Computer Room Booking System

NATHAN KING

Code can be found at https://12nathanking.github.io/

Table of Contents

Analysis

Identifying the Problem

At Meridian School in Hertfordshire, teachers require many resources that help students throughout their education. Computer rooms are a key asset for learning, however many teachers are looking to put them to good use. This leads to complicated situations where the computer rooms are double booked or some are being used when not necessary. I am looking to create a program that is able to book computer rooms, with a selected date and lesson time. To complete this task, I require a customer who understands the benefits of the system and is able to suggest the best approach towards my goal. I approached Mr Rutter, the IT Manager at Meridian School with my idea and organised a meeting where we could discuss potential objectives.

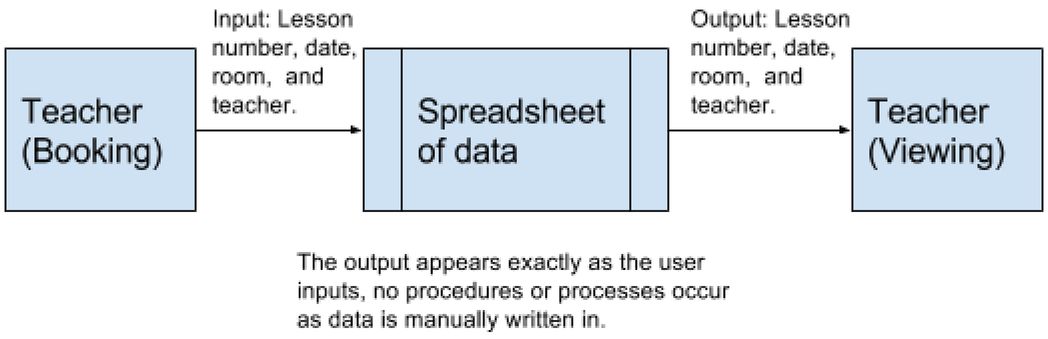
Primary Research

In order to research this problem I have had a meeting with my customer, Mr Rutter (evidence in the appendix - part **A**) where we discussed ideas relating to what he wanted as an outcome. In addition, he described the layout of what the design could possibly look like. From this information I have an idea of the basic concepts that I will need to use.
Another form of research I carried out was viewing the current system that the school uses, which allowed me to analyse potential issues of this type of program and also gain new ideas for my own solution. Throughout the rest of my analysis I will reference the current system and integrate Mr Rutter's ideas to help finalise my key objectives.

Description of the Current System

Mr Rutter suggests that currently, the teachers use a spreadsheet that holds a timetable of the rooms and whether they are booked or not. The main advantage of this is that it’s free to use for the school. Therefore, only one person has to manage and organise the system whilst teachers select the rooms they want on a given date. The disadvantages are that there is a limit on how many weeks in advance a booking can take place. This means that new spreadsheets will have to be manually reset by the manager at least school term, in order to prevent wasting storage. Furthermore, there are often cases of booking rooms that can take up to 30 students by classes of less than 10. This is inefficient as there are assets being wasted by the school (such as the 20 unused computers). Mr Rutter hopes that my program could help make rooms be used to their full potential by recommending the most suitable room for the class size given.
The purpose of my system is to fundamentally make the process of sharing computer rooms within school much more manageable and therefore replace the use of spreadsheets.

Current system data flow diagram:


Input: Lesson number, date, room, and teacher.
Output: Lesson number, date, room, and teacher.
Teacher (Booking)
Spreadsheet of data
Teacher (Viewing)
The output appears exactly as the user inputs, no procedures or processes occur as data is manually written in.

After initially drafting these concepts and what could be improved, I met with the manager of the current system, Mr Hamilton. He informed me that there was an additional complexity to his system where he imports an xml file that includes information of pre booked rooms. These rooms are identified as unbookable as they are used by teachers as a required classroom (such as an IT lesson). Mr Hamilton provided information of how his import function works in the current system and the resources that are required. One of the resources provided was the xml code (this can be found in the appendix - part **B)**, which will have to be read through in order to pull out the correct bookings. Therefore, as an additional requirement to my system, I will have to use this xml code myself and allow the admin to import it. This ensures that the system is accurate and is relevant to our school, which is one of Mr Rutter’s requirements that was implied when discussing the final product. The imported code shall be inserted into the database in the booking table with relevant dates and lesson numbers.

Xml Research

One of my aims will be to read through an xml file and identify the correct data, inserting it into the database. There are different ways to approaching this, so I will research the methods that could be used to implement the desired algorithm. In order to research xml

correctly I must identify the development code I will use in this project, php (this will be discussed in my design phase). The main source I used for research was W3schools (link in the Bibliography).
The Xml language is a way to structure data for sharing across websites. In order to read an xml file, I will require an xml parser. However, there are a number of different methods to use for this and it depends on the structure of the file.
Methods:
→ Tree Based:
These parsers store the document in memory and transforms the XML into a tree structure. It analyses the whole document, and provides access to the tree elements.
This include:

- SimpleXML
- XML DOM

→ Event Based:
These parsers do not store the document in memory, instead, they read and allow you to interact one node at a time. Once you move onto the next node, the old one is thrown away, which means data about the previous node will not be kept.
This includes:

- XML Expat

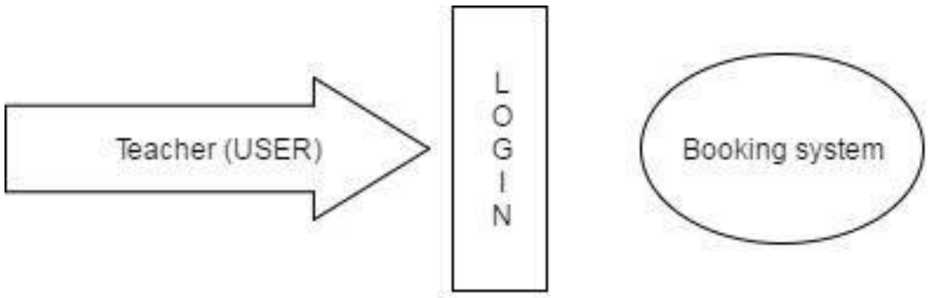
I will be using a tree based parser for my system because within the file (shown in the appendix - part **B**) each node contains many parts to it. This means that data about each part may need to be stored in order to piece together a booking, as event based parsers throw away old nodes, this would not be a good method to use.
The W3Schools site explains that 'SimpleXML provides an easy way of getting an element's name, attributes and textual content if you know the XML document's structure or layout'. SIMS, the school database, will provide the XML files with a consistent structure, which helps identify that Simple XML will be the method I will use to read the files and obtain the relevant data. Although this is a design choice, the research was required to identify and analyse key objectives.

Users of the System

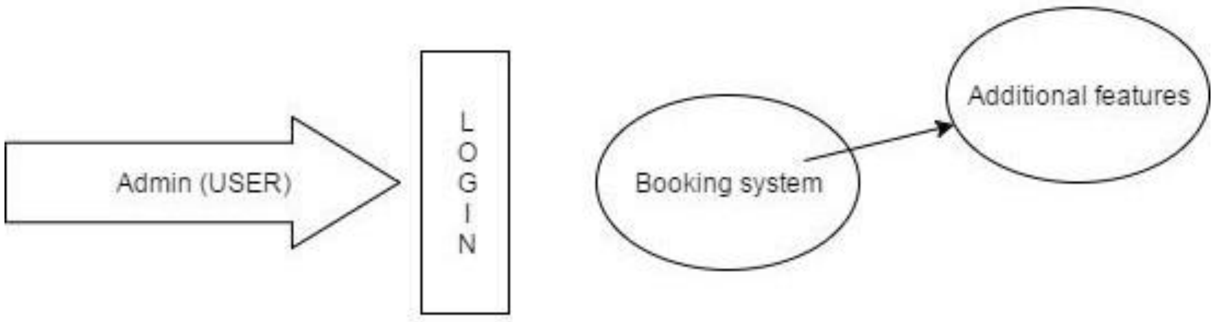
Although this program will benefit Mr Rutter and help him manage the IT in Meridian School, he will not be the main user. Teachers will be using the system to book rooms for their lessons. With a target audience of teachers, I am able to include sufficient detail within the system so that it can be both informative and have optimised utility. However, Mr Rutter informed me that teachers do not want to spend much time having to create bookings as they are very likely to be busy. Therefore the layout of the system must make it quick and easy for each task to be completed. Cascading style sheets will be useful to simplify the layout of the system and make it more organised. This is to allow all teachers within the school to use the system successfully and find/locate each function promptly.
Mr Rutter has stated that he would like to enter the system and have more tasks to do than the normal user, such as the importing of the file. Therefore, the function will provide different functions to different users. This means that the system must be able to identify

which users are admins, which will require me to keep a piece of identification within the database. In addition to this information, users wanting to enter the system will be required to login by entering their details correctly. The login information will be stored within the database.
I have created a diagram that describes the login system and how it will affect different users.

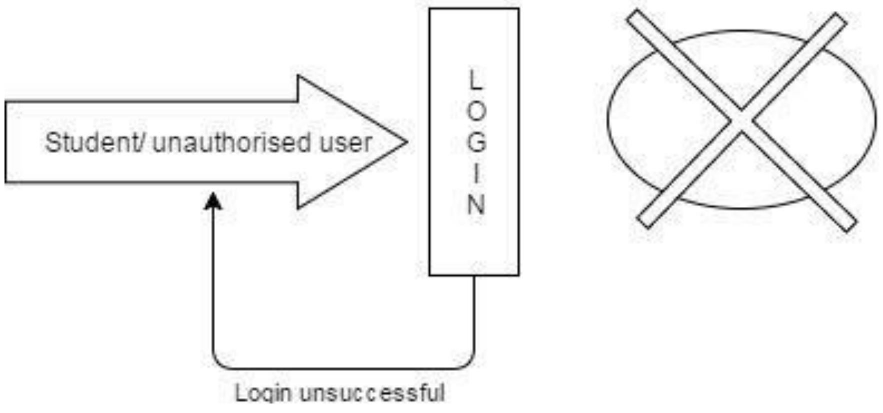
For a normal user (teacher), they will be required to enter their details correctly so that it matches the information within the database. This will enable them to enter the system and enable them to book rooms.



For an admin user, they will also have to successfully pass through the login system. This will enable them to book rooms like a teacher, but will also have access to additional features. This will include signing up teachers, deleting large amounts of bookings at once, and importing xml files.



For unauthorised users, if they have not been signed up they will be unable to pass the login page and access the system and its features. The reason for this is to prevent unwanted bookings that make the system harder to manage.



User Needs

Mr Rutter requires an efficient program that can determines the most appropriate room for the teacher. Therefore, the user may select what they need for the lesson, and the program will adjust to those inputs (order rooms in terms of the most appropriate).
In addition, my customer has focused on a program that helps the user's process of booking a room. This means that the user interface is required to be simplistic and straightforward. The process of booking must be quick so that it improves on the current system and time isn't wasted when using it. Some teachers may always be busy during school times which means that accessing the system from home is important. This implies a web based program is required. Another important requirement is the ability delete bookings. However, there are potential issues to allowing any user to delete any reservation, as the system's purpose is to protect teacher's booking. Therefore, only admins will be allowed to delete any user's bookings, in addition to the creator of the booking having the ability to cancel.
Furthermore, all users will want confirmation that the system is secure. In addition to the login system discussed, I will be required to hash or encrypt data such as passwords before they are stored within the database, so that they cannot be traced backwards by unauthorised users.
Another important factor is the functions that the system provides. For teachers, they will want to be able to book rooms, cancel bookings and potentially find the best room for them. However for an admin, they will want to be able to perform tasks that teachers can do in addition to importing files, deleting large numbers of bookings and creating accounts for new users. Only admins can create new accounts so that it limits and prevents the unwanted members of the public being able to enter the system. Deleting larger numbers of bookings will be used as a tool to clear the database (to prevent performance slowdowns), which would mean that only an admin will be able to access this as it could potentially wipe the system completely.

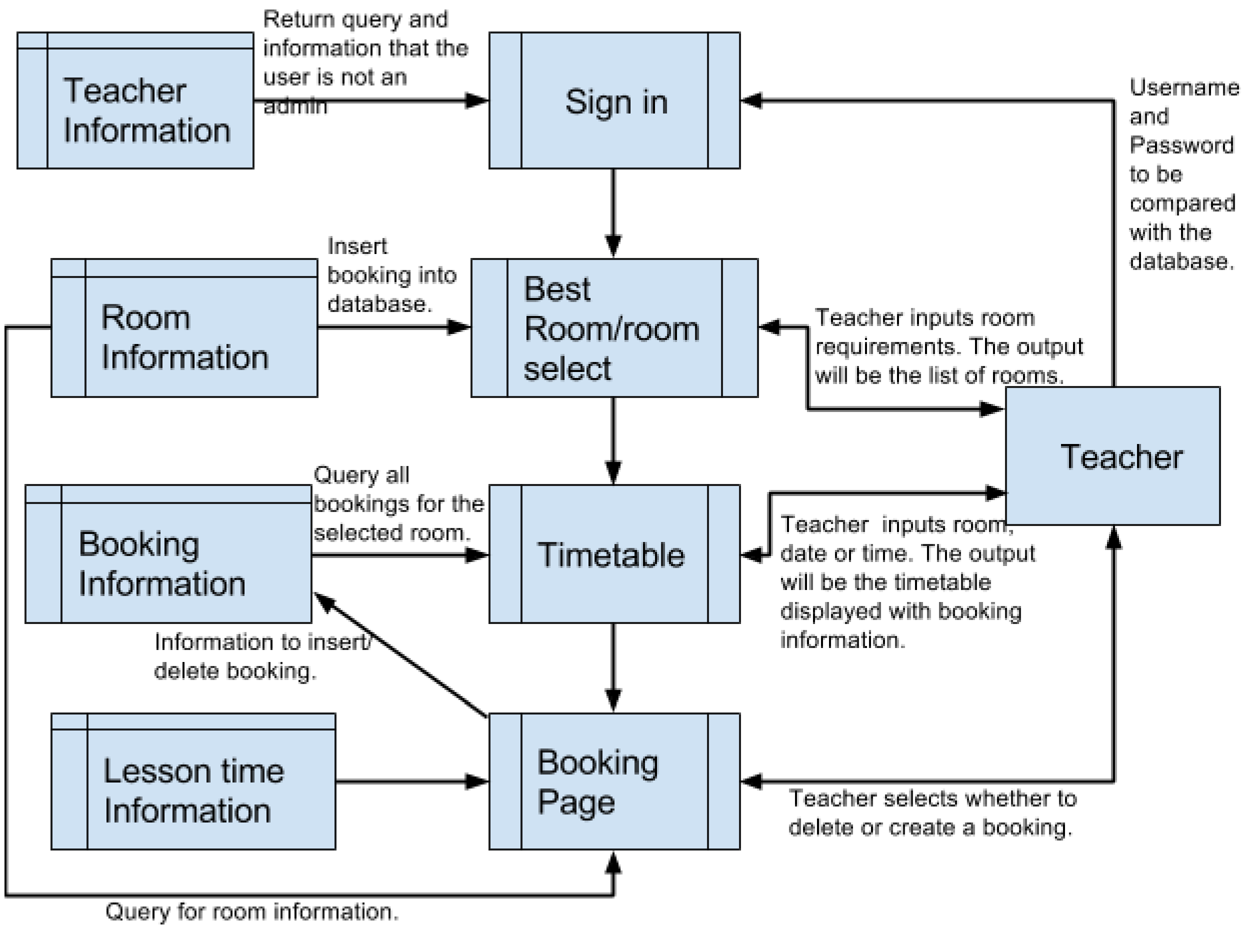
Identifying the Solution

A new computer room booking system associated with web browsing will be effective as it lets teachers gain access at home or at school. The program will need to allow teachers to book rooms for the lesson and date they require, which implies that filters or drop down lists may be necessary. A crowded user interface with multiple tables will make it difficult to work with as the display will become over-complex for use in a short period of time.
The system will be created with a relational database and will update as users input information about the room they require. Each computer room will hold specific information to it that will be used in order to generate the best room for the teacher. The database will therefore be used to store all the information about the computer rooms as well as time and date (that the room is booked for). The system will require many algorithms to perform different tasks that have been discussed through the analysis. Some tasks can only be accessed by teachers but all tasks can be accessed by admins.
In order for users to get around the system, it is likely that it will require many transition buttons as well as a timetable that will list the booking information for specific days and

lessons. The database will therefore need to be referenced many times within the technical solution so that each booking can be selected so that it can be viewed or deleted by a user.

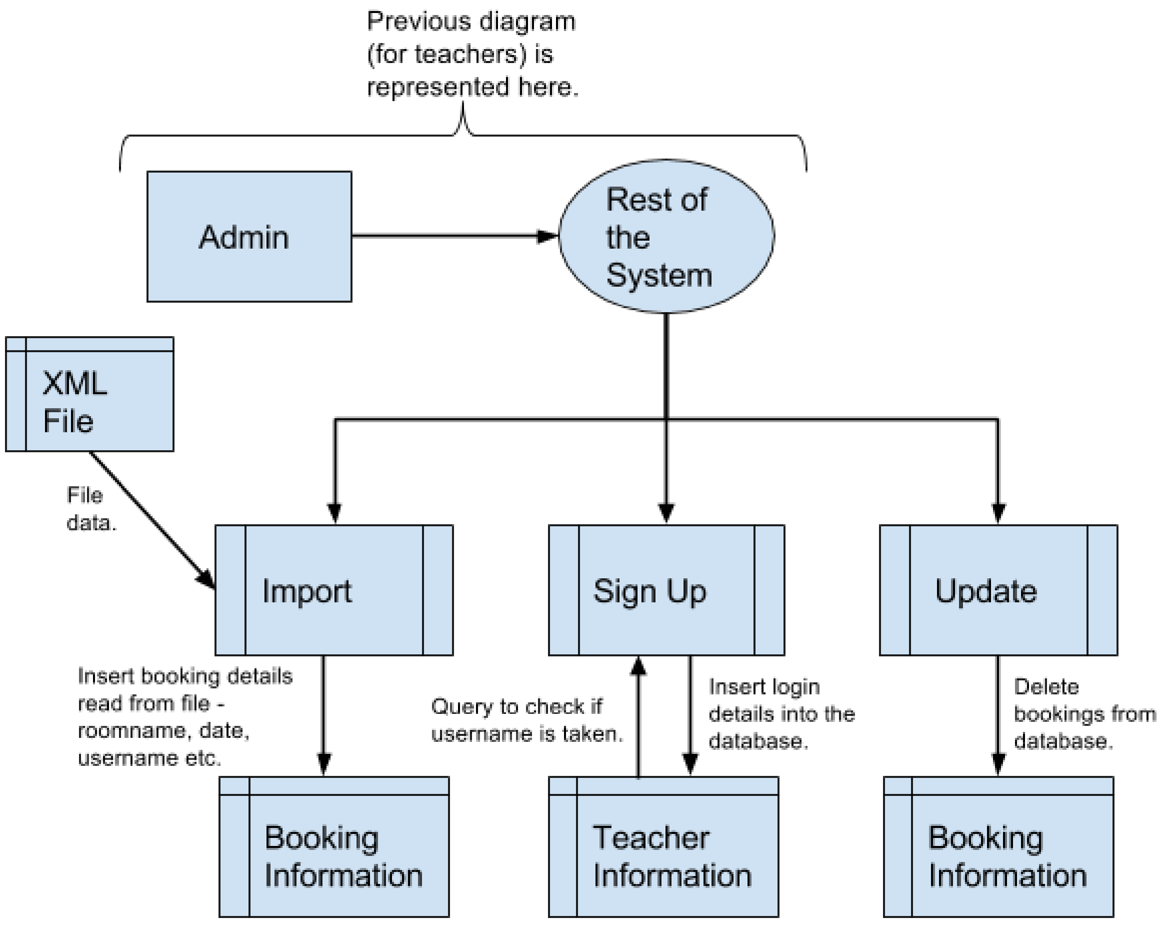
I looked at the potential flow of data to get a better understanding of the main components of the system. It demonstrates how the user will interact with the program and a potential order that events will have to occur. As teachers will not have the same access to the system as admins, I created two separate diagrams to display how the additional features will be implemented.

New system data flow diagram (for a teacher):


Teacher Information
Return query and information that the user is not an admin
Sign in
Username and Password to be compared with the database.
Room Information
Insert booking into database.
Best Room/room select
Teacher inputs room requirements. The output will be the list of rooms.
Teacher
Booking Information
Query all bookings for the selected room.
Timetable
Teacher inputs room, date or time. The output will be the timetable displayed with booking information.
Information to insert/ delete booking.
Lesson time Information
Booking Page
Teacher selects whether to delete or create a booking.
Query for room information.

The data flow diagram identifies that the login process must be completed in order for the correct data to be used in the the other processes. Therefore, the teacher must successfully login before being able to access the rest of the system. The diagram indicates the way in which my database will need to be accessed and provides insight on what tables I may need when I enter the design phase. I will use this plan as I create each function individually before coding the system.

New system data flow diagram (for an admin):


Previous diagram (for teachers) is represented here.
Admin
Rest of the System
XML File
File data.
Import
Sign Up
Update
Insert booking details read from file - roomname, date, username etc.
Query to check if username is taken.
Insert login details into the database.
Delete bookings from database.
Booking Information
Teacher Information
Booking Information

The ‘Rest of the System’ includes all the previous elements within the teacher diagram as admin users will be able to access those features in addition to the three shown in the admin diagram. In addition, the Xml file will be required and the data must be imported correctly so that the parser can be applied to pull the bookings out.

Both diagrams show numerous interactions with external storage in order to retrieve or change data. This confirms that the system will rely heavily on the database and roughly every function will require using it.

Limitations

1. Ideally the system should only be allowed to deal with bookings no greater than 3-4 months in advance so that the database does not get too large. However I decided to leave it open as Mr Rutter did not state that this was at all needed. It is unknown how many bookings can be made before hardware limitations occur, but there is only a specific amount of memory available. The ability for the admin to delete large amounts of bookings should prevent databases having too much data.
2. The system will only work with Meridian School's current timetable. This is because room names, lesson times and potentially the number of lessons will be unique to this school. If Meridian school has large timetable changes, It may prevent the system from performing some functions, so data will manually have to be edited inside the database and code so that they match.
3. The username that the admin creates when signing up a teacher must match the details that are found on the school SIMS database. If the admin creates a user that does not have the same corresponding details, there may be some functionalities that cannot be accessed by this user, such as being able to delete bookings made by the imported xml files (as these bookings are created with the teacher's SIMS username). To get round this, I will implement regular expressions that will help influence the admin to enter the correct details and prevent the input of usernames that do not have the same layout as SIMS.
4. The Meridian school system works on a two week timetable cycle. Therefore, when uploading an xml file, it is possible to book week one information on a week two. This is because there will not be an implementation of a 'numbered week' system within the code as it is an unnecessary complication. However, as long as the xml file starts on the same week as the admin inputs, then this will not be an issue. For example, the file begin with data that is found on a week two, which means that the admin will have to select a starting date that is on a week two. I am not implementing the two week timetable cycle as Mr Rutter stated that it was not an important part of the system, and can still work without it.
5. The algorithm which inserts bookings into the database using the xml file will only work for the layout specified in the example given by Mr Hamilton (the Meridian School timetable). For example if the data included extra rows it will be unlikely to produce a successful outcome. It will be up to the admin to make sure that the xml file is correct and taken from SIMS. I will be able to validate the file so that only the xml type can be imported, but it will be too difficult to check every element within the file.

Final Objectives/Requirements of the Program

The Program Will:

1. Be a Web Based Program.
    - 1.1. Access via web browser.
    - 1.2. Only teachers and admins can use the program.
    - 1.3. Cascading style sheets will be used to organise the layout clearly.
2. Use logins to access the system.
    - 2.1. Login will be built up of the teacher's name and a given password.
    - 2.2. Login details will be stored in a database.
    - 2.3. The passwords will be encoded (MD5SUM).
    - 2.4. The admin will be required to sign up teachers through the program and they will have to pass on the details.
    - 2.5. The username information will have to match the data from the school database (SIMS).
3. Use a relational database that links directly the program.
    - 3.1. Rooms Table
        - 3.1.1. It will store the resources and the number of computers that each computer room contains.
        - 3.1.2. The there will be a fixed number of computer rooms.
    - 3.2. Bookings Table
        - 3.2.1. It will store whether or not a room is booked.
        - 3.2.2. It will store the date and time of when the room is booked for.
        - 3.2.3. It will store who has booked the room.
    - 3.3. Teacher Table
        - 3.3.1. It will store the login details of each teacher.
        - 3.3.2. It will contain data to determine if they are an admin or not.
    - 3.4. Lesson times Table
        - 3.4.1. It will store the times of each lesson time.
        - 3.4.2. Friday will have different lesson times to the rest of the week.
4. Allow teachers to find the most suitable room to book.
    - 4.1. A list of the rooms from most suitable to least suitable will be displayed.
    - 4.2. Filters will be used such as check boxes to determine requirements.
    - 4.3. The algorithm to find the best room will generate scores, which will be displayed to give an idea of the difference between the rooms.
    - 4.4. Student count will be input as an integer.
    - 4.5. Filters will include: Printer(s), whiteboard, interactive whiteboard and scanner.
    - 4.6. The filters and student count value will be compared to the room data within the database to determine the score.
5. Have a timetable of each computer room in the school, for the user to see.
    - 5.1. The timetable will appear on the current week when first entering the page.
    - 5.2. Each cell will include the user who booked the room.
    - 5.3. If a room is not booked it will contain 'empty'.
    - 5.4. The week the timetable is on can be changed with 'skip' buttons that immediately skip to the next or previous week.

