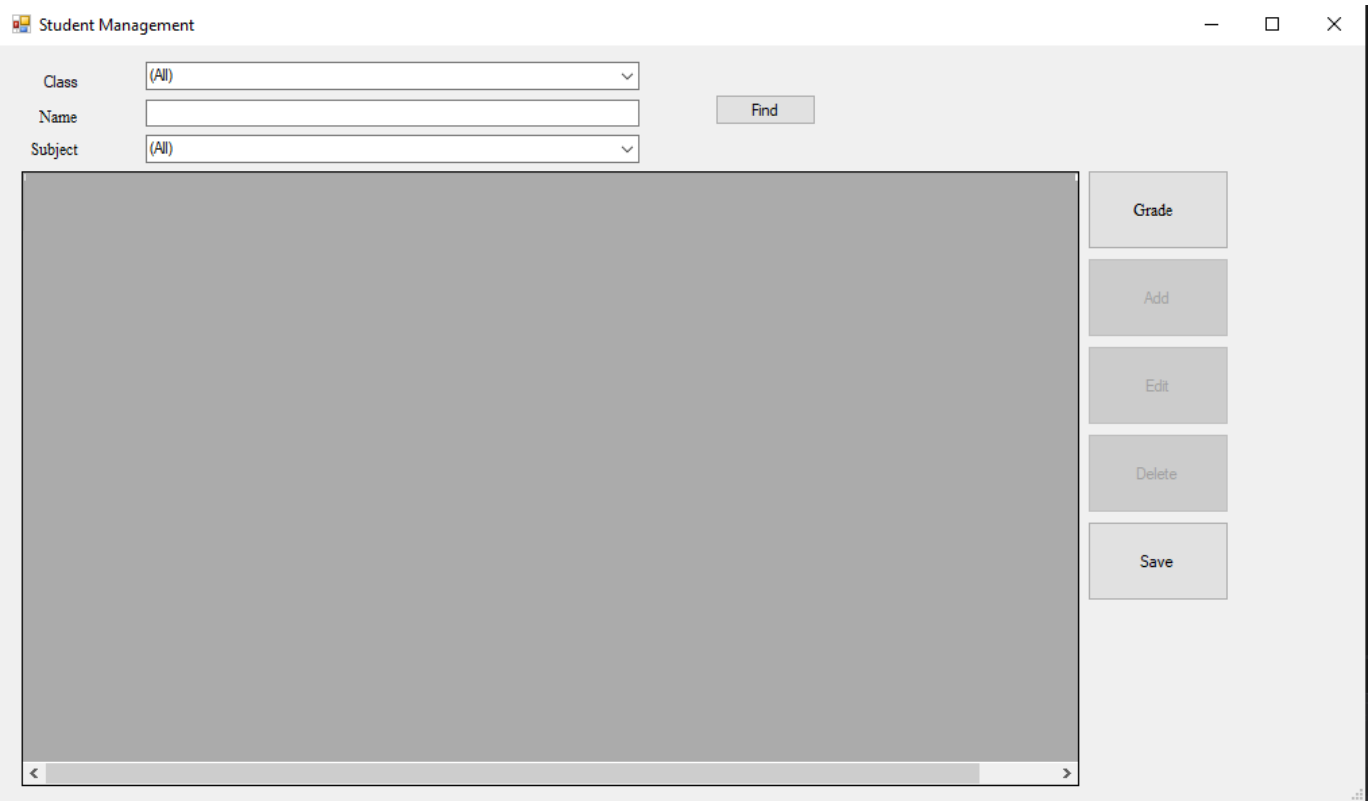


12. Add Course

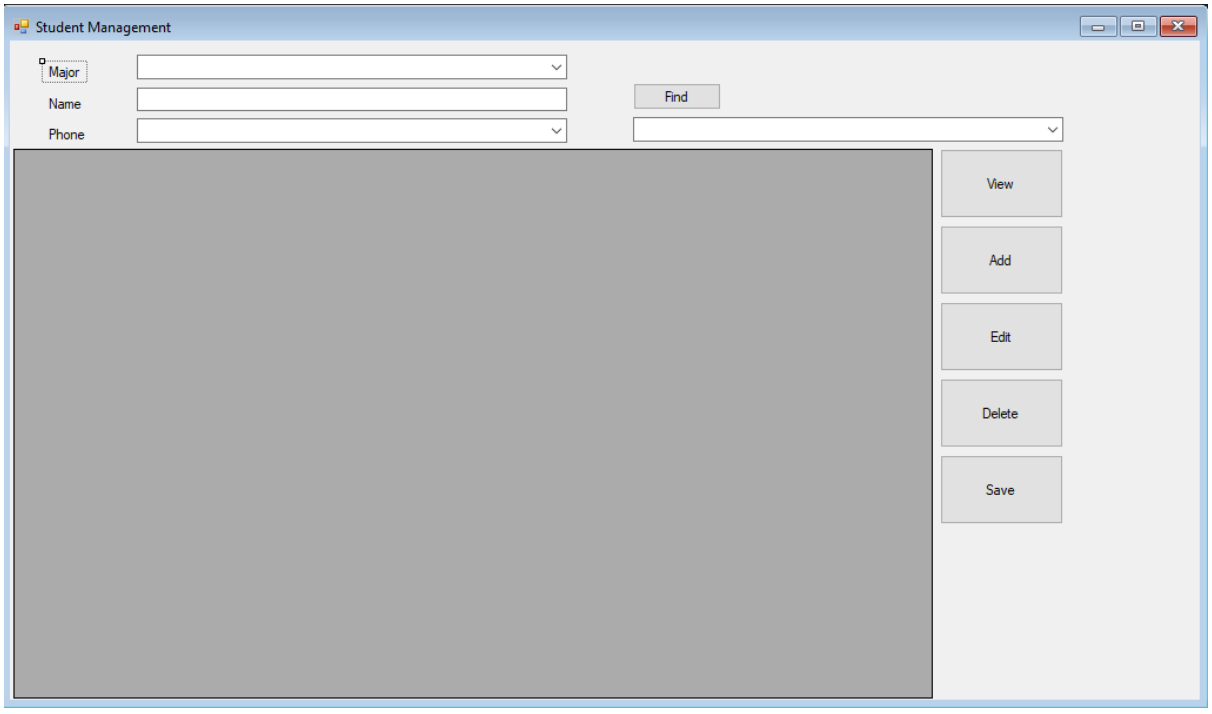


After click to add button in Course Management, New form will be opened. In this form, we will 3 combobox to select class,teacher,subject for one course before create. After finish selection, we can click to Add and view the result in datagridview. After that we can click Back button for back to Course management screen or if we can stay for add more Course. .In case, A course has one teacher already, data wont allow to add a course which have the same subjectid and classid to system.

