Durant van Rensburg, 3125

IT Certification: Database Development

Formative Assessment 2:

MCSE DA: DBD621

# Planning:

## Database and Tables:

I Created a database called ‘FA2’ with three database tables: A student, Campus and Teacher’s table. Each table has 14 different schemas’, because there are 14 campuses nationally and the teachers only have access to their own campus’ details. I created a stored procedure called ‘CreateTheTables’ in order to create these tables.

**Here is the code:**

CREATE PROCEDURE CreateTheTables @schemaName nvarchar(100)

AS

EXEC('CREATE SCHEMA [' + @schemaName + '] AUTHORIZATION [dbo]')

DECLARE @sql nvarchar(MAX)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Campus(

CampusID int primary key,

PhoneNumber nvarchar(10),

CampusName nvarchar(30))'

EXEC(@sql)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Teacher(

TeacherID int primary key,

TeacherName nvarchar(50),

CampusID int,

Course nvarchar(30),

Description nvarchar(MAX))'

EXEC(@sql)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Student(

StudentID int primary key,

StudentName varchar(50),

Mark int,

Course nvarchar(30))

'

EXEC(@sql)

GO

## Dummy data:

I created a stored procedure called ‘CreateDummyData’ in order to add records to each created table.

**Code:**

CREATE PROCEDURE CreateDummyData @schemaName nvarchar(100)

AS

DECLARE @sql nvarchar(MAX)

DECLARE @len int

DECLARE @pos int

DECLARE @count int

SET @pos = 0

SET @len = 0

SET @count = 1

--Add Campus to table

SET @sql = N'INSERT INTO ' + @schemaName + N'.Campus VALUES (@parID, @parNumber, @parName)';

EXEC sp\_executesql @sql,

N'@parID int, @parNumber nvarchar(10), @parName nvarchar(30)',

@parID = @count,

@parNumber = '0726543211',

@parName = @schemaName

--Add Students

SET @count = 4

WHILE @count > 1

BEGIN

--INSERT VALUES INTO Teacher table

DECLARE @Mark int

DECLARE @student nvarchar(30)

--Generate a random mark for student by calling the GenerateStudentMarks() function

SET @Mark = dbo.GenerateStudentMarks()

SET @sql = N'INSERT INTO ' + @schemaName + N'.Student VALUES (@parID, @parStudent, @parMark, @parCourse)';

SET @student = 'student' + CAST(@count as nvarchar)

EXEC sp\_executesql @sql,

N'@parID int, @parStudent nvarchar(30), @parMark int, @parCourse nvarchar(30)',

@parID = @count,

@parStudent = @student,

@parMark = @Mark,

@parCourse = 'MCSE DA'

--Increase counter

SET @count = @count - 1

END

--Add Teachers

SET @count = 4

WHILE @count > 1

BEGIN

--INSERT VALUES INTO Teacher table

SET @sql = N'INSERT INTO ' + @schemaName + N'.Student VALUES (@parID, @parTeacher, 1, @parCourse, @parDescription)';

DECLARE @Teacher nvarchar(30)

DECLARE @Description nvarchar(50)

SET @Teacher = 'Teacher' + CAST(@count as nvarchar)

SET @Description = 'Teaches MCSE DA at ' + @schemaName + ' campus'

EXEC sp\_executesql @sql,

N'@parID int, @parTeacher nvarchar(50), @parCampusID int, @parCourse nvarchar(30), @parDescription nvarchar(30)',

@parID = @count,

@parTeacher = @Teacher,

@parCampusID = 1,

@parCourse = 'MCSE DA',

@parDescription = @Description

--Increase counter

SET @count = @count - 1

END

GO

I Created a function called ‘GenerateStudentMark’ in order to create a randomized student mark for each student and it returns a random student mark (int). It only calculates the average of the student’s year.

**Code:**

CREATE FUNCTION dbo.GenerateStudentMarks()

RETURNS INT

AS

BEGIN

DECLARE @RETURN int

DECLARE @Upper INT;

DECLARE @Lower INT;

DECLARE @Random float;

SELECT @Random = rndResult

FROM rndView

SET @Lower = 50

SET @Upper = 99

SET @RETURN= (ROUND(((@Upper - @Lower -1) \* @Random + @Lower), 0))

RETURN(@Return)

END

GO

## Schemas:

I Created a while loop in order to create a schema for each table. The while loop contains a variable (schemaList) that reads the variable and creates the schema’s name that’s specified in the variable. The while loop calls the two stored procedures: CreateTheTables and CreateDummyData to create the tables and add dummy data to the tables.