- 5.5. The user can also input a date they would like to view. The timetable will still be from monday to friday, but the input date will be contained within it.
- 5.6. When entering a date, they can also use a calendar to click the day they want.
- 5.7. The timetable being viewed can have its room name changed with a drop down menu. For example, the timetable will be displaying bookings for ‘A6’, and this can be changed to ‘B17’.

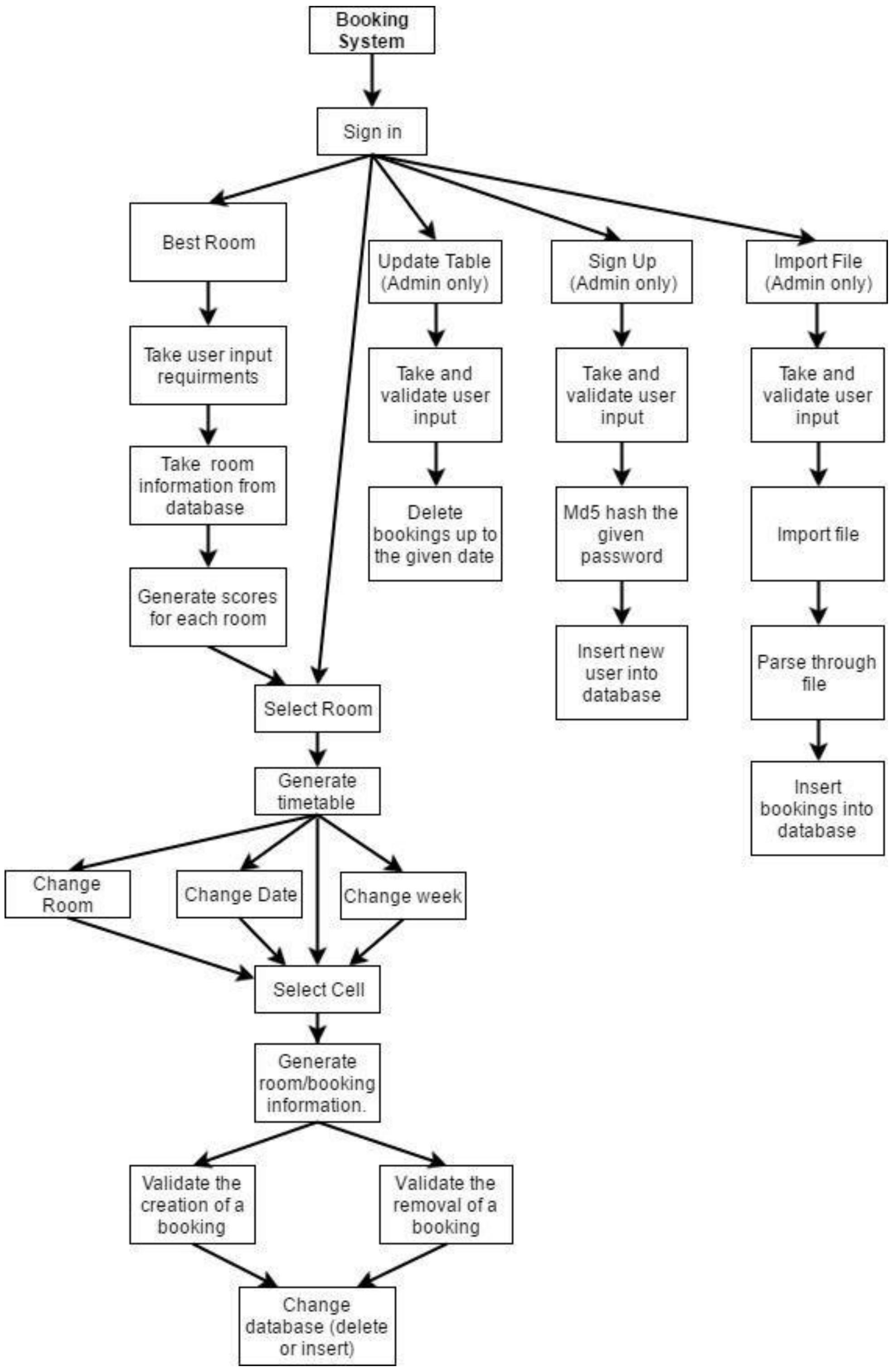
6. Allow for all users to see a booking and it’s information.
   - 6.1. The user will click the cell within the timetable using hyperlinks (they can select it even if it’s empty) and will be taken to a page that displays data about the room.
   - 6.2. One half of the page will display data about the room itself, such as the resources it includes and the number of computers.
   - 6.3. The other half of the page will display data about its booking, such as the time, date and lesson number. It will also tell the user if it is booked or not.
7. Allow for the booking of a computer room.
   - 7.1. Any user can book an empty room by clicking ‘book’, possibly on the ‘viewing’ page (after clicking the cell in the timetable).
   - 7.2. If the room is already booked, and a user attempts to book the room again (with the same date and time), they will be told that the room cannot be booked again and no booking will be made.
   - 7.3. If the room is not booked, and a user selects ‘book’, they will be brought back to the timetable (with the same room name and week beginning). In addition the database will be updated to contain this booking.
   - 7.4. After a successful booking, the cell will go from containing ‘empty’ to including the username of the user who made the booking.
8. Allow for booked rooms to be cancelled.
   - 8.1. Cancellations can only be made by the user who booked the room or by an admin. An admin can cancel any booking.
   - 8.2. Cancellations will be chosen by selecting the booking (on the timetable) and clicking the ‘remove’ button.
   - 8.3. The database will be updated to remove the booking information completely, and its booking status will be ‘unbooked’.
9. Allow for a teacher to be signed up to the system.
   - 9.1. Only the admin will be able to perform this task.
   - 9.2. Regular expression will be used to only allow for a username in the format described in the analysis.
   - 9.3. A password of any length greater than zero can be input.
   - 9.4. If the username already exists, it will inform the admin and the account will not be created.
   - 9.5. If no password is input, it will inform the admin and the account will not be created.
   - 9.6. If the username does not match the regular expression it will inform the admin and the account will not be created.
10. Allow for bookings to be deleted in large numbers.
    - 10.1. Only the admin will be able to perform this task.

- 10.2. The admin will select a date that all booking data within the database will be deleted up to. For example, an input of 02/21/2017 will delete all bookings up until 02/21/2017.

11. Allow for the importing of xml code to determine mandatory bookings.
    - 11.1. Only the admin will be able to perform this task.
    - 11.2. The admin will be able to select the file they require with a function.
    - 11.3. The file will be checked to make sure it is an xml file (if it isn’t the process will not proceed).
    - 11.4. An algorithm will be used to insert bookings into the database.
    - 11.5. The algorithm to insert the data will ignore break/lunch/registration times and will only use lesson times.
    - 11.6. The algorithm will work for a two week timetable and a one week timetable. It will also work with multiple rooms in the file.

Documented Design

Structure Chart


Booking System
Sign in
Best Room
Take user input requirments
Take room information from database
Generate scores for each room
Update Table (Admin only)
Take and validate user input
Delete bookings up to the given date
Sign Up (Admin only)
Take and validate user input
Md5 hash the given password
Insert new user into database
Import File (Admin only)
Take and validate user input
Import file
Parse through file
Insert bookings into database
Select Room
Generate timetable
Change Room
Change Date
Change week
Select Cell
Generate room/booking information.
Validate the creation of a booking
Validate the removal of a booking
Change database (delete or insert)

General Overview

Designing the system will require planning and drafting of the key algorithms that will be essential to making my project a success. The structure chart gives insight to the algorithms I will need to create including: ‘Update table’, ‘Sign up’, ‘Import file’, ‘Generate timetable’, ‘Create booking’, ‘Delete booking’, ‘Best room’. In addition, with php and xml being referenced in my analysis, I will be required to the identify the other software that will be used to develop the system.

Software and Coding Language

The system will be developed in **php** and **html**. The reason for this is that **php** allows a simple, basic manipulation of relational databases, enabling me to meet my objectives. The **html** will enable me to create the relevant forms so that I can obtain the information to manage and interact with the database. Structured query language (**sql**) will be required for querying, inserting and deleting the stored data, which will be important, as I indicated in my analysis that the system will constantly rely on communicating with the database. Cascading style sheets (**css**) will be used to develop a suitable layout so that the pages are not disarranged which could make it more difficult to use. In addition **xml** will be included with the file that must be imported by an admin to generate a handful of bookings at once. ‘Codio’ is a resource provided by the school which gives me access to all these formats and other management tools. It also includes tutorials to new types of code as I will need to learn **sql** in more detail. The tool ‘Phpmyadmin’ will be used to handle the administration of mysql (the database) and will be key to creating the tables to store information for my system.

Security

As a security measure, user passwords will have the md5 hash applied. When the admin creates a teacher’s account the password they enter will have the md5 hash applied to it and the result will be stored in the database. This result is a ‘one way’ value which means theoretically, it is not possible to be used to find the original data. It’s a basic security measure that can prevent unauthorised people accessing any prohibited data. Therefore, when a user signs in with their details, the password entered will also have md5 hash applied and the result will be compared with the stored value in the database to see if they match.

The Database

The data base will hold data including:

Information Stored	Unique (Yes/ No)	Data Type
Teacher Number	Yes	Integer
Teacher Username	No	Variable Character
Teacher Password	No	Variable Character
Is Admin	No	Integer
Room Name	Yes	Variable Character
Room Resources	No	Variable Character
Computer Count	No	Integer
Booking Number	Yes	Integer
Data of Booking	No	Date
Lesson Number	Yes	Integer
Lesson Time (Monday - Thursday)	No	Time
Lesson Time (Friday)	No	Time

At this stage I have added lesson times into the database plan. This will be used by teachers when viewing the timetable to receive more information from the program and make it more beneficial. As Friday has alternate lesson times to the rest of the week, two separate attributes will need to be used to determine the differences.

As the program will revolve around the use of a database, then this needs to be normalised so that no inconsistencies occur, and any potential free space is saved.
All possible data within the database as it appears without attempts of normalisation:

Room Name	Room resources	Computer Count	Teacher Number	Teacher Username	Teacher Password	Is Admin	Booking Number	Booking Date	Lesson Number	Lesson -Time (Mon -Thurs)	Lesson Time (Fri)
-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-

The database in this form will not work as the booking information and teacher information are independent of each other. In addition the teacher information should not reflect any lesson times. Therefore this is inefficient and cannot be used in my program.

In this case, many attributes will be repeated when describing each attribute and the database will be ineffective at looking for data.

First Normal Form

The database in First Normal Form (No repeating attributes):
This database can still be displayed in a more efficient way.

Teacher Number	Teacher Username	Teacher Password	Is Admin
-	-	-	-
-	-	-	-
-	-	-	-

Room Name	Booking Number	Booking Date	Lesson Number	Lesson -Time (Mon -Thurs)	Lesson Time (Fri)
-	-	-	-	-	-
-	-	-	-	-	-
-	-	-	-	-	-

Room Name	Room resources	Computer Count
-	-	-
-	-	-
-	-	-

Third Normal Form

The database in Third Normal Form where it is most efficient:

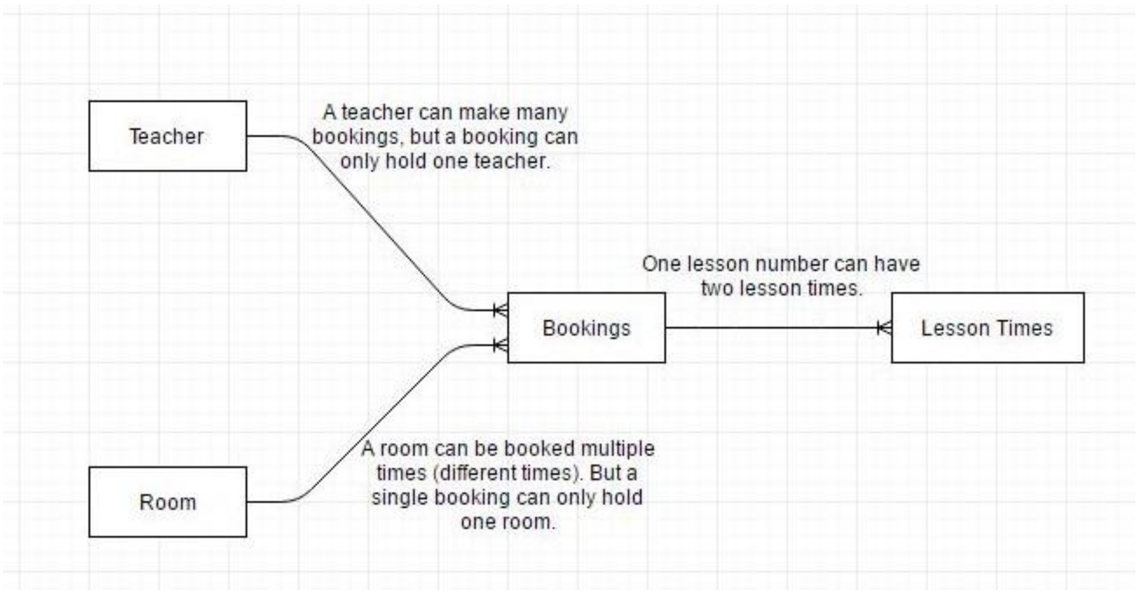
Teacher Number	Teacher Username	Teacher Password	Is Admin
-	-	-	-
-	-	-	-
-	-	-	-

Room Name	Booking Number	Booking Date	Lesson Number
-	-	-	-
-	-	-	-
-	-	-	-

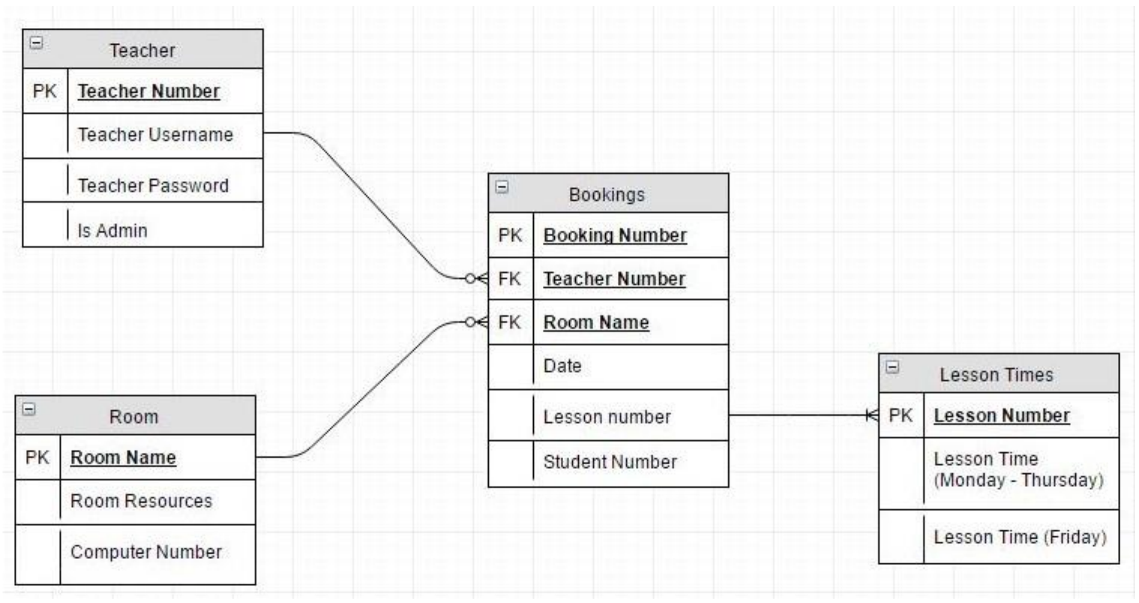
Lesson Number	Lesson Time (Mon -Thurs)	Lesson Time (Fri)
-	-	-
-	-	-
-	-	-

Room Name	Room resources	Computer Count
-	-	-
-	-	-
-	-	-

Entity Relationship Diagram (1st Draft)


Teacher
A teacher can make many bookings, but a booking can only hold one teacher.
One lesson number can have two lesson times.
Bookings
Lesson Times
Room
A room can be booked multiple times (different times). But a single booking can only hold one room.

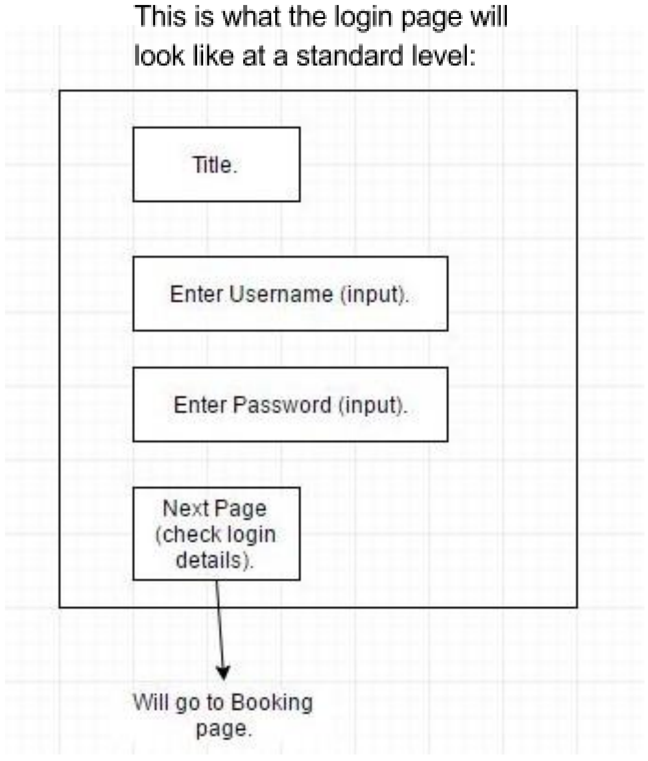
Entity Relationship Diagram (2nd Draft)



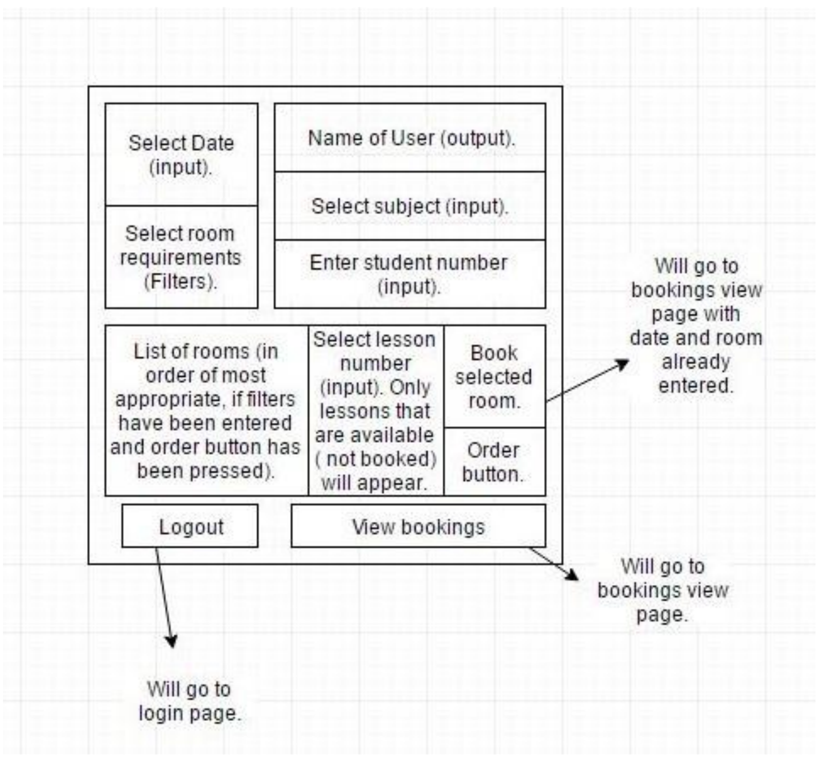
User Interface (1st Draft)

The user interface was described by my customer to be relatively simple to use. However, he wanted a handful of options that teachers can use so that it benefits them more than the current system. I designed the following templates to present what my final program could look like.

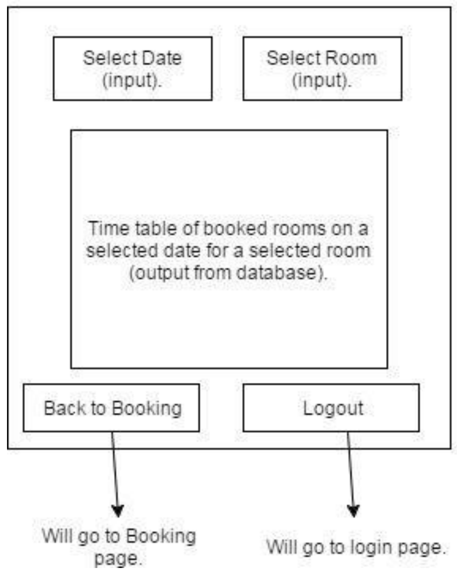
This is what the login page will look like at a standard level:



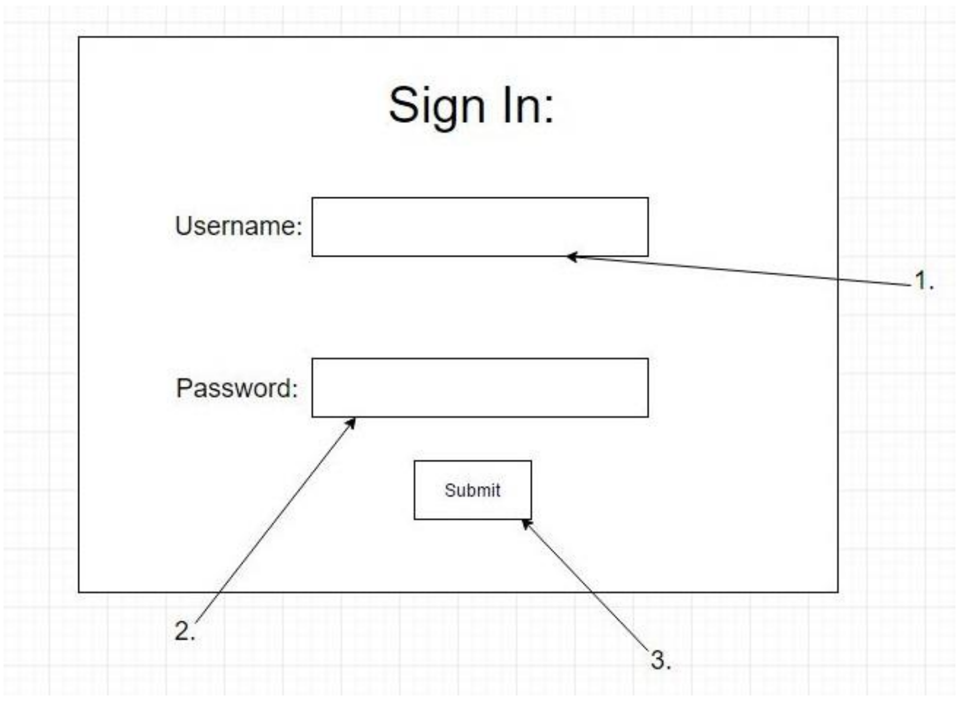
This is what the booking page will look like at a standard level:


Select Date (input).
Name of User (output).
Select subject (input).
Select room requirements (Filters).
Enter student number (input).
List of rooms (in order of most appropriate, if filters have been entered and order button has been pressed).
Select lesson number (input). Only lessons that are available (not booked) will appear.
Book selected room.
Order button.
Will go to bookings view page with date and room already entered.
Logout
View bookings
Will go to bookings view page.
Will go to login page.

This is what the bookings view page will look like at a standard level:



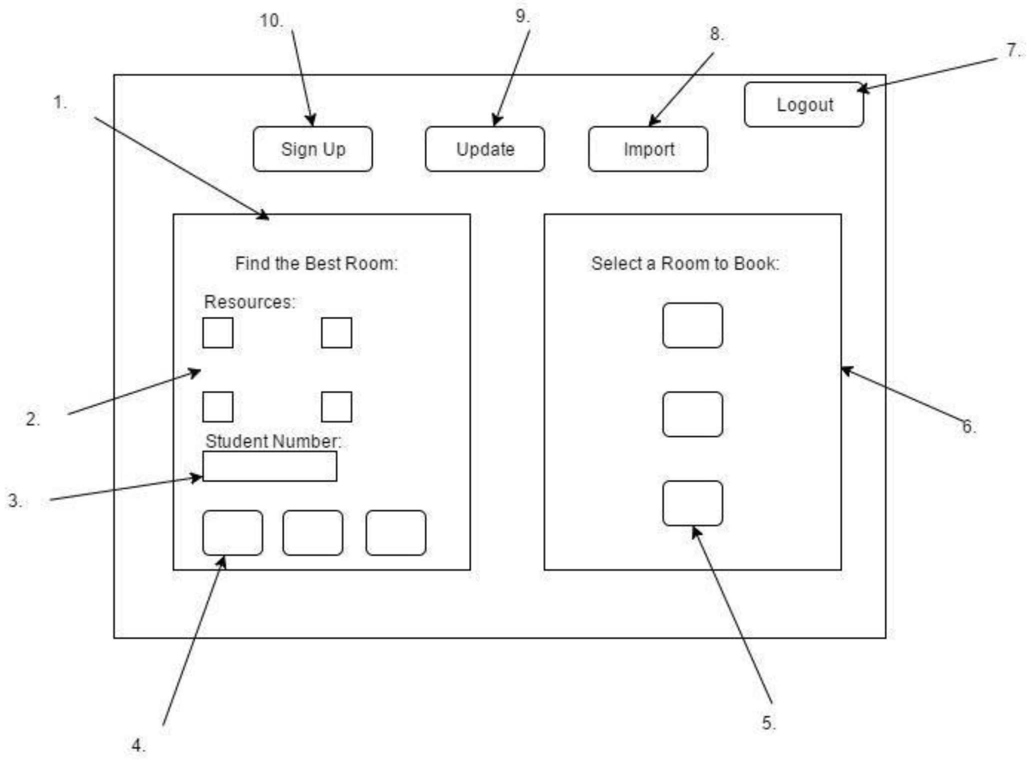
User Interface (2nd Draft)



1. This is an input box where the user will type their username.

2. This is an input box where the user will type their password. The text will not be shown when typing for security measures.