13. Student management
14. Similar for course, teacher only can't use other function exclude Grade

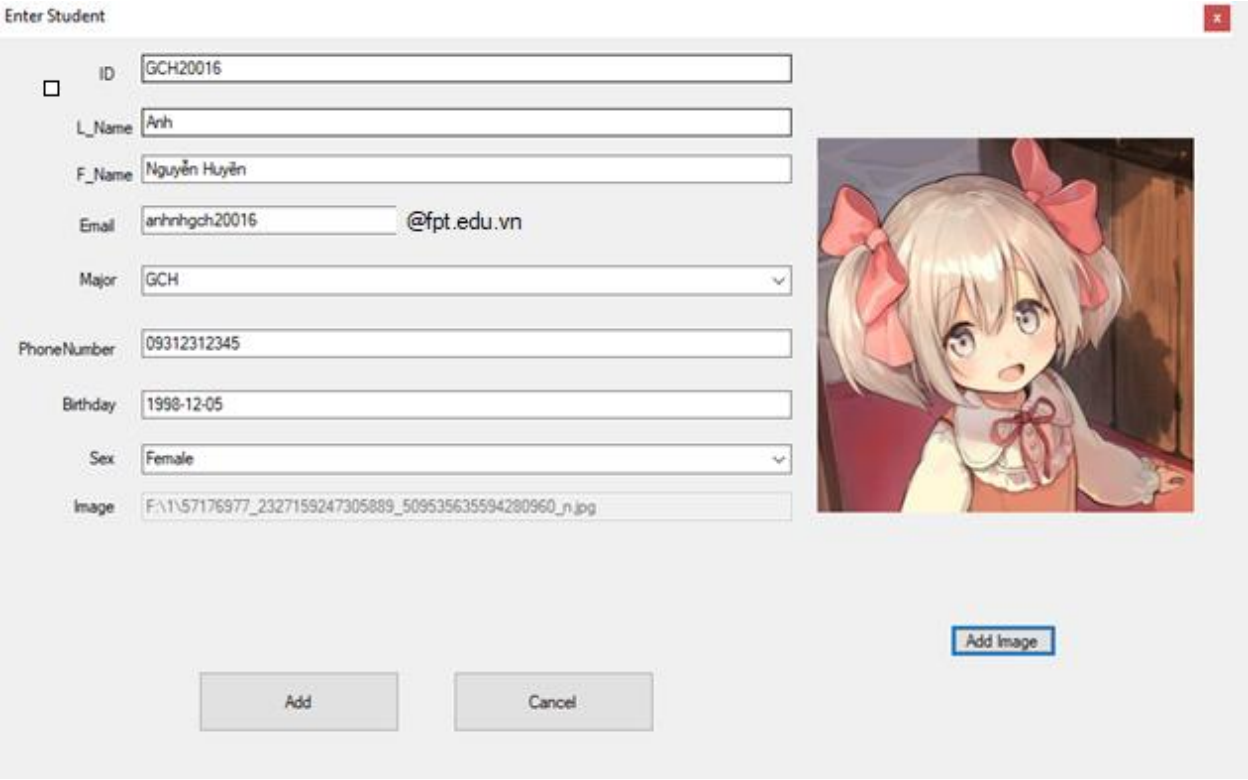


Teacher can find a course by Class and Subject. They can only see course that they are teaching.



