

Lab Scheduling

A dissertation submitted in partial fulfilment of
the requirements for the degree of
BACHELOR OF ENGINEERING in Computer Science

in

The Queen's University of Belfast

by

David Savage

09/05/2017

SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING and COMPUTER SCIENCE

CSC3002 – COMPUTER SCIENCE PROJECT

Dissertation Cover Sheet

A signed and completed cover sheet must accompany the submission of the Software Engineering dissertation submitted for assessment.

Work submitted without a cover sheet will **NOT** be marked.

Student Name:

Student Number:

Project Title:

Supervisor:

Declaration of Academic Integrity

Before signing the declaration below please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

I declare that I have read both the University and the School of Electronics, Electrical Engineering and Computer Science guidelines on plagiarism - <http://www.qub.ac.uk/schools/eeecs/Education/StudentStudyInformation/Plagiarism/> - and that the attached submission is my own original work. No part of it has been submitted for any other assignment and I have acknowledged in my notes and bibliography all written and electronic sources used.

Student's signature

Date of submission

Acknowledgements

I would like to thank my supervisor Dr Barry McCollum whose advice was invaluable to the development of this project. I would also like to thank Sean McKeever who aided me in solving a problem with MySQL that had hindered development for a time.

Abstract

This project is to create a system that employs heuristic techniques to generate a timetable of practical classes using limited lab space and try to increase the utilisation of the labs. All the information for this project will be stored in a MySQL database hosted on a Queens University Belfast web space while all the processing and user interactions will occur through a C# developed application created using Visual Studio 2013 as an IDE.

Contents

Acknowledgements	2
Abstract	3
Introduction and Problem Specification	5
Project Overview	5
Problem Description	5
System Requirements Specification	6
Assumptions	6
Solution Approach	6
Technologies	7
Development Strategy	7
Requirements	7
Use Cases	10
Design	12
Database Design	12
Architectural Design	13
User Interface Design	14
Implementation & Testing	19
System Evaluation and Results	27
Good Aspects of the System & Future Improvements	27
Conclusion	29
References	31
Appendix A: User Guide	32
Appendix B: Minutes of Meetings	38

Introduction and Problem Specification

Project Overview

This project was to design and develop a system that could improve the utilisation of the computer labs in Queens University Belfast. The goal was to develop a system that is functional and easy to use, that uses heuristic methods to allocate space in computer labs for practical classes. An extra aspect to this goal was the concept of an airplane seat booking type system that would allow users to view the status of the computer lab and book a workstation.

The system was made to interact with a database containing information on the modules offered, the number of students on each module and information on each computer in the labs divided up into hour long sessions spanning two weeks. This information is then used to generate a timetable for all modules that require a practical class in a lab and book the computers for it.

Problem Description

The main problem with the current system is that there are many time periods during a day when there are no spare computers available for students to privately use, which results in students coming in only to be turned away. The other extreme also happens, where the labs are almost entirely empty, which is caused in part by students not wanting to spend the time and money to go into the building only to get turned away. This is mostly due to students using them for private work, as opposed to classes which rarely take up the entire lab. The system has a separate aspect to the timetable generation that aimed to make it easier for students to utilise the lab space. In an ideal scenario this new system should try and always leave some free computers in the labs during a practical for private use.

There will be situations where this is simply not possible. Queens University Belfast currently has upwards of 1000 computing students, while only having two large computer labs with 350 computers between them. The physical limits mean that there is no fool proof system to guarantee the labs being able to cope with demand or never being empty.

When trying to develop a system to create a timetable, one problem arises almost instantly. Generating a timetable, even only using bare bones factors of the system, is an NP-hard problem. As such there is currently no way to make a system that can give you a definitive best answer, at least not one that will finish running anytime soon. Therefore what this system will try and do is get any solution to the problem; any scenario where all the classes that need to be booked have been booked will be considered a correct answer.

System Requirements Specification

Assumptions

- Lecturers will be free to take their classes whenever they are booked; there is no already existing schedule of lectures to work around at this stage.
- Allocation of specific students to specific classes is done separately, this system simply makes sure enough computers are reserved to cover the number of students
- The user interaction features are not intended for real world users at this stage, therefore no accounts/log in system will be implemented. Nor is there any validation on text entry fields at this point.

Solution Approach

The main goal for this project was that it can successfully create a schedule of lab classes using data supplied in a database. Once this schedule is generated it is be saved back to the database and can be viewed on a timetable. There is also a simple interface to allow deletion of modules from the database and addition of new ones. Other possibilities exist, such as editing information for already existing modules but that is outside the scope of this project and as such wasn't included at this time. It also goes beyond simply booking an entire room to only booking the number of computers in the room that the class requires. It books specific computers in the lab for the modules timeslot so that the map function is more helpful. This would allow students who need to use a computer for private silent work to see where on the map any free computers might be and come in to work while a practical class is taking place.

The system looks at how many students a module has, and by extension works out how many practical classes it needs. The size of these groupings are controlled by a threshold percentage that creates classes that can't be bigger than X% of the number of lab computers.

Once this functionality was implemented, the system was developed further to open up any computers not booked by a practical class to be manually booked by students. This allows students to know there will be a computer waiting for them to use before physically checking the room. This should drastically cut down the amount of students who go and check the computer labs only to turn away after seeing the room is full.

The system lets the user view the amount of bookings in a lab for the current week and the next week, as well as let them book a computer. The system will have an option to manually progress the system from week to week. This function would essentially scrap the current

week, and copy all of the bookings that are in next weeks schedule before wiping the next week of everything but class bookings.

Technologies

The database for this system is a MySQL database, hosted on a QUB web space. While the interface for all interactions with users was built in Microsoft Visual Studio using C# as a development language. A few other combinations were considered for the project including using Java as the development language with NetBeans as an IDE, or the possibility of using a Microsoft Access database instead of MySQL.

However due to the power and capability of MySQL it was picked for the database and C#/Visual Studio were picked due to the easy of SQL interactions within the IDE as well as other helpful IDE features.

Development Strategy

This project was developed using an evolutionary prototyping methodology. This allowed the system to be continuously refined, adding more of the required and desired features. It also means I am able to improve features that have already been added, if use of the features has revealed scope for improvement.

This did require a good understanding of the requirements before they were implemented, as well as some testing to be done as features were implemented so that future improvements are not being made on top of a bed of flawed features.

Requirements

The system should be able to generate a collection of classes from the list of modules that are small enough groupings to fit inside one of the two lab rooms. – Mandatory

This requirement has been met; the system takes the modules and the students on them and breaks them down into groups that are a percentage of the maximum capacity of the lab rooms.

The system will book the previously created classes in timeslots that have room, and in the event that it cannot find space for a class as it is, it will make a list of those failed classes. – Mandatory

This requirement has been met; the system takes a Dictionary with the classes that were created and goes through a list of randomised timeslots and books the class. In the event it cannot book a class in any timeslot it sets it aside in a separate list.

The system will take the list of failed classes and create new classes which have smaller student counts and try to fit those new smaller classes into the timetable. It should do this automatically. – Mandatory

This requirement has been met; classes are broken up using a threshold percentage against the maximum capacity of one of the lab rooms. In the event the system has any classes in the failed list it will dissolve those classes back to just being groups of the modules. It will then create new classes groupings with a lower threshold percentage, resulting in smaller classes than before and therefore more chance of them fitting into a timeslot.

There will be a list of all the modules in the system and their respective student counts displayed alongside options to delete any of those modules or add a new module. - Mandatory

This requirement has been met; the Module Timetable Generation screen loads up a list of the modules that are in the database and displays them alongside their student count. Below this list there is the option to delete any of those modules by simply copy and pasting their name, as well as adding any new module.

The bookings must be made by booking the required number of computers, as opposed to just booking the entire room, leaving excess computers free for use. – Mandatory

This requirement has been met; if booking a class with 60 students the system will only book 60 computers for it, it can then go on to use the remaining computers in the lab to book another class if needed.

You should be able to view a grid that shows how many computers are booked for that lab in all the hour long blocks of a given week. – Mandatory

This requirement has been met; a user can look at a timetable screen for either lab which shows what percentage of the computers are booked for each time slot. Every day has ten time slots of an hour each and two weeks are stored and displayed.

The status of one of the computer labs at any given timeslot should be displayed as a map, with nodes each representing a single computer laid out in the same way as the physical rooms structure. – Mandatory

This requirement has been met; each lab is represented by a screen of buttons, each representing a single computer, laid out to mimic the physical design of the lab. This means not only can users see how many computers are free in a specific lab at a specific time, but also where in the room the free computers are.