3. This is the submit button which will then take the given information and perform checks of validation. If the data is correct then the program will take the user to the Best Room page. However if the data is incorrect it remains on the same page.



1. This part of the page will allow the user to see the rooms from most suitable to least suitable, and make their booking based off of the results. This is only optional and they can use [6.] to instantly select a room and go straight to the table.

2. These are check boxes that the user can select/ deselect to inform the program what they need out of the computer room.

3. The is an input box that will only allow for integers. This will act as the number of students in the class. When submit with the resources wanted, then [4.] will appear as the list of rooms.

4. These will appear as the rooms in order of suitability. They also act as buttons and when pressed it will take the user to the table with that specific room.

5. These are the rooms in no particular order. They are submit buttons that take the user directly to the table when one is selected.

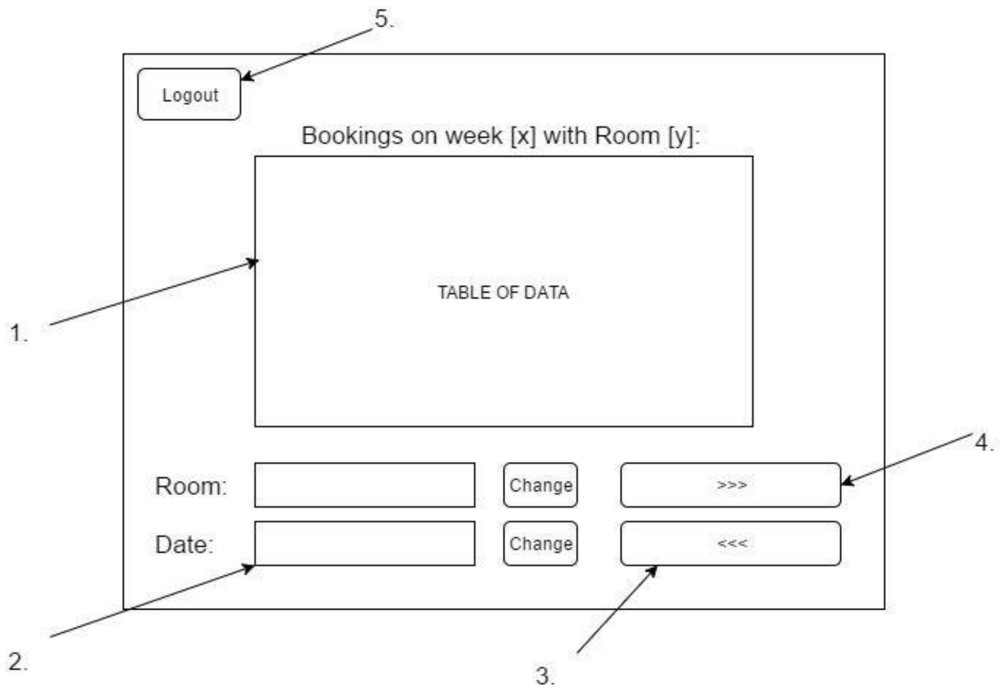
6. This area will just contain rooms and gives the user the option to book much faster, without having to find the best room.

7. This is a submit button, and when pressed it will delete the session and return the user back to the login page.

8. This can only be selected by the admin, and will take them to a new page where they can import xml data about unbookable rooms.

9. This can only be selected by the admin, and will take them to a new page where they can delete any out of a date bookings, returning them to this page.

10. This can only be selected by the admin, and will take them to a new page where they can create new accounts for other users.



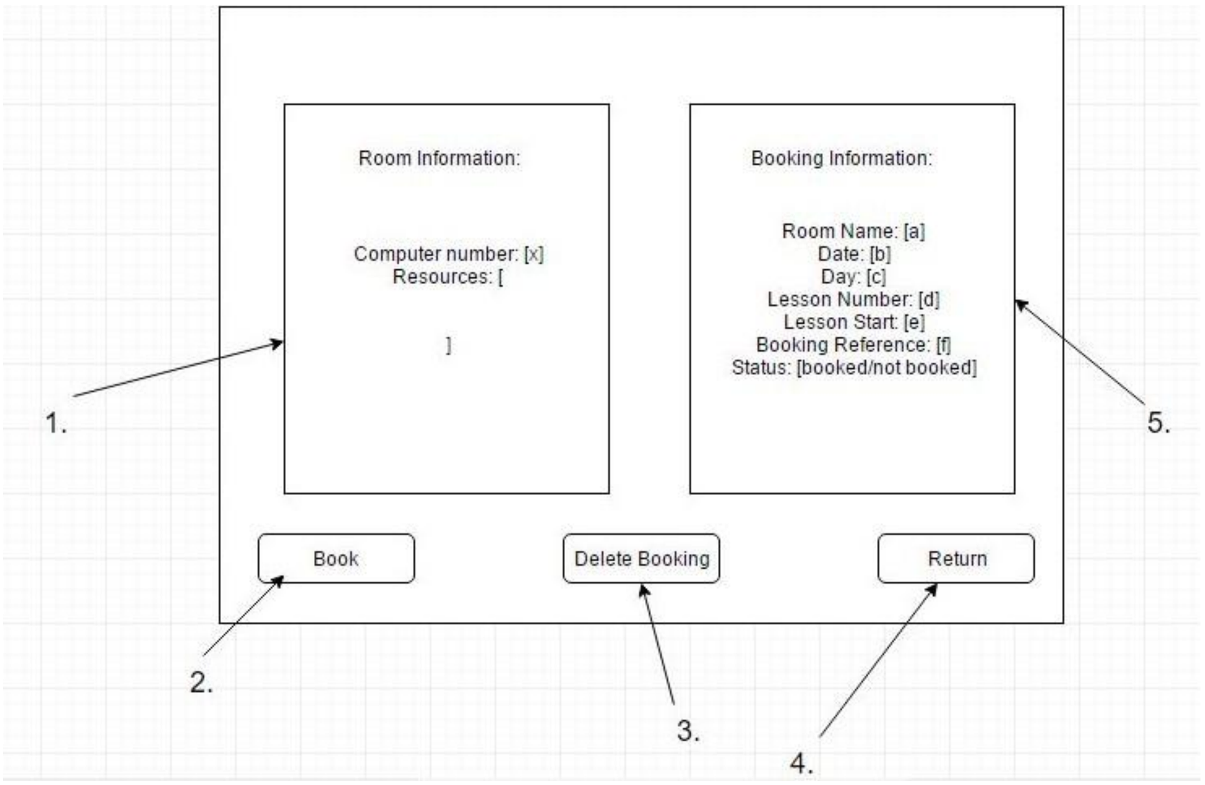
1. The bookings made will be shown in a table format and will have dates as headers, with lesson number as the first column. Each cell will contain empty (if unbooked) or the username of the booker. The text will be a hyperlink that takes the user to the View page.

2. These are two input boxes that act independently to allow the user to change the date or room. This will change how the the table appears.

3. This button will move the timetable data **back** a week. It is designed to make managing the timetable much faster for users and is an alternative to selecting a specific date.

4. This button will move the timetable data **forward** a week. It is designed to make managing the timetable much faster for users and is an alternative to selecting a specific date.

5. This is a submit button, and when pressed it will delete the session. It will return the user back to the login page.



1. This section of the page includes data relevant to the room, such as how many computers it contains and what resources. This data will be taken out of the database and the resources will be presented in a list (bullet point) format.

2. This is a submit button that will allow the user to select the room (by inserting data into the database). However, if there is already data there, the booking cannot be overwritten without using [3.] first. This will return the user to the table page if the booking is successful.

3. This is another submit button that allows the user who made the booking or an admin, to delete it so that it can be overwritten by anyone. This will return the user to the table page.

4. This is another submit button that takes the user back to the table page without making any changes to the database.

5. This section of the page includes data relevant to the booking, such as the date, room, lesson number, lesson start time etc.It will hold a booking reference only if the room has been booked.

Key Algorithms

No.	Algorithm name	Purpose	Basic Overview
1	Check	To make sure the user inputs the correct information when signing in.	The program will query the 'teachers' table in database and compare the details input by the user with the rows taken to see if they match.
2	Best Room	To list the most suitable rooms for the teacher (not compulsory to use).	The algorithm will use inputs and compare with the information in the database to create a score for each room. The best room will have the highest score, so it will be output in descending order.
3	Define Start/ Generate timetable	To define the the dates at which the table will range from, even if the user has not yet input a required date.	The program uses the date that was input to identify when Monday is (start of the table), and when Friday is (end of the table). If no date was input, then the program will use the current time.
4	Book Room	To allow the user to book a room.	The program will use sql code to check if there is already a booking (query the specific date and room to see if it is empty or not). Then it will insert the required data into the database which will be the booking.
5	Delete Booking	To allow the user (who booked the room) or an admin to delete a booking.	The program will use sql code to delete any booking that matches the input room name and input date. This will only work if they are the same user who made the booking.
6	Update Table - admin only	To allow the admin to delete out of date bookings.	When the admin selects the option, the current date will be calculated using a php function. Then, the start of the current

			week will be calculated. Any booking before this date will be deleted using sql code.
7	Upload Xml - admin only	To upload an xml file, then read through it and define rooms that need to be booked.	The algorithm will parse through the xml data and insert into the database the relevant bookings. It will require a standard that can be used for different files.
8	Sign Up - admin only	To create a new account with a unique username. This will allow other teachers to enter and use the program.	The algorithm will contain checks to make sure that the entered username meets the requirements set and that a password has been entered.

1. Check Details

For this algorithm, the user will input their login details and the system will check if these are valid. It will do this by creating a query from the database to select the records with the same username and password. If the query created does not have any data inside then the details input were invalid and the user should not be able to enter the system. If a record has been found, then the user will enter the system with their username stored globally.

An example of this algorithm can be displayed as:

```
username ← User input
password ← User input
Select all from teachers (with the username and password)
Number of rows ← Number of rows from query
If Number of rows >0
      Login ← True
      globaluser ← username
      move to next page
Else
      Login ← False
```

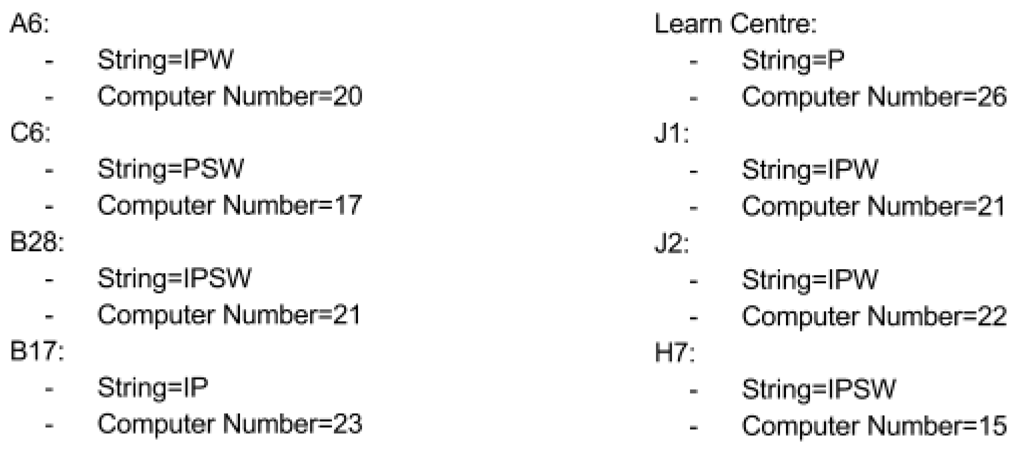
2. Best Room Algorithm

The key algorithm that the system will use determines the most appropriate rooms for the user. The result should list the rooms in order of highest suitability so that the user has an indication of what the most appropriate room is.
In order to do this each room being used must be defined with its resources and the number of computers it contains.This will be manually implemented into the database and will remain constant.

The rooms table, as shown when normalising the database, will contain the room name (i.e A6), the resources it includes (i.e IPW) and the number of computers it includes (i.e 20). The room's resources will be stored as a string, where each letter correlates to an item.

```
V → Interactive Whiteboard
P → Printer
W → Whiteboard
S → Scanner
```

The values which will be stored in the database include:



A6:
- String=IPW
- Computer Number=20

C6:
- String=PSW
- Computer Number=17

B28:
- String=IPSW
- Computer Number=21

B17:
- String=IP
- Computer Number=23

Learn Centre:
- String=P
- Computer Number=26

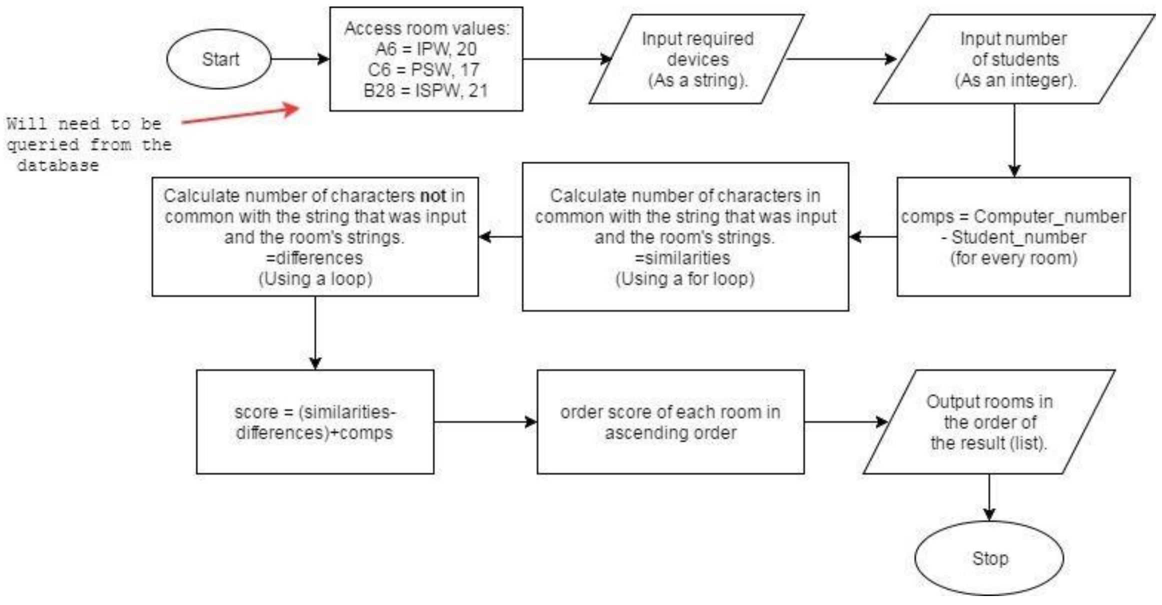
J1:
- String=IPW
- Computer Number=21

J2:
- String=IPW
- Computer Number=22

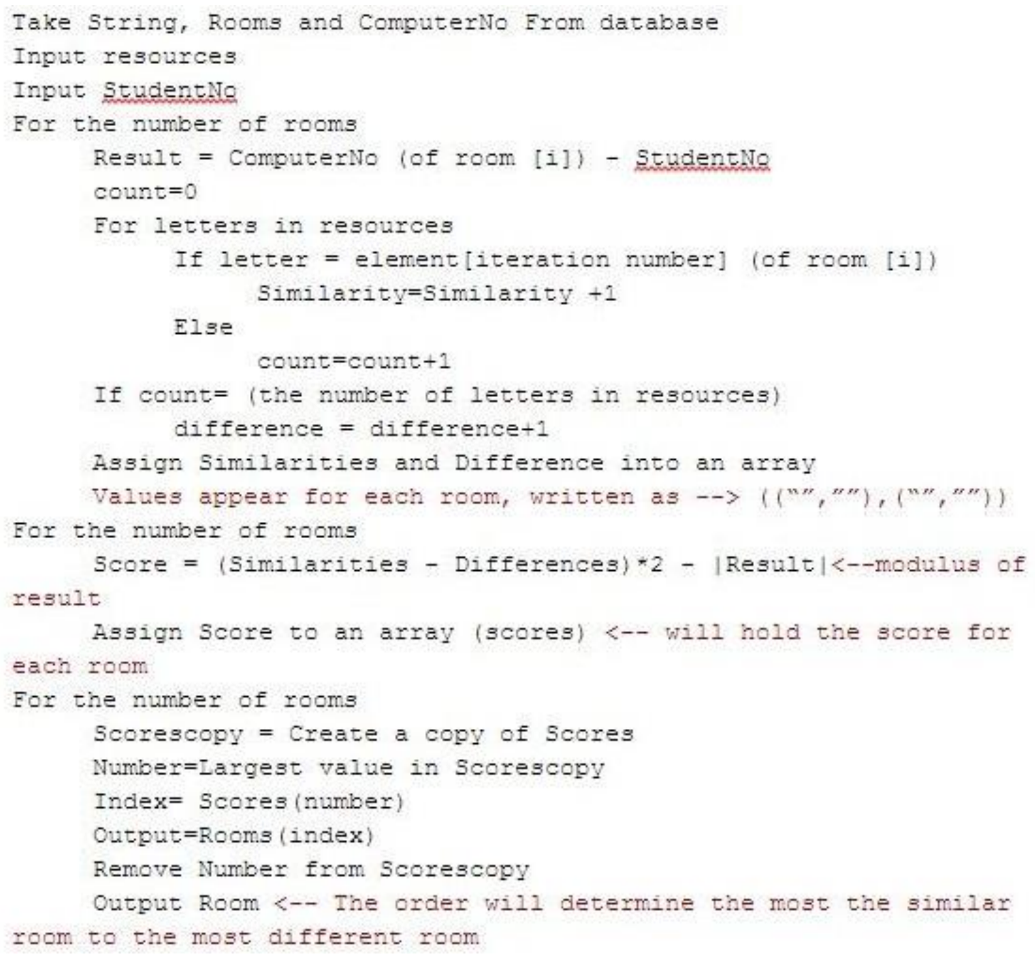
H7:
- String=IPSW
- Computer Number=15

The user will select the resources they want as well as the number of computers. The algorithm will find the number of similarities (integer) and the number of differences (integer) between the resources input and the strings stored within the database. In addition, the system will work out the difference (as an integer) between the number of computers the room includes and the number input, by subtracting them. From these values, a score will be calculated for each room and they will be ordered into a list. The rooms with the highest scores will appear at the top of the list, indicating that they are the most suitable, and the rooms with lowest scores will appear at the bottom of the list.
To generate the ordered list, the algorithm will search for the highest value (in the scores list) and place the corresponding room at the front. The identified value will be removed from the scores list and the next highest value will be found so that its corresponding room can then be enqueued onto the ordered list, until all of the rooms have been positioned. This could mean that a room with the same score as another may appear one place higher in the list generated. To prevent the user getting false information, I will display the scores next to each room so that it is more clear which rooms are better for the user.
In order to code my algorithm I must decide what functions and operations it will contain. I began with a basic flowchart to express the step by step method I want to approach. The final version will contain nested loops, however they were left out deliberately in the first stage of planning to reduce complexities.

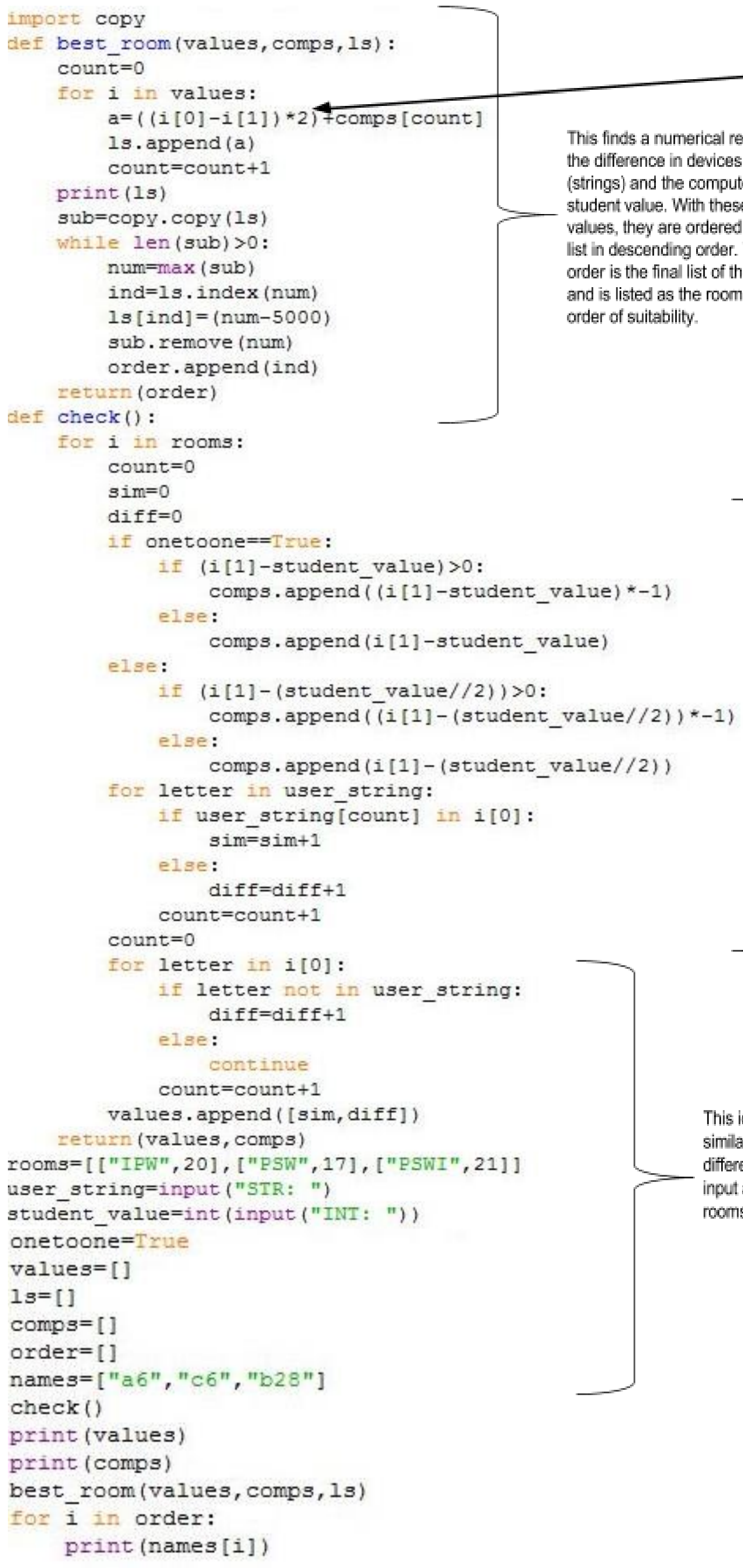
Flow chart for best room algorithm:
This demonstrates the order of the processes required to find the correct result. It does not go into the the detail of each individual function as I am going to create a prototype that is able to display this more directly.


Start
Access room values: A6 = IPW, 20 C6 = PSW, 17 B28 = ISPW, 21
Will need to be queried from the database
Input required devices (As a string).
Input number of students (As an integer).
comps = Computer_number - Student_number (for every room)
Calculate number of characters in common with the string that was input and the room's strings. =similarities (Using a for loop)
Calculate number of characters **not** in common with the string that was input and the room's strings. =differences (Using a loop)
score = (similarities-differences)+comps
order score of each room in ascending order
Output rooms in the order of the result (list).
Stop

Pseudo Code that was created to display more detail of the functions required:



```
Take String, Rooms and ComputerNo From database
Input resources
Input StudentNo
For the number of rooms
      Result = ComputerNo (of room [i]) - StudentNo
      count=0
      For letters in resources
            If letter = element[iteration number] (of room [i])
                  Similarity=Similarity +1
            Else
                  count=count+1
      If count= (the number of letters in resources)
            difference = difference+1
      Assign Similarities and Difference into an array
      Values appear for each room, written as --> ((“”,””),(“”,””))
For the number of rooms
      Score = (Similarities - Differences)*2 - |Result|<--modulus of
result
      Assign Score to an array (scores) <-- will hold the score for
each room
For the number of rooms
      Scorescopy = Create a copy of Scores
      Number=Largest value in Scorescopy
      Index= Scores(number)
      Output=Rooms(index)
      Remove Number from Scorescopy
      Output Room <-- The order will determine the most the similar
room to the most different room
```



```python
import copy
def best_room(values,comps,ls):
    count=0
    for i in values:
        a=((i[0]-i[1])*2)+comps[count]
        ls.append(a)
        count=count+1
    print(ls)
    sub=copy.copy(ls)
    while len(sub)>0:
        num=max(sub)
        ind=ls.index(num)
        ls[ind]=(num-5000)
        sub.remove(num)
        order.append(ind)
    return(order)
def check():
    for i in rooms:
        count=0
        sim=0
        diff=0
        if onetoone==True:
            if (i[1]-student_value)>0:
                comps.append((i[1]-student_value)*-1)
            else:
                comps.append(i[1]-student_value)
        else:
            if (i[1]-(student_value//2))>0:
                comps.append((i[1]-(student_value//2))*-1)
            else:
                comps.append(i[1]-(student_value//2))
        for letter in user_string:
            if user_string[count] in i[0]:
                sim=sim+1
            else:
                diff=diff+1
            count=count+1
        count=0
        for letter in i[0]:
            if letter not in user_string:
                diff=diff+1
            else:
                continue
            count=count+1
        values.append([sim,diff])
    return(values,comps)
rooms=[["IPW",20],["PSW",17],["PSWI",21]]
user_string=input("STR: ")
student_value=int(input("INT: "))
onetoone=True
values=[]
ls=[]
comps=[]
order=[]
names=["a6","c6","b28"]
check()
print(values)
print(comps)
best_room(values,comps,ls)
for i in order:
    print(names[i])
```

This value is doubled within the calculations so that the result favours matching devices over student to computer similarities. This is because sharing a computer is easier than allocating devices that are unique to certain rooms.

