Timetable Constructor

Contents

[Analysis 5](#_Toc480557569)

[The problem 5](#_Toc480557570)

[The plan 5](#_Toc480557571)

[Objectives 7](#_Toc480557572)

[Design 8](#_Toc480557573)

[initial design 8](#_Toc480557574)

[Assigning teachers 8](#_Toc480557575)

[Teacher allocation 8](#_Toc480557576)

[assigning students 8](#_Toc480557577)

[view timetable 9](#_Toc480557578)

[Final gui designs 10](#_Toc480557579)

[IPSO diagram 15](#_Toc480557580)

[flowcharts 16](#_Toc480557581)

[Teacher sign up 16](#_Toc480557582)

[Entity relastionship diagram 17](#_Toc480557583)

[database design 17](#_Toc480557584)

[Block period day 17](#_Toc480557585)

[blocks 17](#_Toc480557586)

[course table 17](#_Toc480557587)

[period settings 17](#_Toc480557588)

[room table 18](#_Toc480557589)

[student course 18](#_Toc480557590)

[teacher class room course 18](#_Toc480557591)

[SQL statements 19](#_Toc480557592)

[create tables 19](#_Toc480557593)

[Timetable 20](#_Toc480557594)

[Settings 20](#_Toc480557595)

[Teacher allocation 20](#_Toc480557596)

[teacher sign up 21](#_Toc480557597)

[student SIGN up 21](#_Toc480557598)

[PseudoCode 22](#_Toc480557599)

[teacher sign up 22](#_Toc480557600)

[timetable 22](#_Toc480557601)

[student sign up 22](#_Toc480557602)

[Bibliography 23](#_Toc480557603)

[technical solution 24](#_Toc480557604)

[teacher Sign up 24](#_Toc480557605)

[teachersignup.php 24](#_Toc480557606)

[teacher allocation 27](#_Toc480557607)

[teacherAllocation.php 27](#_Toc480557608)

[getblockteacherallocation.php 31](#_Toc480557609)

[getroomidTeacherAllocation.php 31](#_Toc480557610)

[getteacheridteacherallocation.php 31](#_Toc480557611)

[isroomfullteacherallocation.php 32](#_Toc480557612)

[student sign up 32](#_Toc480557613)

[studentsignup.php 32](#_Toc480557614)

[timetable 36](#_Toc480557615)

[timetable.php 36](#_Toc480557616)

[getblocktimetable.php 39](#_Toc480557617)

[getcourseidtimetable.php 40](#_Toc480557618)

[getroomidtimetable.php 40](#_Toc480557619)

[getteacheridTimetable.php 40](#_Toc480557620)

[settings 41](#_Toc480557621)

[settings.php 41](#_Toc480557622)

[header.html 44](#_Toc480557623)

[footer.html 44](#_Toc480557624)

[dbconnect.php 44](#_Toc480557625)

[style.css 44](#_Toc480557626)

[interface 46](#_Toc480557627)

[teacher sign up 46](#_Toc480557628)

[teacher allocation 46](#_Toc480557629)

[student sign up 47](#_Toc480557630)

[time table 47](#_Toc480557631)

[settings 48](#_Toc480557632)

[Testing 49](#_Toc480557633)

[objective 2. 49](#_Toc480557634)

[website 49](#_Toc480557635)

[database 50](#_Toc480557636)

[objective 3. 50](#_Toc480557637)

[website 50](#_Toc480557638)

[objective 4. 51](#_Toc480557639)

[website 51](#_Toc480557640)

[objective 5. 51](#_Toc480557641)

[website 52](#_Toc480557642)

[database 52](#_Toc480557643)

[objective 6. 52](#_Toc480557644)

[website 53](#_Toc480557645)

[database 53](#_Toc480557646)

[objective 7. 53](#_Toc480557647)

[website 54](#_Toc480557648)

[objective 8 54](#_Toc480557649)

[website 54](#_Toc480557650)

[objective 9. 54](#_Toc480557651)

[website 54](#_Toc480557652)

[objective 10 54](#_Toc480557653)

[website 55](#_Toc480557654)

[database 55](#_Toc480557655)

[evaluation 55](#_Toc480557656)

[third party feedback 55](#_Toc480557657)

# Analysis

## The problem

School timetables are needed for schools to schedule teachers, students, rooms, classes and blocks (the periods that will be used for a class). In secondary and private schools in England the whole day is filled up but in most colleges there are blocks that are empty.

There are many issues with creating software that generates timetables because different schools have different ways of laying out their days for example some use blocks of lesson and some do not (Anon., 2016). The software also can get more confusing when multiple teachers teach a class and break times are taken into account because some schools do very different approaches to how they split their lunch breaks.

Software that generates timetables costs hundreds of pounds and even after spending a lot of money can still have problems that need to be fixed.

