Introduction

The aim of the project I am working on here is to create a design and then implement a computer-based solution for an application that will allow personal supervisors to monitor their students under their care more easily. Additionally, it is to improve the wellbeing of students to be able to get in contact with their personal supervisors which will reduce some stress. In general, there is a strong need for more cooperation and coordination between students and their personal supervisors. To create a good working relationship will not only increase wellbeing but also productivity.

A computer-based solution is being developed as it is much easier for many to voice their concerns when not face to face. More introverted students for example may feel intimidated from talking to their supervisor directly and hence this program will provide a solution to this problem and most likely improve coordination between students and their supervisors. The solution should be made to be easy to use so any Student, Personal Supervisor or Senior Tutor can navigate their way through the system

Overall Description

The Brief

The brief in place is as follows: “The Department of Computer Science and Technology has in place a personal supervisor (PS) for each student. Supporting our students has always been a key part of their journey. The operation of the personal supervisor system is a key area of the department we are wishing to improve. The department is looking for a digital solution to help both students and staff monitor engagement and be able to action support at vital points in time where needed. There are number of engagement metrics available but sometimes these do not present a true picture. You need to design, develop and test a program that will allow users/stakeholders of the system to perform the usage scenarios described below.”

Stakeholders

The stakeholders in this system are as follows: The students, personal supervisor (PS) and senior tutor (ST).

Key Usage Scenarios

The usage scenarios given are as follows:

- A student should be able to self-report how they are feeling / progressing at fixed time intervals.
- The PS should be able to review the status of all their students
- The PS should be able to book a meeting with students
- The student should be able to book a meeting with the PS
- The ST should be able to see the status of all students and how the PS are interacting with the students

Problem Area

The main issues with student support systems at University are as follows:

- Alienation felt by students due to universities taking a more passive approach to assigning work and assisting with work
- A general lack of communication between faculties
- Not enough time where students and staff meet 1 to 1