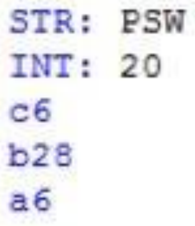
This finds a numerical result for the difference in devices (strings) and the computer to student value. With these values, they are ordered into a list in descending order. This order is the final list of the rooms and is listed as the rooms in order of suitability.

This code was created in python as a prototype of this algorithm.

This compares the student number with the number of computers in each computer room by subtracting the values. It multiplies the result by -1 if it's greater than zero so that all results have the same sign.

This identifies the number of similarities and the number of differences between the string input and the string that the rooms hold.

Inputs and Outputs example:

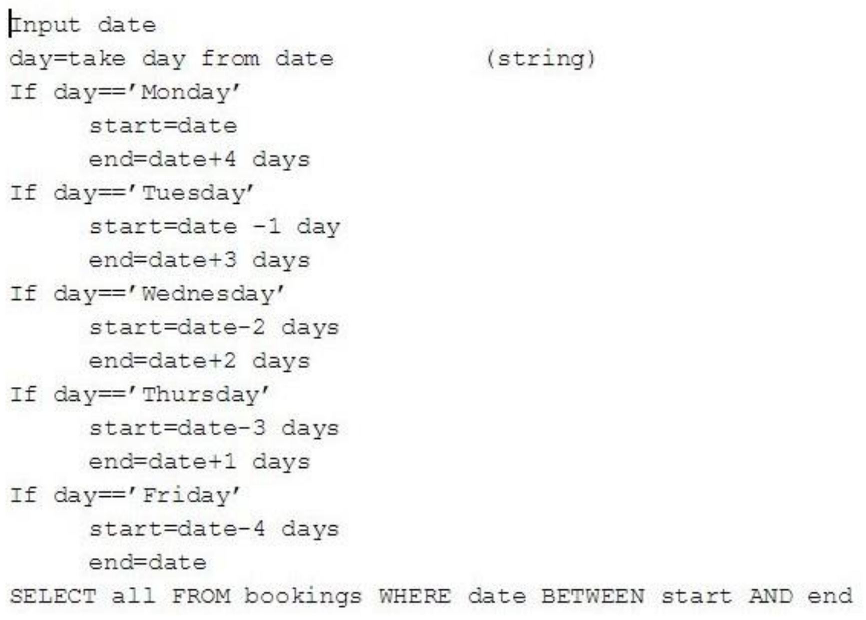


```
STR: PSW
INT: 20
c6
b28
a6
```

3. Define Start / Generate timetable

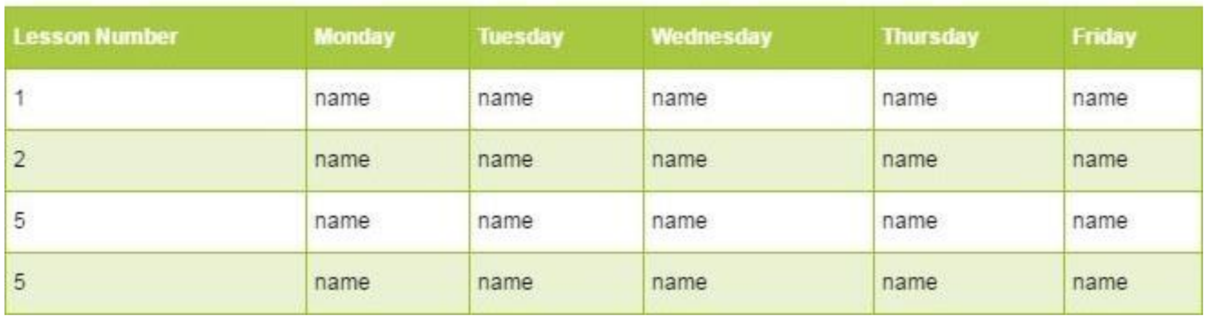
When the user first selects a computer room, they are taken to another page which displays a timetable for the chosen rooms bookings. The timetable will initially be displayed for the current week (Monday to Friday). If the the user decides to change the date then the timetable will have to update and calculate the new values for Monday to Friday. In both cases, an algorithm must be formed in order to calculate the start of the week and the end of the week. For example if I entered the date 10/3/2017 (Friday, as d/m/y), then the monday will be calculated by taking 4 days off to give us 6/3/2017 (Monday, as d/m/y). If the input date was a Saturday or Sunday then it would find the dates for the week following.

The following code is a representation of the concept I want to apply within my php script so that I can calculate the start and end points of the week, from the date that was given.
Pseudo Code:



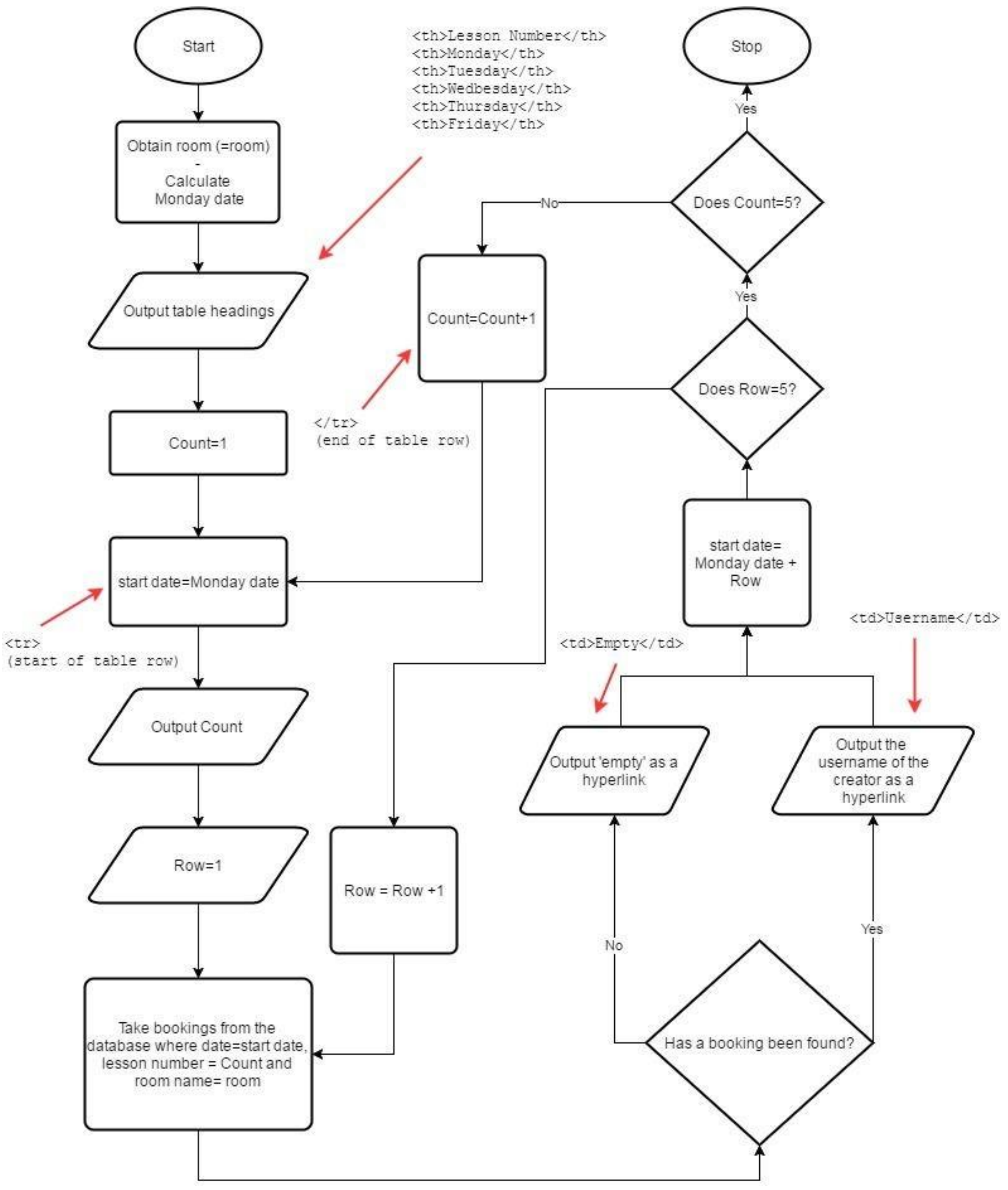
```
Input date
day=take day from date          (string)
If day=='Monday'
      start=date
      end=date+4 days
If day=='Tuesday'
      start=date -1 day
      end=date+3 days
If day=='Wednesday'
      start=date-2 days
      end=date+2 days
If day=='Thursday'
      start=date-3 days
      end=date+1 days
If day=='Friday'
      start=date-4 days
      end=date
SELECT all FROM bookings WHERE date BETWEEN start AND end
```

The image below represents the table that will be created on the website. The cell values containing ‘name’ indicate where a booking has been created. In the final solution, this will be displayed as the username of the booking creator. The username will also be a hyperlink which will transfer the user onto another page so that anyone can get more detail about the room and the time it has been booked for. The hyperlink will contain data about the booking and will be used for the Book Room and Delete Booking algorithm.
For this part of the system I will require html tags that will generate the timetable onto the webpage. I followed the example provided by Mr Hamilton that shows the table structure being used in the current system (appendix - part **C**). This is the table I will generate:



Lesson Number	Monday	Tuesday	Wednesday	Thursday	Friday
1	name	name	name	name	name
2	name	name	name	name	name
5	name	name	name	name	name
5	name	name	name	name	name

The following is what tags will be used:
`<th></th>` Will be used to display each table heading. There will be six tags in succession, starting with 'Lesson Number' and then 'Monday' to 'Friday' as shown in the table image.
`<tr></tr>` Will create a new row that will contain the cells of data. These tags will have to be placed at the start of a loop so that the cells can be inserted one at a time.
`<td></td>` Will create the table data. These will be each individual cell and will include either 'empty' or the username of the booking creator.


Start
Obtain room (=room) - Calculate Monday date
<th>Lesson Number</th> <th>Monday</th> <th>Tuesday</th> <th>Wedbesday</th> <th>Thursday</th> <th>Friday</th>
Output table headings
Count=1
</tr> (end of table row)
<tr> (start of table row)
start date=Monday date
Output Count
Row=1
Take bookings from the database where date=start date, lesson number = Count and room name= room
Row = Row +1
Count=Count+1
Stop
Yes
Does Count=5?
No
Yes
Does Row=5?
start date= Monday date + Row
<td>Empty</td>
<td>Username</td>
Output 'empty' as a hyperlink
Output the username of the creator as a hyperlink
No
Yes
Has a booking been found?

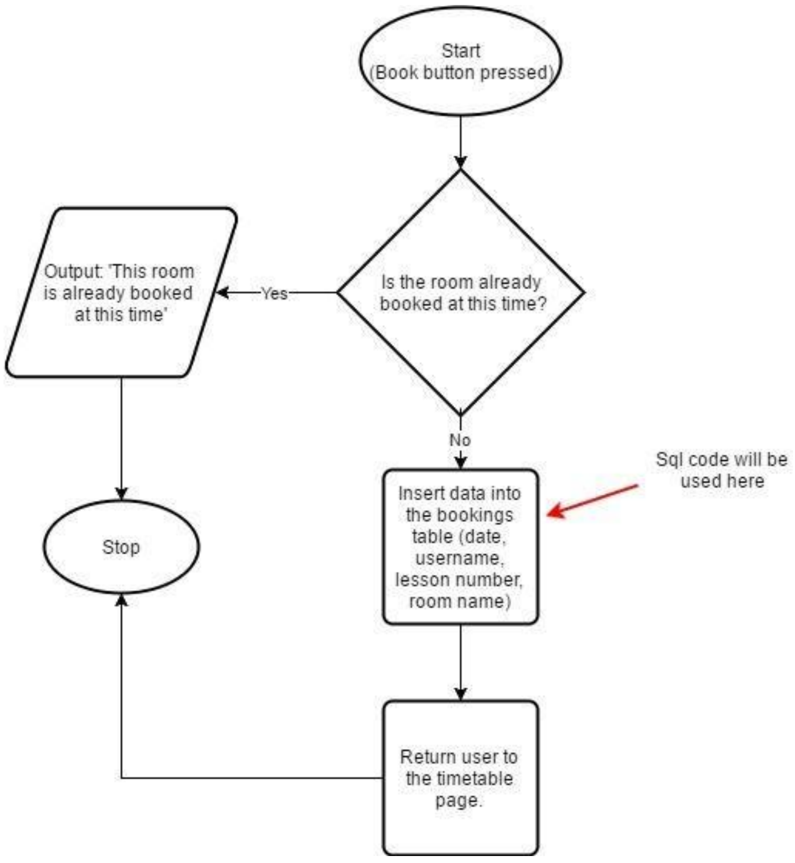
4. Book Room

After the user selects a cell within the timetable, they are able create a booking. When the user selects the 'book' button, the algorithm will check if this room is already booked for this time. It will do this by retrieving the data sent from the hyperlink made from the timetable page. If the room is already booked, then the user will receive an output informing them that they cannot make the booking. However, if is available, then data will be inserted into the database that will form a booking. The user will then be returned back to the timetable page where they will be able to see the booking they have just created.
In order to insert data into the database, an sql statement will be required to store the booking so that it can be utilised in other parts of the system.

Variables required:

- Booking Number (integer) - This will be used to validate if a room is booked or not. If this variable has been set, then a booking is present and the user can not book the room again.
- Username (varchar) - This will be used to create the booking when inserting into the database.
- Date (date) - This will be used to create the booking when inserting into the database.
- Lesson Number (integer) - This will be used to create the booking when inserting into the database.
- Room Name (varchar) - This will be used to create the booking when inserting into the database.

The following is a flow chart which displays the steps of this algorithm:


Start
(Book button pressed)
Is the room already booked at this time?
Yes
Output: 'This room is already booked at this time'
No
Insert data into the bookings table (date, username, lesson number, room name)
Sql code will be used here
Stop
Return user to the timetable page.

```
If book button has been pressed
      If this room is booked
            Output 'This room is already booked at this time'
      Else
            Insert into Bookings
            Return user to the timetable
Else
      -
```

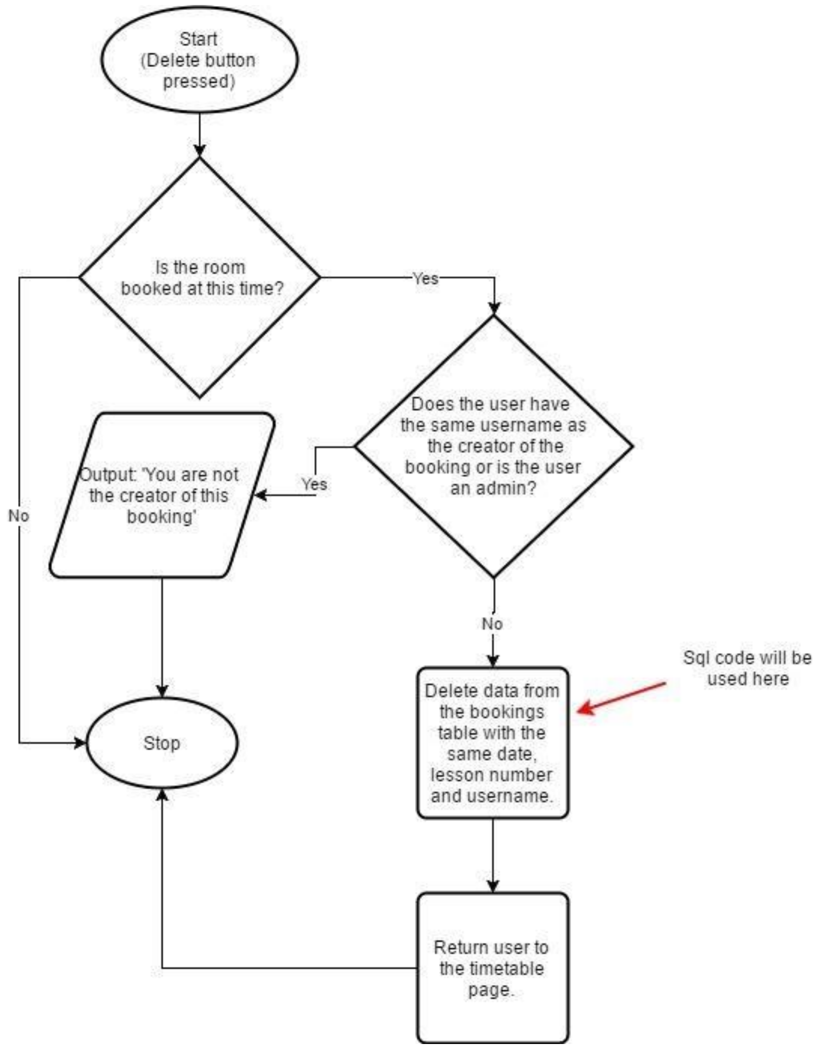
5. Delete Booking

After the user selects a cell within the timetable, they are able to delete a booking. When the user selects the 'delete' button, the algorithm will check if this room is booked for this time. It will do this by retrieving the data sent from the hyperlink made from the timetable page. If the room is not booked, then the algorithm will stop. For a booked room, the algorithm will check if the usernames of the booking creator and the current user (making the cancellation) match. Matching usernames will allow the system to delete the booking from the database, however, I will also implement the procedure that will allow admins to delete any booking.
In order to delete data from the database, an sql statement will be required to remove the booking. Therefore I will need variables taken from the hyperlink that associate with the booking details (date, lesson number etc.).
Overall, this algorithm contains many elements from the Book Room algorithm as they both require checks to see if the room is booked or not before changing the database. Therefore, I will expand on the previous flow chart created.
The following is a flow chart which displays the steps of this algorithm:
Variables required:

- Booking Number (integer) - This will be used to validate if a room is booked or not. If this variable has not been set, then this algorithm will stop because there is no booking to delete.
- Username (varchar) -

Current user - This will be used to compare with the other username (booking). If they match, then the algorithm can continue as the creator of the booking has the ability to delete their own reservation.
Booking - This username will be used to delete the booking as part of the sql statement.

- Booking Date - This will be used to delete the booking as part of the sql statement.
- Lesson Number - This will be used to delete the booking as part of the sql statement.
- Room Name - This will be used to delete the booking as part of the sql statement.
- Admin - This will be used to identify if the user is an admin. If they are then they have the ability to delete the booking even if they are not the creator of it.


Start
(Delete button pressed)
Is the room booked at this time?
Yes
No
Does the user have the same username as the creator of the booking or is the user an admin?
Yes
Output: 'You are not the creator of this booking'
No
Delete data from the bookings table with the same date, lesson number and username.
Sql code will be used here
Stop
Return user to the timetable page.

An example of this algorithm is produced below:

```
If delete button has been pressed
      If this room is booked
            If username (user) equals username (booking) or Admin
            equals true
                  Delete from Bookings where Date equals Booking
                  Date and Lesson equals Lesson Number
                  Return user to the timetable
            Else
                  Output 'You did not create this booking'
      Else
            -
Else
      -
```

6. Update Table

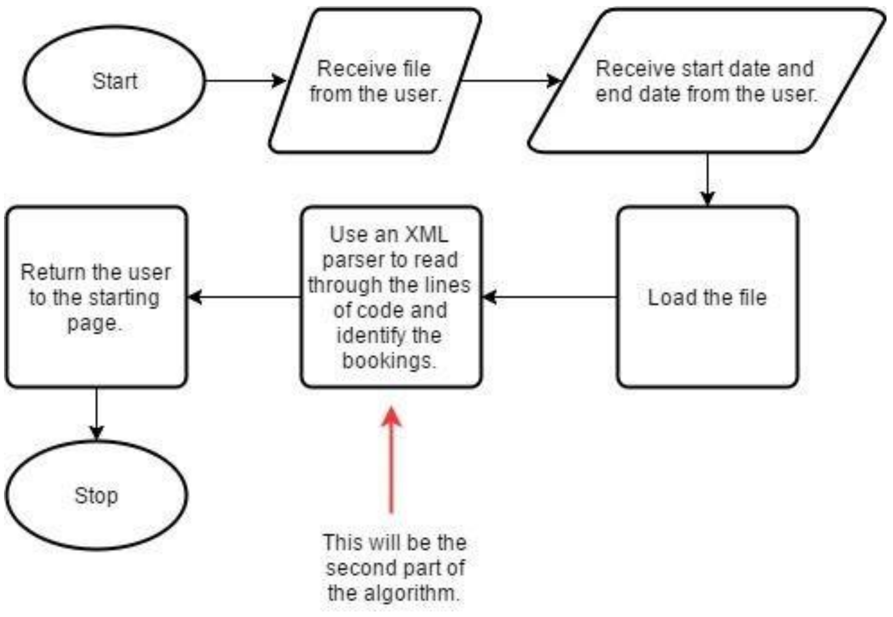
Deleting a large number of bookings will first require the user to input a date. All bookings before this date will then be removed from the database. In order to do this, an sql statement will be required. The user will then be returned to the previous page.
When they look at the timetable, they should be able to see that bookings up to the date they input have been removed.

An example of this algorithm is produced below:

```
Remove date ← User input
Delete from Bookings where Date < Remove date
```

7. Upload Xml

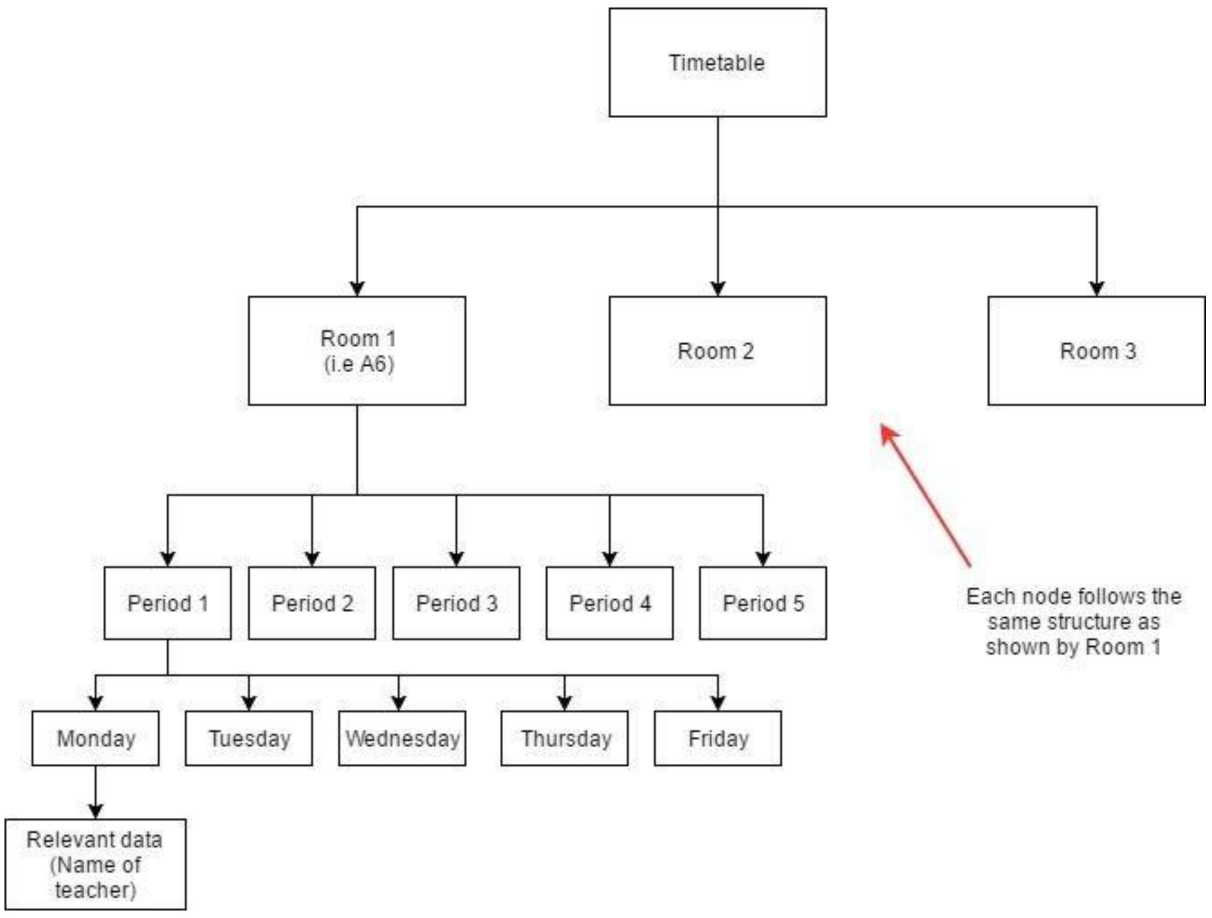
This algorithm will be built up of two main parts. The first part will be obtaining the file from the user, as well as other inputs. The second part will be the XML parser which will be required to identify the bookings embedded within the file.
I designed the structure of this algorithm by creating a flowchart to show me where the XML parser will need to be implemented.