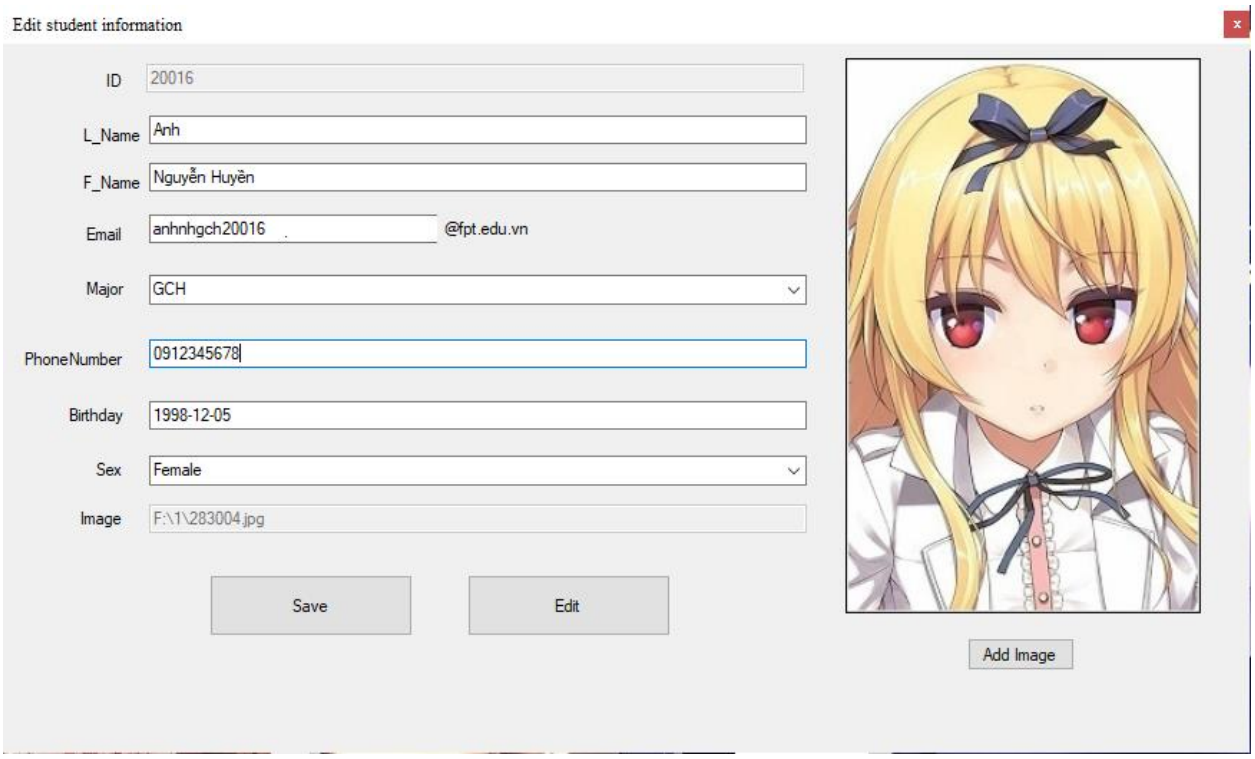
15. Add student to list

After choose “Add” option in Student management screen. Program will appear one form which allow user to enter Student information. After completing all required fields, we can enter to input them to system or cancel to close form. In this form, we can enter information with 5 textbox. We also have 2 combobox for Sex and Major to make sure the user doesn't enter wrong one. Beside, user can enter student image with add image button. Moreover email domain will be define and user only need to enter mailbox name. In database, image is allow null field, so we can enter it or not.



We have query with getMajor and getSex procedure for fill the major combobox. When loading form program will exec there two procedure.

16. Edit student information



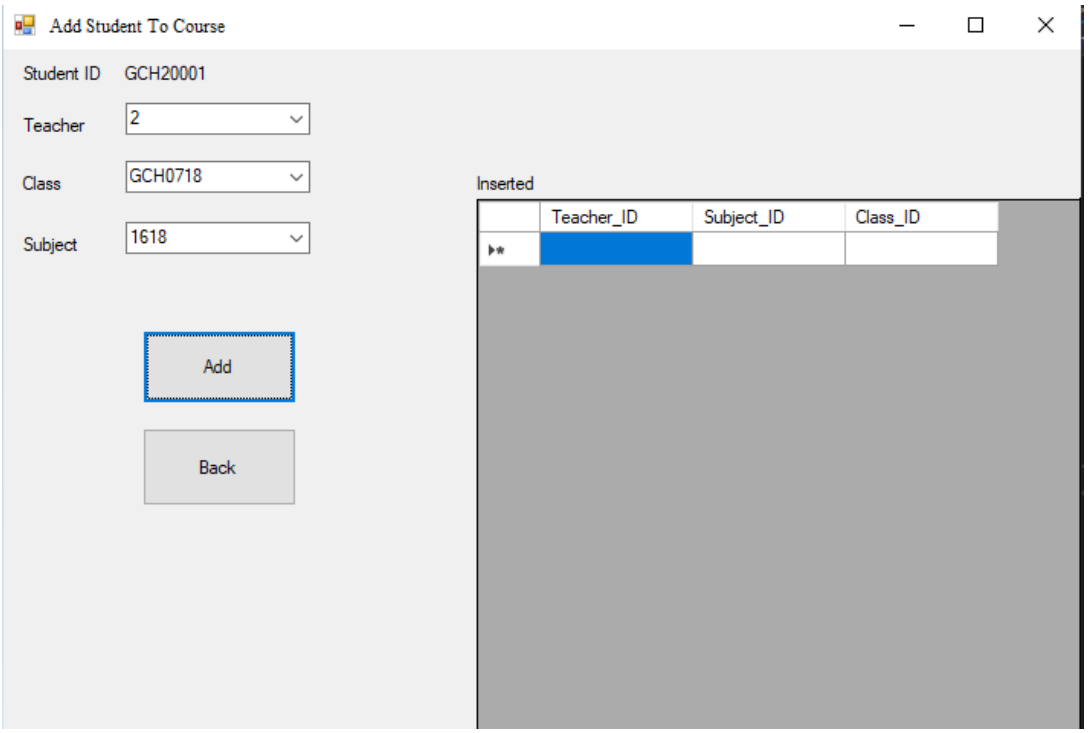
If we click to edit button, Edit student form will appear. This form is very similar to the "Add student form". But we will Save button instead Add. All information of choosen student will be fill in the form exclude Studentid, because studentid is not accepted to change. After we edit it, we can click to save button to save the change or click exit.

17. Delete Student in list

We can select one student in datagridview (student list) then click to delete button. All information about this student will be remove from list.