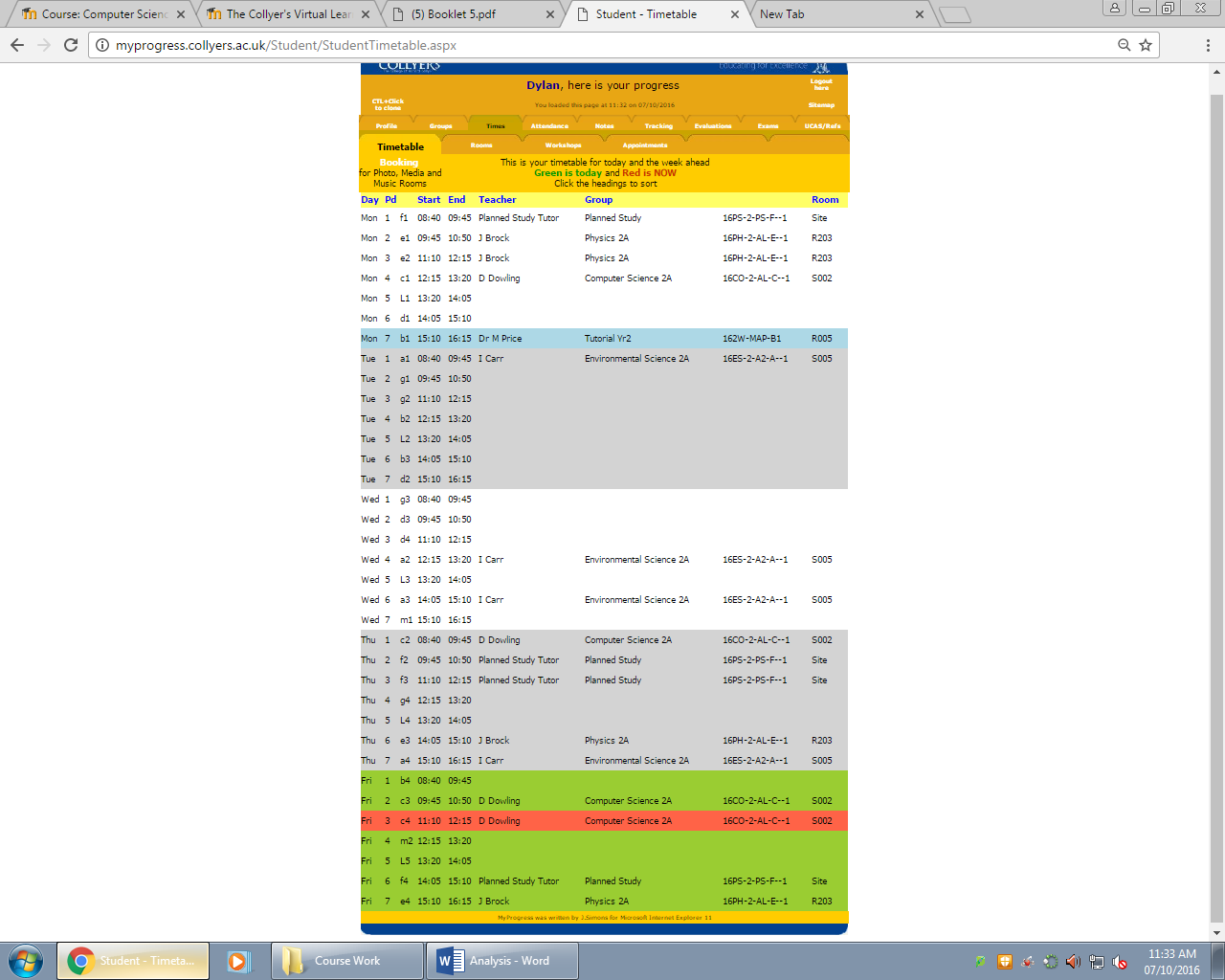
My program will be used by staff of a school to help allocate students and teachers to subjects, classes, rooms etc. Students will also use this software to pick what subjects they want to attend.

To help me with this project I can talk to staff/teachers at college and teachers out of college who use this type of software to sort their school’s timetables.

## The plan

To avoid running into extremely difficult problems I am going to create a time table constructor for the college I attend (Collyer’s), I will not be implementing split lessons were two teachers teach the same class or a class that is taught in different rooms. At Collyer’s each class is put into one of six blocks (A, B, C etc.). There are six periods every day. The periods each block fall in is decided by a person. I have decided I will let the user pick how many blocks and periods there are in a day and what time they are. The users will then input the block a teacher teaches a subject in, the subjects the student has chosen. There will be a way for the user to check what classes are full, what teacher they have, what room it is in and the what block it is in and the user should be able to check a student’s timetable. A teacher and room should only be able to be assigned to one class per block.

I interview a teacher who uses this type of software, the current software they use is SIMS it allocates everything automatically but I am going to make mine all manual.

This picture below is an example of a student’s timetable from college I attend

## Objectives

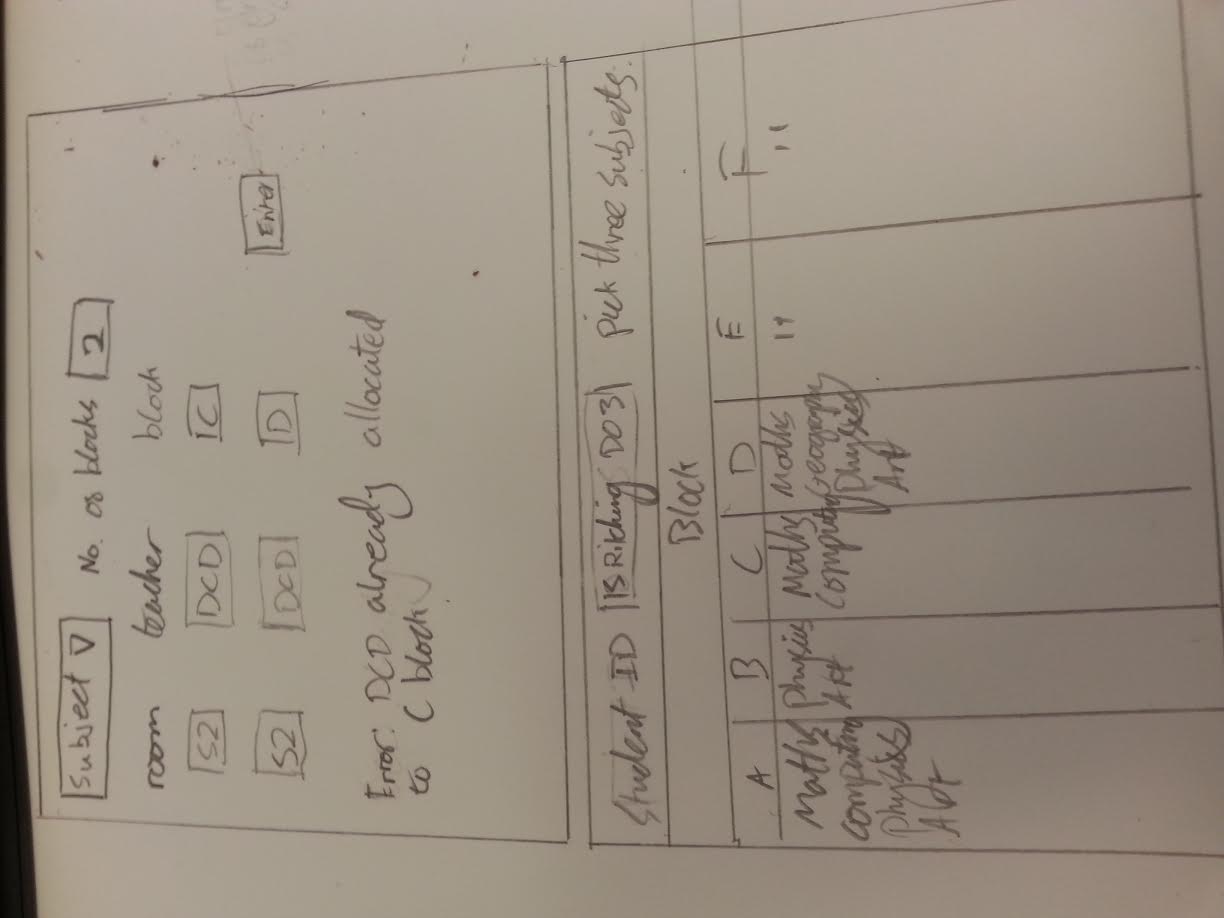
The system must:

1. Store:
   1. teacher information
   2. student information
   3. student subject choices
   4. subject information
   5. period and block times
2. Include a window that allows the user to input what subject a block is in, what teacher teaches that block, what room it’s in, the year group of the class and the max size of the class
   1. if the data is valid the information should be stored
   2. If there is clash between the teacher and block or room and block display error message and the information can’t be processed until corrected
3. Include a window that shows the teacher allocation that contains a table which will display on each line the teacher, subject, year, block and room. Allow the user in this window to be able to:
   1. double click to delete lines of information from the table which will also delete information from the database
4. Include a window that allows a student to view their timetable
   1. each cell includes the subject, block, room and teacher
   2. x-axis will be days and y-axis is the period and what time the period is in
5. Include a settings window were the number of periods there are in a day, the times of each period, the number of blocks and the times of each block can be entered.
   1. if the enter button is clicked the information should be sent to the database
6. Include a window that allows a student to pick three subjects from multiple different blocks and when completed stores the information
   1. if the enter button is clicked the information should be sent to the database and overwrite any lessons that the student has been assigned to before
7. Include a way to see what class is full
8. Include a way for the user to navigate between different pages
9. Include on each page the user (teacher or student) needs to input their userID (if needed) so the information can be assigned to the correct person
10. When a class is full don’t allow any more students to sign up to that class.