Requirements Modelling

Scenarios

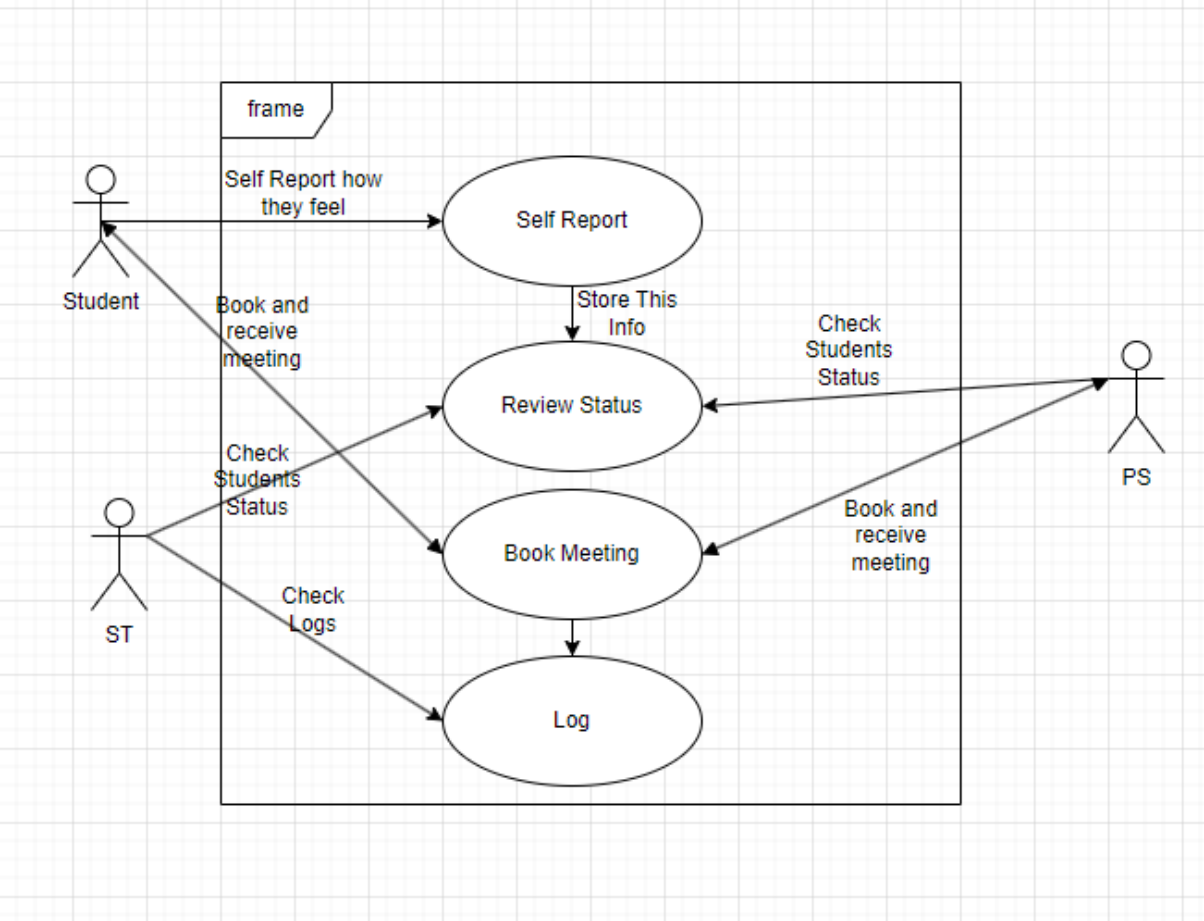
Student Scenario

Dmitri is a first-year student at Hull University getting ready for his exams in January. He has struggled to understand some of the coursework and feels overwhelmed. He has gone over much of the course notes online, but he feels like he needs to talk with someone directly, so he tries to contact his personal supervisor. He ends up getting some advice regarding his exams but with only limited time he only saw slight improvement before his exams. He wished that he could have got in contact with his personal supervisor earlier, but he did not realise this at the time he was struggling in his first term. Being able to receive more structured supervision he feels would have been helpful to him each week.

Personal Supervisor Scenario

Benedict is a personal supervisor who has been working at Hull University for 5 years. Often, he finds it tricky to help the students under his watch as they seem reluctant to reach out to him. It is also often chaotic trying to find times to book meetings with his students that do not conflict with each other's timetables. He knows some of his students have been struggling looking at their overall marks deteriorate over time but there is not much communication between him and his students and he would like to get to know them better.

Use Cases



Activity Diagrams

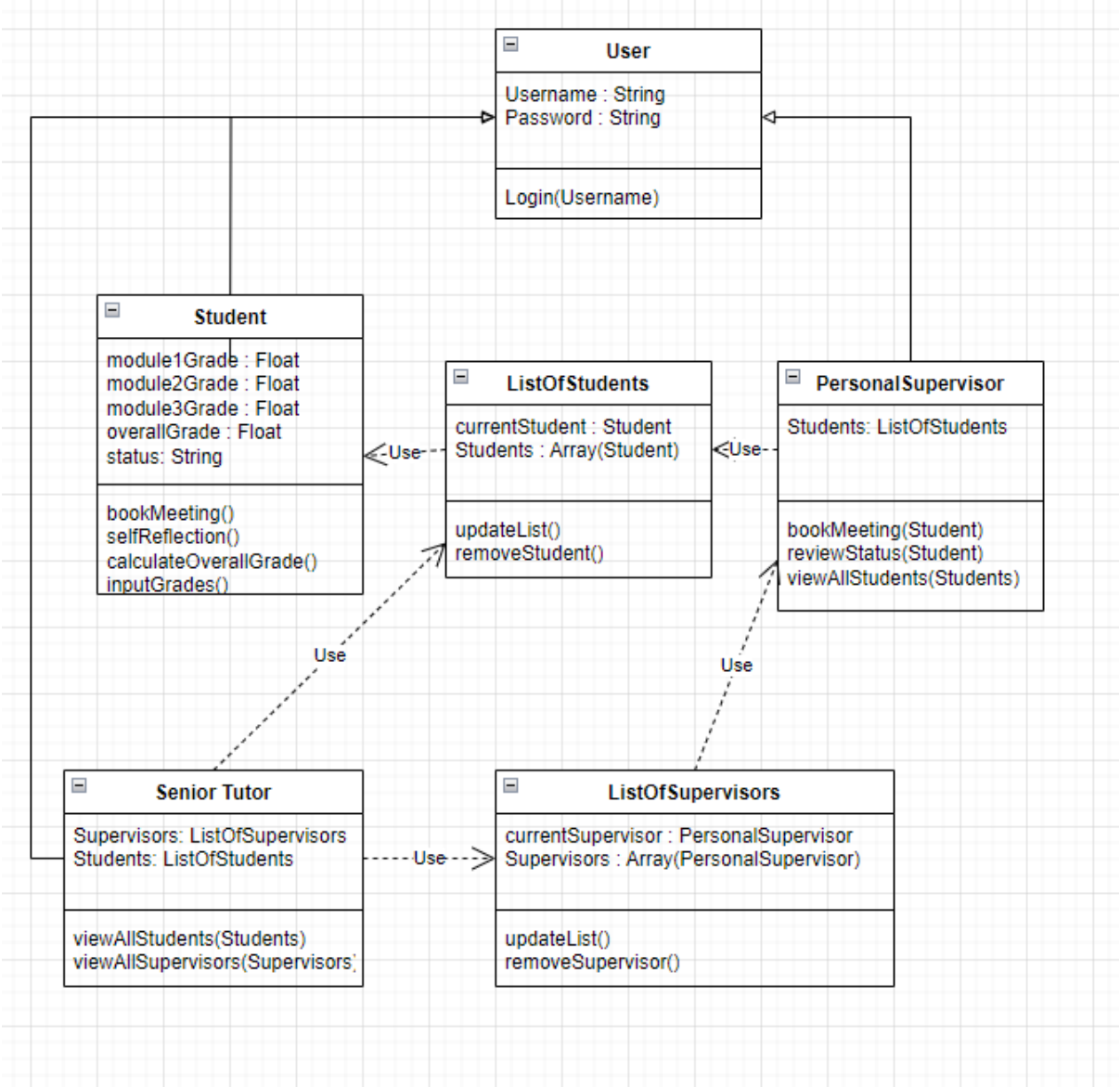
Requirements

- Students can book a meeting with their personal supervisor.
- Personal supervisors can book a meeting with their students
- Students can type out their thoughts and feelings to self-report
- Personal Supervisors can see the statistics of all their students
- Personal Supervisors can see the reports of all their students
- Senior Tutors can see the statistics of every student
- Senior Tutors can see every student's reports
- Personal Supervisors can login
- Students can login
- Senior Tutors can login
- The System allows users to exit
- The Program can run on a university computer
-