Start
Receive file from the user.
Receive start date and end date from the user.
Load the file
Use an XML parser to read through the lines of code and identify the bookings.
Return the user to the starting page.
Stop
This will be the second part of the algorithm.

In order to read the XML file, the simple XML parser will be required as discussed within my analysis phase. However, as the XML file is unique to most structures, I will have to create the algorithm so that it can successfully read all files that have the same layout, but different data.

I attempted to design a flowchart for this part of the algorithm but it would be too time consuming and not beneficial due to the large number of variables being changed at every step. I therefore approached this algorithm differently as I was required to identify the structure of the file provided.

The XML file has the following structure:



The algorithm must be able to read this tree structure. It must also be able to read it multiple times must also increment the ‘day’ variable after each read iteration. When this variable is equal to five, then it can be reset back to one for the start of the next week. This can be shown as:

```
→ Iterate continuously (while loop) until end date has been proceeded.
        → Iterate through each room. (SimpleXML statement)
                →Iterate through each lesson. (SimpleXML statement)
                        → Iterate through each day. (SimpleXML statement)
                                → Iterate through each cell. (SimpleXML statement)
```

When the two weeks have been iterated through, then the start date will be increased by fourteen so that it will begin the next two week cycle with the correct dates.

When the booking data calculated is equal to or greater than the end date. Then the variable ‘Break’ will become true. This means that when that iteration inside the while loop is finished, then it will not create another iteration and the algorithm can end. This can be shown as:

```
If bookdate > or equal to end date:
      Break equals True
Else
```

```
      -
```

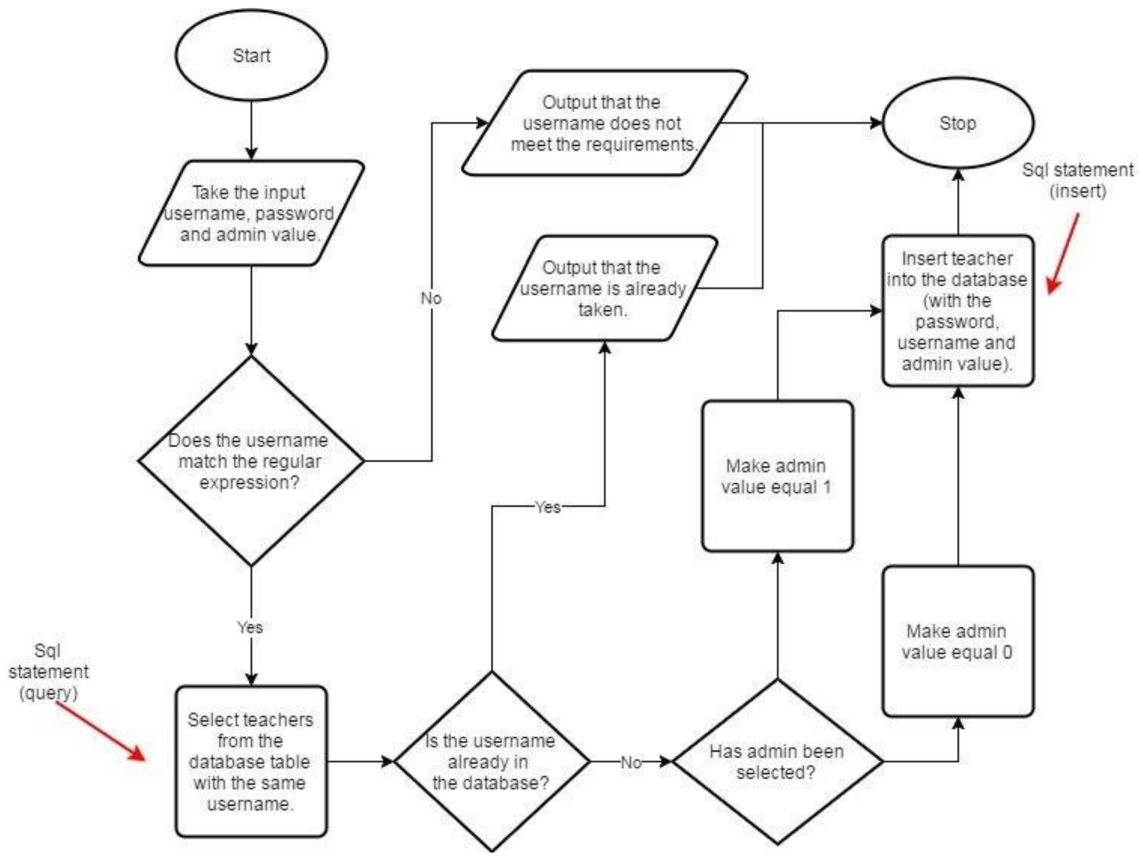
Moreover, a separate, but similar function will be required when creating the bookings, this can be shown as:

```
If book date < startdate and bookdate > end date:
      -
Else
      Delete from Bookings table
      Insert into Bookings table
```

It shows that a booking will only be inserted into the database if it between the two dates entered by the user (inclusive). To prevent a double booking, it will overwrite data (delete a record with the same room, lesson number and date).

8. Sign Up

For this algorithm, the user must enter a username, password and an admin value before submitting. The username will be checked with a regular expression, so that it can only match the SIMS layout. The password will be checked to make sure it has been entered. The admin value will be set with a check box, so the user will have to tick or leave unticked to determine whether they want to make the new account have access to admin features. The value will be stored in the database as a 1 (admin equals true) or a 0 (admin equals false).The username must also be unique, which means that the system must query the database to check if it is already in use, similar to the check algorithm. When the input requirements have been met then the username, password and admin value will be inserted into the teacher table so that it can be used to perform a login.


Start
Take the input username, password and admin value.
Does the username match the regular expression?
No
Output that the username does not meet the requirements.
Stop
Sql statement (insert)
Output that the username is already taken.
Insert teacher into the database (with the password, username and admin value).
Make admin value equal 1
Yes
Make admin value equal 0
Yes
Sql statement (query)
Select teachers from the database table with the same username.
Is the username already in the database?
No
Has admin been selected?

The regular expression must accept:
→ A single letter (capital), followed by a single space, followed by a single letter (capital) followed by a single space, followed by followed by a single letter (capital), followed by one or more letters.
E.g.

- ‘W T Hamilton’ will be accepted.
- ‘N J King’ will be accepted.
- ‘Nathan J King’ will not be accepted.

An expression for this can be written as:
`[A-Z]\s[A-Z]\s[A-Z]\w+`

OR
→ A single letter (capital), followed by a single space, followed by a single letter (capital), followed by one or more letters.
E.g.

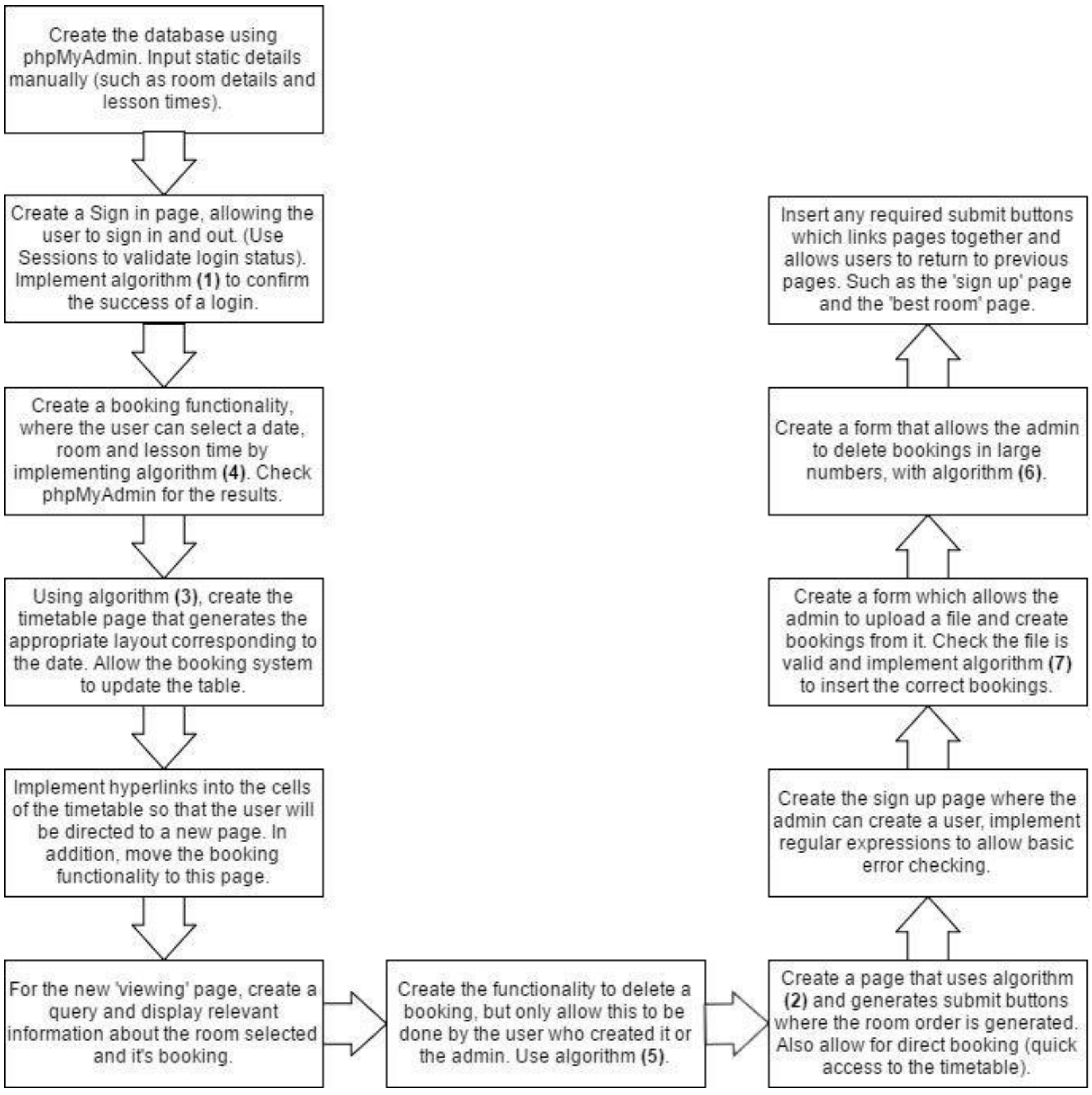
- ‘N Rutter’ will be accepted.
- ‘N King’ will be accepted.
- ‘Nathan King’ will not be accepted.

An expression for this can be written as:
`[A-Z]\s[A-Z]\w+`
The expressions can therefore be combined into a single statement:
`([A-Z]\s[A-Z]\s[A-Z]\w+)|([A-Z]\s[A-Z]\w+)` I used a regular expression tester (linked in the bibliography - **5**) to check that my examples match the prototype statement.

Coding the System

The following is the structure in which I will develop and code the system. The diagram displays the order I will take at developing each function and when my key algorithms I created (table on page **21**) will be required.

Order of Development Diagram:


Create the database using phpMyAdmin. Input static details manually (such as room details and lesson times).
Create a Sign in page, allowing the user to sign in and out. (Use Sessions to validate login status). Implement algorithm **(1)** to confirm the success of a login.
Create a booking functionality, where the user can select a date, room and lesson time by implementing algorithm **(4)**. Check phpMyAdmin for the results.
Using algorithm **(3)**, create the timetable page that generates the appropriate layout corresponding to the date. Allow the booking system to update the table.
Implement hyperlinks into the cells of the timetable so that the user will be directed to a new page. In addition, move the booking functionality to this page.
For the new 'viewing' page, create a query and display relevant information about the room selected and it's booking.
Create the functionality to delete a booking, but only allow this to be done by the user who created it or the admin. Use algorithm **(5)**.
Create a page that uses algorithm **(2)** and generates submit buttons where the room order is generated. Also allow for direct booking (quick access to the timetable).
Create the sign up page where the admin can create a user, implement regular expressions to allow basic error checking.
Create a form which allows the admin to upload a file and create bookings from it. Check the file is valid and implement algorithm **(7)** to insert the correct bookings.
Create a form that allows the admin to delete bookings in large numbers, with algorithm **(6)**.
Insert any required submit buttons which links pages together and allows users to return to previous pages. Such as the 'sign up' page and the 'best room' page.

Technical Solution

The final code for this project is found in the appendix - part **G** where each file is named and annotated with comments.
The Following is a table that displays all the filenames and their core purpose. The same information can be found within the code (appendix - part **G**).

Filename	Purpose
Login.php	This file acts as a login page for users where they will enter their username and password to be sent to Check.php to check for validity.
Check.php	This file is used to validate the user's input details given in Login.php. If the details are correct, then a 'login' occurs and the user can be transferred to the next web page.
Bestroom.php	This file displays a list of the computer rooms to the user and allows them to select the most suitable room to access the timetable with. It also contains other submit buttons to take admins to other pages.
Table.php	This file is used to display the bookings for each computer room in a table so that the user can select and interact with the cells. The timetable can be managed using different menus.
View.php	This file is used to allow the user to view the selected cell in the timetable, where they can then create a booking, delete a booking or return without any changes made.
Update.php	This file is used to allow an admin to delete all bookings up to a given date.
UploadForm.php	This file is used to allow an admin to select a file to upload, where it will be validated in FileUpload.php.
FileUpload.php	This file is used to validate the file selected by the user from UploadForm.php and will then send the data to AdminData.php.
AdminData.php	This file presents the algorithm that must read the xml language and inserts relevant booking data onto the database. It uses the file given from FileUpload.php.
Signup.php	This file is used to allow the admin to create a new user account. It validates the input details to make sure it meets the requirements.
Logout.php	This file is used to sign a user out of the system.

Testing the Systems

In order to determine the success of this program I must test the functions I have coded. Due to the system creating many different web pages, screen shots would make it difficult to demonstrate each test in action. Therefore I have decided to create videos in which the test and its result can be seen. Each table has a reference to the test number and the links can be used to locate each video.

Test plan

Tests 1-5 include:

- Signing in with a valid login details.
- Logging out after signing in.
- Attempting to sign in with invalid login details.

Video link to view tests: https://youtu.be/zGx3wXyl1Ks

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
1	Login.php --->Bestroom.php	To test that a **valid** login details will direct the user to the next page.	I will enter 'N J King' as the username and '1234' as the password.	The user will be directed to the Bestroom.php page	Success	0:00
2	Bestroom.php --->Logout.php --->Login.php	To test that a user can log out.	I will select the logout button while on Bestroom.php.	I should be directed to Login.php and should have to re-enter my login details in order to enter the system again.	Success	0:09
3	Login.php	To test that **valid** login details will direct the user to the next page.	I will enter 'J A Hamilton' as the username and 'businessstudies' as the password.	The page will refresh and the user will have to re-enter the details.	Success	0:13
4	Login.php	To test that **invalid** login details will prevent the user from proceeding to the next page.	I will enter 'J A Hamilton' as the username and 'invalid' as the password.	The page will refresh and the user will have to re-enter the details.	Success	0:32

5	Login.php	To test that **invalid** login details will prevent the user from proceeding to the next page.	I will enter 'invalid' as the username and 'invalid' as the password.	The page will refresh and the user will have to re-enter the details.	Success	0:40

Tests 6-13 include:

- Best room algorithm testing (Compare results with python code).
- To test that selecting a room will direct the user to the table page with that room information.

Video link to view tests: https://youtu.be/8aWv7_lgjOQ

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
6	Bestroom.php	To test that the best room algorithm generates a list of rooms in the same order as the prototype python code. (Appendix - part **D)**	I will check Scanner and Printer and input 23 students.	The output will be: LearnCentre B17 (or B28) B28 (or B17) C6 J2 A6 (or H7) H7 (or A6) J1	Success	0:00
7	Bestroom.php	To test that the best room algorithm generates a list of rooms in the same order as the prototype python code. (Appendix - part **D)**	I will check Interactive whiteboard and input 15 students.	The output will be: C6 (or H7) H7 (or C6) A6 (or J1) J1 (or A6) B17 J2 B28 LearnCentre	Success	0:11
8	Bestroom.php	To test that the best room algorithm generates a list of rooms in the same order as the prototype python code. (Appendix - part **D)**	I will check Whiteboard and Printer and input 19 students.	The output will be: A6 J2 B28 H7 (or J1) J1 (or H7) B17 (or C6) C6 (or B17) LearnCentre	Success	0:22

9	Bestroom.php	To test that the best room algorithm generates a list of rooms in the same order as the prototype python code. (Appendix - part **D)**	I will check Scanner, printer, whiteboard and interactive whiteboard and input 16 students.	The output will be: H7 B28 (or C6) C6 (or B28) A6 J2 J1 B17 LearnCentre	Success	0:33
10	Bestroom.php --->Table.php	To test that the buttons generated after using the best room finder will direct the user to next page with the correct table.	I will click on the button ‘B17’ that the algorithm generated.	The user will be directed to the Table.php page and the table should have B17 in the table heading with the correct table layout.	Success	0:46
11	Bestroom.php --->Table.php	To test that the buttons generated after using the best room finder will direct the user to next page with the correct table.	I will click on the button ‘J1’ that the algorithm generated.	The user will be directed to the Table.php page and the table should have J1 in the table heading with the correct table layout.	Success	0:58
12	Bestroom.php --->Table.php	To test that clicking one of the buttons under ‘Direct Booking’ with room values will direct the user to next page with the correct table.	I will click on the button ‘B28’ under ‘Direct Booking’.	The user will be directed to the Table.php page and the table should have B28 in the table heading with the correct table layout.	Success	1:08
13	Bestroom.php --->Table.php	To test that clicking one of the buttons under ‘Direct Booking’ with room values will direct the user to next page with the correct table.	I will click on the button ‘LearnCentre’ under ‘Direct Booking’.	The user will be directed to the Table.php page and the table should have ‘LearnCenter’ in the table heading with the correct table layout.	Success	1:15

Tests 14-19 include:

- Uploading a file.
- Parsing through the xml data correctly.
- Updating the database to contain the file data.

Video link to view tests: https://youtu.be/bO-jdz_zJec

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
14	UploadForm.php --->FileUpload.php --->AdminData.php --->Bestroom.php	To test whether the algorithm to read imported files can successfully book the rooms with the correct details. It must be able to book for the correct time frame (two weeks).	Starting with a blank timetable, I will select the rooms_all.xml file and use a starting date of 4/10/2017 and an end date of 4/21/2017. Then I will submit and check the timetable.	The data in the table when selecting B17 should match the table provided by Mr Hamilton. The timetable should only have dates from 4/10/2017 to 4/21/2017.	Success	0:00 (Checking timetable at 0:32)
15	UploadForm.php --->FileUpload.php --->AdminData.php --->Bestroom.php	To test whether the algorithm to read imported files can successfully book the rooms with the correct details. It must be able to book for the correct time frame (three days).	Starting with a blank timetable, I will select the rooms_all.xml file with a starting date of 4/12/2017 and an end date of 4/14/2017. Then I will submit and check the timetable.	The data in the table when selecting B28 should match the table provided by Mr Hamilton. The timetable should only have dates from 4/12/2017 to 4/14/2017.	Success	0:50 (Checking timetable at 1:09)
16	UploadForm.php --->FileUpload.php --->AdminData.php --->Bestroom.php	To test whether the algorithm to read imported files can successfully book the rooms with the correct details. In addition, It must be able to book for the correct time frame (three weeks).	Starting with a blank timetable, I will select the j1room.xml file with a starting date of 4/3/2017 and an end date of 4/21/2017. Then I will submit and check the timetable.	The data in the table when selecting J1 should match the table provided by Mr Hamilton. The timetable should only have dates from 4/3/2017 to 4/21/2017.	Success	1:26 (Checking timetable at 1:53)

17	UploadForm.php	To test whether not inserting a date value into the boxes will prompt the admin to do so.	I will enter a correct file, but leave the two date values blank, then press submit.	The program should stop and tell the user to insert the data first.	Success	2:09
18	UploadForm.php	To test whether attempting to upload a file without entering the file name will prompt the admin to do so.	I will enter correct dates, but leave the file input blank, then press submit.	The program should stop and tell the user to insert the data first.	Success	2:18
19	UploadForm.php --->FileUpload.php	To test whether attempting to upload a file but using a non xml file as the input will stop the insert algorithm and return the user to the upload page.	I will enter correct dates, but also enter a file with a filetype py (python)	The program should return me back to UploadForm.php and no attempt to upload the file should be made.	Success	2:28

Tests 20-27 include:

- Changing/ updating table preferences (Date/Room).
- Using the submit buttons correctly.
- Transferring from Table.php to View.php via hyperlinks.

Video link to view tests: https://youtu.be/3lkO3cVSlHA

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
20	Table.php	To test whether the user can change the room that the given timetable applies to.	Starting with table 'A6' I will use the drop down menu and select 'J2', then I will select the 'change room' button.	The table should change the heading to include J2, and have relevant booking information.	Success	0:00
21	Table.php	To test whether the user can change the date that the given timetable applies to when selecting a school day (Monday to Friday).	I will use the calendar function and select 3/29/2017 (a Wednesday), then I will select the 'change date' button.	The table should change to have the a monday date of 3/27/2017 and a friday date of 3/31/2017. No potential bookings should appear as no bookings should be made for this time.	Success	0:12
22		To test whether the user can change the date that the given timetable applies to when selecting a weekend day (Saturday or Sunday).	I will use the calendar function and select 6/4/2017 (4th June - a Sunday), then I will select the 'change date' button.	The table should change to have the a monday date of 6/5/2017 and a friday date of 6/9/2017. No potential bookings should appear as no bookings should be made for this time.	Success	0:26
23	Table.php	To test whether attempting to change the date of the timetable while inserting an irrelevant date (before 6th March 2017) will not change the timetable.	I will use the calendar function and select 1/6/2017 (6th January 2017) then I will select the 'change date' button.	The table should tell me that this date cannot be used and will not change the values within the timetable.	Success	0:42

24	Table.php	To test whether the user can proceed one week in the timetable using the right shortcut.	While on Table.php I will select the right shortcut button ('>>>>>') twice.	The table should advance two weeks with the correct heading values. Any bookings should maintain the correct dates.	Success	0:57
25	Table.php	To test whether the user can go one week backwards in the timetable using the left shortcut.	While on Table.php I will select the left shortcut button ('<<<<<') one time.	The table should go back three weeks with the correct heading values. Any bookings should maintain the correct dates.	Success	1:08
26	Table.php --->View.php	To test that selecting an empty slot in the timetable will direct the user appropriately (take them to the View.php with the correct data).	I will select any empty cell whilst on the table with room 'B28'.	I should be redirected to the View.php page and it should display data about 'B28', as well as informing me that the room is not booked for the selected time.	Success	1:15
27	Table.php --->View.php	To test that selecting a booked slot in the timetable will direct the user appropriately (take them to the View.php with the correct data).	I will select any booked cell whilst on the table with room 'B28'.	I should be redirected to the View.php page and it should display data about 'B28', as well as informing me that the room is booked for the selected time.	Success	1:28

Tests 28-34 include:

- Attempting to book a room.
- Attempting to delete a booking.
- Using the submit buttons correctly for the view.php page.

Video link to view tests: https://youtu.be/0apTALadalA

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
28	View.php --->Table.php	To test that an unused room can be booked by a user.	Signed in as 'D M Rooney', I will select an empty room, then I will select the 'book room' button.	I should be redirected back to the timetable page and I should be able to see that the selected cell is now booked under 'D M Rooney'.	Success	0:00
29	View.php	To test that a room that is already booked cannot be booked again.	Signed in as 'D M Rooney', I will select a booked room under 'R G Milne', then I will select the 'book room' button.	I should remain on the same page and no booking should be made. It should inform me that the room is already booked.	Success	0:20
30	View.php --->Table.php	To test that a room that is already booked can be unbooked by the teacher who created the booking.	Signed in as 'R G Milne', I will select a booked room under 'R G Milne', then I will select the 'delete booking' button.	I should be redirected back to the timetable page and be able to see the unbooked cell is now empty.	Success	0:34
31	View.php	To test that a room that is already booked cannot be unbooked by a teacher who did not create the booking.	Signed in as 'R G Milne', I will select a booked room under 'J A Hamilton', then I will select the 'delete booking' button.	I should remain on the same page and no deleting of a booking should be made. It should inform me that I did not create the booking.	Success	0:53
32	View.php --->Table.php	To test that any already booked room can be unbooked by an admin.	Signed in as 'N J King', I will select a booked room under 'J A Hamilton', then I will select the 'delete	I should be redirected back to the timetable page and be able to see the unbooked cell is now empty.	Success	1:06

			booking' button.			
33	View.php --->Table.php	To test the the user can return to the previous page without booking or deleting a booking.	I will select any cell on room 'J2' with week beginning 4/17/2017 the 'return to table' button on View.php page.	I should be redirected back to the timetable page and remain on the week beginning 4/17/2107 with room 'J2'.	Success	1:23
34	View.php	To test that attempting to remove a booking when the room is already empty should not attempt a change in the timetable.	With an unbooked room, I will select the 'remove booking' button.	I should remain on the same page and no change should be made.	Success	1:40

Tests 35-40 include:

- Successfully signing up a teacher.
- Unsuccessfully signing up a teacher.
- Checking if required fields have been entered.