The system should record statistics of the status of computers. -Mandatory

This requirement has been met; when the system progresses forward a week it first cycles through all of that weeks' time slots and works out what percentage of computers in each lab remained open, were booked for private use, or were booked for a class. Once it has the percentages for every time slot of the week it saves them all to a text file with the title "Statistics File *Current Date*".

The timetable view for a lab should have a colour scheme that shows, in increments of ten, what percentage of the computers are already booked at that time. Hovering over the button should reveal the exact percentage. – Desirable

This requirement was met; there is a ten step colour gradient from green if the lab is entirely free to a dark red if the lab is 100% booked up. Hovering over the button for a specific timeslot has text appear revealing the exact percentage of the lab that is booked up.

The nodes in the map of the lab should have a colour scheme to show what that computer is booked for, or whether it is free. – Desirable

This requirement was met; upon loading the map of a lab for a specific timeslot it retrieves the status of each computer from the database and colours the buttons to show the status of a computer. Red means that computer is booked for a class, green means the computer is currently free, and blue means that the computer has been booked by a student for private use. This distinction is important because while computers booked for a class will be used for the entirety of the time, a computer booked for private use could actually be available. This could be due to the user who booked it not showing up, or not needing the full timeslot they booked.

Clicking on one of the nodes in the map should allow that computer to be booked, unless it is already booked in which case the user should be warned as such. – Desirable

This requirement has been met; clicking on any of the buttons representing a computer node will change the nodes status to "Private Use". As this was only a desirable requirement it is

very simple but the future scope for the feature would be to have users log in using an idNumber which would then be recorded with the booking. This would also open up avenues such as recording how many computers each user has booked and how many of them they have actually arrived and logged into the computer for.

You should be able to search a module title once classes have been generated and be told when its classes were booked and in which lab. – Desirable

This requirement has been met; once classes are generated a user can either type in the name of a module or copy it from the table and a message box will appear and tell the user which timeslot the module has classes in, and which lab each booking is in.

A statistics panel that lets user see the average usage for the labs plotted as a graph, possibly divided up by what the computer was used for. – Luxury

This requirement has not been met; the system does generate statistics that could be used to develop this feature. However during development it was decided to focus on other higher priority requirements first, and as such this requirement was never gotten round to.

Use Cases

Use Case #1 – Adding a Module

Methods Used: btnAddModule_Click

A user opens the system and is faced with the StartMenu form. They use the “Lab Classes Generation” button to close the StartMenu form and open the ModuleTimetableGeneration form. Once on this form they use the area marked “Add New Module” by entering the name of the module, its course code, student count and level into the labelled text boxes and pressing the “Add Module” button. This causes the system to take the entered information from the text boxes and enters them into an SQL Insert Into query which is then sent to the database. A message box pops up to inform the user that the module was added and the table displaying modules and their student counts is updated to show the new module.

Use Case #2 – Deleting a Module

Methods Used: btnDeleteModule_Click

A user opens the system and is faced with the StartMenu form. They use the “Lab Classes Generation” button to close the StartMenu form and open the ModuleTimetableGeneration form. Once on this form they see all of the modules that exist in the system in a table on the

right side of the screen, along with the number of students in them. The user then types in the name of one of these modules or copies it from the table and pastes it in and presses the “Delete Module” button. This buttons takes the name the user entered and creates a SQL Delete query with it, before sending it to the database. A message box appears to let the user know that the module they entered has been deleted and the table displaying the modules and student counts updates to reflect this change.

Use Case #3 – Searching for Classes of a Module

Methods Used: btnFindClasses_Click

A user opens the system and is faced with the StartMenu form. They use the “Lab Classes Generation” button to close the StartMenu form and open the ModuleTimetableGeneration form. On this form the user can either type the name of a module into the appropriate textbox or copy paste it from the table on the form. Once they press the “Find Classes” button a message box will appear and tell the user which days the module has classes, when the classes are, and in which lab the classes are.

Use Case #4 – Progressing the week of the system

Methods Used: btnProgressTheWeek_Click, generateStatistics

A user opens the system and is faced with the StartMenu form. They use the “Lab Classes Generation” button to close the StartMenu form and open the ModuleTimetableGeneration form. One this form the user clicks the “Progress the week” button. This clears the first week of any private use bookings and then moves any private use bookings that are in the second week up into the first week while leaving any class bookings alone. Before the bookings in the first week are cleared it first finds out a percentage breakdown of the booking status of all the computers across all the time slots and saves it into an external file with is titled “Statistics File” appended with the current date. A message box appears to inform the user that the changes were saved into the database successfully.

Use Case #5 – Clearing the bookings table and generating new classes

Methods Used: btnResetTheBookingTable_Click, btnRunCode_Click, GeneratePracticalClasses, BookPracticalClasses, copyTheBookingsToNextWeek (Optional: GeneratePracticalClassesFailed, BookPracticalClassesFailed)

A user opens the system and is faced with the StartMenu form. They use the “Lab Classes Generation” button to close the StartMenu form and open the ModuleTimetableGeneration

form. On this form the user has two options. They can press the “Reset The Booking Table” button which will set every computer in every timeslot to Open, and then pressing “Run The Code” will make new classes and book them. Alternatively if the system already has private bookings then the user can press the “Clear the booking table of classes and generate new classes”. This will clear the table of any previously generated classes and replace them with new ones, but will not get rid of any private use bookings that are stored in the table. In the event that the system cannot book all of the generated classes it will use two fall-back methods to modify the process and try again.

Use Case #6 – Checking the status of a lab/Booking a Computer for private use

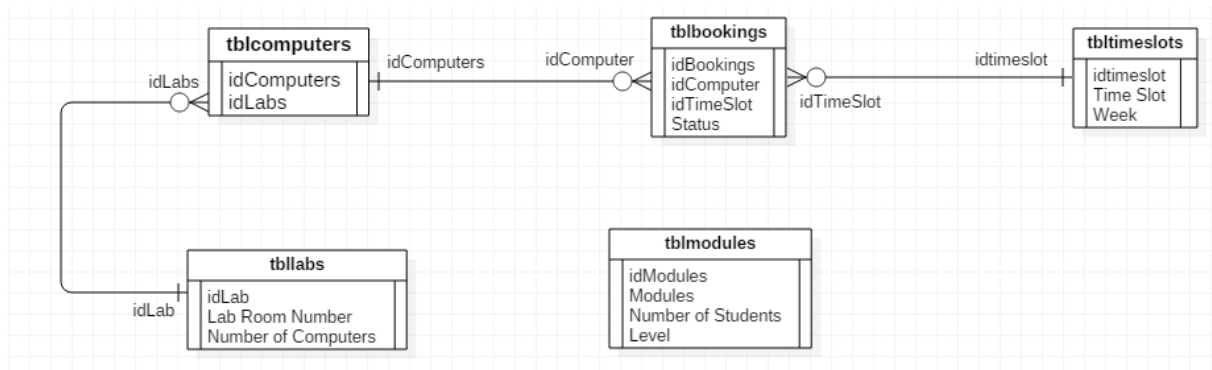
Methods Used: colourTheButtons, MapLab1_Load, fillButtonArray, findButton, btnComp_Click

A user opens the system and is faced with the StartMenu form. They use the “Lab 1 Timetable” button to close the StartMenu and open the TimetableLab1 form. On this form they can see a status of the lab for each time slot. Once they have decided on the time slot they want they click it and it opens up the MapLab1 form. On this form they see 150 buttons, some of which may be red or yellow depending on which time slot they picked. They click a green node and that node changes status to book for private use. Had they clicked a red node they would have seen a pop up box telling them that computer has already been booked for a specific class. If they clicked a yellow node they would have been told that computer has already been booked by another student for private use.

Design

Database Design

The database design is rather simple as there isn't a lot of information that needs stored. 350 computers are recorded as belonging to either lab 1 or lab 2 using a one to many relationship between tbllabs and tblcomputers. The table tbltimeslots then contains information on the 100 timeslots and both this table and tblcomputers are related to tblbookings by a one to many relationship. Finally tblmodules stores the information on the modules which will have classes booked but it shares no relationship with any of the other tables at this time. A simple UML diagram is displayed below.



The relationships between the tables have almost no bearing on the rest of the system, but during the initial set up of creating all the time slots and computers it was useful as it ensured all the rows created in the tblbookings table were valid without having to manually check all 35,000 rows.

It was considered that the booking table would have a relationship to tblmodules and it would store a module code for every booking. However as the relationships aren't going to be important once the data is taken out of the database and processed in the application it was decided that this wouldn't be worth it.

Architectural Design

The system contains six forms.