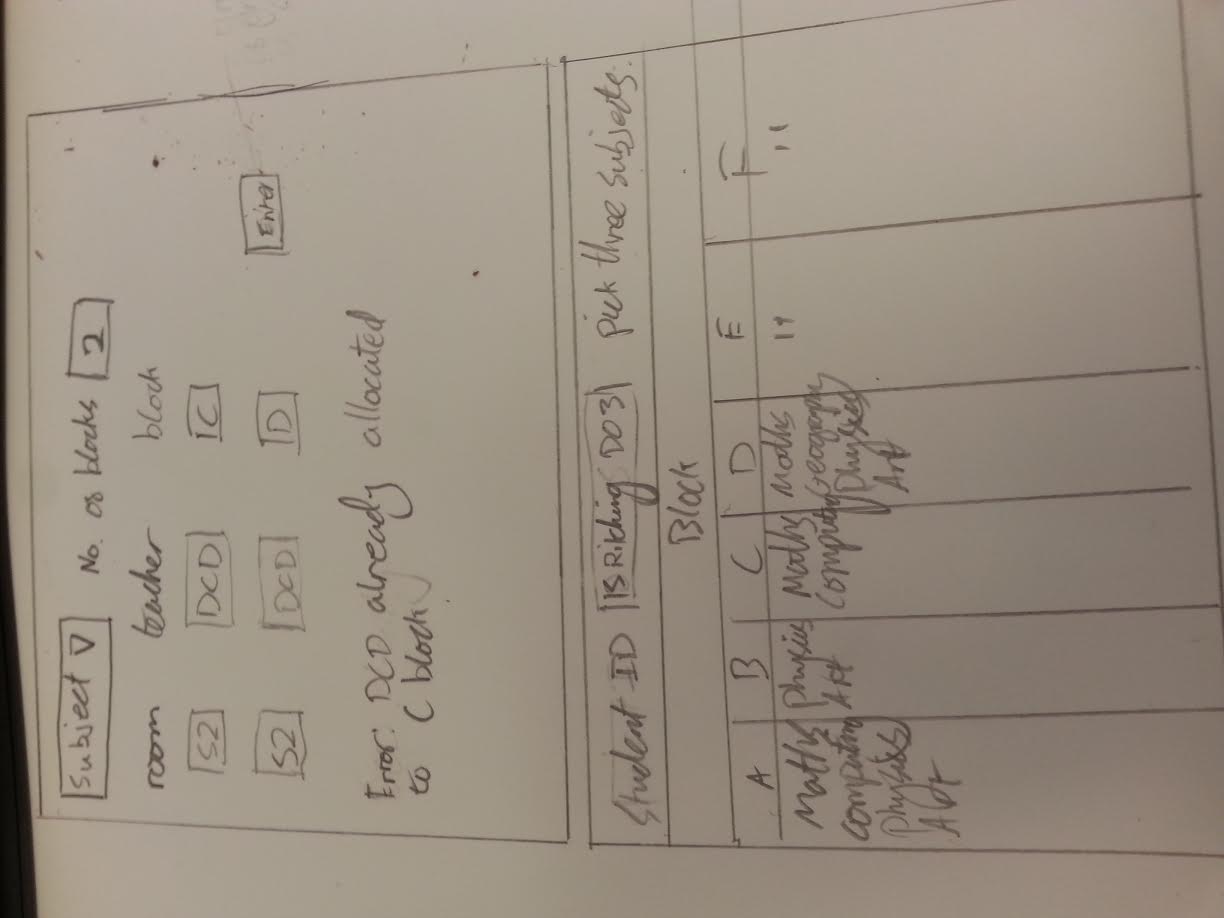
# Design

## initial design

### Assigning teachers

In this window the user will choose how many blocks they wish to add and a drop down menu with the subjects the lesson could be for. When both the subject and number of blocks fields have been filled in a table with input fields will be generated with the same amount of rows as number of blocks chosen. There will be three columns. The first column will have a room field, the second column will have the teacher field, and the third column will be the block field and a fourth column that contains a field that the amount of students that can fit in the room will be input. There will be an enter button that when pressed will check if the room and teacher are free in that block. If both are free then the data will be stored in the database if either aren’t free then an error message will appear showing the user what isn’t free.

### Teacher allocation

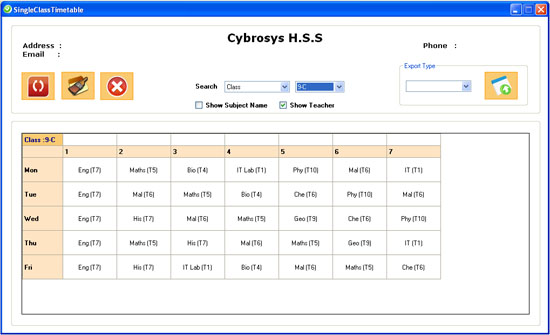
In the teacher allocation window there will be a drop down menu of subjects. When a subject has been picked a table will generate with each column titled teacher, subject, year group, block and room. This window will be used to view what teachers, for a specified subject, teach what class. The user will be able to delete lines of information which will also delete the information from the database.

### assigning students

At the top the student will input their student username. There will be a table with the columns for each block underneath each block will be a subject that occupies that block, the student will click three subjects in different blocks that they would like to attend. The subjects and blocks the student picked will be stored in the database.

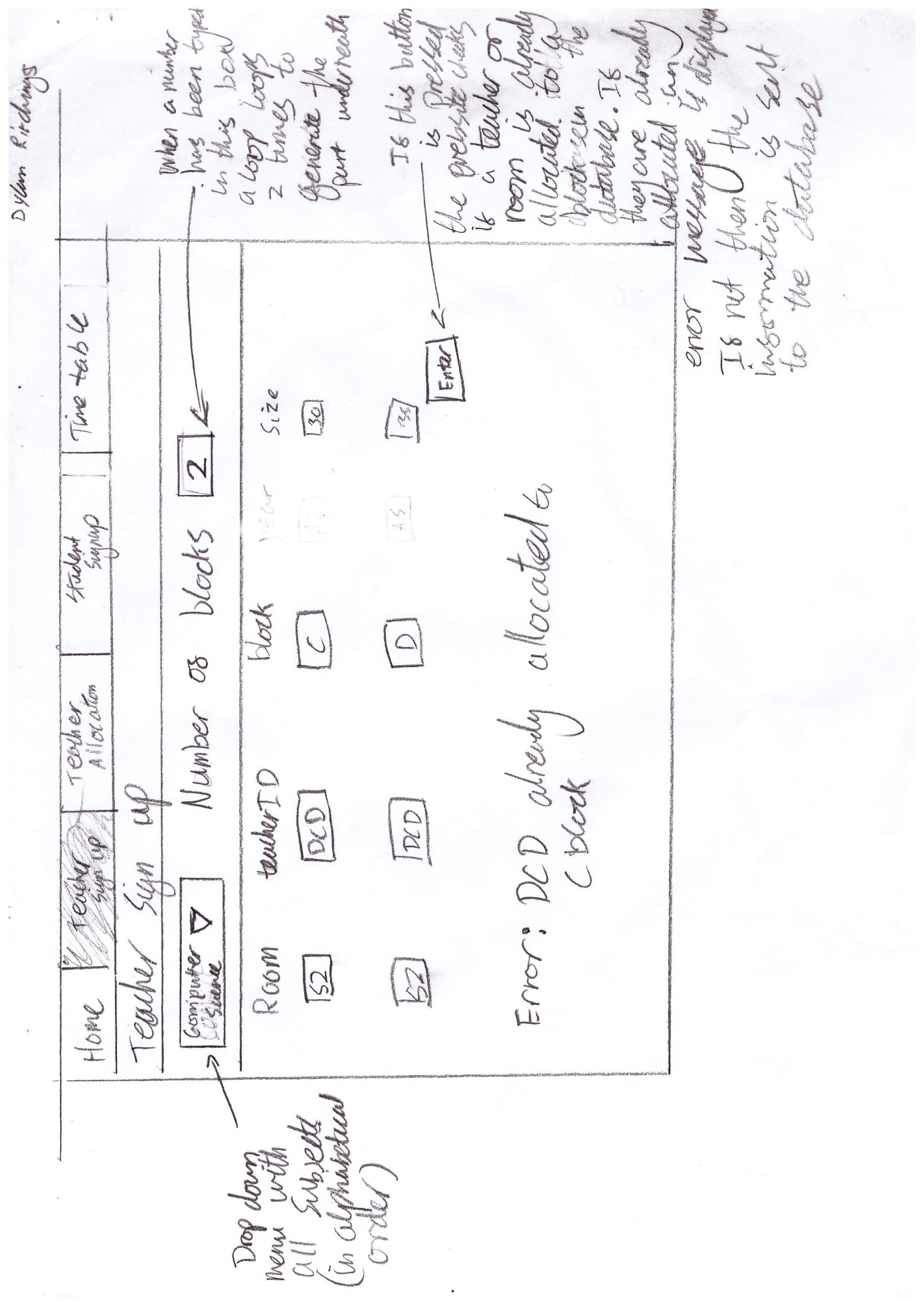
The program will then move students to other blocks of the same subject if the current block is full or automatically send them a message saying they have to change a subject.