Video link to view tests: https://youtu.be/zqKJBRG-0Ik

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
35	Signup.php	To test whether the admin can add a new teacher to the system. (Unique username and password).	I will enter 'C Gratton' as the username and 'testpassword' as the password.	I should be directed to the Bestroom.php page. Once I attempt to login with the username 'C Gratton' and password 'testpassword' then I should be able to enter the program.	Success	0:00
36	Signup.php	To test whether entering a username that is being used will prevent the admin from creating a new account.	I will enter 'J A Hamilton' as the username and 'test' as the password.	The output 'Username already in use.' should appear and no sign up should be made.	Success	0:34
37	Signup.php	To test whether entering a username	I will enter 'Nathan J King'	The output 'Invalid input for username.'	Success	0:46

		that that does not match the regular expression will prevent the admin from creating an account.	as the username and ‘test’ as the password.	should appear and no sign up should be made.		
38	Signup.php	To test whether entering a username that does not match the regular expression will prevent the admin from creating an account.	I will enter ‘Computingaccount’ as the username and ‘test’ as the password.	The output ‘Invalid input for username.’ should appear and no sign up should be made.	Success	0:58
39	Signup.php	To test whether entering a username that does match the regular expression but does not enter a password will prevent the user from creating an account.	I will enter ‘R A King’ as the username and leave the password box empty, then submit.	No sign up should be made and I should be prompted to input the password.	Success	1:12
40	Signup.php	To test whether not entering the username but entering a password will prevent the user from creating an account.	I will leave the username box empty and enter ‘test’ as the password, then submit.	No sign up should be made and I should be prompted to input the username.	Success	1:24

Tests 40-43:

- Updating the database/ deleting old bookings.
- Another attempt of signing out.
- Attempting to enter the system without logging in (with given links).

Video link to view tests: https://youtu.be/tsSLXHhkh04

Test No.	Page/Link	Purpose	Test	Expected Result	Result	Video Timestamp
41	Update.php	To test whether the admin can delete out of date bookings.	Starting with a filled timetable, I will enter a date of 4/13/2017, then submit.	I should be redirected back to the Bestroom.php page. When checking the timetable I should see that all bookings up until 4/13/2017 should be deleted.	Success	0:00

42	Logout.php	Another test to see that users can log out.	I will select the logout button while on Table.php.	I should be directed to Login.php and should have to re-enter my login details in order to enter the system again.	Success	0:25
43	Bestroom.php	To test that only an admin can access the signup page, the upload a file page, and the update database page.	I will login as an admin and select the three buttons on the Bestroom.php page. I will then login as a normal user and select the three buttons on the Bestroom.php page.	As an admin, I should successfully be directed to the pages for the buttons pressed. As a normal user, I should not be allowed to be directed to the three pages, I should remain on the current page.	Success	**Admin test** - 0:36 **Normal user test** - 0:58
44	All pages	To test that attempting to enter the system without successfully logging in will direct the user to the login page.	I will run through my pages listed and select each of them relating to my project. However I will not have signed in first.	They should direct me to the login page.	Success	1:27

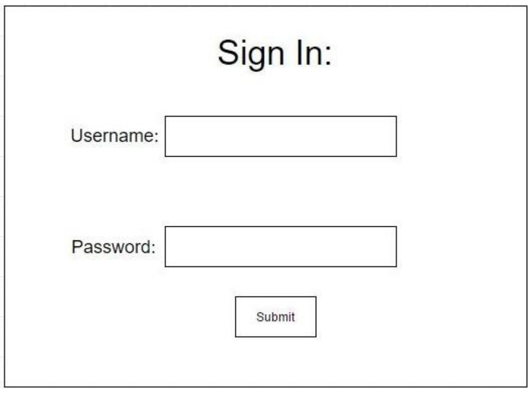
An Example of the System in Use

The following video is a final representation of my project, putting forward a random number of previous tests. This will also demonstrate my submit/return buttons that move the user through different pages. It represents an example of how a teacher could use my program when deciding to book computer rooms.
Video link: https://youtu.be/2wdg4Nx_P5E

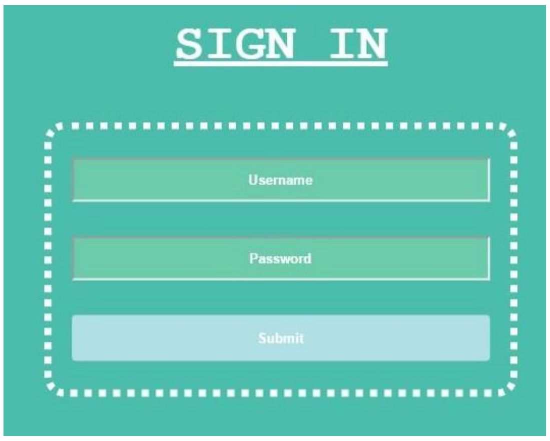
Appraisal and Evaluation

Overview

When designing my system, I approached each function in different ways and started at a low level plan so that I could build up a final solution. My use of flowcharts was very useful throughout this project as I was able to identify and group smaller components to achieve a larger goal. In some cases, I would begin with a flow chart and then proceed to creating pseudo code or even a prototype. This was highly beneficial and proved to be useful when coding my final solution as I often was only required to translate what I had already created onto my project. The user interface design was very useful when creating my final solution, due to the level of detail I went to. This gave me a much better understanding of what I needed to include to get the outcome I wanted. The descriptions given to each part of the UI were convenient as they reminded me of where each function needed to be placed.
The sign in page created in my design phase:



The sign in page created in my final solution:



Section A - Objectives

Objective 1: Be a Web Based Program that can be accessed in and outside of school.
The program is accessed on a website and therefore can be used outside of school, therefore this objective has been met.

Part 1.1: Only teachers and admins can use the program.
Unauthorised users are unable to create an account, and therefore cannot get passed the login screen. Only admins can create accounts and as a result this part has been implemented.

Part 1.2: Cascading style sheets (css) will be used to organise the layout clearly.
CSS has been implemented within the html code and the user interface design has been closely followed.This part has been implemented. Css did however take up a large amount of time and therefore may not have been the best method to create the required user interface.

Objective 2: Use logins to access the system.
A login system for teachers was created and test number **(1)** demonstrates it in use. Therefore this objective has been met.

Part 2.1: Login details will be built up of the teacher's name and a given password.
This part has been implemented. The user must enter their username and password to enter the system. This is a standard security measure that I believe worked successful users

Part 2.2: Login details will be stored in a database.
Throughout the technical solution there are examples of connecting to the database in order to obtain, insert or delete data. Therefore this part has been implemented.

Part 2.3: The passwords will be encoded (MD5).
After comparing the passwords stored in the database with the inputs from test (**35**) it is clear that this part has been implemented as no errors occur when logging in. An example of the hashed passwords can be found in the appendix - part **E**. Although this was successful and adds security to my system, the MD5 algorithm can create collisions when the same value is generated for two different inputs. This could suggest that an alternative method could be a much better way of storing passwords such as another hash function SHA-512.

Part 2.4:The admin will be required to sign up teachers through the program and they will have to pass on the details.
Test number **(43)** demonstrates that this part has been implemented. I believe that this method is better than the teacher's creating their own individual account because there is a possibility that unauthorised users can create an account and cause complications within the system. In addition, this is a similar method to what schools do to most systems with security, they generate my login details and pass them on to me.

Part 2.5: The username information will have to match the data from the school database (SIMS).
The use of regular expressions to regulate the usernames input, (shown with test number **(35)**) has demonstrated that this part has been implemented. This was a successful method as it was very quick to create and only allows data that matches the SIMS layout. However, there is still a possibility that the username does not match the SIMS data. This may indicate that a better method could be to directly search for the username inside of SIMS so that it can be imported onto my system. This would make sure that there are no differences in names.

Objective 3: Use a relational database that links directly the program.
This objective has been met. The database was implemented early into the development of this project and allows me to store relevant information to make the system work.

Rooms Table:
Part 3.1.1: It will store the resources that each computer room contains and the number of computers.
This has been implemented.

Bookings Table:
Part 3.2.1: It will store whether or not a room is booked.
This has been implemented.

Part 3.2.2: It will store the date and time of when the room is booked for.
This has been implemented.

Part 3.2.3: It will store who has booked the room.
This has been implemented.

Teachers Table:
Part 3.3.1: It will store the login details of each teacher.
This has been implemented.

Part 3.3.2: It will contain data to determine if they are an admin or not.
This has been implemented.

Lesson Times Table:
Part 3.4.1: It will store the times of each lesson time.
This has been implemented.

Part 3.4.1: Friday will have different lesson times to the rest of the week.
This has been implemented.

Objective 4: Allow teachers to find the most suitable room to book.
Using algorithm **(2)** during my design phase and with tests **(6)** , **(7)**, **(8)**, **(9)** there is evidence that this objective has been met.
Using associative arrays to store the values and names of each room was useful in my technical solution as it allowed me to use the score as an index to identify the next room name in the final output. This was a much faster method than having two separate lists and identifying the index individually, which I did in my prototype.

Part 4.1: A list of the rooms from most suitable to least suitable will be displayed.
Tests **(6)** , **(7)**, **(8)** and **(9)** demonstrate that this has been implemented.

Part 4.2: Filters will be used such as check boxes to determine requirements.
The boxes are defined within the technical solution and are essential to obtaining information on the room requirements. This part has been implemented.

Part 4.3: The algorithm to find the best room will generate scores, which will be displayed to give an idea of the difference between the rooms.
This part has been implemented, demonstrated by tests **(6)** , **(7)**, **(8)** and **(9)**.

Part 4.4: Student count will be input as an integer.
The technical solution includes html code which generates an input box that only allows the user to enter an integer. This demonstrates that this has been implemented.

Part 4.5: Filters will include: Printer(s), whiteboard, interactive whiteboard and scanner.
Using algorithm **(2)**, the technical solution contains check boxes that have the labels Printer(s), whiteboard, interactive whiteboard and scanner. This demonstrates that this part has been implemented.

Part 4.6: The filters and student count value will be compared to the room data within the database to determine the score.
Algorithm **(2)** which was created within the design phase compared the user inputs with the data for stored for each room. This has allowed for the success of tests **(6)** , **(7)**, **(8)** and **(9)** and shows that this has been implemented.

Objective 5: Have a timetable of each computer room in the school, for the user to see.
The screenshots shown within the evaluation below as well as tests **(20-27)** demonstrate that this objective has been met and the user is able to view the timetable of each computer room within the school.

Part 5.1: The timetable will appear on the current week when first entering the page.
Test **(20)** demonstrates that when selecting a room, the system takes the user to the timetable for the current week. Therefore this part has been implemented.

Part 5.2: Each cell will include the user who booked the room.

Test **(20)**, and the screenshot from the evaluation below shows that each cell within the timetable includes the username of the teacher. When a booking is deleted during this time, the cell will update accordingly by removing the username. Therefore, this has been implemented.

Part 5.3: If a room is not booked it will contain ‘empty’.
Test **(20)**, and the screenshot from the evaluation below shows that each empty cell within the timetable includes the value ‘empty’. When a booking is created for this time, the cell will update accordingly by adding the associated username (to the creator of the booking). Therefore, this has been implemented.

Part 5.4: The week the timetable is on can be changed with ‘skip’ buttons that immediately skip to the next or previous week.
Tests **(24)** demonstrates that when the ‘right’ skip button is selected it will change the timetable to have the date for the following week. Test **(25)** demonstrates that when the ‘left’ skip button is selected it will change the timetable to have the date for the previous week. Therefore, this part has been implemented.

Part 5.5: The user can also input a date they would like to view. The timetable will still be from monday to friday, but the input date will be contained within it.
This has been implemented. Tests **(21)** and **(22)** demonstrate the timetable changing when the user inputs a different date than the current week.

Part 5.6: When entering a date, they can also use a calendar to click the day they want.
Tests **(21)** and **(22)** show the calendar function in use and demonstrate that this has been implemented.

Part 5.7: The timetable being viewed can have its room name changed with a drop down menu.
Test **(20)** show that this part has been implemented. Therefore the user is able to change the room when selecting a different room within the menu.

Objective 6: Allow for all users to see a booking and it’s information
Tests **(28-34)** show the ‘viewing’ page where the user is able to see a booking and all it’s relevant information. Therefore this objective has been met.

Part 6.1: The user will click the cell within the timetable using hyperlinks (they can select it even if it’s empty) and will be taken to a page that displays data about the room.
This has been implemented, as demonstrated by tests **(26)** and **(27)**

Part 6.2: One half of the page will display data about the room itself, such as the resources it includes and the number of computers.
Using the UI for the ‘viewing’ page created during the design phase, I have been able to closely follow the structure to implement this part. This is shown throughout tests **(28-34)**, where the ‘viewing’ page is shown.

Part 6.3: The other half of the page will display data about its booking, such as the time, date and lesson number. It will also tell the user if it is booked or not.
Similarly to the previous part of this objective, tests **(28-34)** display the 'viewing' page and demonstrates the other half of the page. Tests **(26)** and **(27)** indicate that the system tells the user whether the room is booked or not.

Objective 7: Allow for the booking of a computer room.
Using algorithm **(4)**, I have created a function within the system that will allow any user to book an empty room. This is demonstrated within test **(28)** and shows that this objective has been met.

Part 7.1: Any user can book an empty room by clicking 'book', possibly on the 'viewing' page (after clicking the cell in the timetable).
This has been implemented, as shown by test **(28)**.

Part 7.2: If the room is already booked, and a user attempts to book the room again (with the same date and time), they will be told that the room cannot be booked again and no booking will be made.
Test **(29)** shows that if a teacher attempts to book a room at a time where a booking is already present, then the system will inform the user that the room is booked. Therefore, as double bookings are being prevented, this part has been implemented.

Part 7.3: If the room is not booked, and a user selects 'book', they will be brought back to the timetable (with the same room name and week beginning). In addition the database will be updated to contain this booking.
When a successful booking takes place, the user is then redirected back into the timetable where they are able to see the booking they have just made. This is demonstrated in test **(28)** and shows that this part has been implemented.

Part 7.4: After a successful booking, the cell will go from containing 'empty' to including the username of the user who made the booking.
Test **(28)** demonstrates the changing of the cell value after a successful booking is made. Therefore this has been implemented.

Objective 8: Allow for booked rooms to be cancelled.
Using algorithm **(5)** during the design phase, the system allows a booking to be deleted when certain criteria is met. Tests **(30-32)** demonstrate the cancellation system in use, and show that this objective has been met.

Part 8.1: Cancellations can only be made by the user who booked the room or by an admin. An admin can cancel any booking.
Test **(30)** indicates that a user who created the booking has the ability to cancel it. Test **(31)** indicates that a user who did not create the booking cannot cancel it. Test **(32)** indicates that an admin is able to delete any booking. Therefore this part has been implemented.

Part 8.2: Cancellations will be chosen by selecting the booking (on the timetable) and clicking the 'remove' button.
Tests **(30-32)** demonstrate the whole cancellation progress when a cell is selected from the desired room. The user must select the 'remove' button in order to cancel, which indicates that this has been implemented.

Part 8.3: The database will be updated to remove the booking information completely, and its booking status will be 'unbooked'.
Within the technical solution for 'view.php', there is use of the mysql language which deletes data from the relevant table to remove the booking completely. This part has been implemented.

Objective 9: Allow for a teacher to be signed up to the system.
Using algorithm **(8)**, I have been successful in created a function that is able to add new users onto the system. Tests **(35-40)** demonstrate the signing up function in use and indicate that this objective has been met.

Part 9.1: Only the admin will be able to perform this task.
Test **(43)** demonstrates that only an admin user is able to select the submit button that transfers them onto the page where they are able to sign up teachers. Therefore this has been implemented.

Part 9.2: Regular expression will be used to only allow for a username in the format described in the analysis.
Using some of the design discussed within algorithm **(8)**, I have created a regular expression that is included within the technical solution. Therefore, only inputs specified by the expression are valid, so this part has been implemented. An evaluation of the regular expression was made in part 2.5 of the objectives appraisal.

Part 9.3: A password of any length greater than zero can be input.
Test **(35)** demonstrates a successful sign up, whereas test **(39)** demonstrates an unsuccessful sign up because no password was input. Therefore this has been implemented as not entering any characters for the password resulted in the system informing the user that an error has been made.

Part 9.4: If the username already exists, it will inform the admin and the account will not be created.
Test **(36)** demonstrates this successfully as an existing username is used for the input. The user is therefore informed about the error and will have to try a new input. This has therefore been implemented.

Part 9.5: If no password is input, it will inform the admin and the account will not be created. This has been implemented, as discussed during Part 9.3.

Part 9.6: If the username does not match the regular expression it will inform the admin and the account will not be created.

Tests **(37-38)** demonstrate the regular expression in use. As the inputs do not match the expression, the test accounts are not created. This shows that this part has been implemented.

<u>Objective 10:</u> Allow for bookings to be deleted in large numbers.
Using algorithm **(6)**, I have created a function that is able to delete bookings up to a desired date. This is demonstrated within test **(41)** which deletes bookings up to 4/13/2017. Therefore this objective has been met.

<u>Part 10.1:</u> Only the admin will be able to perform this task.
Test **(43)** demonstrates that only an admin user is able to select the submit button that transfers them onto the page where they are able to delete many bookings at once. Therefore this has been implemented.

<u>Part 10.2:</u> The admin will select a date that all booking data within the database will be deleted up to.
Test **(41)** demonstrates the user selecting the date they would like, which indicates that this part has been implemented.

<u>Objective 11:</u> Allow for the importing of xml code to determine mandatory bookings.
Using algorithm **(7)**, I have created a function that is able to read through imported xml files (such as those provided by Mr Hamilton). During the reading of the file, booking data is inserted into the database when specified by the user. Tests **(14-19)** demonstrate this part of the system in use, and indicate that this objective has been met. After researching Xml parsers, I concluded that I would use simpleXml to read through the data. This was a successful. However, as the file size was quite large there were some performance issues as it took a while to read the file repeatedly when booking for greater than two months. This suggests that an attempt at an alternative method may be an improvement, such as the event based parser, Xml reader, which reads faster and consumes less memory (as each node is thrown away).

<u>Part 11.1:</u> Only the admin will be able to perform this task.
Test **(43)** demonstrates that only an admin user is able to select the submit button that transfers them onto the page where they are able to upload a file. Therefore this has been implemented.

<u>Part 11.2:</u> The admin will be able to select the file they require with a function.
The technical solution includes php code which allows the user to submit from within their saved files. This indicates that this part has been implemented.

<u>Part 11.3:</u> The file will be checked to make sure it is an xml file (if it isn’t the process will not proceed).
The technical solution includes php code which takes the type of the file that was imported, and either proceeds or stops the function. Test **(19)** demonstrates that this has been implemented.

Part 11.4: An algorithm will be used to insert bookings into the database.
Algorithm **(7)** created within the design phase was used within the technical solution and demonstrates that this part has been implemented.

Part 11.5: The algorithm to insert the data will ignore break/lunch/registration times and will only use lesson times.
The comments within the technical solution for reading the file reference where break/lunch/registration times are being ignored as they are not relevant to this system. Therefore this has been implemented.

Part 11.6: The algorithm will work for a two week timetable and a one week timetable. It will also work with multiple rooms in the file.
Tests **(14)** and **(16)** demonstrate that the algorithm can be used for all rooms , or just a single room (J1). Test **(15)** shows the algorithm working for a much shorter time frame.Therefore this has been implemented.

Section B - Feedback

Obtaining Feedback from the Teachers:

To get feedback from the teachers, I gave them some time to spend with the system using their own accounts I created for them. They each filled in a short questionnaire (shown in the appendix - part **F**). After the teachers filled out their questionnaire, I had a short discussion with them to ask for any other feedback that they thought would be useful. Due to time constraints, only four teachers could be used, however they offered a substantial number of feedback.

Analysis of the Results from the Questionnaires:

All of the the teachers ticked yes to whether the system allowed them to sign in with their given username and password. This indicates that the login system is successful in allowing a teacher to enter the program which therefore creates a higher level of security as discussed within my analysis phase. When creating this system. security was a very important factor because if anyone could access the it then it would be much more difficult to manage and maintain, due to the potential of unauthorised people booking at unnecessary times.

All of the teachers ticked yes to whether the system allowed them to create a booking. In addition, all of the teachers ticked yes to whether the system allowed them to remove a booking they created/own. This indicates that each teacher has the ability to use the system for its core purpose, booking computer rooms. For the school’s current system, individual teachers must speak to the admin in order to book a room. This was time consuming and as my system allows all teachers to create reservations on their own, this is a huge improvement in time management due to the increase in simplicity. Now, all users will benefit as they will not be spending as much time selecting and bookings a computer room.

All of the teachers ticked yes to whether the system allowed them to find the most suitable room for their needs. However there are ways to improve this that will be referenced in the analysis of feedback from the discussion.

All of the teachers ticked no to being an admin, and they also all ticked no to being able to access the admin features. This demonstrates that teacher accounts are not able to access admin features, which was important during my analysis

All of the teachers agree that this system is fully functioning and they would be happy to use it in the current school environment.

Analysis of Feedback from the Discussion:

All of the teachers commented about the layout of the system saying that it was very well structured and was very easy to get a grasp of. In addition, they liked using the timetable to interact and direct them to each booking. This suggests that having the timetable as an individual page is beneficial as it does not overcomplicate the system. If most of the functions were to be included on the same page as the timetable then it is likely that the system will feel too convoluted, as they are given too many options at a single time.

They thought that being able to adjust the timetable date with a single button was a useful tool that would make booking much faster. Furthermore, one teacher suggested that they would most likely use the button more than the calendar function to select the week they want.
For booking a computer room, all of the teachers were happy that selecting a cell would take them to another page with more detail about the room and potential booking. However one teacher made a suggestion to make a tag appear when hovering over the cell which provides further information. The teacher did not specify what information would be on the tag, but it could be assumed that it includes a faster way to book or some room information.

There were mixed responses when using the function to generate a list of the most suitable rooms. Two of the teachers identified that the score allocated to each room could be used better, but understood why it was included. One teacher commented that although it was useful to be given a list of the best rooms, there was no true way of knowing if the algorithm works to generate the room they actually want, as someone may favour the number of computers over resources.
Another teacher suggested that it would be easier to see the resources of the rooms on the same page so that they can make their own decision.
It is clear that the output of the best room algorithm could be modified to focus on the top of the list (only the best room) as the rest of the list will be overlooked if it does not match the user’s needs. The score generated was not that helpful and was only used to indicated differences between rooms (incase they have the same value), which suggests that it may be more appropriate to display the resources of the best room generated so that the user can decide if the algorithm has given them what they wanted.

They also commented that after the list of the rooms was generated, they did not need to see the rooms with the lowest scores. This is because rooms at the bottom of the list were never going to be selected, and if they were, the option to book any room is on the other side of the screen (direct booking section).

Obtaining Feedback from my Customer:

To get feedback from Mr Rutter, I showed him through the program, informing him of the different tasks he could perform as an admin. He then had his own time to spend with the system and filled in the same questionnaire that was given to each of the teachers (shown in the appendix - part **F**). Afterwards, I had a discussion with him about what works well with the system and what I could do to improve it.

Analysis of Feedback from My Customer:

When showing Mr Rutter the program he was very pleased to see all of the ideas he discussed with me, during our first meeting, being used successfully. When answering the questionnaire, he identified that all of the implemented functions worked correctly and that he had access to all of the admin features. He indicated that the program meets the requirements for a computer room booking system, which ensures that the project has been a success.

When selecting a room to book, he mentioned that the layout was very easy to understand and the overall interface was impressive. In comparison to the current system he believed that this would make bookings much faster and less tedious. However Mr Rutter did suggest that the current system is based off of the two week timetable, so this project would benefit if it could calculate and inform the user of which timetable is week one and which is week two. This would be a nice addition that would specifically be useful when importing files. For example, there could be an implementation where importing a week one timetable will only insert into the database for every week one for the next two months (or any time the user chooses).

When using the import function, he was very impressed with how quick and easy it was. He commented that this will be one of the most useful tools for admins as they have the ability to insert data where bookings have the most priority (such as an ICT lesson requiring B28). In addition, having the ability to state when the bookings begin and end could allow for him to insert all prioritised times at the start of the year. This will therefore be crucial to keeping the system accurate in reflecting the whole school and each lesson.

When signing up a teacher, he was impressed with the implementation of the regular expression to only allow for certain username layouts. He commented that the overall method was helpful for him as an amin and only takes a minute to sign up a teacher which is particularly convenient if, for example, many new teachers begin working at the school in a (new) year. He did mention that in the unlikely event of two users having very similar or almost the same SIMS name, it would be very difficult to manage this in this program. Although unsure of how SIMS gets round this, he suggested that I could implement a user number (unique key) that will identify differences in two similar users when interacting with the system. Another adjustment could be to have two separate usernames for each user, where one relates directly to SIMS and the other is one used to sign in with. Mr Rutter stated that this would be a useful change but not mandatory at there are no problems with the system.

Mr Rutter also commented that only allowing admins to create accounts for new users is practical and does a good job of maintaining security elements. Without it any user could create an account and enter the system which would not only make the overall program less manageable but could also overload the database. With this in mind, he suggested that being able to delete users from the database would help prevent poor database performance in the future as old information can be removed. He did also say that this not a priority as it would take a long time for performance to dip.

When discussing more about database performance, Mr Rutter commented that it was logical to allow admins to delete bookings up to a given date. He was impressed with how quickly it could be done and that it allowed him to not only delete information that is no longer required, but potentially delete everything when the new school year begins.

He was particularly impressed when using the function to find the most suitable rooms, commenting that it is a really useful tool that can often be overlooked. He praised being able to instantly view the list and interact with the rooms after selecting the resources and the number of students. However he did suggest that it could become more useful if there was a way to directly change what the room includes. He gave an example saying that if a computer is out of order on a given day, he could decrease the computer number for that room by one so that the data is more accurate. It can also be assumed that this applies to resources, so if a printer becomes out of service, the database could be updated to reflect the changes. This would be a beneficial change that builds up on the overall usefulness of this system as it becomes more accurate.