18. Insert student to course

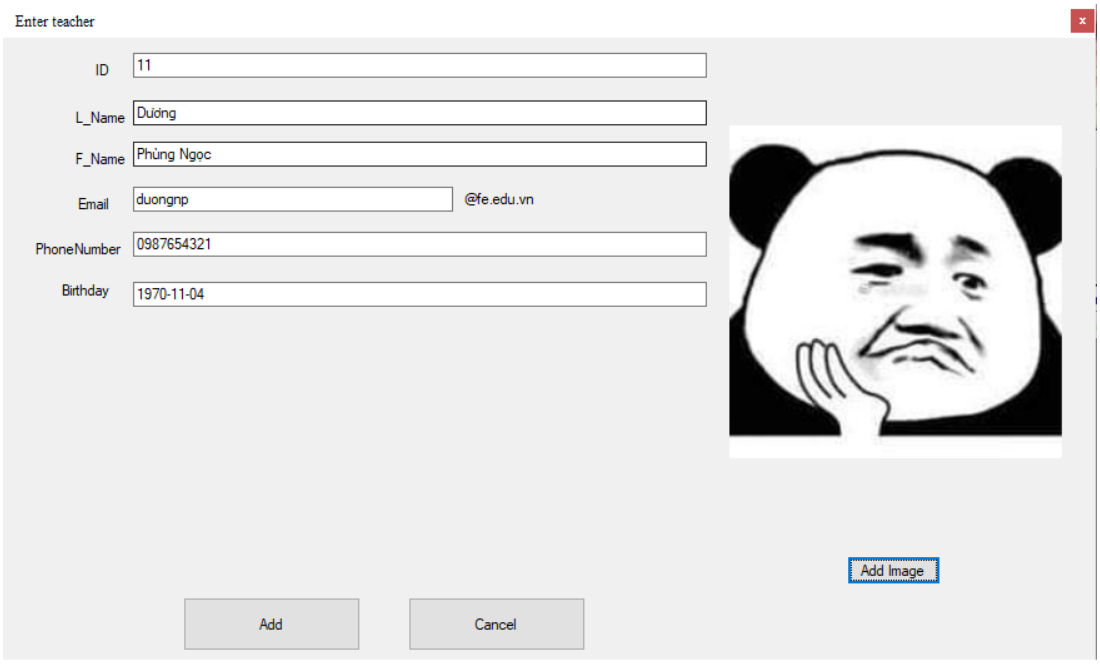
To open this form, we need click one row (a student) then click view button student management. In this form, we are adding new record for RegisterStudent as Registration of Student who is clicked in student management form. In this form, we can choose course information with 3 data field in combobox or we can select the available course in the table Inserted.



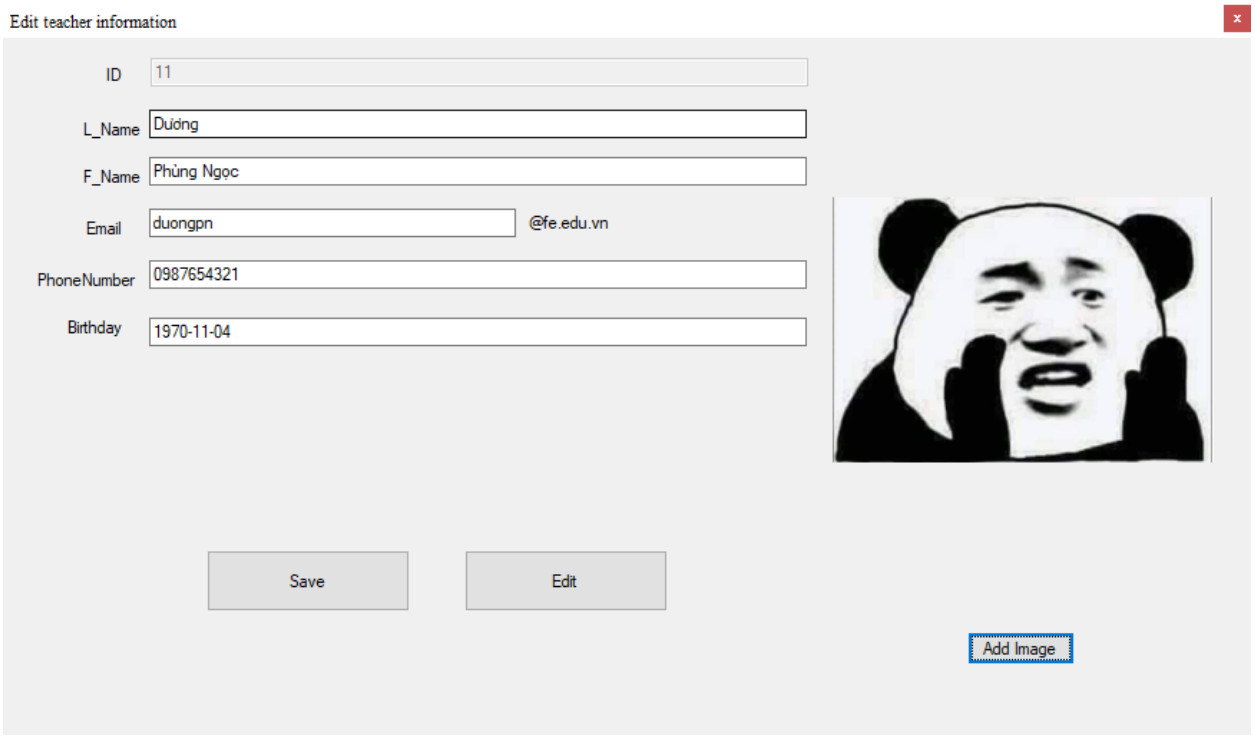
In case a student has relearn a subject more than 3 times. Program wont allow to add him to any course that have these subject.