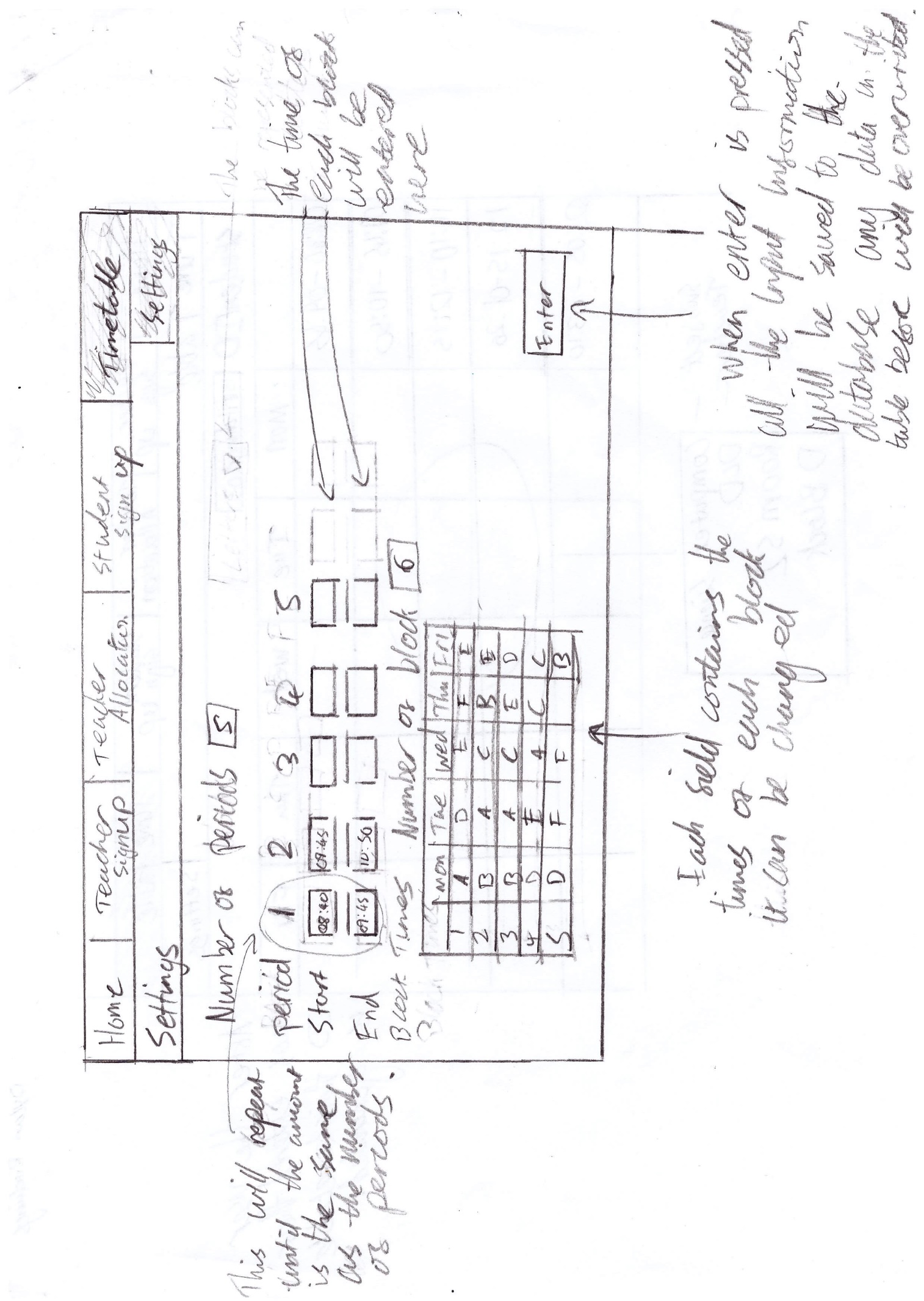
### view timetable

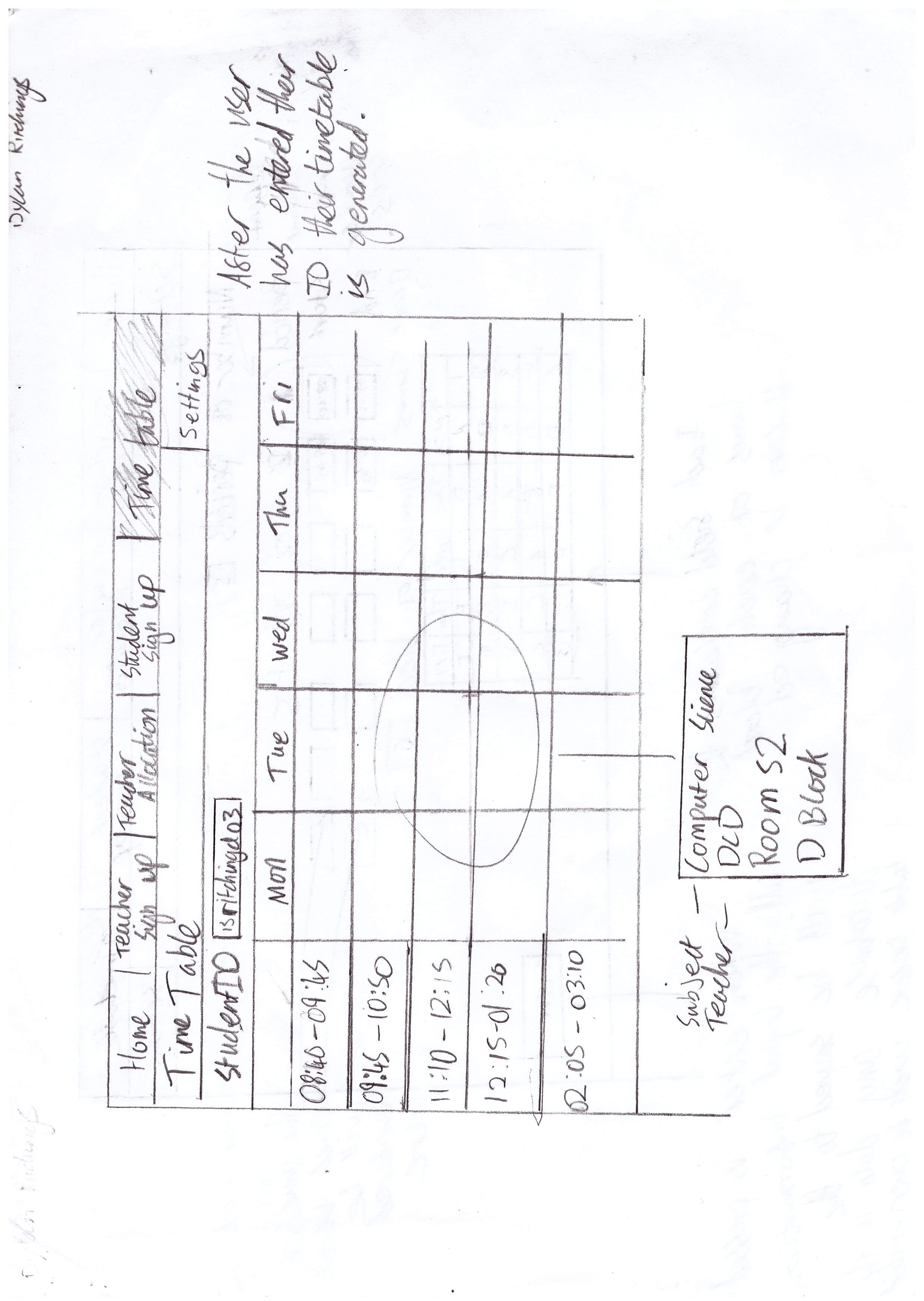
The user will input their studentID and the timetable will be