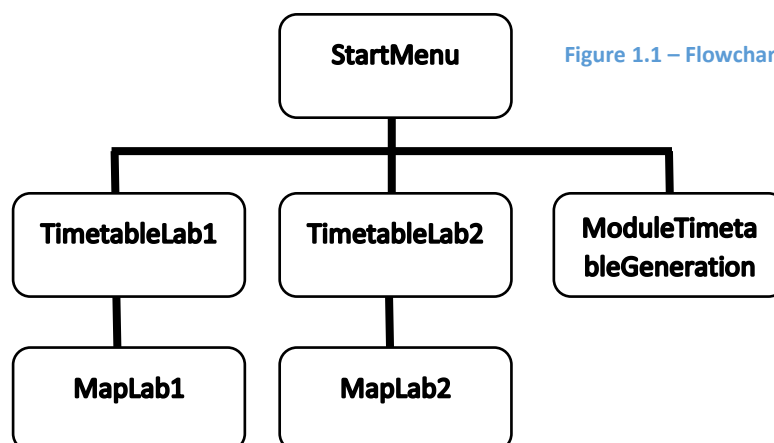


Figure 1.1 – Flowchart of form connections

The flow of the system is quite simple. For any administrative features, such as generating practical classes or adding a new module to the schedule, the ModuleTimetableGeneration form is used. Otherwise if you are looking to either book a computer or check what the status of one of the labs will be at any given time you chose to go to TimetableLab1 or TimetableLab2. This screen will then lead you onto MapLab1 or MapLab2 once you have selected a time slot.

Currently there is no log in system so there is only one user who can access every form. If a log in feature was added in the future, certain forms would be locked off to some users. Students would have zero access to the ModuleTimetableGeneration form, for example. How a user interacts with a form would be a change I would implement as well. A staff member on the MapLab1/2 form, for example, should be able to disable a computer as well as book one. This would make the computer unable to be booked until such a time as it was reversed and would be for situations where a computer station has broken.

User Interface Design

ModuleTimetableGeneration

This form serves two main functions. The first is run the heuristic methods to generate and book a class schedule, and the other is to allow various actions to be made with the list of modules in the database. As such there are three separate areas to contain the controls a user needs to; find the classes of a specific module, delete a module from the database, and add a new module to the database. Above these are the buttons that affect the timetables and any bookings in the database, and finally to the right of all of this is a display showing the modules that are in the table. The design of this form only changed very slightly from my original idea. Below the original idea and the final form are shown together. The biggest changes between these stages was the addition of the find classes areas which wasn't originally planned, and the add modules area needing more space.

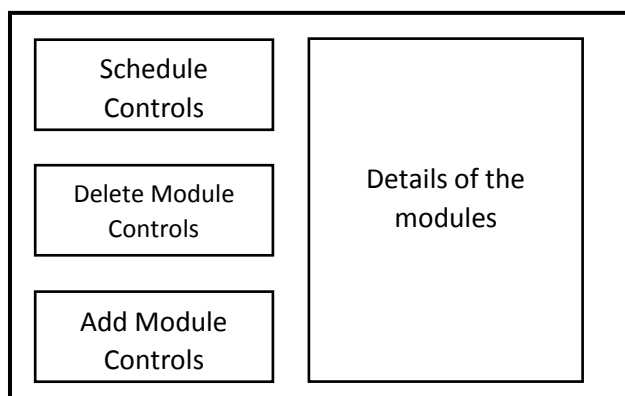


Figure 1.2 – Initial design of ModuleTimetableGeneration form

The screenshot shows the 'ModuleTimetableGeneration' application window. It contains several functional areas:

- Top Left:** Buttons for 'Run The Code', 'Reset The Booking Table', 'Clear the Booking Table of classes and generate new classes', and 'Progress the week'.
- Find A Modules Classes:** A section with a text input 'Name of module to find:' and a 'Find Classes' button.
- Delete A Module:** A section with a text input 'Name of module to delete:' and a 'Delete Module' button.
- Add New Module:** A section with inputs for 'Name of module to add:', 'Course code:', 'Number of students:', and 'Course Level:', along with an 'Add Module' button.
- Back to Menu:** A button located at the bottom right of the form.
- Table:** A table listing modules with columns: idModules, Modules, Number of Students, and Level. The table contains 20 rows of data.

idModules	Modules	Number of Students	Level
CSC1009	Introduction To Software Engineering & Project Management	399	Level 1
CSC1017	Reasoning for Problem Solving	316	Level 1
CSC1018	Foundations of Computing Systems	341	Level 1
CSC1020	Programming	244	Level 1
CSC1021	Fundamentals of Programming	203	Level 1
CSC2011	Professional Computing Practice	339	Level 2
CSC2038	User Experience Design	132	Level 2
CSC2039	Architecture and Networks	227	Level 2
CSC2040	Data Structures, Algorithms & Programming Languages	230	Level 2
CSC2041	Information Management	125	Level 2
CSC2042	Information Modelling	214	Level 2
CSC2043	Software Development - Processes and Practice	135	Level 2
CSC2046	System Administration and Maintenance	81	Level 2
CSC3021	Concurrent Programming	153	Level 3
CSC3030	Intelligent Information Systems	104	Level 3
CSC3031	Software Design Principles & Patterns	54	Level 3

Figure 1.3 – Final design of ModuleTimetableGeneration form

Message boxes will appear whenever a user tries to delete or add a module to inform the user that the changes either succeeded or failed. Similarly when a user tries to find the classes for a given module the result is returned in a message box.

This figure shows a close-up of the 'Find A Modules Classes' section of the form. The text input 'Name of module to find:' contains the word 'Programming'. The 'Find Classes' button is highlighted. To the right, a message box is open, displaying the following text:

This module is booked in the follow times:
Thursday 10-11 in Lab 1
Thursday 12-13 in Lab 1

The message box has a red close button in the top right corner and an 'OK' button at the bottom right.

Figure 1.4 – Example of the find classes message

TimetableLab1/TimetableLab2

The design of the timetable screens is very simple and it is basically identical to the original design.

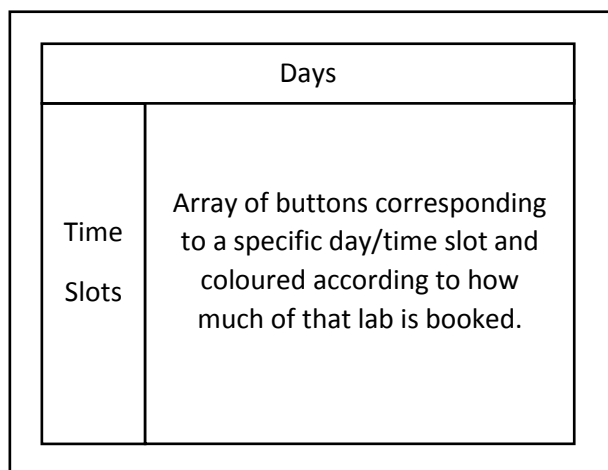


Figure 1.5 – Initial design of the Timetable forms



Figure 1.6 – Final design of the timetable forms

There is an eleven step colour gradient for these buttons that ranges which changes every 10%






0-10%	 #0EE500	10-20%	 #37E100	20-30%	 #60DD00
30-40%	 #87D900	40-50%	 #ADD500	50-60%	 #D1D200
60-70%	 #CEA800	70-80%	 #CA7F00	80-90%	 #C65700
90-100%	 #C23100	100%	 #BF0C00		

Table 1 – Colour gradient for the timetable forms

The two extremes of this colour gradient, green (#0EE500) and red (#BF0C00), were chosen because of the accepted social convention that green is positive and red is negative. Therefore when the lab is almost totally empty and you are sure to get a computer, or a group of computers, it's represented by a bright green and when you stand a very low chance of a computer being available it is a darker red.

Additionally when hovering over one of the time slots you will be shown the exact percentage of computers that are booked.

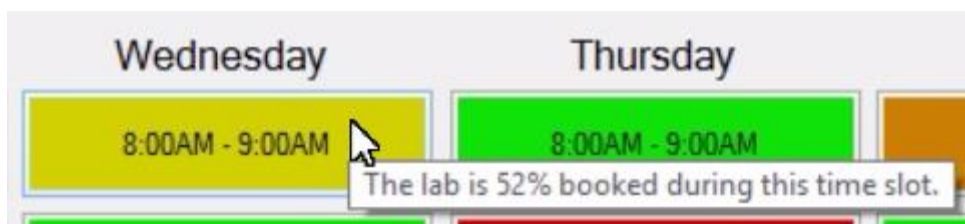


Figure 1.7 – Exact percentage of a given time slot

MapLab1/MapLab2

These forms were designed to mimic the physical layout of the computer labs. Due to this there was no design for the screen drawn. It would simply be a series of buttons laid out in rows of five just like the computer lab. The Lab1 form has 150 of these nodes while the Lab2 form has 200 nodes, but they are laid out exactly the same, just spread out across a larger area.