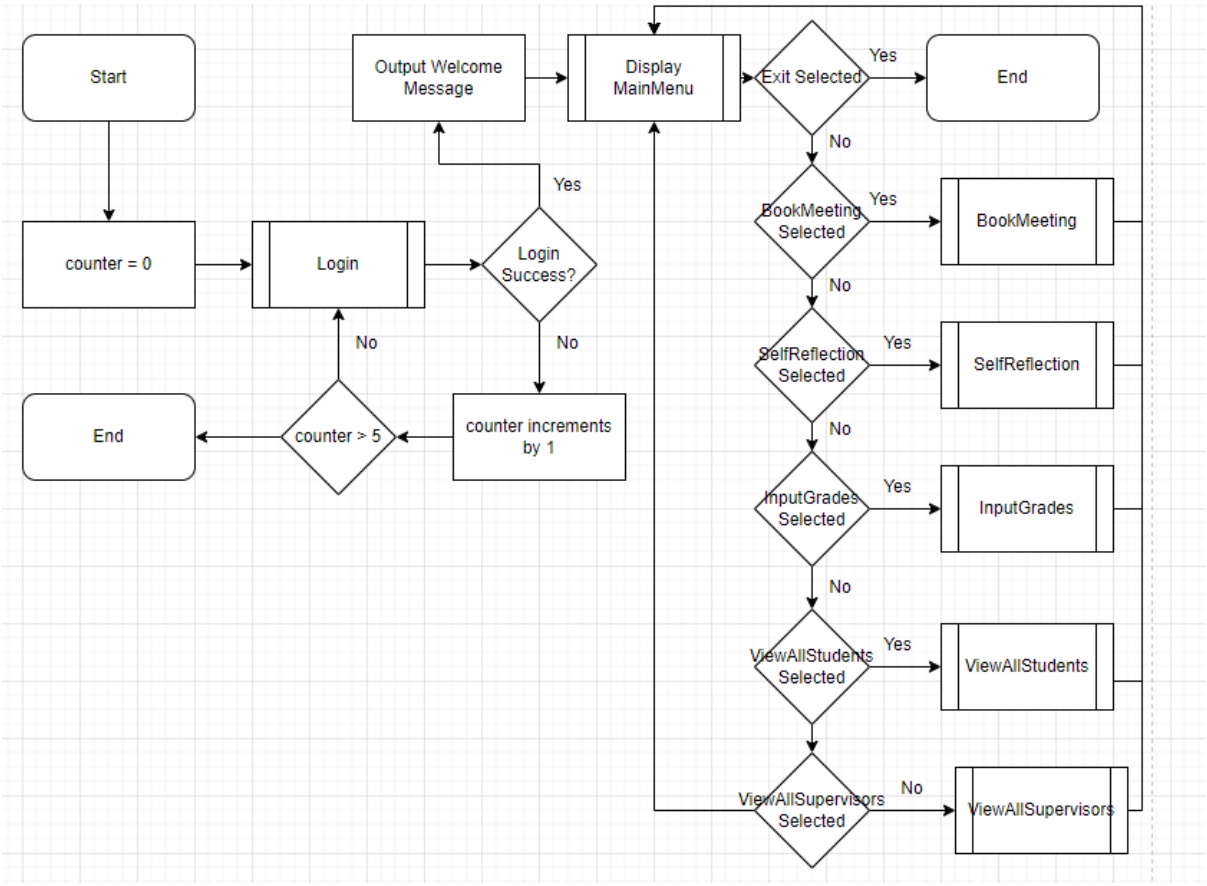
Test Plans

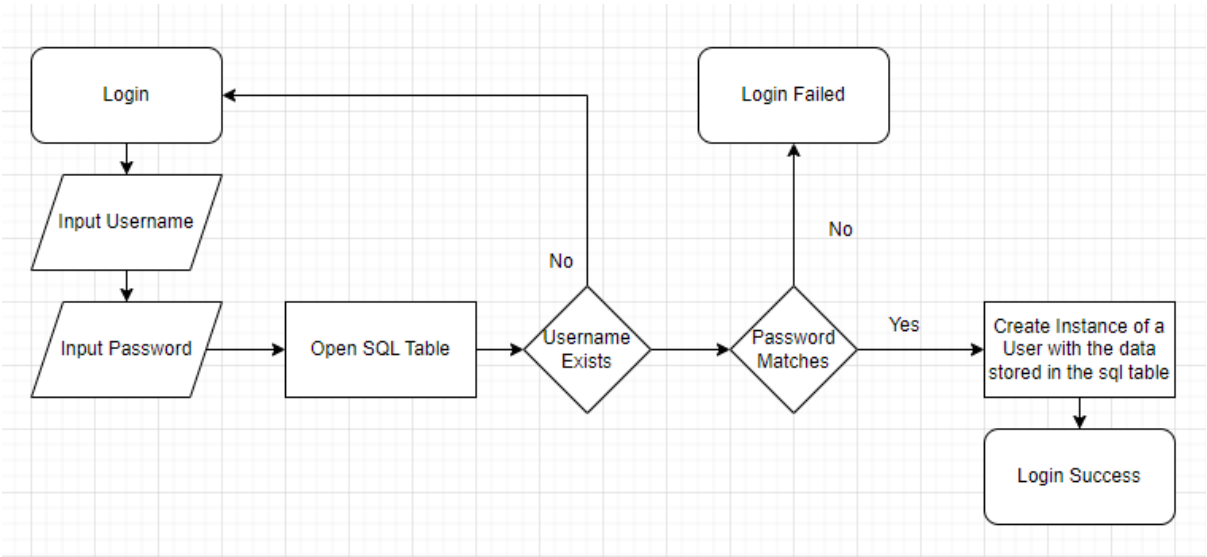
Design

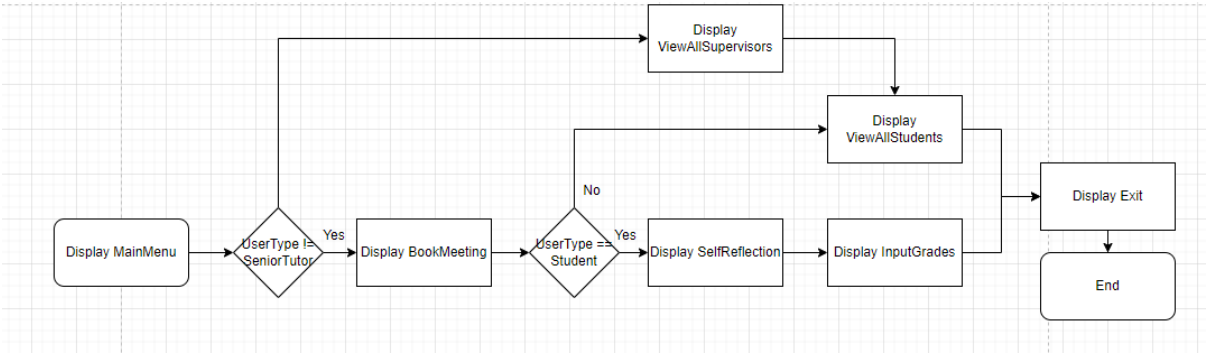
Class Modelling

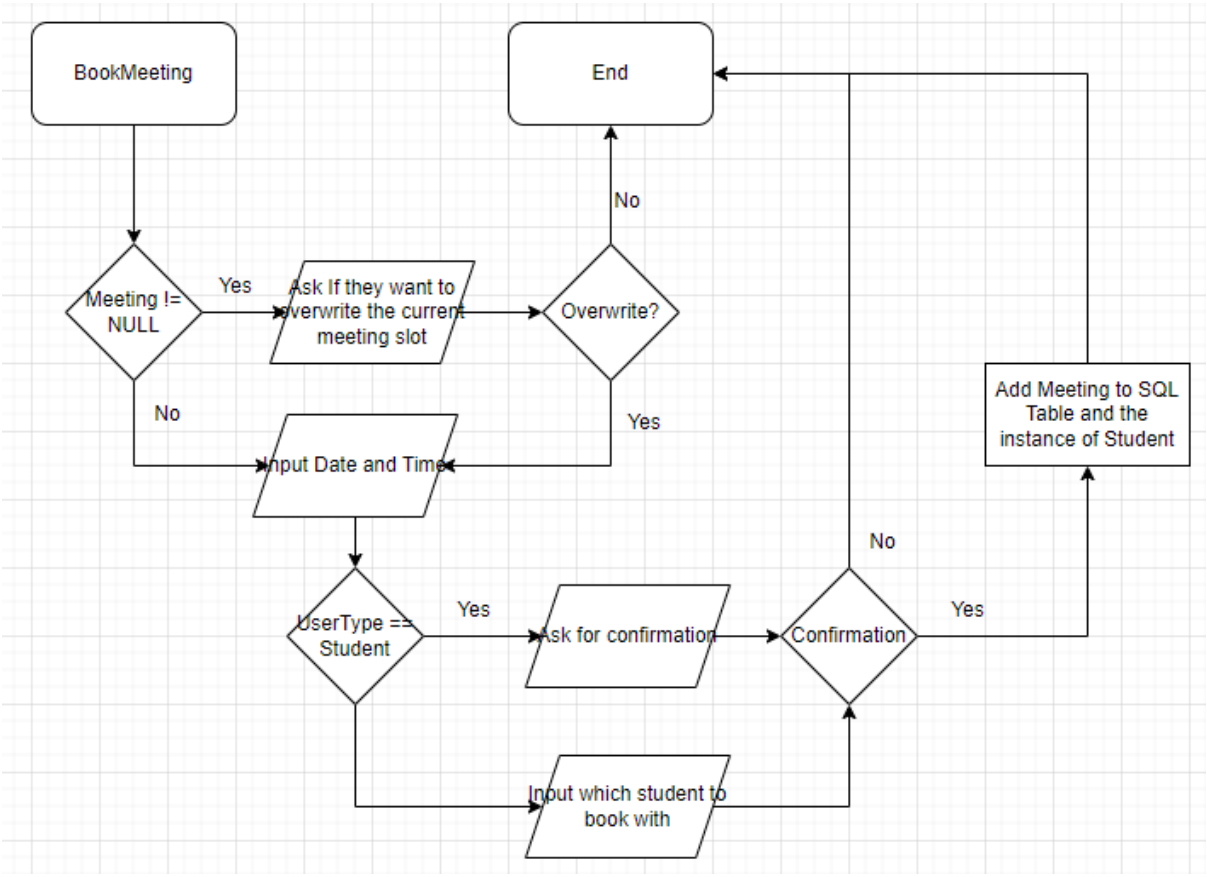


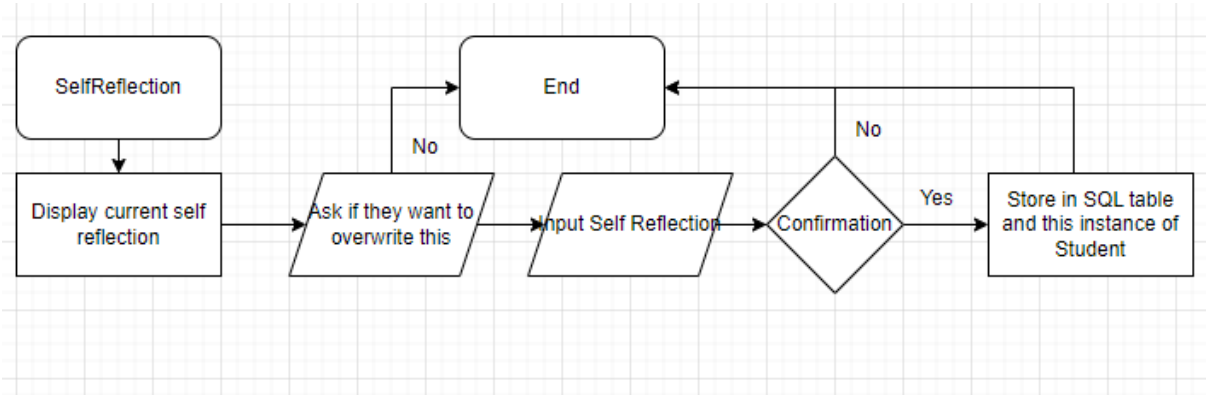