displayed in a grid layout like the one below but each cell will contain: subject, teacher, room and block.

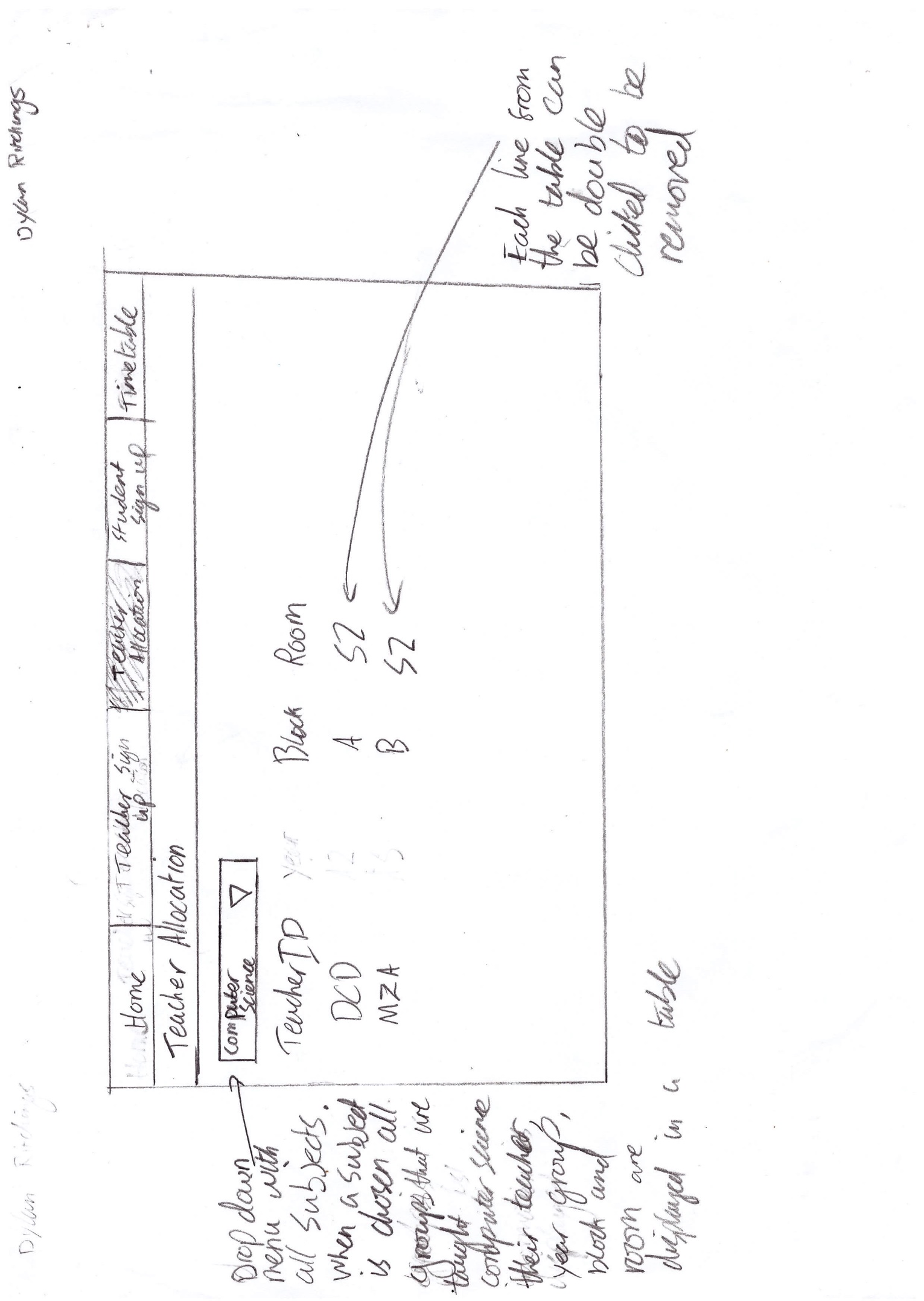
(Anon., 2009)