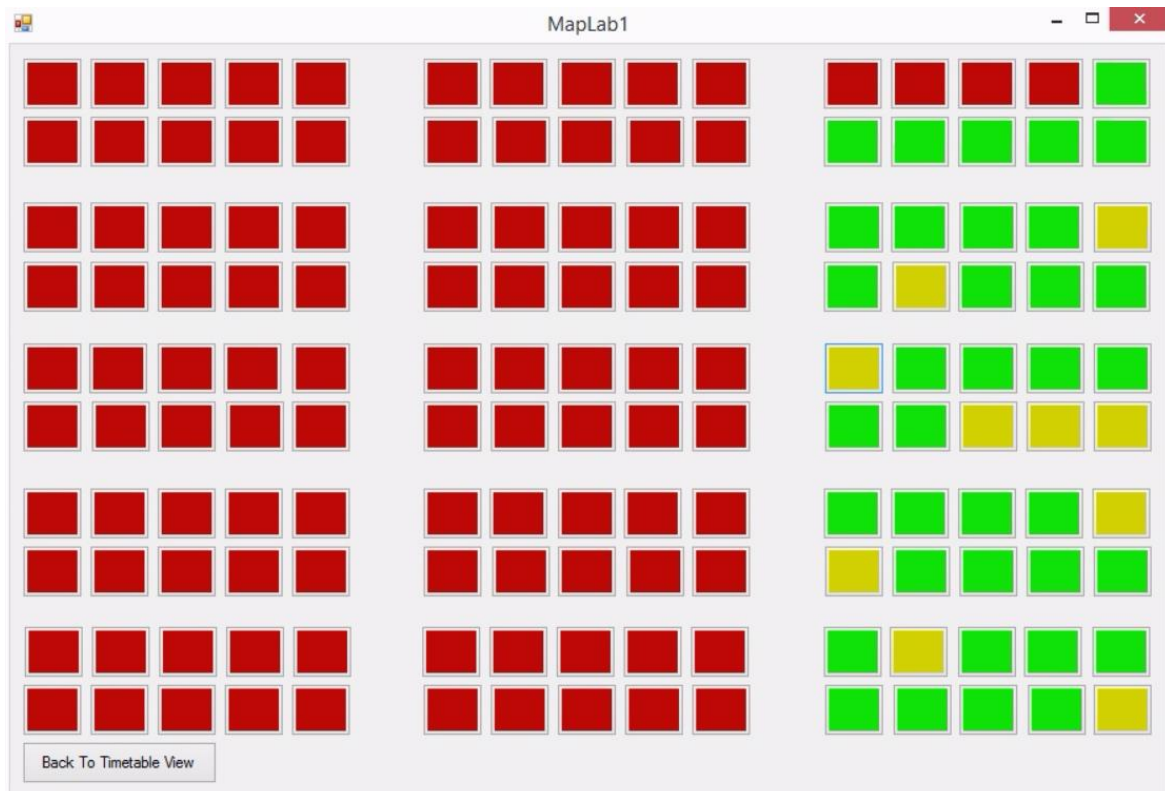
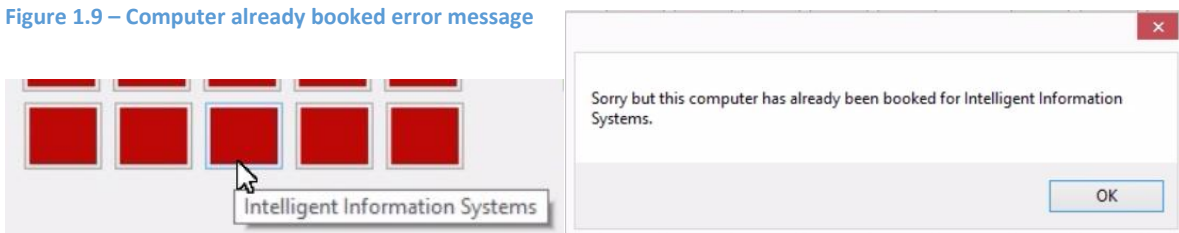


Figure 1.8 – Final design of the Map forms

This has a similar colour scheme to the Timetable forms. It uses three colours from that gradient scheme; green (#0EE500), red (#BF0C00), and the 50-60% yellow (#D1D200). The green colour indicates that the node in question is currently open and can be booked for private use. A red node has been booked for a specific class and the class can be seen by hovering over the node. In the event a user clicks a red node a pop up box will appear and inform them that they can't book this computer.

Figure 1.9 – Computer already booked error message



Meanwhile the yellow colour indicates the computer has been booked for private use. As with a computer booked for a class hovering over the node will inform the user that yellow means the computer is booked for private use and if they attempt to book it will be warned that they cannot.

Implementation & Testing

This section will look at all of the methods created for this project, going form by form. It will explain what these methods do and were written as the methods were tested, ensuring they do what they are described to in this section correctly.

TimetableLab1/TimetableLab2

The TimetableLab1/2 forms have two methods. The first is colourTheButtons which takes in a button as a parameter. This method goes through each of the time slot buttons in turn and gets the status of all the computers in the applicable lab for that time slot. It then works through these records and counts how many computers are booked and divides this number into a percentage. It then attaches this to the current button as a ToolTip and using a series of nested if statements it finds the correct colour for the button and changes it to said colour.

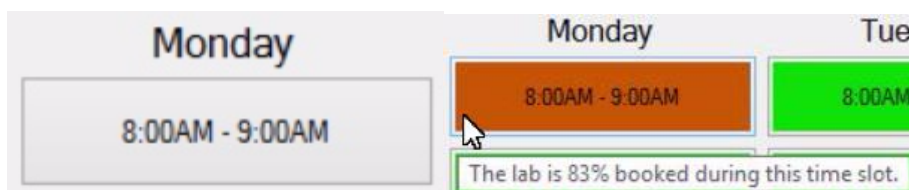


Figure 2.0 – Before and after colourTheButtons

The other method used on the timetable forms is simply a form transition that was slightly modified to pass through a variable. btnTimeSlotPicked_Click uses a substring of the current button name to isolate what the select time slot is and passes it through to the map forms.

Both these methods were passively tested through use of the system as one deals with the form loading and the other is a modified form transition, both of which were ran over and over again during development of the system,

MapLab1/MapLab2

When the map form opens it receives a time slot value from the previous form, but if anything goes wrong and it fails to get one it defaults to the first time slot. Following this it creates an array of all the computers in the current lab and then looks through the database, using a

DataTable, to find the rows that relate to the specific computers and timeslot that is in question.

Then using a double nested if it goes through each button and checks if the status is Open, in which case it sets the colour to green and the ToolTip to “Open”. If the status is Private Use then it sets the colour to yellow and the ToolTip to “Private Use”. As these are the only two standard options if the status is not one of the two then the status must be a module name, meaning a class has been booked using this computer. Therefore the colour is set to red and the ToolTip is set to the module name.

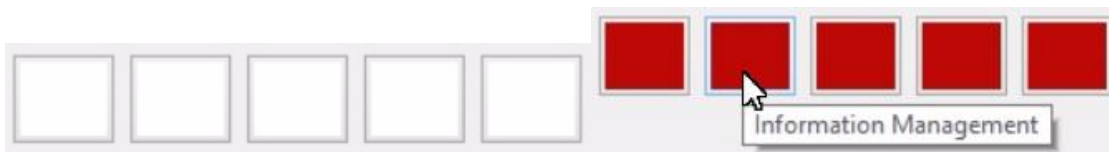


Figure 2.1 – A row of computers in MapLab1 before and after the page has loaded

To do this I have to have an array containing all 150/200 buttons so I can manipulate their colour and ToolTip text. In order to create this array of buttons I used two methods called fillButtonArray which takes in no parameters and findButton which takes in a string parameter. FillButtonArray runs through a loop creating strings which match up to the names of the buttons using the index of the loop to complete the ID, and then uses findButton with the string that was created to find the button and store it in the array.

```
for (int index = 0; index < 150; index++)
{
    if (index < 9)
    {
        Lab1ComputerButtons[index] = findButton("btnComp000" + computerIndex);
    }
    else if (index < 99)
    {
        Lab1ComputerButtons[index] = findButton("btnComp00" + computerIndex);
    }
    else
    {
        Lab1ComputerButtons[index] = findButton("btnComp0" + computerIndex);
    }
    computerIndex++;
}
```

Figure 2.2 – Using the loop index to create references to the button controls

The final functionality of the map forms occurs when a user clicks a computer node to try and book the computer. This will activate btnComp_Click which will get a substring of the buttons name to find out what computer it is the user has selected. It will then fetch the status of the computer from the database again, the first time being when the page was loaded, just to make sure it is still the same status.