Flow Charts

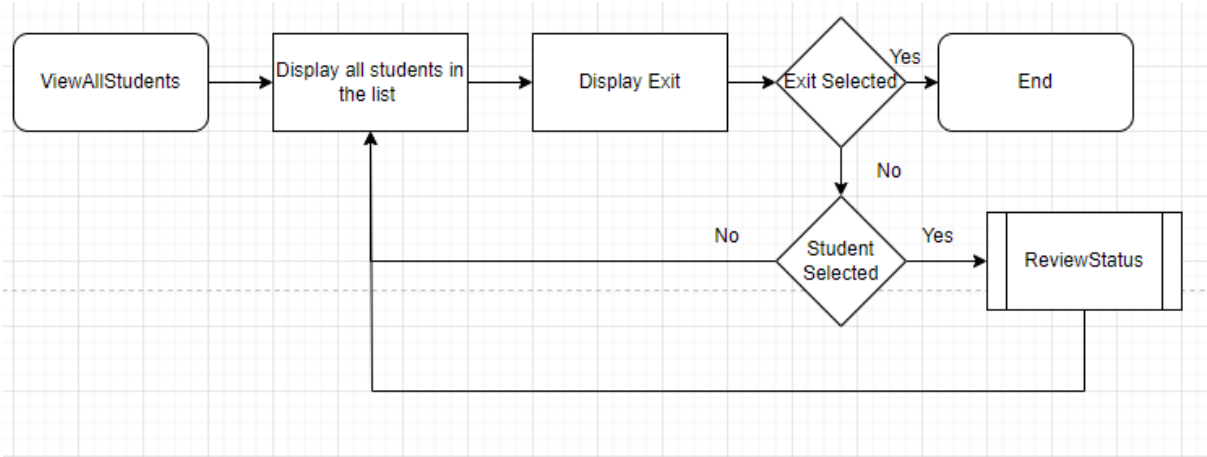

Start
Output Welcome Message
Display MainMenu
Exit Selected
Yes
End
No
BookMeeting Selected
Yes
BookMeeting
No
SelfReflection Selected
Yes
SelfReflection
No
InputGrades Selected
Yes
InputGrades
No
ViewAllStudents Selected
Yes
ViewAllStudents
ViewAllSupervisors Selected
No
ViewAllSupervisors
counter = 0
Login
Login Success?
Yes
No
No
End
counter > 5
counter increments by 1