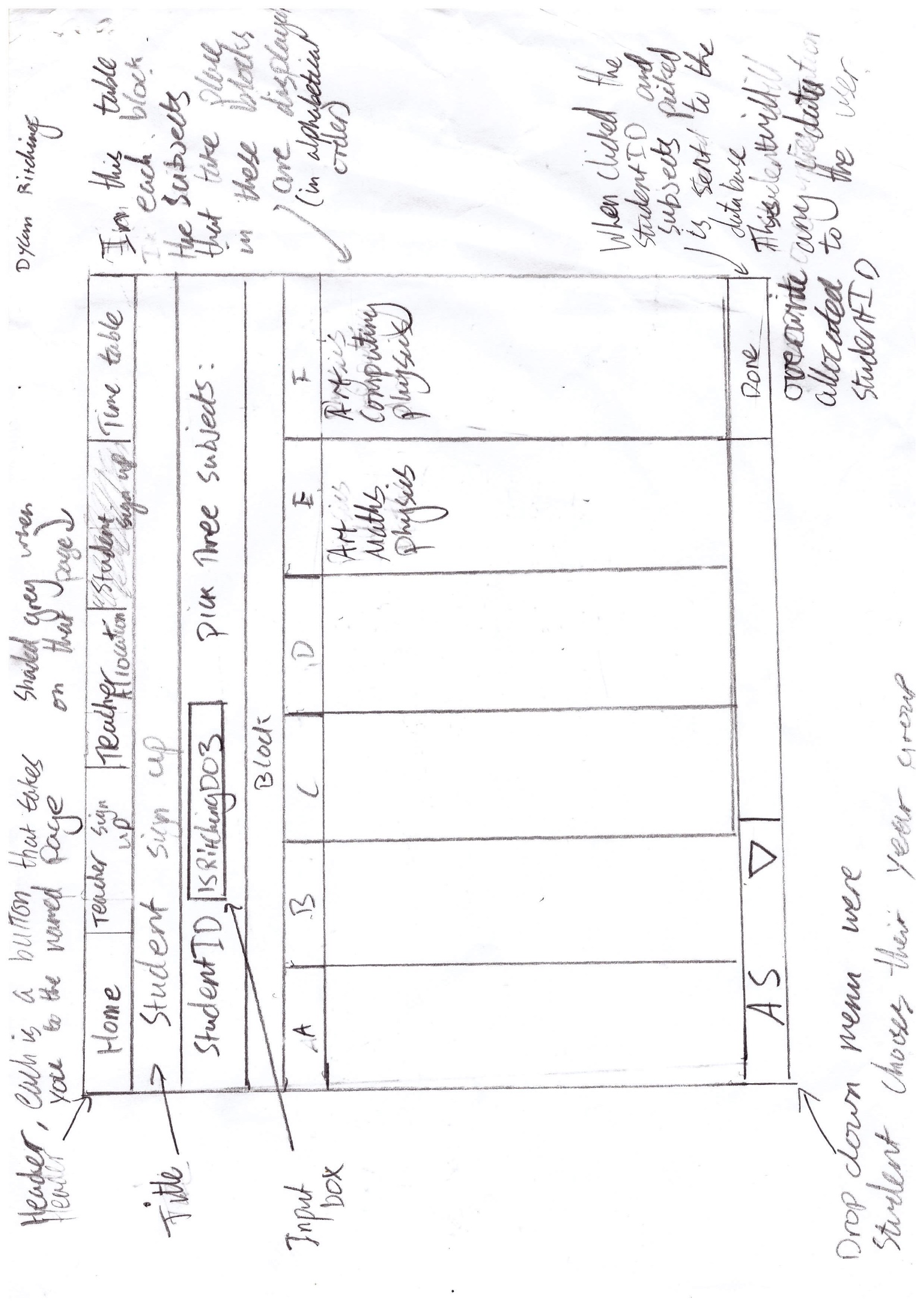
## Final gui designs











## IPSO diagram

Inputs Processes

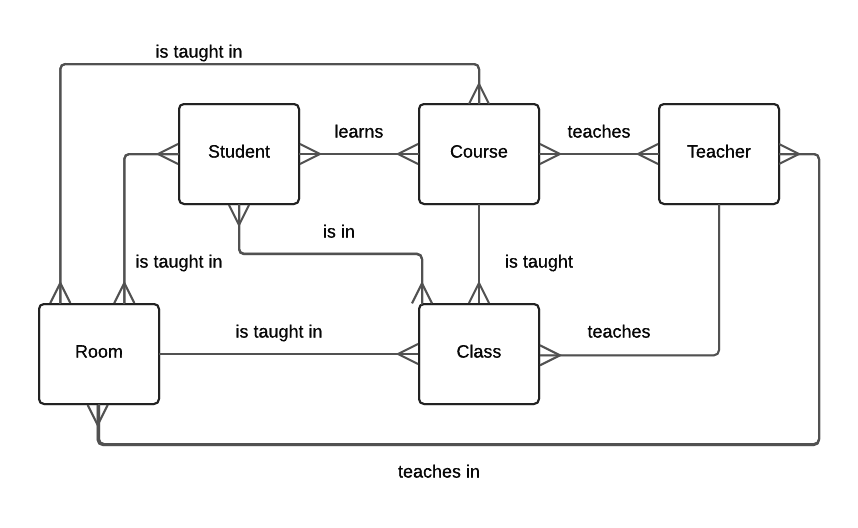
|  |  |
| --- | --- |
| * Class details to database (teacherID, room, block, subject, year group, class size) * Student details to database(studentID, year, subjects, blocks) * Double click to delete selected record (multi table) * Period times * Number of periods * Block times * Number of blocks * Student sign up page includes a column of blocks with the subjects underneath that are in those blocks * Teacher sign up | * Check if teacherID or a room is already allocated to a specified block in database * Search for class details for a specific subject from database * Validate block input * Validate year input * Sort a specific students details from database into a timetable * Generate input fields based on number of blocks on teacher sign up page * Generate input fields based on number of periods on timetable settings page * Generate table with column length same size as number of periods on timetable settings page * Generate table with all subjects that fill each block |
| External   * BlockPeriodDay (block, period, day) * blocks (block) * courseTBL (courseID, courseName) * periodSettings (period, periodStart, periodEnd) * roomTBL (roomID, size, block) * studentCourse (studentID, courseID, block, roomID) * studentTBL (yearGroup, studentID)   teacherClassRoomCourse (teacherID, block, roomID, courseID, year) | * Teacher allocation page includes class details from database for a subject (teacherID, year, block, room) * Time table page that has a table that includes a single students details for each subject they picked from database, each cell will include subject, teacherID, room and block. (rows: days of week, columns: time of each lesson) * Table that displays the time of each block that can be edited |
| Internal   * Number of blocks (teacher sign up and settings) * Number of periods (settings) * class list * student list |

Storage Output

## flowcharts

### https://documents.lucidchart.com/documents/55b4f666-54a8-4b81-ae71-4ff2d2f2aef5/pages/0_0?a=1258&x=5&y=22&w=1221&h=2165&store=1&accept=image%2F*&auth=LCA%2023b47ac83fe286957314457146d2f7dced8f6a11-ts%3D1479468799Teacher sign up

## Entity relastionship diagram



## database design

### Block period day

|  |  |