**Code:**

DECLARE @len int

DECLARE @pos int

DECLARE @schemaName varchar(30)

DECLARE @sql varchar(500)

DECLARE @schemaList varchar(500)

SET @pos = 0

SET @len = 0

SET @schemaList = 'Auckland\_Park,Bloemfontein,Boksburg,Cape\_Town,Durban,Nelspruit,Polokwane,Potchefstroom,Port\_Elizabeth,Pretoria,Randburg,Sandton,Roodepoort,Stellenbosch,Vereeniging,'

WHILE CHARINDEX(',', @schemaList, @pos+1)>0

BEGIN

SET @len = CHARINDEX(',', @schemaList, @pos+1) - @pos

SET @schemaName = SUBSTRING(@schemaList, @pos, @len)

-- Call the stored procedure to create the tables and records

EXEC CreateTheTables @schemaName

EXEC CreateDummyData @schemaName

SET @pos = CHARINDEX(',', @schemaList, @pos+@len) +1

END

GO

## Users:

I Created two users for this project: FA2Admin and FA2StellenboschTeacher. FA2Admin has full access to the database properties and can edit anything on the database, but the FA2StellenboschTeacher can only have access to the Stellenbosch schema tables, because the teachers are only allowed to edit their own campus’ information.



# Testing Manual:

## SQL Server:

### Error Report:

* Could not find stored procedure 'CreateDummyData'.

Msg 102, Level 15, State 1, Line 136

**Solution:**

The function was underneath the procedure that called it, So I moved the function to the top of the Query.

* Invalid use of a side-effecting operator 'rand' within a function.

Msg 102, Level 15, State 1, Line 136

**Solution:**

The RAND() function has a strict policy that the creator must follow. I ignored the syntax properties of this statement and I corrected it.

* Msg 245, Level 16, State 1, Procedure CreateDummyData, Line 12 [Batch Start Line 131]

Conversion failed when converting the nvarchar value 'INSERT INTO Auckland\_Park.Campus

VALUES (1, 072654321' to data type int.

**Solution:**

When the INSERT INTO values where inserted into the table, I forgot to add a quotation mark and I corrected it immediately.

* Msg 213, Level 16, State 1, Line 167

Column name or number of supplied values does not match table definition.

**Solution:**

When I inserted records to the Teacher’s table, I forgot to add a value for the CampusID column and I quickly corrected it.

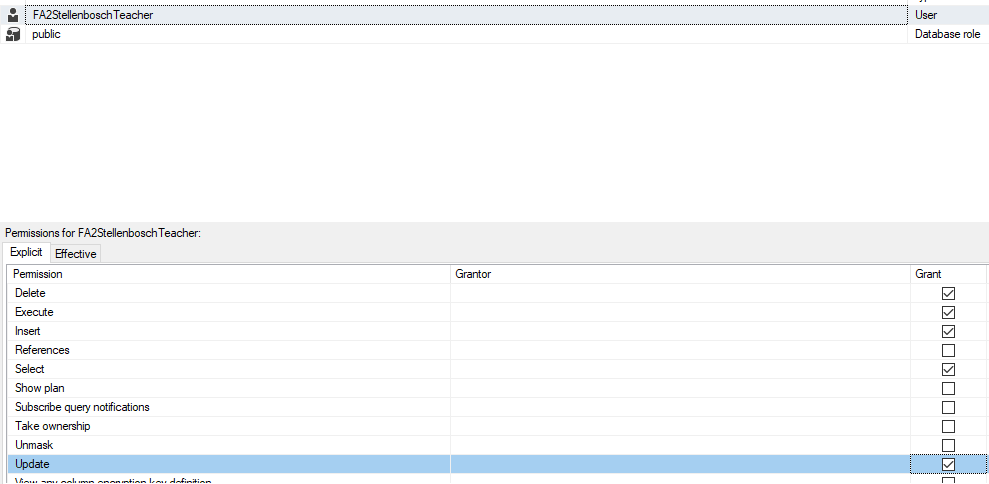
### Users and their permissions:

I created two users for the database: FA2Admin and FA2StellenboschTeacher

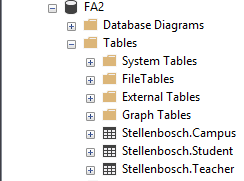


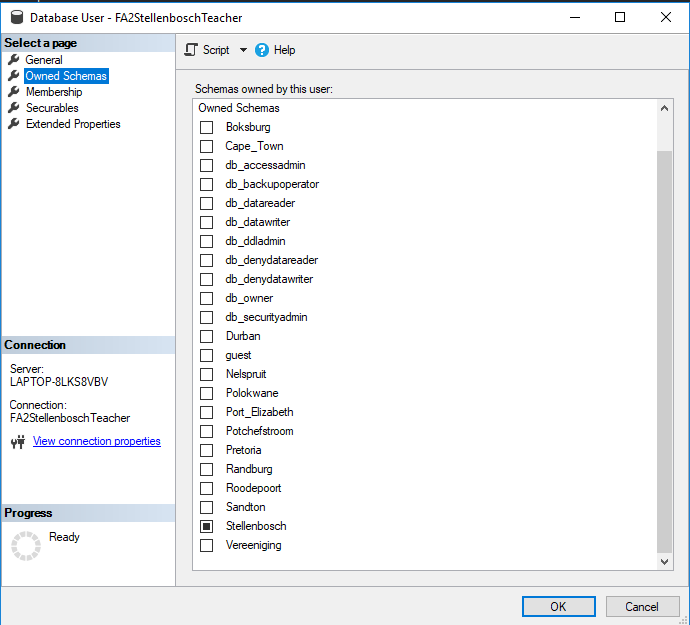
#### Permissions:

##### FA2StellenboschTeacher:



The teacher can insert, update, select, delete and execute, but only on his owned schema (Stellenbosch).





I used the following code to test the teachers permissions:

--Show Results for teacher:

SELECT \*

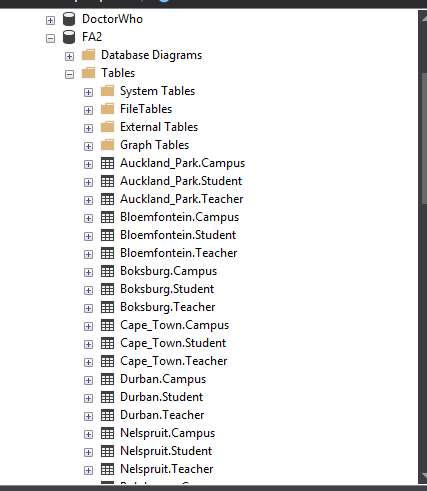
FROM Nelspruit.Student

SELECT \*

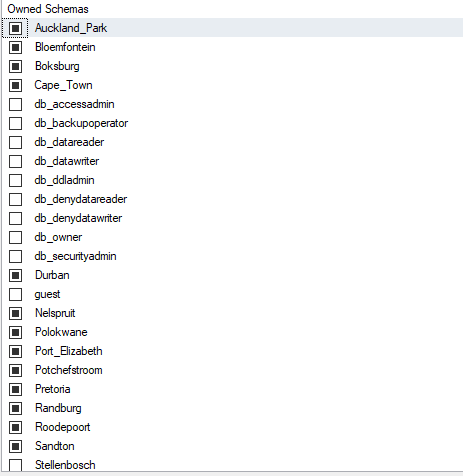
FROM Stellenbosch.Student

##### FA2Admin:

This user can do anything to the database, but only to the FA2 database.

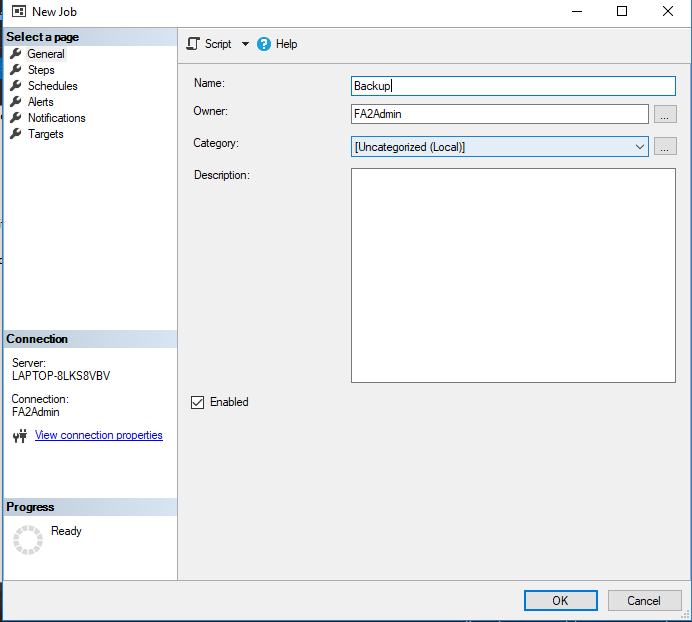


He can edit all of the schemas except for Stellenbosch:

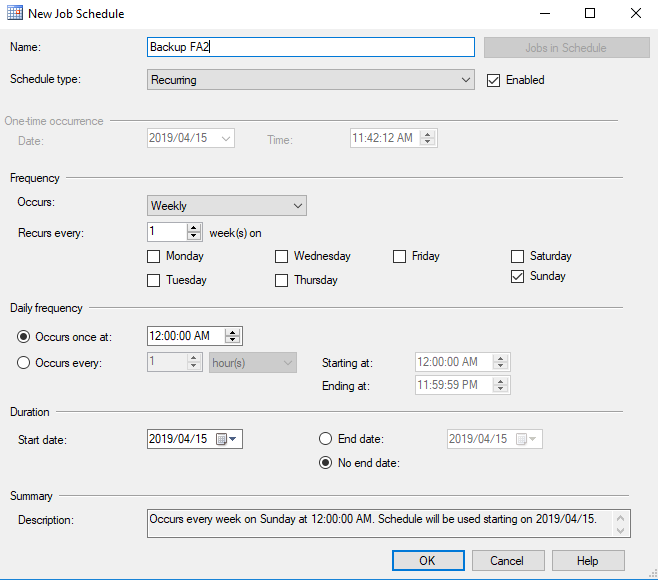


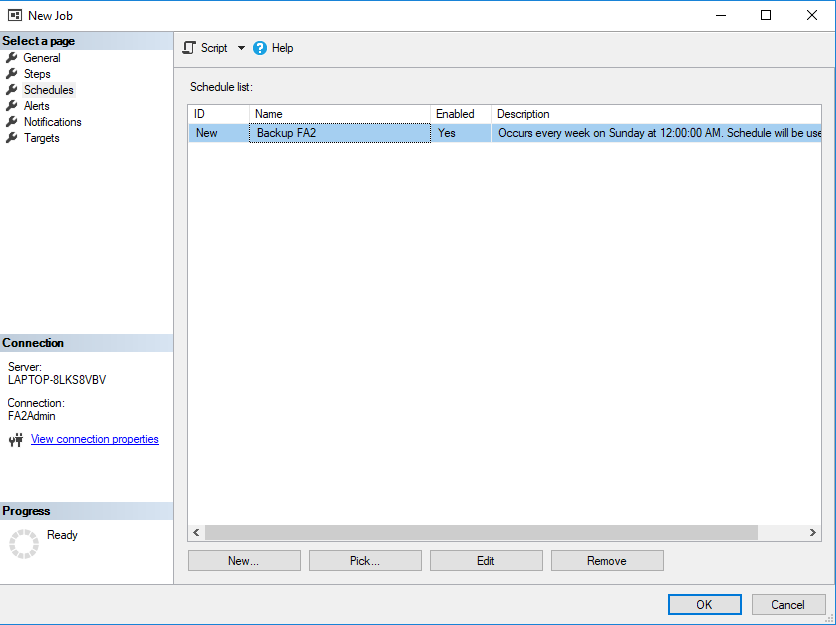
### Backup:

I Went to the SQL Server Agent and created a new job called Backup:

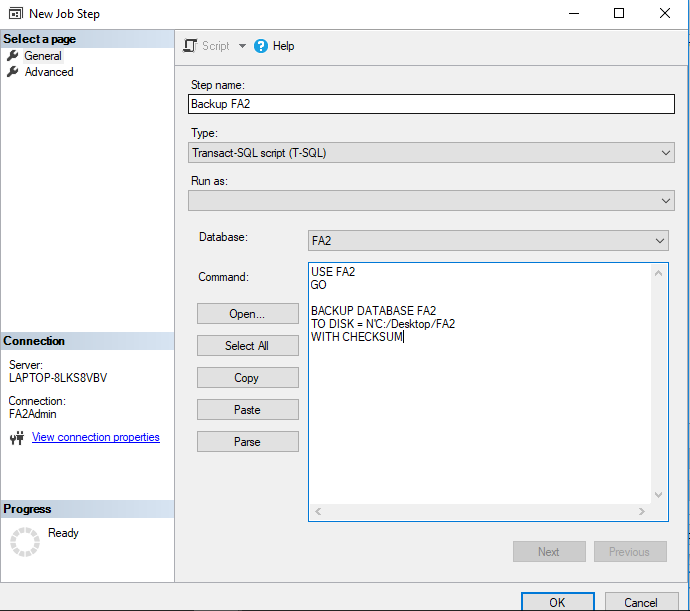


Then I created a schedule:





Then I created steps for the backup:



### Query Code:

--Use other DB in order to Delete existing DB

USE MASTER

GO

-- Delete DB

IF db\_id('FA2') IS NOT NULL

BEGIN

ALTER DATABASE FA2 SET single\_user WITH ROLLBACK IMMEDIATE

DROP DATABASE FA2

END

GO

-- Create DB

CREATE DATABASE FA2

GO

USE FA2

GO

--Change owner of DB(to Public)

ALTER DATABASE FA2 SET TRUSTWORTHY ON;

GO

EXEC dbo.sp\_changedbowner @loginame = N'sa', @map = false

GO

sp\_configure 'show advanced options', 1;

GO

RECONFIGURE;

GO

sp\_configure 'clr enabled', 1;

GO

RECONFIGURE;

GO

--Create the tables

CREATE PROCEDURE CreateTheTables @schemaName nvarchar(100)

AS

EXEC('CREATE SCHEMA [' + @schemaName + '] AUTHORIZATION [dbo]')

DECLARE @sql nvarchar(MAX)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Campus(

CampusID int primary key,

PhoneNumber nvarchar(10),

CampusName nvarchar(30))'

EXEC(@sql)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Teacher(

TeacherID int primary key,

TeacherName nvarchar(50),

CampusID int,

Course nvarchar(30),

Description nvarchar(MAX))'

EXEC(@sql)

SET @sql = '

CREATE TABLE ' + @schemaName + '.Student(

StudentID int primary key,

StudentName varchar(50),

Mark int,

Course nvarchar(30))

'

EXEC(@sql)

GO

CREATE VIEW rndView

AS

SELECT RAND() rndResult

GO

--Generate Marks for student

CREATE FUNCTION dbo.GenerateStudentMarks()

RETURNS INT

AS

BEGIN

DECLARE @RETURN int

DECLARE @Upper INT;

DECLARE @Lower INT;

DECLARE @Random float;

SELECT @Random = rndResult

FROM rndView

SET @Lower = 50

SET @Upper = 99

SET @RETURN= (ROUND(((@Upper - @Lower -1) \* @Random + @Lower), 0))

RETURN(@Return)

END

GO

CREATE PROCEDURE CreateDummyData @schemaName nvarchar(100)

AS

DECLARE @sql nvarchar(MAX)

DECLARE @len int

DECLARE @pos int

DECLARE @count int

SET @pos = 0

SET @len = 0

SET @count = 1

--Add Campus to table

SET @sql = N'INSERT INTO ' + @schemaName + N'.Campus VALUES (@parID, @parNumber, @parName)';

EXEC sp\_executesql @sql,

N'@parID int, @parNumber nvarchar(10), @parName nvarchar(30)',

@parID = @count,

@parNumber = '0726543211',

@parName = @schemaName

--Add Students

SET @count = 4

WHILE @count > 0

BEGIN

--INSERT VALUES INTO Teacher table

DECLARE @Mark int

DECLARE @student nvarchar(30)

--Generate a random mark for student by calling the GenerateStudentMarks() function

SET @Mark = dbo.GenerateStudentMarks()

SET @sql = N'INSERT INTO ' + @schemaName + N'.Student VALUES (@parID, @parStudent, @parMark, @parCourse)';

SET @student = 'student' + CAST(@count as nvarchar)

EXEC sp\_executesql @sql,

N'@parID int, @parStudent nvarchar(30), @parMark int, @parCourse nvarchar(30)',

@parID = @count,

@parStudent = @student,

@parMark = @Mark,

@parCourse = 'MCSE DA'