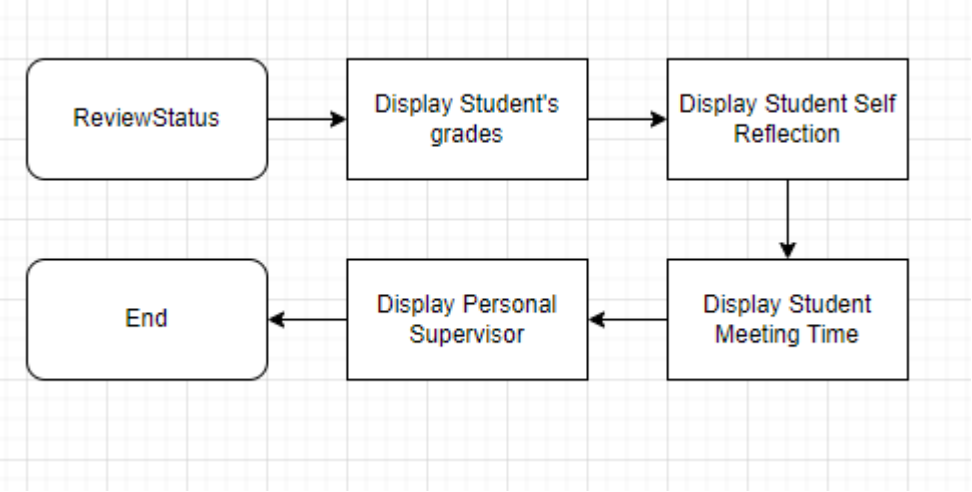

Login
Input Username
Input Password
Open SQL Table
Username Exists
No
Password Matches
No
Yes
Login Failed
Create Instance of a User with the data stored in the sql table
Login Success


Display MainMenu
UserType != SeniorTutor
Yes
Display BookMeeting
UserType == Student
Yes
No
Display SelfReflection
Display InputGrades
Display ViewAllStudents
Display ViewAllSupervisors
Display Exit
End


BookMeeting
End
Meeting != NULL
Yes
Ask If they want to overwrite the current meeting slot
Overwrite?
No
Yes
No
Input Date and Time
Add Meeting to SQL Table and the instance of Student
UserType == Student
Yes
Ask for confirmation
Confirmation
Yes
No
Input which student to book with


SelfReflection
End
No
Display current self reflection
Ask if they want to overwrite this
Input Self Reflection
Confirmation
No
Yes
Store in SQL table and this instance of Student





Database Models

Username	Password	User Type	Grade 1	Grade2	Grade3	AvgGrade	SelfReflection	Meeting Times	Supervisor
Admin	password	ST	NULL	NULL	NULL	NULL	NULL	NULL	NULL
JohnSmith	abcdefg	PS	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NathanielRoper	Hello123	Student	100	50	75	75	I am happy with the work so far	7:30 01/01/24	JohnSmith
RobinJones	examplePassword	Student	21	76	54	50.33	I am strugglin g	NULL	JohnSmith
SarahWhite	123456	Student	99	99	99	99	NULL	6:15 01/02/24	BobRoberts