19. Add teacher

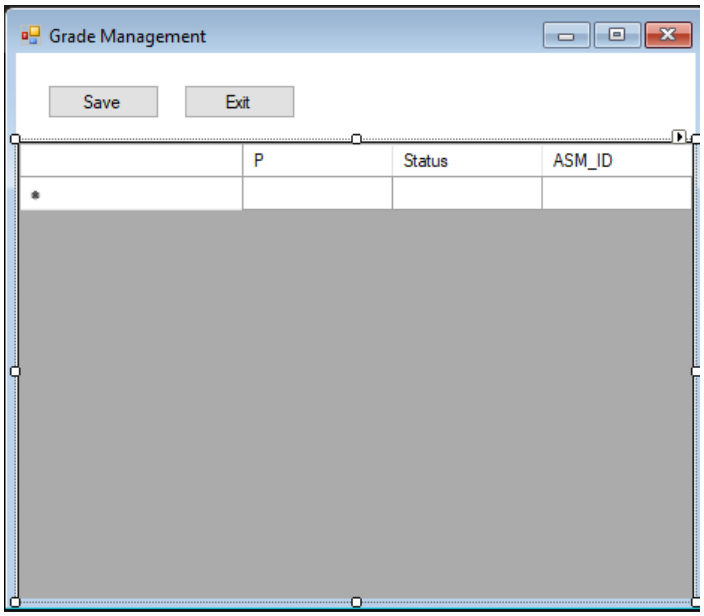
"Teacher add form" has a similar mechanism of operation for students. It only different in domain name of email.