Then it simply uses a nested if statement to show the user a message box saying it cannot book this computer, if the status was anything other “Open”. If the status is open then it will change the status of that computer in the database and then update the ToolTip and colour of the node.

The functionality of both the load form method and the btnComp_Clicked method were tested repeatedly through the development of the system as they were used to verify if other methods, such as progressing the weeks, had had the desired effect.

Module Timetable Generation

There are various different functions on this form so I’m going to explain them in separate sub sections.

Finding a Modules Classes

Once the user types the name of the module they’re looking for, or copies it from the display, and presses the “Find Classes” button this triggers the btnFindClasses_Click method. This method takes the text that the user has entered and looks through the tblbookings database table using a DataTable and when it finds a computer that has been booked for that module it puts the timeslot ID into an array. It also keeps the IDs of computers that it finds, which is used to find out which lab the bookings are in. It then uses the time slot ID from earlier to return the day and time of the timeslot before finally creating a string that displays all of the time slots that were found, followed by the lab that they’re booked in and then displaying it in a message box.

```
computerID = computerID.Where(x => !string.IsNullOrEmpty(x)).ToArray();
string [] strLabs = new string [computerID.Length];
int intLabFoundIndex = 0;
foreach (DataRow row in dtComputers.Rows)
{
    for (index = 0; index < computerID.Length; index++)
    {
        if (row["idComputer"].ToString().Equals(computerID[index]))
        { strLabs[intLabFoundIndex] = row["idLab"].ToString();
          intLabFoundIndex++;
        }
    }
}
```

Figure 2.1 – Finding the class location through a single booked computer

This method was tested simply by entering the name of a module and seeing what time slots the system returned, before checking through phpMyAdmin to see whether that class was indeed booked in that time slot. It was also tested by searching for an invalid module name, in which case the system informs the user that it couldn’t find any classes.

Adding a new Module

Once the user enters the four pieces of information about the module they want to add and press the “Add Module” button the `btnAddModule_Click` starts. It simply takes the text from the four textboxes and puts it into a SQL Update query which it then sends to the database and clears the text boxes before telling the user their module was added successfully using a message box.

This functionality was tested simply by using this method to add a new module and then viewing the database through phpMyAdmin to check if the module was added correctly.

Deleting a Module

Similar to adding a module once the user enters the name of the module they wish to delete, either through typing or copying from the display, and presses the “Delete Module” button it starts the `btnDeleteModule` method. This method takes the text that was entered and uses it to create a SQL Delete query which it then sends to the database and informs the user the deletion was successful using a message box.

Similarly to adding a module this was tested by deleting a module through the systems interface and then checking the database through phpMyAdmin to see if that specific module was indeed deleted.

Progressing the timetable by a week

Once the user presses the “Progress the week” button the method `btnProgressTheWeek_Click` starts. The first thing this method does is call another method called `generateStatistics` which goes through every time slot of the current week and for each lab works out what percentage of the computers in it were booked for private use, booked for a class, or left open. Once it has these percentages for all the time slots it saves these to an external file and then terminates, sending the program back to the `btnProgressTheWeek_Click` method. This method then creates a series of SQL statements and amends them into a string before sending it to the database all in one go at the end. The first of these is to reset the status of every computer for every time slot in the first week of the system to “Open”.

After this it starts looping through all of the timeslots in the second week and looks at the status of each individual computer. If the computers status is open then nothing happens, if the computer is booked for a class it creates a SQL query to book that computer in that time slot for that class in the first week. Finally if the computer is booked for private use it creates

a query to carry that booking forward to the first week, the same as a class booking, only it also creates a second query which resets the original private use booking to open.

Once it has all of the SQL statements made it sends them all to the database and alerts the user that the changes have been saved to the database successfully with a message box.

In order to test this functionality a series of private bookings were made in various time slots throughout the next week range of time slots. Then after this method was ran it was checked that those bookings no longer appeared in their original time slot and instead appeared in the same place under the current week, while also making sure all the class bookings had remained the same.

Fully clearing the timetable

When a user presses the “Reset The Booking Table” button it triggers the btnResetTheBookingTable_Click method. This method simply creates a single SQL statement which changes the status of every record in the table to open. Afterwards a message box appears to inform the user that the table was reset successfully.

To test this method it was simply ran and then the table was checking through phpMyAdmin to see if the status of every computer had been reset to open.

Partially clearing the timetable and generating new classes

In the event that you want to get rid of the current timetable of classes and generate a brand new one, but there is private use bookings on the system then a user should use the “Clear the booking table of classes and generate new classes” button. This triggers the btnResetAndRun_Click method which first clears the database of any scheduled classes, but in no way changes any private use bookings. Once this is done is simply generates a new class schedule around the private use bookings using the same methods described in the next section.

This method was tested by running the first half of it, which clears only class bookings from the table, and then checking through phpMyAdmin and the timetable/map forms that the private bookings have not been affected. This is because the second half of this action is the same as the normal btnRunCode_Click that is covered on it’s own in the next section.

Generating a schedule of classes

To generate a schedule of classes and book those into the database the user must press the “Run The Code” button. This starts the btnRunCode_Click method which prepares two

arrays, one for the modules and one for their student counts, a dictionary to store the class groupings and a hardcoded threshold percentage which is set to 90%. It then calls the `GeneratePracticalClasses` method and passes in the aforementioned arrays, dictionary, and threshold percentage.

The first thing the `GeneratePracticalClasses` method does is populate the modules and student count arrays with the information saved in the database. These arrays are then sorted by their student counts, largest to smallest using a bubble sort. It then moves into a while loop which runs until there are no groupings that are too large to fit in the largest lab. This code goes through the modules array and first checks the student count against the maximum limit for class size. This maximum limit is the size of the largest lab multiplied by the threshold percentage, currently this is 90% of 200. If the module has over that amount of students then it halves the number of students and checks again. It will keep doing this until the number can pass this check.

Once it knows that the current class grouping could fit into lab two it moves and checks if it could find into lab 1, with a maximum capacity of 150. This figure is also multiplied by the earlier threshold percentage to ensure that a class won't ever take up 100% of a lab. Once it figures out which of the two labs this class grouping should be scheduled for it attaches a number on the end and saves it into the dictionary. Thus if a Programming module has three practical classes they are saved as `Programming1`, `Programming2`, `Programming3`.

Once the class is decided it adds up how many students from the current module have already been covered by this class. It then goes back and gets the total amount of students in the module, removes the amount it has already booked a computer for and starts again. This runs on the one module until that module has zero students left. Once a module is completely broken up into classes it then moves onto the next module and will continue until the dictionary contains all the classes that are to be booked.

This method was simply tested by writing the contents of the dictionary out to an external text file where the full breakdown of how many classes each module was broken up into and how many students were in each class could be seen.

The code now returns to `btnRunCode_Click`, which calls another method called `BookPracticalClasses` which passes in the dictionary of classes and the threshold percentage used to generate them. The first thing this method does is set up a series of arrays. Two one dimensional string arrays are set up to store the IDs of all of the computers, divided up by lab.

Then a 2D string array is set up and populated with the information on the time slots. The system covers two weeks, but the class schedule should only spread across one week so it only takes in the first 50 time slots, which is the first week.

Then a short for loop takes each of the classes that is stored in the dictionary and removes them, moving them into a 2D array. This happens because next the 2D array is sorted so the classes are listed highest to lowest, again using a bubble sort. The final step before the booking of classes begins is the array of time slots is sent to another method called `RandomiseArrayOrder`. This method simply takes in this array and shuffles it around using the C# function `Random.Next`.

This loop runs through all of the classes that need to be booked and the first thing it does is check whether the `TimeSlotIndex` is over 50. As that would indicate that this module failed to get booked in any of the time slots and should be added to an array of modules that couldn't be booked this time, before moving on to the next module and starting again from the first time slot.

If the loop is in a valid timeslot it will check which lab the class in question was assigned back at the `GeneratePracticalClasses` stage. Following this is a switch statement depending on whether it's going to try booking the module in lab 1 or lab 2. The explanation of these cases is almost identical so this will just assume it will be booked in Lab 1. The first step of this is an array gets the status of all of the computers in the lab at the current time slot. Once it has got the status of all the relevant computers from the database it counts how many of these have the status open. This allows the program to check if the lab has enough free computers for this class at this time slot. If the class cannot fit, the booking in this time slot has failed and the program will move to the next time slot and try again.