Overview of Feedback:

The overall feedback from the questionnaires suggest that the components of the system (such as the login and booking functions) work correctly. In addition, it provides further confirmation that only admins have access to the admin features This indicates that the technical solution is of a good standard and can successfully maintain a working system. The feedback from the discussion was also very positive with many users suggesting how every part of the program was made to be useful and that it heavily surpasses the current system. There were also many parts to the system that could be further expanded upon to benefit the users and improve the overall functionality.

Section C - Extensions

With the feedback given from Mr Rutter and the sample of teachers I am able to determine changes to the system that could be made to refine this project.

As already discussed with Mr Rutter, being able to delete user accounts would be a useful minor change as it will help prevent performance slowdowns and keeps the overall system more manageable as there would be less unnecessary data.

As discussed within my teacher feedback, when finding the best room, some teachers suggested that it was not necessary to see the least suitable classrooms, as they would never be the ones selected. This indicates that it may be more useful to only list the best and most suitable room (or rooms if there are duplicate scores). Displaying the resources next to the best room generated would give the user a better understanding of the score created, as some users suggested that the score was not a very useful tool to relate to the suitability of the room on its own. This will allow them to make their own final judgement if the algorithm has actually given them the best room.

A more advanced extension could allow for teachers to import a timetable/schedule so they can book multiple rooms at once. This could be accessed by all teachers but will not overwrite bookings already created. Therefore, the program will making it much faster for teachers who want to book in large quantities.

Confirmation emails could be implemented to email the user when they have created or deleted a booking. This would require more space within the database to include their email address. Overall this would be useful to remind users of their booking time and date and for which room.

As suggested by Mr Rutter, giving accounts two seperate usernames, one being used to sign in with and the other only being used to identify who owns a bookings could be a useful improvement. Not restricting usernames to the names found in SIMS could give more options to admins when creating accounts as they are not being limited.

Implementing the two week cycle to this system would prevent users getting confused when creating bookings and would also make importing files more convenient (as they can book for specific weeks). The change would also more closely reflect the school timetable. The adjustment would require a calculation to determine the week, so the admin may have to input an example date for week one so that it can be stored within the database and referenced when displaying the timetable.

As discussed, Mr Rutter suggested that having the ability to change room information would create a more accurate system. However, in addition to this, admins could also be able to create or delete rooms all together. This could be used if a computer room is getting redecorated or a new classroom is built with computers.

As a final change, when creating the technical solution, cascading style sheets (css) were crucial to creating a strong user interface that made using the system simple and understandable. However, with little knowledge of css, this took a lot of time as I used w3schools tutorials (shown in the bibliography - **6**) to get the program to appear as designed. In the future, I may aim to use a template css as this would be much faster and would not require nearly as much time when coding. Overall, this would benefit the project as I could focus on more important components such as importing and reading files.

Conclusion

Section A of my evaluation indicates that I have been successful in meeting every objective and implementing all my additional requirements created during the analysis phase. Therefore, I have confidence in the completeness of this project and that it can be used as a fully functioning booking system. The feedback given in section B provides confirmation that teachers would be happy to use the system, and that it heavily surpasses what the school currently uses. It is also clear from the feedback given that the program is very practical and has many useful properties (such as being able to move forward or backwards a week with one button). Moreover, Mr Rutter was very pleased with the outcome of the system and was impressed with the level of detail included. It is evident that the system will make computer room bookings much faster and easier to manage, which was my initial goal. Minor changes could be made to the system in order to increase its usefulness further, however these are very obtainable extensions and can be implemented reasonably.

Bibliography

1. Xml research link:
https://www.w3schools.com/php/php_xml_parsers.asp

2. Database tool provider:
https://www.phpmyadmin.net/

3. Development software/resource:
https://codio.com/

4. Sql research link:
https://www.w3schools.com/sql/

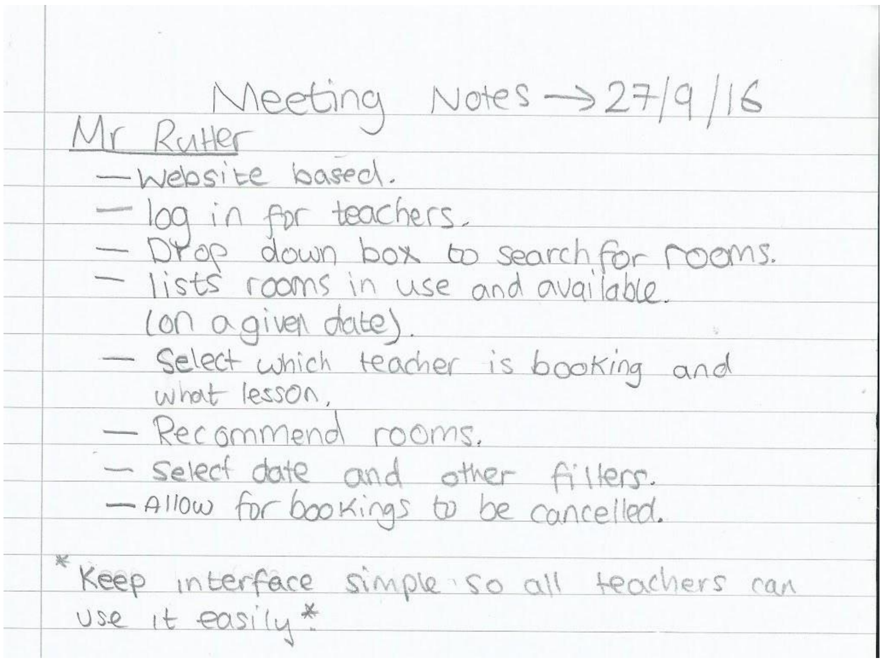
5. Regular Expression Tester
https://regex101.com/

6. Css tutorials
https://www.w3schools.com/css/

Appendix

Part A

Initial notes written down during the discussion with Mr Rutter:



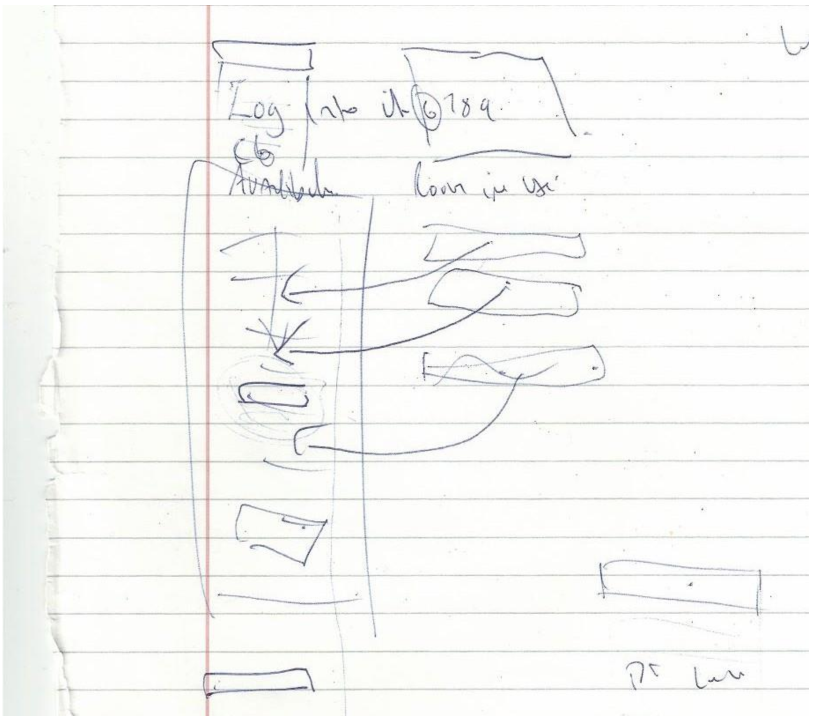
Meeting Notes → 27/9/16

Mr Rutter

- Website based.
- log in for teachers.
- Drop down box to search for rooms.
- lists rooms in use and available. (on a given date).
- Select which teacher is booking and what lesson.
- Recommend rooms.
- Select date and other filters.
- Allow for bookings to be cancelled.

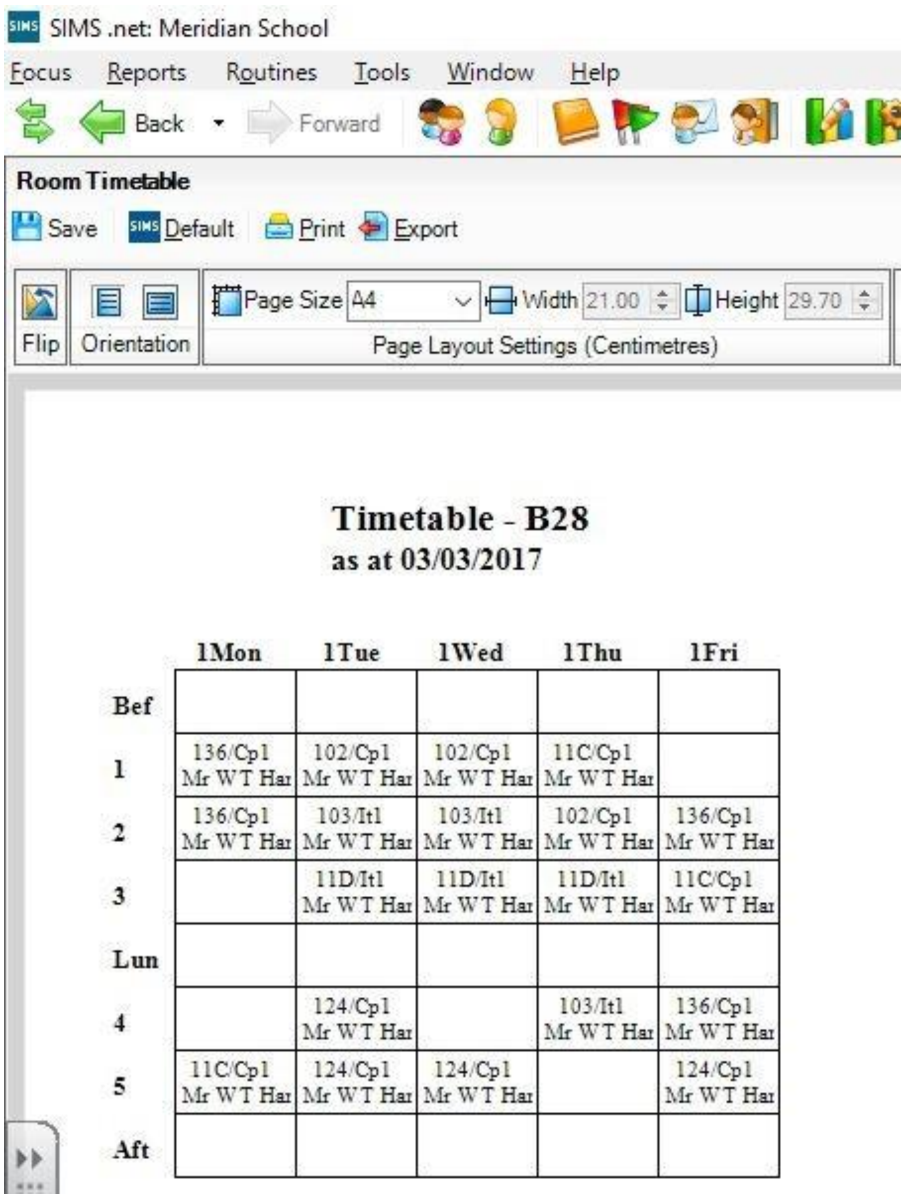
Keep interface simple so all teachers can use it easily

Drawings made by Mr Rutter when describing his ideas during the meeting:



Part B

An example of a teacher's timetable that is generated within the SIMS system. This is the program that generates the xml code that I export onto my system: (The current view is only for week one)

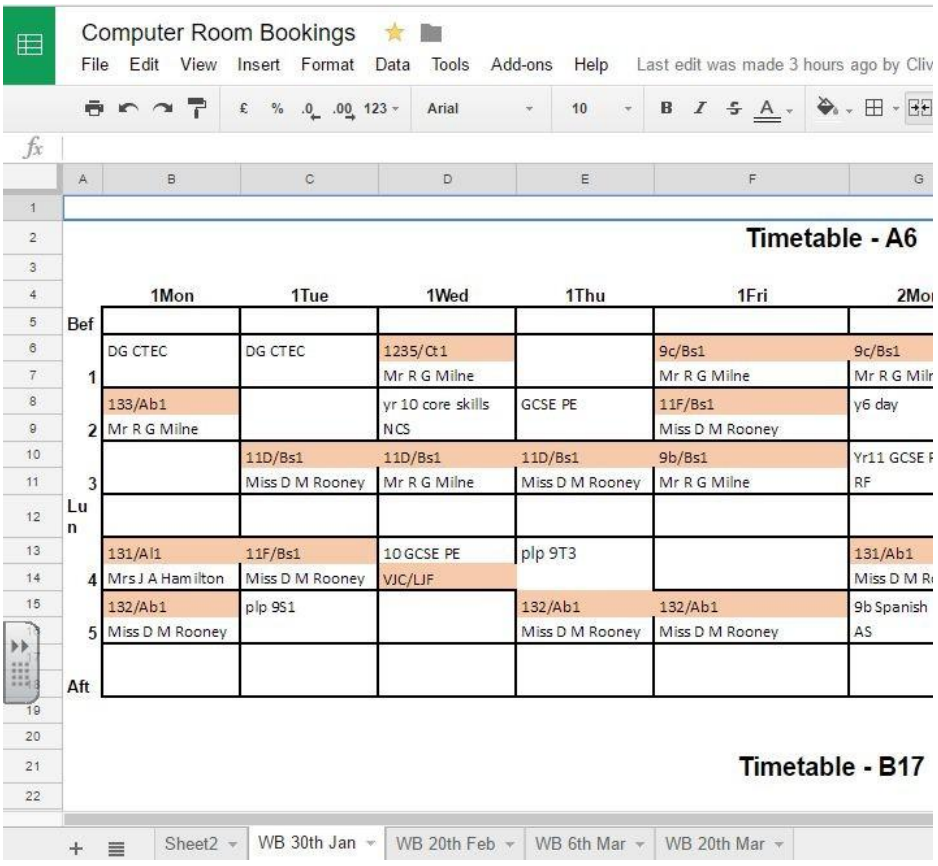

SIMS .net: Meridian School
Focus Reports Routines Tools Window Help
Back Forward
Room Timetable
Save Default Print Export
Flip Orientation
Page Layout Settings (Centimetres)

Page Size A4 Width 21.00 Height 29.70

Timetable - B28
as at 03/03/2017

	1Mon	1Tue	1Wed	1Thu	1Fri
Bef					
1	136/Cp1 Mr W T Har	102/Cp1 Mr W T Har	102/Cp1 Mr W T Har	11C/Cp1 Mr W T Har	
2	136/Cp1 Mr W T Har	103/It1 Mr W T Har	103/It1 Mr W T Har	102/Cp1 Mr W T Har	136/Cp1 Mr W T Har
3		11D/It1 Mr W T Har	11D/It1 Mr W T Har	11D/It1 Mr W T Har	11C/Cp1 Mr W T Har
Lun					
4		124/Cp1 Mr W T Har		103/It1 Mr W T Har	136/Cp1 Mr W T Har
5	11C/Cp1 Mr W T Har	124/Cp1 Mr W T Har	124/Cp1 Mr W T Har		124/Cp1 Mr W T Har
Aft					

An example of the current system layout after the xml code has been implemented (highlighted cells indicate the bookings made by the xml code):

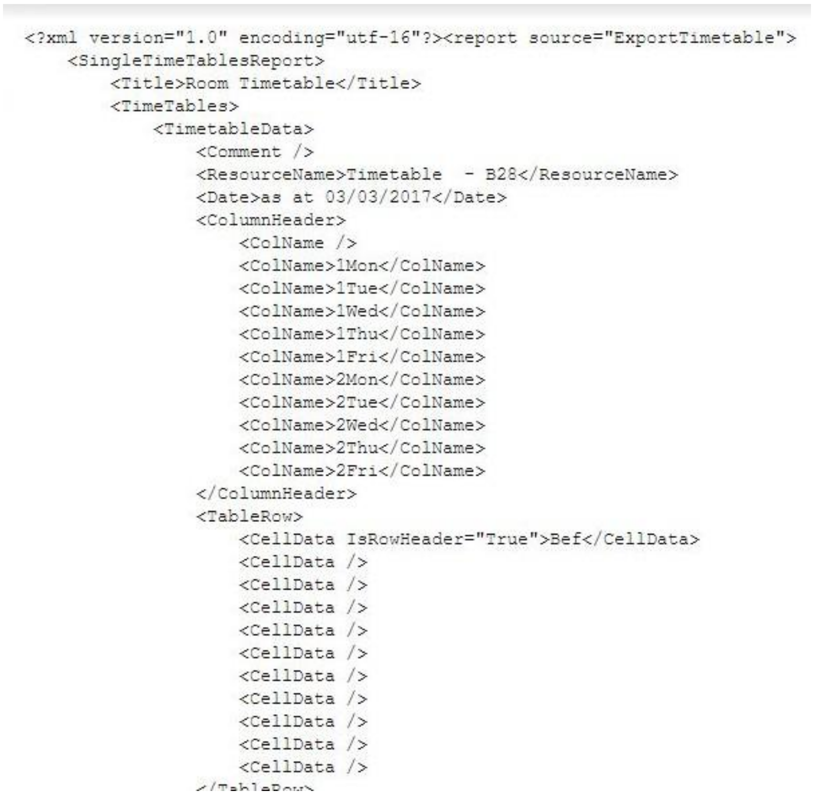

Computer Room Bookings
File Edit View Insert Format Data Tools Add-ons Help
Last edit was made 3 hours ago by Cliv
Arial
10
Sheet2
WB 30th Jan
WB 20th Feb
WB 6th Mar
WB 20th Mar

Timetable - A6

	1Mon	1Tue	1Wed	1Thu	1Fri	2Mo
Bef						
1	DG CTEC	DG CTEC	1235/Ct1 Mr R G Milne		9c/Bs1 Mr R G Milne	9c/Bs1 Mr R G Mil
2	133/Ab1 Mr R G Milne		yr 10 core skills NCS	GCSE PE	11F/Bs1 Miss D M Rooney	y6 day
3		11D/Bs1 Miss D M Rooney	11D/Bs1 Mr R G Milne	11D/Bs1 Miss D M Rooney	9b/Bs1 Mr R G Milne	Yr11 GCSE RF
Lun						
4	131/Al1 Mrs J A Hamilton	11F/Bs1 Miss D M Rooney	10 GCSE PE VJC/LJF	plp 9T3		131/Ab1 Miss D M R
5	132/Ab1 Miss D M Rooney	plp 9S1		132/Ab1 Miss D M Rooney	132/Ab1 Miss D M Rooney	9b Spanish AS
Aft						

Timetable - B17

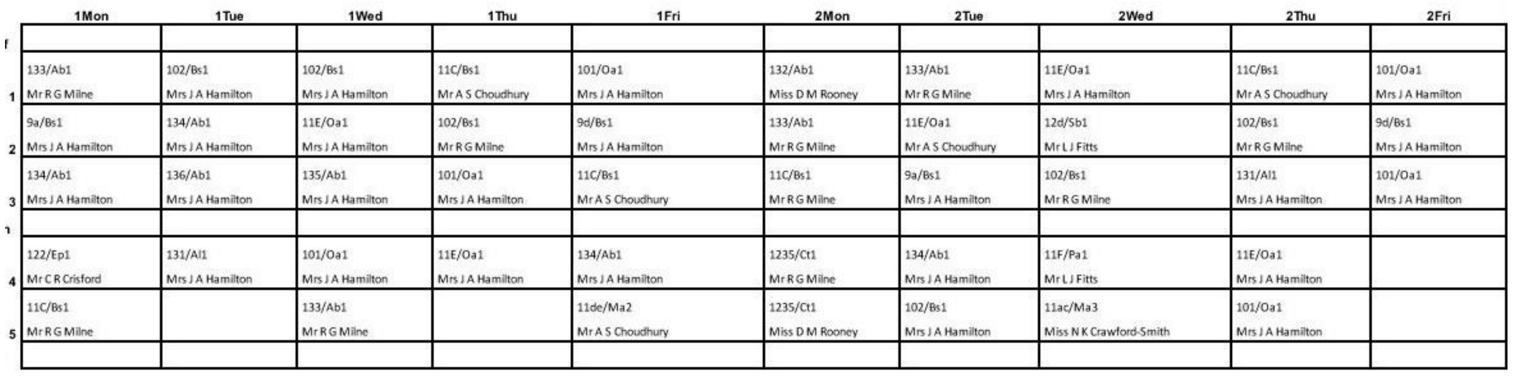
An extract of the xml code provided by Mr Hamilton:



```
<?xml version="1.0" encoding="utf-16"?><report source="ExportTimetable">
    <SingleTimeTablesReport>
        <Title>Room Timetable</Title>
        <TimeTables>
            <TimetableData>
                <Comment />
                <ResourceName>Timetable  - B28</ResourceName>
                <Date>as at 03/03/2017</Date>
                <ColumnHeader>
                    <ColName />
                    <ColName>1Mon</ColName>
                    <ColName>1Tue</ColName>
                    <ColName>1Wed</ColName>
                    <ColName>1Thu</ColName>
                    <ColName>1Fri</ColName>
                    <ColName>2Mon</ColName>
                    <ColName>2Tue</ColName>
                    <ColName>2Wed</ColName>
                    <ColName>2Thu</ColName>
                    <ColName>2Fri</ColName>
                </ColumnHeader>
                <TableRow>
                    <CellData IsRowHeader="True">Bef</CellData>
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                    <CellData />
                </TableRow>
```

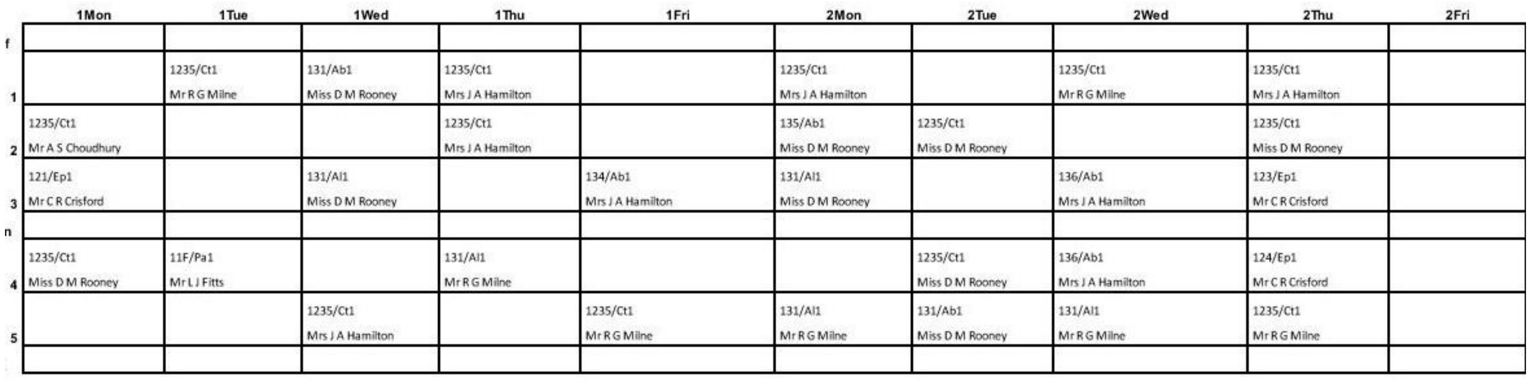
Part C

Booking timetable J1 (provided by Mr Hamilton as excel)



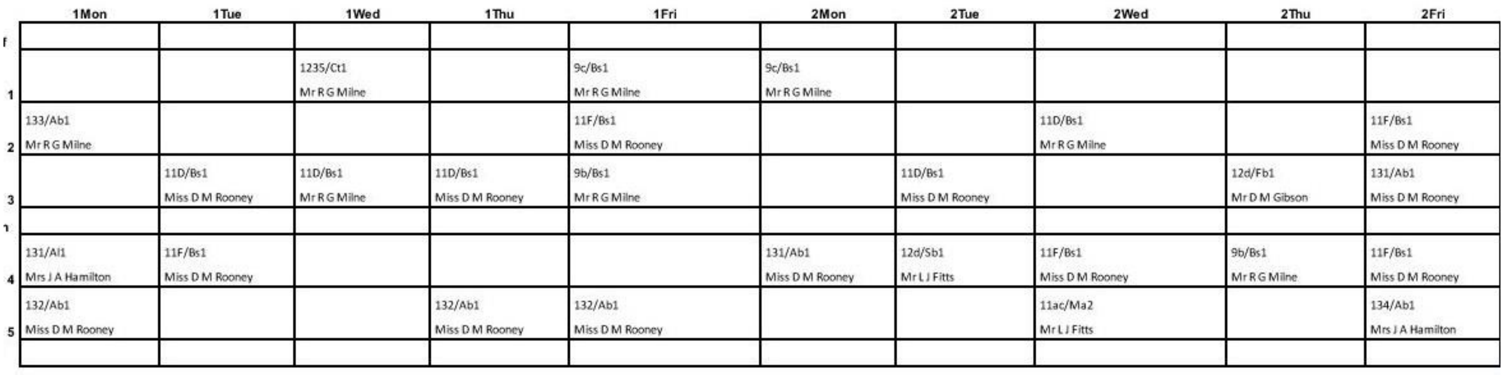
	1Mon	1Tue	1Wed	1Thu	1Fri	2Mon	2Tue	2Wed	2Thu	2Fri
f										
1	133/Ab1 Mr R G Milne	102/Bs1 Mrs J A Hamilton	102/Bs1 Mrs J A Hamilton	11C/Bs1 Mr A S Choudhury	101/Oa1 Mrs J A Hamilton	132/Ab1 Miss D M Rooney	133/Ab1 Mr R G Milne	11E/Oa1 Mrs J A Hamilton	11C/Bs1 Mr A S Choudhury	101/Oa1 Mrs J A Hamilton
2	9a/Bs1 Mrs J A Hamilton	134/Ab1 Mrs J A Hamilton	11E/Oa1 Mrs J A Hamilton	102/Bs1 Mr R G Milne	9d/Bs1 Mrs J A Hamilton	133/Ab1 Mr R G Milne	11E/Oa1 Mr A S Choudhury	12d/Sb1 Mr L J Fitts	102/Bs1 Mr R G Milne	9d/Bs1 Mrs J A Hamilton
3	134/Ab1 Mrs J A Hamilton	136/Ab1 Mrs J A Hamilton	135/Ab1 Mrs J A Hamilton	101/Oa1 Mrs J A Hamilton	11C/Bs1 Mr A S Choudhury	11C/Bs1 Mr R G Milne	9a/Bs1 Mrs J A Hamilton	102/Bs1 Mr R G Milne	131/Al1 Mrs J A Hamilton	101/Oa1 Mrs J A Hamilton
ı										
4	122/Ep1 Mr C R Crisford	131/Al1 Mrs J A Hamilton	101/Oa1 Mrs J A Hamilton	11E/Oa1 Mrs J A Hamilton	134/Ab1 Mrs J A Hamilton	1235/Ct1 Mr R G Milne	134/Ab1 Mrs J A Hamilton	11F/Pa1 Mr L J Fitts	11E/Oa1 Mrs J A Hamilton	
5	11C/Bs1 Mr R G Milne		133/Ab1 Mr R G Milne		11de/Ma2 Mr A S Choudhury	1235/Ct1 Miss D M Rooney	102/Bs1 Mrs J A Hamilton	11ac/Ma3 Miss N K Crawford-Smith	101/Oa1 Mrs J A Hamilton	