BobRoberts	Bottle1	PS	NULL	NULL	NULL	NULL	NULL	NULL	NULL

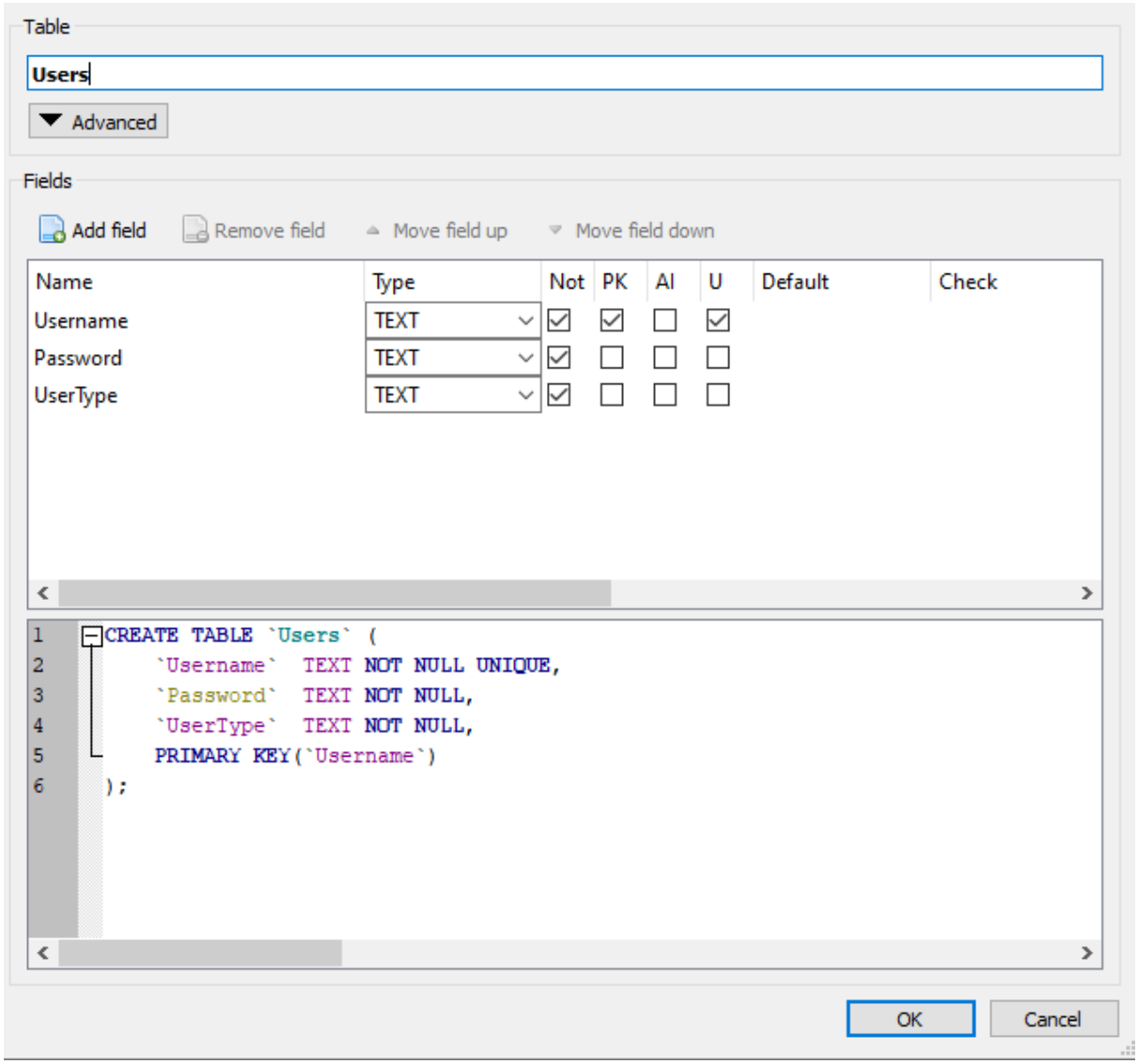
Username	**Password**	**UserType**
Admin	password	ST
JohnSmith	abcdefg	PS
NathanielRoper	Hello123	Student
RobinJones	examplePassword	Student
SarahWhite	123456	Student
BobRoberts	Bottle1	PS

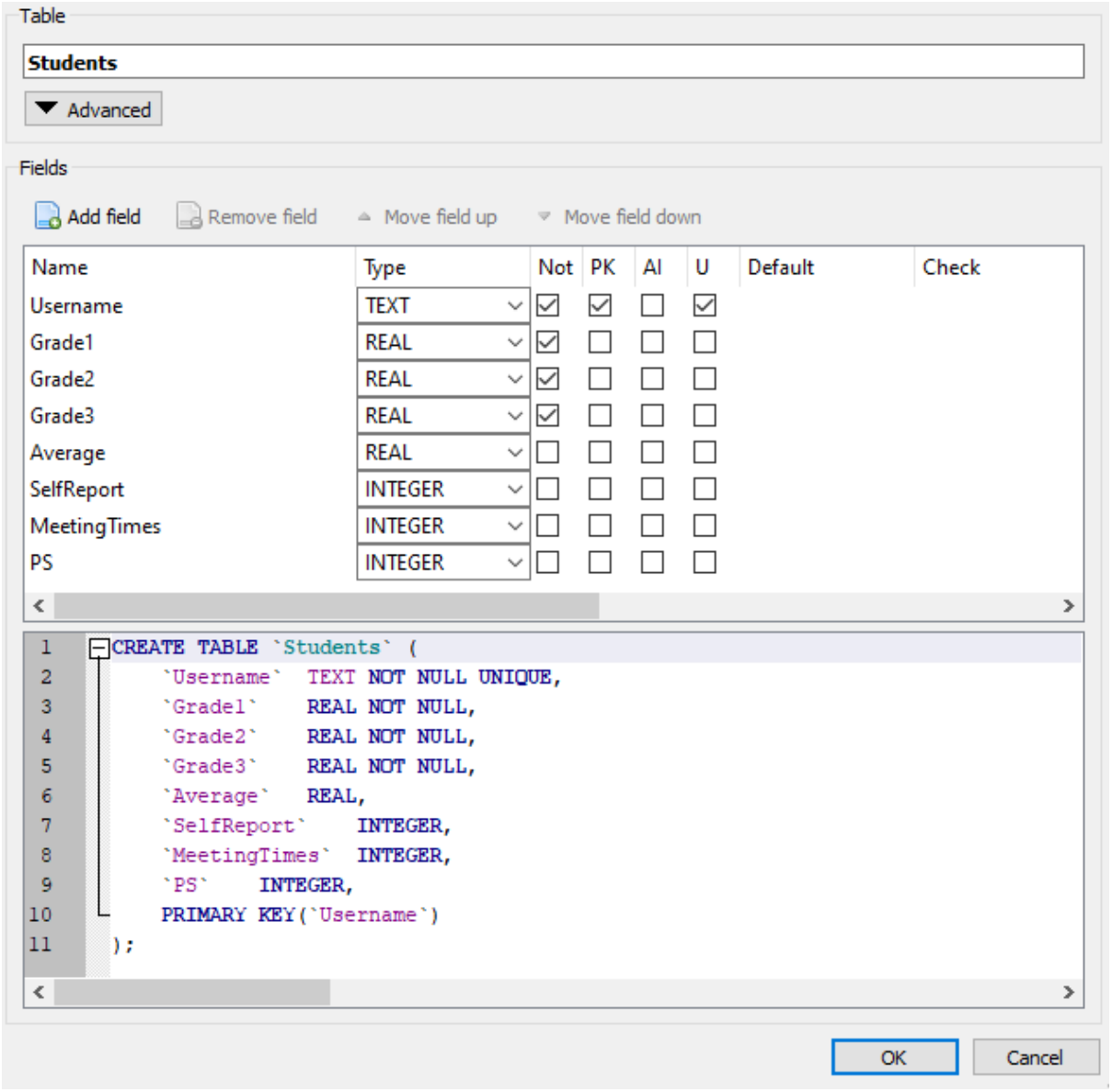
Username	**Grade 1**	**Grade 2**	**Grade 3**	**AvgGrade**	**SelfReflection**	**MeetingTimes**	**Supervisor**
NathanielRoper	100	50	75	75	I am happy with the work so far	7:30 01/01/24	JohnSmith
RobinJones	21	76	54	50.33	I am struggling	NULL	JohnSmith
SarahWhite	99	99	99	99	NULL	6:15 01/02/24	BobRoberts