If the class can fit in the lab and the current time slot then it creates an array and stores the information of all the computers that are free in that array. Now that this is done the program is ready to book the current class. The time slot is within a one week window, and there is room in the primary lab for the class during the current time slot. Now a simple loop just creates SQL queries that are all appended into a string that books computers for the current class. This is then sent to the database and the current class has been booked.

Once it has ran through every class that was generated it checks a Boolean value to find out if any of the classes failed to be booked. If there was any classes that failed it check how many times something has failed to book, at the minute if something can't be booked after three

passes then the user is told that it cannot be booked. If it hasn't failed three times already it amalgamates the classes it generated back into just modules.

Once we have a list of how many students couldn't be booked for each module that had any failures we send that information to a slightly modified version of `GeneratePracticalClasses` called `GeneratePracticalClassesFailed`. The only difference in the 'Failed' version of this method is that it doesn't retrieve the list of modules and student counts from the database. The program also reduces the threshold percentage by 10%. This results in smaller classes, which have more chance of fitting in to the time table than a large group.

Once these new classes are generated they are then sent to a `BookPracticalClassesFailed` method which tries a different method to book the now smaller classes. The 'Failed' version of `BookPracticalClasses` is identical to the normal version at the start, up until we randomise the order of the time slots. However where the first approach takes a module and tries to find a suitable time slot to book it in, only checking the lab that it was assigned to earlier in the process, this method does the opposite. Using a for loop it goes through the time slots one by one and for each time slot it works out how many free computers there are in each of the labs during the current time slot.

Now that it knows how many free computers are in each lab, it checks if the current class will fit in lab 1 in the current time slot, and if it doesn't it will check if it will fit in lab 2. If the current class can't fit in either lab then it moves on and checks if the next class will. The system will now run `GeneratePracticalClassesFailed` and `BookPracticalClassesFailed` until it has either booked all the classes that failed the first time or has hit the limit.

The `BookPracticalClasses` method was tested, to check if it reached a different solution each time, by repeatedly running the run and viewing the results in both phpMyAdmin and through the timetable/map forms of the system. The 'failed' versions of both this method and the `GeneratePracticalClasses` method were tested by first running the original methods multiple times so that the labs were full enough that some classes wouldn't be booked on the first pass.

Once this was done a breakpoint was inserted at the end of `GeneratePracticalClassesFailed` and the code was ran again. When the breakpoint was hit the contents of the dictionary was inspected to see if the system had correctly reassembled modules from their original class groupings and created new smaller groupings.

Then the system was allowed to continue and the amount of free computers was compared before and after to check if the ‘failed’ method had succeeded in booking these classes that had initially failed booking.

Finally once the class schedule has been generated it needs to be copied to the second week. This simply uses a method called `copyTheBookingsToNextWeek` which takes no parameters. It simply looks at the table of bookings and looks at the status of every computer during every timeslot. It then copies every instance where the computer is booked during that week, back to the second week. This means the same computers at the same times will be booked for the same classes every week. Finally, once the whole process is finished, a message box appears to inform the user that the operation has completed.

The `copyTheBookingsToNextWeek` method was very simple to test as before it is run there are zero bookings in time slots 51 to 100. After the method has been ran it was checked through phpMyAdmin that the bookings in time slots 51 to 100 mirrored the bookings that appeared in time slots 1 to 50.

System Evaluation and Results

This section will look at the system produced and evaluate whether this system has the potential to increase the utilisation of the lab space, as well as what further development and improvements would need to be made.

Good Aspects of the System & Future Improvements

This system has two separate parts to it that both contribute to making it easier for students to use the lab space in Queens University Belfast. The generation of the class timetable is the core of this system and it aims to generate classes in such a way that when possible a class will never take up the entirety of a lab. This means that in hopefully all cases there will always be some computers free for students who wish to quietly progress on their own work.

It will even book two separate classes into the same lab during the same time period if both classes are small enough, though this would likely require the approval of lecturers who would need to be okay with having two small classes share the large rooms before being implemented in the real world. The system may do this before it is running out of space in which to fit class bookings, or it may decide that two classes are so small that they can share the lab for a time slot and leave the lab entirely free for either another class or private use during a different time slot.

The other big aspect of this system is the timetable and map view of the lab rooms. Temporarily ignoring the booking function of this aspect, it would serve mainly as a useful tool to students which could increase the amount of students using the lab for their own private work.

To come in to the university costs most students both time and money, and when a student comes in and finds the labs so full they have to leave because they can't get a computer it lowers the motivation of that student to try it again. This has a knock on effect that this clear timetable view hopes to eliminate.

In the past when a student gets turned away when the lab is full, they might leave for the day, even though the lab may become half empty in two hours. Now we have a situation where the lab is half empty, and there are student who would want to be using the computers but because they couldn't when they first arrived, they aren't here now.

This situation shows that a clear view of how busy the labs are, or are expected to be in the future, would assist students who wish to use the lab space use it with less hassle and chance. A number of students talked to said that multiple times during their time at Queens University Belfast they would have gone in and worked in the labs instead of at home, but didn't because if they had gone in and been turned away they would have pointlessly wasted time.

Even just being able to see when some computers might become available would be helpful to the effort to trying to ensure that as little of the lab space is wasted at any given time. As an extension of this idea the system also features the beginnings of a booking system for computers. Clicking on a computer in the map view books that computer for private use.

This is a step up from a student being able to see a lab might have some free computers and lets a student know that if they come in at a certain time they will definitely get a computer to use in the labs.

Overall I feel both the main aspects of this system perform their function well. The generation successfully generates a schedule of classes, which certain nuances to try and improve lab usage, while the user interface demonstrates a useful property that would benefit students wishing to use the lab space.

Conclusion

Overall I feel this project was partially successful, with the system developed having met all of the mandatory and desirable requirements laid out for it and only failing to meet one luxury requirement.

When looking at how the classes generation performs, if it was paired with some real world usage statistics about the labs, the system could be further developed by adding a weighting to all of the time slots in the system. Time slots that generally have less students using a computer for private use would be used to book the larger classes, while time slots that generally have a lot of students doing their own work would be left fully open where possible and have the smallest classes booked in them if not.

For example when progressing the weeks of the system it currently works out what percentage of computers in each time slot were booked for each type of use and saves these percentages into a .txt file. However, if further developed it could be saved back into the database using an equation to take the weighting currently stored and modifying it using the new statistic.

Part of the reason this idea was not developed further in this project was that getting real usage stats was impossible and generating a month or so of fake but realistic usage would have been too long of an endeavour.

The lack of this feature is the biggest weakness with this project, however if it had been implemented using unrealistic test data it would have weakened the rest of the generation methods as a whole and it was decided this wasn't worth it.

The timetable and booking aspect of the system should modify the habits of students when it comes to seeking space in the labs, increasing the amount of time overall that computers are used. However this cannot actually be tested without further development to combine this aspect of the system with the computer network currently in place at Queens University Belfast. If these functions could be developed as part of the current systems the idea becomes a lot more powerful; students booking computers via their student numbers, tracking how often each student actually uses a computer they booked, notifications on the screen of the computers, prevention of abuse of the system by having a limit on how single students can book computers.

As an example of one of these if a student was using a computer that someone else had booked from 2PM it could be used to alert them through a pop up message that another student has booked this computer and will be arriving in the next ten minutes, so the current user should be prepared to move if they do show up.

The primary change that would be made, if this project was to be done again, would be to create a more detailed work plan/Gantt chart and stick to it more rigidly. Work on this project was rather stop/start for a period of time at the start of the year, which the project suffered for. Help would also have been sought for the MySQL problem much earlier than it was, as many attempts to solve it without external help ended up only wasting time that would have been better spent on further developments.

References

Software Code Repository: <https://gitlab.eeecs.qub.ac.uk/40078844/CSC3002.git>

Appendix A: User Guide

This system must be ran on a Queens University Belfast network computer. It uses a MySQL database hosted on a QUB webspace which will only allow connections from within the network. As such the system will not function on any computer not connected to Queens University Belfast.

Adding/Deleting/Finding Classes for a Module

In order to add a new module, delete an old one or find all the classes of any given one you should first navigate to the ModuleTimetableGeneration form using the following button.