Booking timetable J2 (provided by Mr Hamilton as excel)



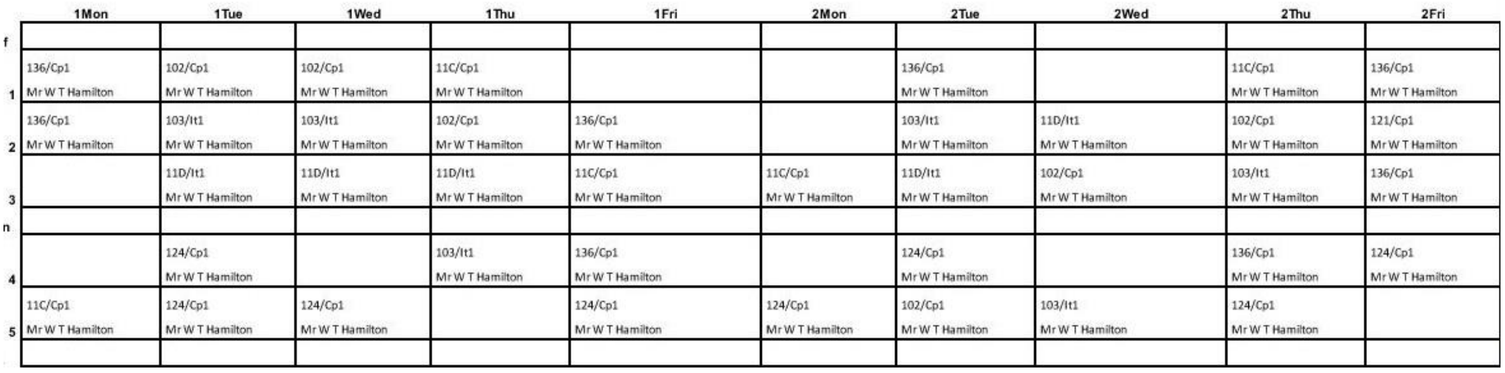
	1Mon	1Tue	1Wed	1Thu	1Fri	2Mon	2Tue	2Wed	2Thu	2Fri
f										
1		1235/Ct1 Mr R G Milne	131/Ab1 Miss D M Rooney	1235/Ct1 Mrs J A Hamilton		1235/Ct1 Mrs J A Hamilton		1235/Ct1 Mr R G Milne	1235/Ct1 Mrs J A Hamilton	
2	1235/Ct1 Mr A S Choudhury			1235/Ct1 Mrs J A Hamilton		135/Ab1 Miss D M Rooney	1235/Ct1 Miss D M Rooney		1235/Ct1 Miss D M Rooney	
3	121/Ep1 Mr C R Crisford		131/Al1 Miss D M Rooney		134/Ab1 Mrs J A Hamilton	131/Al1 Miss D M Rooney		136/Ab1 Mrs J A Hamilton	123/Ep1 Mr C R Crisford	
n										
4	1235/Ct1 Miss D M Rooney	11F/Pa1 Mr L J Fitts		131/Al1 Mr R G Milne			1235/Ct1 Miss D M Rooney	136/Ab1 Mrs J A Hamilton	124/Ep1 Mr C R Crisford	
5			1235/Ct1 Mrs J A Hamilton		1235/Ct1 Mr R G Milne	131/Al1 Mr R G Milne	131/Ab1 Miss D M Rooney	131/Al1 Mr R G Milne	1235/Ct1 Mr R G Milne	

Booking timetable A6 (provided by Mr Hamilton as excel)



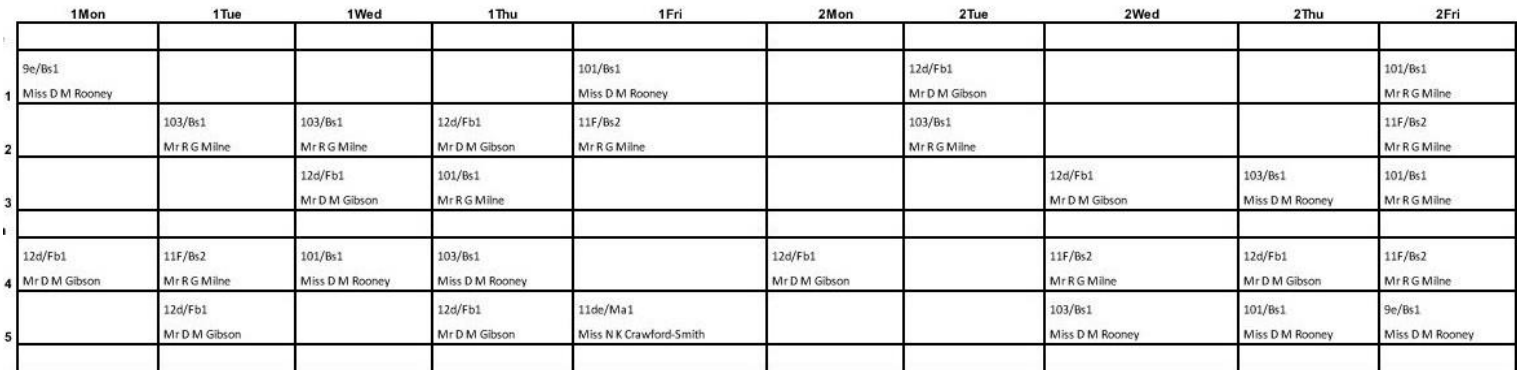
	1Mon	1Tue	1Wed	1Thu	1Fri	2Mon	2Tue	2Wed	2Thu	2Fri
f										
1			1235/Ct1 Mr R G Milne		9c/Bs1 Mr R G Milne	9c/Bs1 Mr R G Milne				
2	133/Ab1 Mr R G Milne				11F/Bs1 Miss D M Rooney			11D/Bs1 Mr R G Milne		11F/Bs1 Miss D M Rooney
3		11D/Bs1 Miss D M Rooney	11D/Bs1 Mr R G Milne	11D/Bs1 Miss D M Rooney	9b/Bs1 Mr R G Milne		11D/Bs1 Miss D M Rooney		12d/Fb1 Mr D M Gibson	131/Ab1 Miss D M Rooney
ı										
4	131/Al1 Mrs J A Hamilton	11F/Bs1 Miss D M Rooney				131/Ab1 Miss D M Rooney	12d/Sb1 Mr L J Fitts	11F/Bs1 Miss D M Rooney	9b/Bs1 Mr R G Milne	11F/Bs1 Miss D M Rooney
5	132/Ab1 Miss D M Rooney			132/Ab1 Miss D M Rooney	132/Ab1 Miss D M Rooney			11ac/Ma2 Mr L J Fitts		134/Ab1 Mrs J A Hamilton

Booking timetable B28 (provided by Mr Hamilton as excel)



	1Mon	1Tue	1Wed	1Thu	1Fri	2Mon	2Tue	2Wed	2Thu	2Fri
f										
1	136/Cp1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	11C/Cp1 Mr W T Hamilton			136/Cp1 Mr W T Hamilton		11C/Cp1 Mr W T Hamilton	136/Cp1 Mr W T Hamilton
2	136/Cp1 Mr W T Hamilton	103/It1 Mr W T Hamilton	103/It1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	136/Cp1 Mr W T Hamilton		103/It1 Mr W T Hamilton	11D/It1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	121/Cp1 Mr W T Hamilton
3		11D/It1 Mr W T Hamilton	11D/It1 Mr W T Hamilton	11D/It1 Mr W T Hamilton	11C/Cp1 Mr W T Hamilton	11C/Cp1 Mr W T Hamilton	11D/It1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	103/It1 Mr W T Hamilton	136/Cp1 Mr W T Hamilton
n										
4		124/Cp1 Mr W T Hamilton		103/It1 Mr W T Hamilton	136/Cp1 Mr W T Hamilton		124/Cp1 Mr W T Hamilton		136/Cp1 Mr W T Hamilton	124/Cp1 Mr W T Hamilton
5	11C/Cp1 Mr W T Hamilton	124/Cp1 Mr W T Hamilton	124/Cp1 Mr W T Hamilton		124/Cp1 Mr W T Hamilton	124/Cp1 Mr W T Hamilton	102/Cp1 Mr W T Hamilton	103/It1 Mr W T Hamilton	124/Cp1 Mr W T Hamilton	

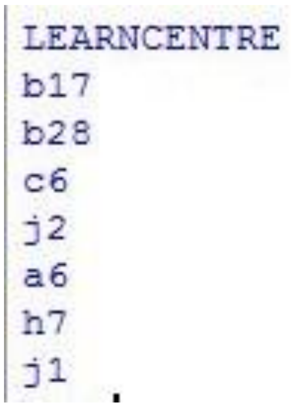
Booking timetable B17 (provided by Mr Hamilton as excel)



	1Mon	1Tue	1Wed	1Thu	1Fri	2Mon	2Tue	2Wed	2Thu	2Fri
1	9e/Bs1 Miss D M Rooney				101/Bs1 Miss D M Rooney		12d/Fb1 Mr D M Gibson			101/Bs1 Mr R G Milne
2		103/Bs1 Mr R G Milne	103/Bs1 Mr R G Milne	12d/Fb1 Mr D M Gibson	11F/Bs2 Mr R G Milne		103/Bs1 Mr R G Milne			11F/Bs2 Mr R G Milne
3			12d/Fb1 Mr D M Gibson	101/Bs1 Mr R G Milne				12d/Fb1 Mr D M Gibson	103/Bs1 Miss D M Rooney	101/Bs1 Mr R G Milne
ı										
4	12d/Fb1 Mr D M Gibson	11F/Bs2 Mr R G Milne	101/Bs1 Miss D M Rooney	103/Bs1 Miss D M Rooney		12d/Fb1 Mr D M Gibson		11F/Bs2 Mr R G Milne	12d/Fb1 Mr D M Gibson	11F/Bs2 Mr R G Milne
5		12d/Fb1 Mr D M Gibson		12d/Fb1 Mr D M Gibson	11de/Ma1 Miss N K Crawford-Smith			103/Bs1 Miss D M Rooney	101/Bs1 Miss D M Rooney	9e/Bs1 Miss D M Rooney

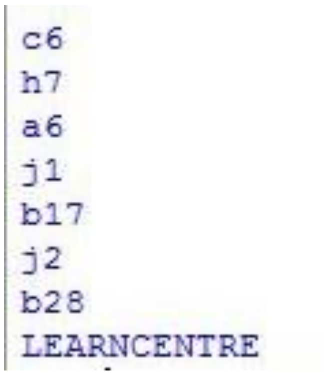
Part D

The output of the prototype python code (for the best room algorithm) when attempting test **(6)**.



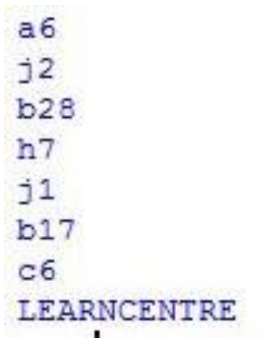
```
LEARNCENTRE
b17
b28
c6
j2
a6
h7
j1
```

The output of the prototype python code (for the best room algorithm) when attempting test **(7)**.



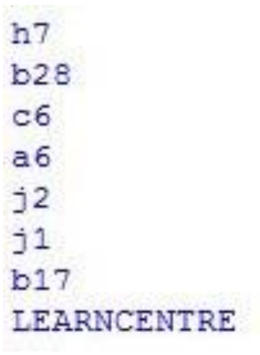
```
c6
h7
a6
j1
b17
j2
b28
LEARNCENTRE
```

The output of the prototype python code (for the best room algorithm) when attempting test **(8)**.



```
a6
j2
b28
h7
j1
b17
c6
LEARNCENTRE
```

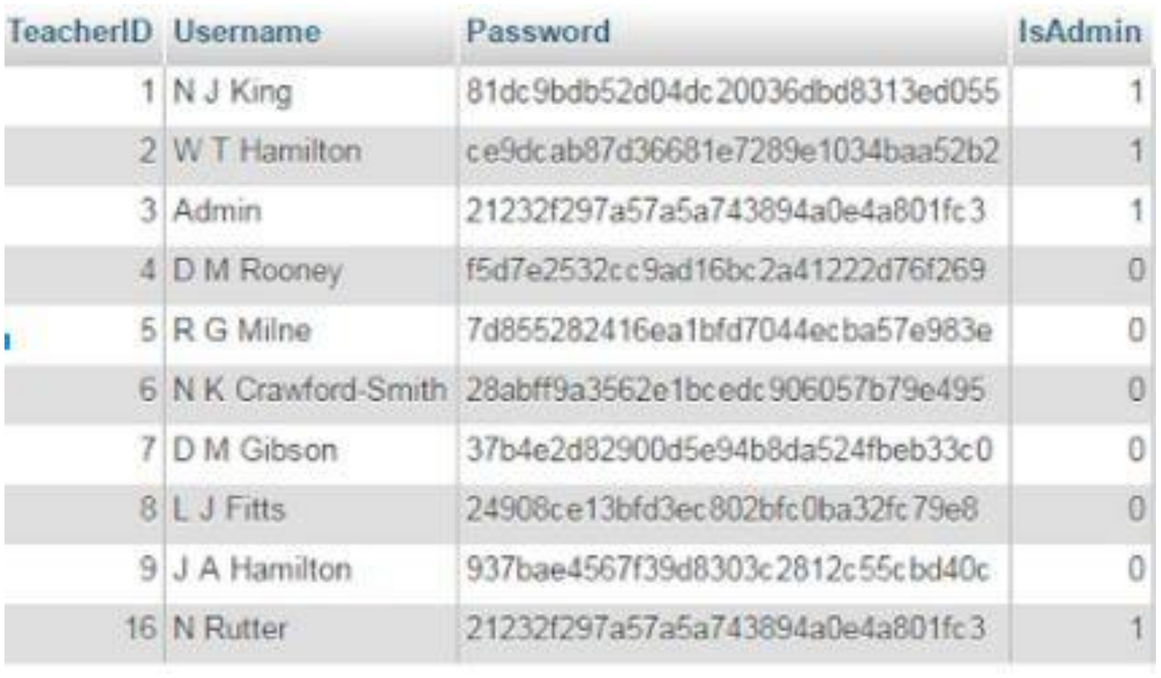
The output of the prototype python code (for the best room algorithm) when attempting test **(9)**.



```
h7
b28
c6
a6
j2
j1
b17
LEARNCENTRE
```

Part E

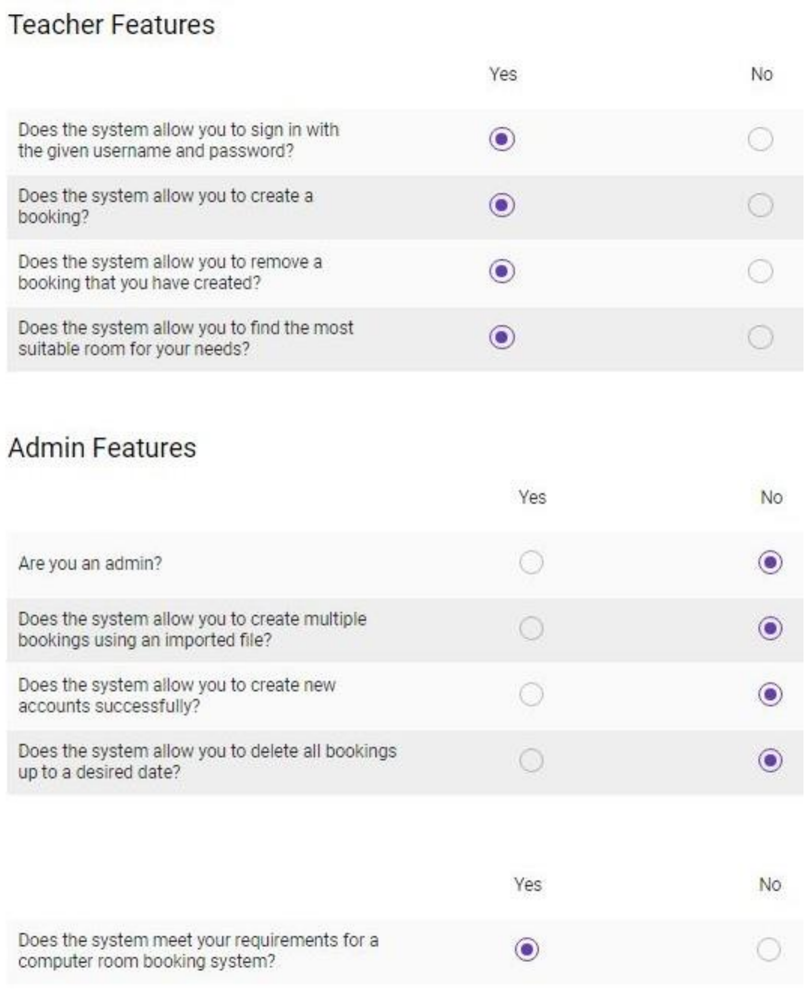
The database displaying teacher login information with the md5 algorithm applied on the passwords.



TeacherID	Username	Password	IsAdmin
1	N J King	81dc9bdb52d04dc20036dbd8313ed055	1
2	W T Hamilton	ce9dcab87d36681e7289e1034baa52b2	1
3	Admin	21232f297a57a5a743894a0e4a801fc3	1
4	D M Rooney	f5d7e2532cc9ad16bc2a41222d76f269	0
5	R G Milne	7d855282416ea1bfd7044ecba57e983e	0
6	N K Crawford-Smith	28abff9a3562e1bcedc906057b79e495	0
7	D M Gibson	37b4e2d82900d5e94b8da524fbeb33c0	0
8	L J Fitts	24908ce13bfd3ec802bfc0ba32fc79e8	0
9	J A Hamilton	937bae4567f39d8303c2812c55cbd40c	0
16	N Rutter	21232f297a57a5a743894a0e4a801fc3	1

Part F

Questionnaire completed by a teacher:



Teacher Features

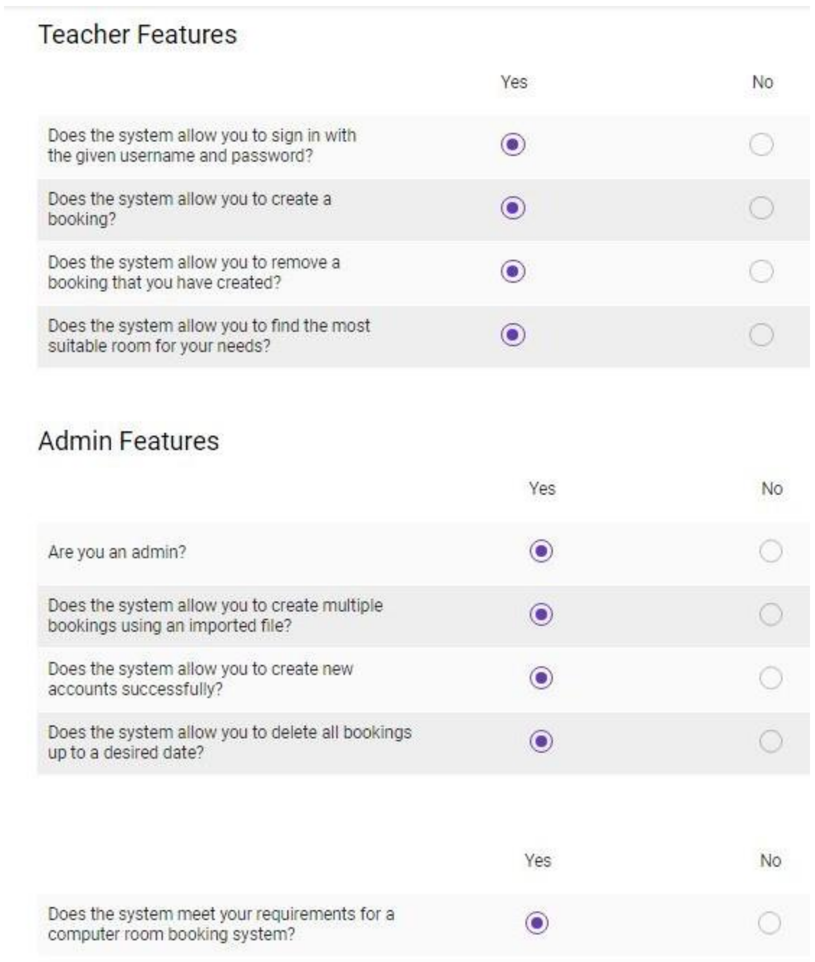
	Yes	No
Does the system allow you to sign in with the given username and password?	[✓ Yes]	No
Does the system allow you to create a booking?	[✓ Yes]	No
Does the system allow you to remove a booking that you have created?	[✓ Yes]	No
Does the system allow you to find the most suitable room for your needs?	[✓ Yes]	No

Admin Features

	Yes	No
Are you an admin?	Yes	[✓ No]
Does the system allow you to create multiple bookings using an imported file?	Yes	[✓ No]
Does the system allow you to create new accounts successfully?	Yes	[✓ No]
Does the system allow you to delete all bookings up to a desired date?	Yes	[✓ No]

	Yes	No
Does the system meet your requirements for a computer room booking system?	[✓ Yes]	No

Questionnaire completed by my customer, Mr Rutter:



Teacher Features

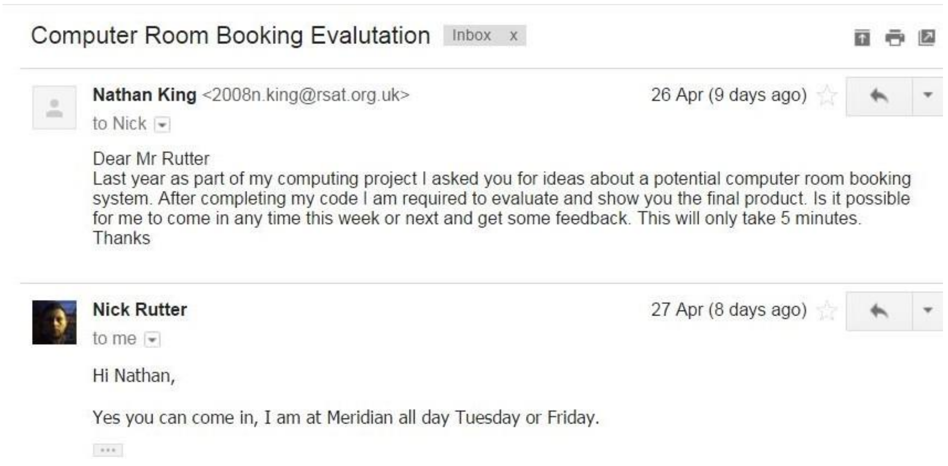
	Yes	No
Does the system allow you to sign in with the given username and password?	●	○
Does the system allow you to create a booking?	●	○
Does the system allow you to remove a booking that you have created?	●	○
Does the system allow you to find the most suitable room for your needs?	●	○

Admin Features

	Yes	No
Are you an admin?	●	○
Does the system allow you to create multiple bookings using an imported file?	●	○
Does the system allow you to create new accounts successfully?	●	○
Does the system allow you to delete all bookings up to a desired date?	●	○

	Yes	No
Does the system meet your requirements for a computer room booking system?	●	○

Evidence of my interactions with Mr Rutter:



Computer Room Booking Evalutation Inbox x

Nathan King <2008n.king@rsat.org.uk> 26 Apr (9 days ago)
to Nick

Dear Mr Rutter
Last year as part of my computing project I asked you for ideas about a potential computer room booking system. After completing my code I am required to evaluate and show you the final product. Is it possible for me to come in any time this week or next and get some feedback. This will only take 5 minutes.
Thanks

Nick Rutter 27 Apr (8 days ago)
to me

Hi Nathan,

Yes you can come in, I am at Meridian all day Tuesday or Friday.

Part G

The following pages are the technical solution for my project.