--Increase counter

SET @count = @count - 1

END

--Add Teachers

SET @count = 4

WHILE @count > 0

BEGIN

--INSERT VALUES INTO Teacher table

SET @sql = N'INSERT INTO ' + @schemaName + N'.Student VALUES (@parID, @parTeacherName, @parCampusID, @parCourse, @parDescription)';

DECLARE @Teacher nvarchar(30)

DECLARE @Description nvarchar(50)

SET @Teacher = 'Teacher' + CAST(@count as nvarchar)

SET @Description = 'Teaches MCSE DA at ' + @schemaName + ' campus'

EXEC sp\_executesql @sql,

N'@parID int, @parTeacherName nvarchar(50), @parCampusID int, @parCourse nvarchar(30), @parDescription nvarchar(MAX)',

@parID = @count,

@parTeacherName = @Teacher,

@parCampusID = 1,

@parCourse = 'MCSE DA',

@parDescription = @Description

--Increase counter

SET @count = @count - 1

END

GO

--Create Schema's

DECLARE @len int

DECLARE @pos int

DECLARE @schemaName varchar(30)

DECLARE @sql varchar(500)

DECLARE @schemaList varchar(500)

SET @pos = 0

SET @len = 0

SET @schemaList = 'Auckland\_Park,Bloemfontein,Boksburg,Cape\_Town,Durban,Nelspruit,Polokwane,Potchefstroom,Port\_Elizabeth,Pretoria,Randburg,Sandton,Roodepoort,Stellenbosch,Vereeniging,'

WHILE CHARINDEX(',', @schemaList, @pos+1)>0

BEGIN

SET @len = CHARINDEX(',', @schemaList, @pos+1) - @pos

SET @schemaName = SUBSTRING(@schemaList, @pos, @len)

-- Call the stored procedure to create the tables and records

EXEC CreateTheTables @schemaName

EXEC CreateDummyData @schemaName

SET @pos = CHARINDEX(',', @schemaList, @pos+@len) +1

END

GO

--Show Results for teacher:

SELECT \*

FROM Nelspruit.Student

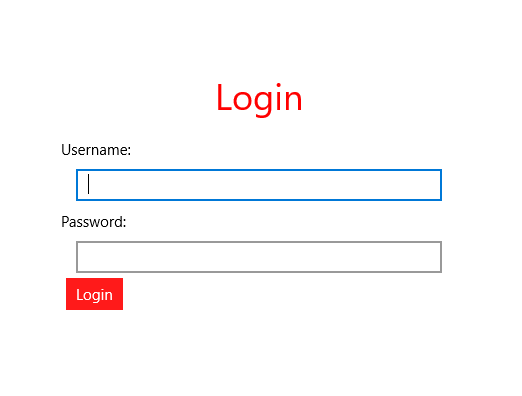
SELECT \*

FROM Stellenbosch.Student

## App:

### Login Page:

**Layout:**

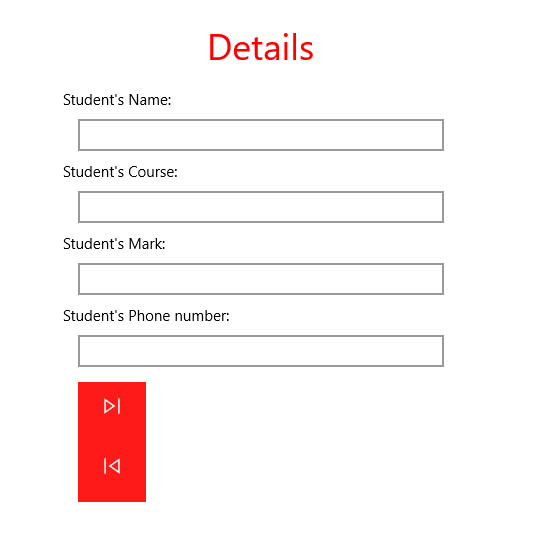


#### How it works:

The user must enter their account username and password and the application will do the rest. If the user is a teacher, they’ll only be able to see their own campuses information, if the user is an admin, the user can see all of the campuses and student’s information.

### Details Page:

**Layout:**



#### How it works:

The application only displays one student’s information at a time. The teacher can edit the student’s information by editing the displayed information in the textboxes. The teacher can navigate through the student objects by pressing the navigate forward and backwards buttons as displayed in the layout photo.