The image shows two screenshots of a web application. The top screenshot is the 'Start Menu' with three buttons: 'Lab 1 Timetable', 'Lab 2 Timetable', and 'Lab Classes Generation'. The 'Lab Classes Generation' button is highlighted with a red rounded rectangle. The bottom screenshot is the 'ModuleTimetableGeneration' form. It contains several sections: 'Run The Code', 'Reset The Booking Table', 'Clear the Booking Table of classes and generate new classes', and 'Progress the week'. There are three main functional areas highlighted with colored rounded rectangles: a green one for 'Find A Modules Classes' (containing a text input and a 'Find Classes' button), a red one for 'Delete A Module' (containing a text input and a 'Delete Module' button), and a blue one for 'Add New Module' (containing text inputs for 'Name of module to add', 'Course code', 'Number of students', and 'Course Level', along with an 'Add Module' button). A table of existing modules is visible in the background. Annotations with text labels are placed over the highlighted areas: 'Find Classes' in green, 'Delete Module' in red, and 'Add Module' in blue. A 'Back to Menu' button is also present at the bottom right.

idModules	Modules	Number of Students	Level
CSC1009	Introduction To Software Engineering & Project Management	399	Level 1
CSC1017	Reasoning for Problem Solving	316	Level 1
CSC1018	Foundations of Computing Systems	341	Level 1
CSC1020	Programming	244	Level 1
CSC1021	Programming	203	Level 1
CSC1022	Programming Practice	339	Level 2
CSC1023	Design	132	Level 2
CSC2039	Architecture and Networks	227	Level 2
CSC2040	Data Structures, Algorithms & Programming Languages	230	Level 2
CSC2041	Information Management	125	Level 2
CSC2042	Information Management	214	Level 2
CSC2043	Information Management - Processes and Practice	135	Level 2
CSC2044	Information Management and Maintenance	81	Level 2
CSC3021	Concurrent Programming	153	Level 3
CSC3030	Intelligent Information Systems	104	Level 3
CSC3031	Software Design Principles & Patterns	54	Level 3

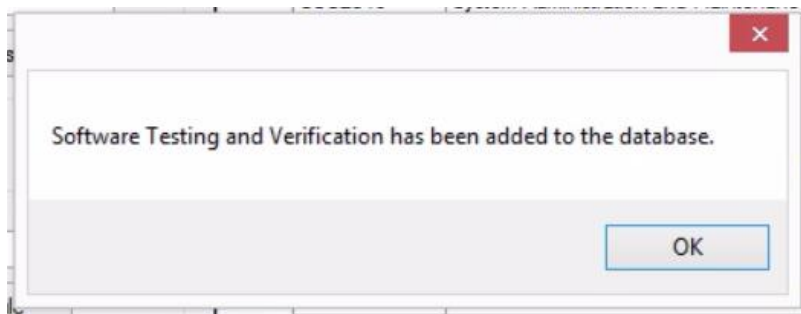
Once on this should you want to add a module you should go to the add modules section and enter the name, course code, student count and level of the module like so. Once you press the button you will see the following message.



Add New Module

Name of module to add: Course code:

Number of students: Course Level:



If you want to delete a module then you should find the module that you wish to delete in the data view, to check that it hasn't already been deleted, and then either copy the module name into the delete area field or type it in. Once you have done this press the delete module button and you will see the following message.



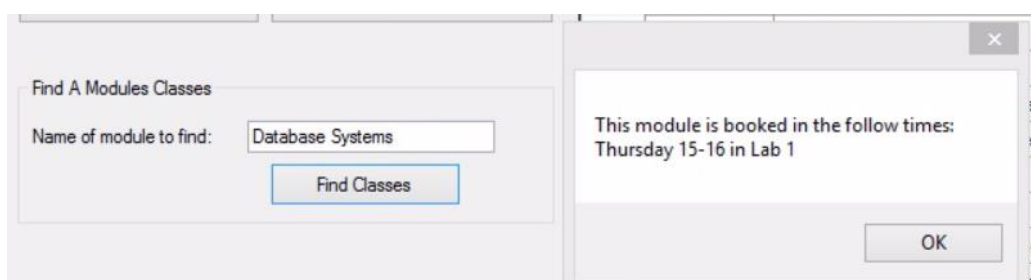
Delete A Module

Name of module to delete:

Software Testing and Verification has been deleted.

If you wish to find the times and locations of all the classes that have been booked for a given module simply locate the module in the data view and copy its title into this text box.

You can also type the modules name if you wish, but you must type it exactly. Once you have done this press the find classes button and you will see a message similar to this one telling you when this module has classes and where those classes are booked.



Find A Modules Classes

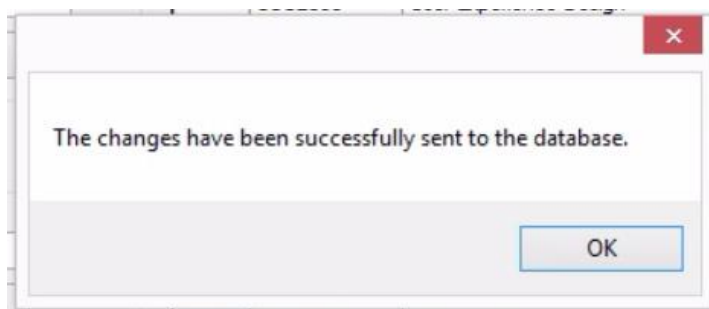
Name of module to find:

This module is booked in the follow times:
Thursday 15-16 in Lab 1

Progressing the weeks/clearing the class schedule/clearing all bookings/generating a new schedule

This is designed to be done after the timetable ends on a Friday evening. Just like the previous section you should open the ModuleTimetableGeneration form using the button labelled as such. Once on this for, if you wish to progress the weeks all you need to do is press the button labelled “progress the week” located in the bottom right of the circled area. Shortly after you will be shown a message informing you that the weeks have been progressed.

idModules	Modules	Number of Students	Level
CSC1009	Introduction To Software Engineering & Project Management	399	Level 1
CSC1017	Reasoning for Problem Solving	316	Level 1
CSC1018	Foundations of Computing Systems	341	Level 1
CSC1020	Programming	244	Level 1
CSC1021	Fundamentals of Programming	203	Level 1
CSC2011	Professional Computing Practice	339	Level 2
CSC2038	User Experience Design	132	Level 2
CSC2039	Architecture and Networks	227	Level 2
CSC2040	Data Structures, Algorithms & Programming Languages	230	Level 2
CSC2041	Information Management	125	Level 2
CSC2042	Information Modelling	214	Level 2
CSC2043	Software Development - Processes and Practice	135	Level 2
CSC2046	System Administration and Maintenance	81	Level 2
CSC3021	Concurrent Programming	153	Level 3
CSC3030	Intelligent Information Systems	104	Level 3
CSC3031	Software Design Principles & Patterns	54	Level 3



To clear the class schedule but leave any private bookings alone you need to press the button labelled “clear the booking table of classes and generate new classes” located in the bottom left of the circled area. This will clear the database of bookings for classes only and then

generate a new schedule. Please be patient as this can take over a minute in some circumstances.

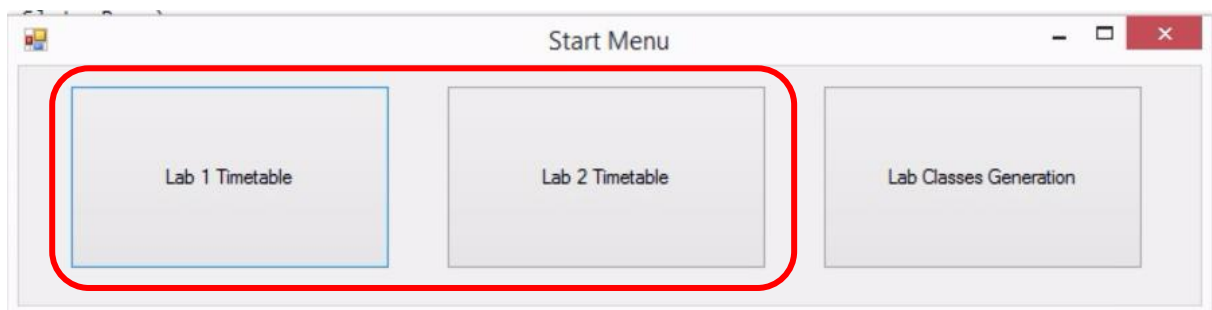
If you wish to clear the bookings table entirely and reset it to its default state all you need to do is press the “reset the booking table” button located in the top right of the circled area.

And to generate a new class schedule without getting rid of the one currently generated, simply press the “run the code” button located in the top left of the circled area.

If you have nine modules scheduled and want to add a tenth into the schedule you will need to delete the ones that have already been scheduled from the database and add the one you wish to book. The system will book every module that is sitting in the modules table at the time this button is pressed.