20. Edit teacher information

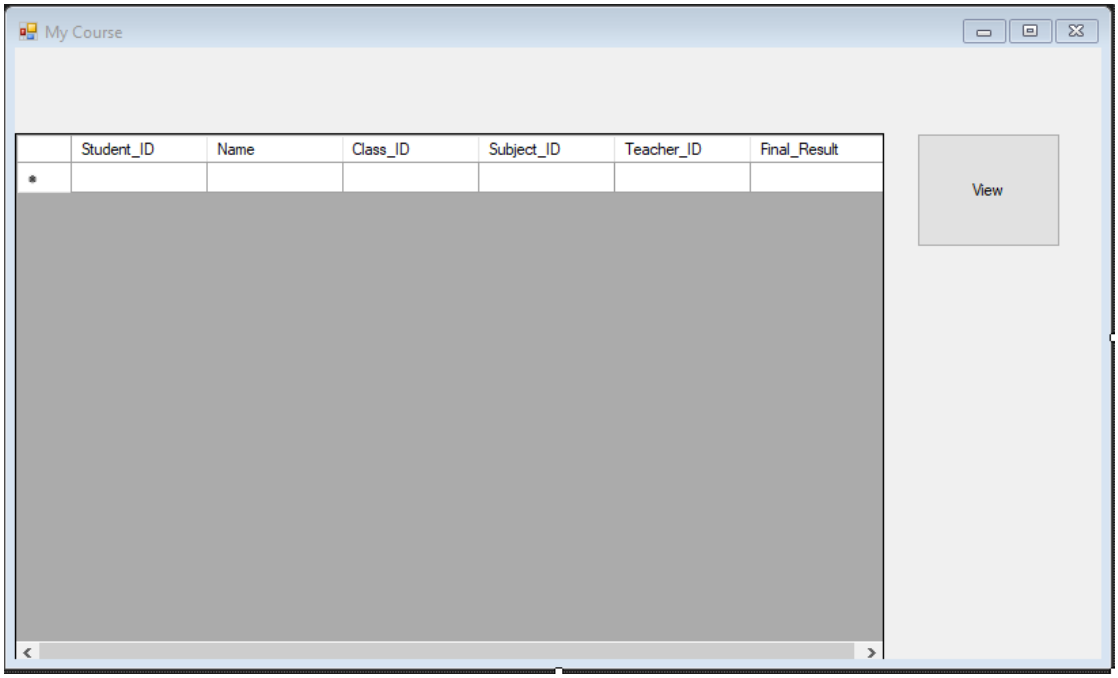


21. Add Grade for StudentRegister



In this form, user can enter grade for each criterion of student. 0 is false and 1 is pass. After that user must enter asm_id to finish one criterion.

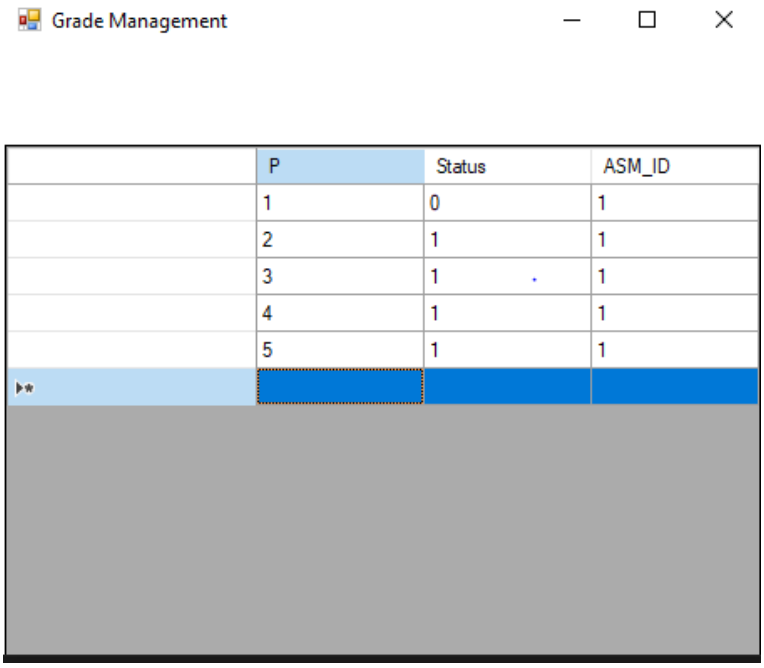
c) Student



In this form, student can see list of all their course and final result. It will show NULL if student is still studying or not grading.

Query

They can click to one course in datagridview and click button View to see detail about criteria.



Index of comments

2.1 Implementation: statements used to create tables, insert data into tables are written properly. The tables and links between them are compatible with the design.

Input and output interface: Nice interface design

SQL statement: There are statements corresponding to the function buttons on the interface. This statement queries appropriate management information from multiple tables.

Testing: Normal and data validation test cases and critical review about actual outcomes and test against expected outcomes are realized.

Documentation: There are appropriate user and technical documents with an assessment of the possibility and realistic improvements of the system.

Your assignment achieves well all the criteria about implementing, testing, interface designing, making technical and user documents. Well done!

However, you should not put all the codes of ASM in procedures. It makes your system become unnecessarily complicated. The presentation of the report also has some shortcomings. Specifically: The criteria M5 and D3 have been achieved but you did not put in the given heading. This is likely to cause misunderstanding.