Implementation

Creating Databases

Firstly, I created some SQL queries to create the 2 tables needed. I did this using SQLite (https://www.sqlite.org) and using the Database models I created earlier in the design phase.





Next, I added data to both tables and made sure they corresponded to each other.

	Username	Password	UserType
	Filter	Filter	Filter
1	admin	password	ST
2	AlexG	iuahiad2452	Student
3	BobRoberts	987654321	Student
4	BorisB	HelloWorld1	Student
5	DrStevenBlack	Arrow3532	PS
6	GeorgeMichae...	DeadlyDeadly...	Student
7	HarryKane	mongoose1424	Student
8	JackSmith	example	PS
9	JackWhite	glasses12	Student
10	JessicaHyde	hwlkjkl123	Student
11	Malachi123	Control85	Student
12	MikeMyers	password123	Student
13	NathanielRoper	123456789	Student
14	OliverTwist	123	Student
15	RachelGrace	laptop34	Student
16	SarahSmith	example87	PS

	Username	Grade1	Grade2	Grade3	Average	SelfReport	MeetingTimes	PS
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	GeorgeMichae...	98.0	65.0	44.0	NULL	I need some support I think with module 3	15:00 03/12/23	SarahSmith
2	BorisB	67.0	51.0	68.0	NULL	I am struggling to meet the workload	9:30 14/12/23	JackSmith
3	JessicaHyde	98.0	43.0	78.0	NULL	I am struggling a bit on module 2	11:30 14/12/23	SarahSmith
4	JackWhite	34.0	82.0	76.0	NULL	I am fine so far	NULL	JackSmith
5	NathanielRoper	82.0	45.0	67.0	NULL	I am content with the work so far	8:30 17/12/23	JackSmith
6	OliverTwist	67.0	52.0	34.0	NULL	I am content	14:25 10/01/24	DrStevenBlack
7	HarryKane	67.0	43.0	60.0	NULL	Everything seems ok	NULL	SarahSmith
8	AlexG	57.0	51.0	88.0	NULL	Everything is ok	14:25 18/12/23	DrStevenBlack
9	BobRoberts	99.0	100.0	53.0	NULL	NULL	14:50 11/01/24	DrStevenBlack
10	Malachi123	80.0	52.0	34.0	NULL	NULL	NULL	DrStevenBlack
11	MikeMyers	43.0	76.0	59.0	NULL	NULL	NULL	JackSmith
12	RachelGrace	98.0	43.0	78.0	NULL	NULL	NULL	SarahSmith

Now I have created 2 tables and filled them with some data I can start working on the login system at the start of the main program. The first step to implementing them in my program is to install the sqlite drivers on Visual Studio. I installed them and then stated this at the start of my program.

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.SQLite;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

SQLiteConnection sqlite_conn;
sqlite_conn = CreateConnection();

Console.WriteLine("Login");
Console.WriteLine("--------------------------");

static SQLiteConnection CreateConnection()
{

    SQLiteConnection sqlite_conn = new SQLiteConnection("Data Source=CourseworkDatabase.db; Version = 3; New = True; Compress = True;");
        // Open the connection:
    try
    {
        sqlite_conn.Open();
        Console.WriteLine("Database connection open"); //remove after testing
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    return sqlite_conn;
}
```

I created a method to create a connection and will test this to see if it connects properly. It should return the message 'Database connection open' if it does.

```
Microsoft Visual Studio Debug Console
Database connection open
Login
--------------------------

C:\Users\Nathaniel\Documents\GitHub\Nathaniel-Roper-Individual-Software-project-portfolio\PS Application\PS Applicat
bin\Debug\net6.0\PS Application.exe (process 15144) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the c
le when debugging stops.
Press any key to close this window . . .
```

This means that the connection was a success, next I will get the username and password inputs from the user and compare them to the data in the user table.