Booking a computer

In order to book a computer for private use you will need to decide which lab you want try and book a computer in, once you have made your decision select the appropriate of these two options.



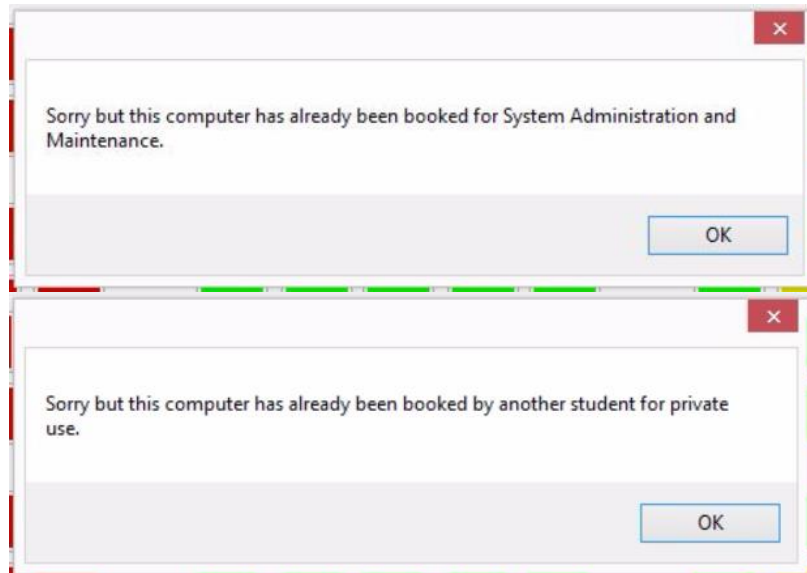
You will now be faced with a screen like this which shows you the status of the labs throughout the week.

Monday	Tuesday	Wednesday	Thursday	Friday
8:00AM - 9:00AM	8:00AM - 9:00AM	8:00AM - 9:00AM	8:00AM - 9:00AM	8:00AM - 9:00AM
9:00AM - 10:00AM	9:00AM - 10:00AM	9:00AM - 10:00AM	9:00AM - 10:00AM	9:00AM - 10:00AM
10:00AM - 11:00AM	10:00AM - 11:00AM	10:00AM - 11:00AM	10:00AM - 11:00AM	10:00AM - 11:00AM
11:00AM - 12:00PM	11:00AM - 12:00PM	11:00AM - 12:00PM	11:00AM - 12:00PM	11:00AM - 12:00PM
12:00PM - 1:00PM	12:00PM - 1:00PM	12:00PM - 1:00PM	12:00PM - 1:00PM	12:00PM - 1:00PM
1:00PM - 2:00PM	1:00PM - 2:00PM	1:00PM - 2:00PM	1:00PM - 2:00PM	1:00PM - 2:00PM
2:00PM - 3:00PM	2:00PM - 3:00PM	2:00PM - 3:00PM	2:00PM - 3:00PM	2:00PM - 3:00PM
3:00PM - 4:00PM	3:00PM - 4:00PM	3:00PM - 4:00PM	3:00PM - 4:00PM	3:00PM - 4:00PM
4:00PM - 5:00PM	4:00PM - 5:00PM	4:00PM - 5:00PM	4:00PM - 5:00PM	4:00PM - 5:00PM
5:00PM - 6:00PM	5:00PM - 6:00PM	5:00PM - 6:00PM	5:00PM - 6:00PM	5:00PM - 6:00PM

You must now select what time you want to book a computer for, in the event that there are no computers free at the time of the day you want you should check the other lab. Once you have found the time you wish to book a computer for, click the option and you will see a screen like this.

Row 1	Row 2	Row 3	Row 4	Row 5	Row 6
Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red
Red	Red	Red	Red	Red	Red

Each of these buttons represents a computer in your selected lab. You want to find a computer that is coloured green as that means it's open for you to book. If you try to book a computer of another colour you will be told that it has already been booked.



Once you have booked your computer you will see it's status and colour have been changed, the computer is now booked for you to arrive and use.

Appendix B: Minutes of Meetings

QUEEN'S UNIVERSITY OF BELFAST COMPUTER SCIENCE

Minute of Project Supervision Meeting

Student Name:	David Savage		
Project Module Code:	CSC3002		
Project Supervisor:	Barry McCollum		
Meeting Number:	1	Date of Meeting:	Monday 17/10/16

Progress since last meeting, and decisions arrived at during meeting:

- Discussed frequency of meetings and decided we would only hold meetings when it was felt they were needed
- Discussed talking with the reception hatch staff of CSB as they see the issues with the labs every day.
- Discussed the overall aims of the project

Action Points:

- Meet with Brian from the reception hatch to find out his view on the computer lab

Date of next meeting:

N/A

Agreed minute should be signed by the student and initialled by the supervisor.

Student's Signature: _____ Date: _____

Supervisor's Initials: _____ Date: _____

Supervisor's Comments:

--

**QUEEN'S UNIVERSITY OF BELFAST
COMPUTER SCIENCE**

Minute of Project Supervision Meeting

Student Name:	David Savage		
Project Module Code:	CSC3002		
Project Supervisor:	Barry McCollum		
Meeting Number:	2	Date of Meeting:	Monday 31/10/16

Progress since last meeting, and decisions arrived at during meeting:

- Met with Brian from the reception hatch and got information on the usage of the computer labs, as well as information about how many students are enrolled on modules that need to book space in a practical lab
- Discussed how some aspects of the problem cannot be solved by a computer system and talked about requirements for the system that would help it increase the overall utilisation of the lab space.

Action Points:

- Start the documentation by creating the problem description and system requirements
- Start constructing the basic blocks of the system, first the MySQL database that will store all of the information on the modules, computers and bookings. Followed by the C# application that will serve as the interface and processing side of the system.

Date of next meeting:

N/A

Agreed minute should be signed by the student and initialled by the supervisor.

Student's Signature: _____ **Date:** _____

Supervisor's Initials: _____ **Date:** _____

Supervisor's Comments:

--

QUEEN'S UNIVERSITY OF BELFAST
COMPUTER SCIENCE

Minute of Project Supervision Meeting

Student Name:	David Savage		
Project Module Code:	CSC3002		
Project Supervisor:	Barry McCollum		
Meeting Number:	3	Date of Meeting:	Monday 21/11/16

Progress since last meeting, and decisions arrived at during meeting:

- Had begun creating the database schema and was almost ready to implement it and actually construct the database.
- Discussed how best to handle the division of time slots. How long would each slot be, from what time in the morning to what time in the evening, would it only cover two weeks or more.

Action Points:

- Create the database and decide which options for the time slot I think would be best
- Further develop the C# application for the upcoming interim demo

Date of next meeting:

N/A

Agreed minute should be signed by the student and initialled by the supervisor.

Student's Signature: _____ **Date:** _____

Supervisor's Initials: _____ **Date:** _____

Supervisor's Comments:

QUEEN'S UNIVERSITY OF BELFAST
COMPUTER SCIENCE

Minute of Project Supervision Meeting

Student Name:	David Savage		
Project Module Code:	CSC3002		
Project Supervisor:	Barry McCollum		
Meeting Number:	4	Date of Meeting:	Monday 13/03/17

Progress since last meeting, and decisions arrived at during meeting:

- C# application was almost fully fleshed out, had recently resolved a problem with the MySQL side of the system after having been put in touch with Sean McKeever, a Queens University Belfast IT Manager.
- Discussed where the system still had to go and the MySQL problem.

Action Points:

- Continue development on the C# application, furthering the heuristic methods used to generate the class schedule and the feedback the system gives users on the state of the labs.

Date of next meeting:

N/A

Agreed minute should be signed by the student and initialled by the supervisor.

Student's Signature: _____ Date: _____

Supervisor's Initials: _____ Date: _____

Supervisor's Comments:

--

QUEEN'S UNIVERSITY OF BELFAST
COMPUTER SCIENCE

Minute of Project Supervision Meeting

Student Name:	David Savage		
Project Module Code:	CSC3002		
Project Supervisor:	Barry McCollum		
Meeting Number:	5	Date of Meeting:	Friday 28/04/17

Progress since last meeting, and decisions arrived at during meeting:

- Met to discuss heuristics and how the methods implemented in the system met the criteria of being a heuristic solution to scheduling classes and trying to utilise lab space better.

Action Points:

- Refine the system and add some finishing touches, focus on finishing the accompanying report.

Date of next meeting:

N/A

Agreed minute should be signed by the student and initialled by the supervisor.

Student's Signature: _____ Date: _____

Supervisor's Initials: _____ Date: _____

Supervisor's Comments:

--