| --- | --- |
| **Name** | **Type** |
| block | Char(1) |
| period | Int |
| day | Varchar(9) |

### blocks

|  |  |
| --- | --- |
| Block | char(1) |

### course table

|  |  |
| --- | --- |
| courseID (PRIMARY) | Varchar(6) |
| courseName | Varchar(30) |

### period settings

|  |  |
| --- | --- |
| Period (PRIMARY) | Int |
| periodStart | periodStart |
| periodEnd | periodEnd |

### room table

|  |  |
| --- | --- |
| roomID | Varchar(6) |
| Size | Int |
| Block | Char(1) |
| roomPKey (AUTO\_INCREMENT) (PRIMARY) | int |

### student course

|  |  |
| --- | --- |
| studentID (PRIMARY) | Varchar(6) |
| courseID | Varchar(6) |
| Block | Char(1) |
| roomID | Varchar(6) |

### teacher class room course

|  |  |
| --- | --- |
| teacherID (PRIMARY) | Varchar(6) |
| Block | Char(1) |
| roomID | Varchar(6) |
| courseID | Varchar(6) |
| Year | Int |
| classID | Varchar(20) |

## SQL statements

### create tables

CREATE TABLE `BlockPeriodDay` (  
 `block` char(1) NOT NULL,  
 `period` int(11) NOT NULL,  
 `day` varchar(9) NOT NULL  
)

CREATE TABLE `blocks` (  
 `block` varchar(1) NOT NULL,  
 PRIMARY KEY (`block`)  
)

CREATE TABLE `courseTBL` (  
 `courseID` varchar(6) NOT NULL,  
 `courseName` varchar(30) NOT NULL,  
 PRIMARY KEY (`courseID`,`courseName`))

CREATE TABLE `periodSettings` (  
 `period` int(11) NOT NULL,  
 `periodStart` time NOT NULL,  
 `periodEnd` time NOT NULL,  
 PRIMARY KEY (`period`))

CREATE TABLE `roomTBL` (  
 `roomID` varchar(6) NOT NULL,  
 `size` int(11) NOT NULL,  
 `block` char(1) NOT NULL,  
 `roomPKey` int(11) NOT NULL AUTO\_INCREMENT,  
 PRIMARY KEY (`roomPKey`))

CREATE TABLE `StudentCourse` (  
 `studentID` varchar(6) NOT NULL,  
 `courseID` varchar(6) NOT NULL,  
 `block` char(1) NOT NULL,  
 `roomID` varchar(6) NOT NULL,  
 PRIMARY KEY (`studentID`,`courseID`),  
 KEY `courseID` (`courseID`))

CREATE TABLE `studentTBL` (  
 `yearGroup` char(2) NOT NULL,  
 `studentID` varchar(6) NOT NULL,  
 PRIMARY KEY (`studentID`))

CREATE TABLE `TeacherClassRoomCourse` (  
 `teacherID` varchar(6) NOT NULL,  
 `block` varchar(1) NOT NULL,  
 `roomID` varchar(6) NOT NULL,  
 `courseID` varchar(6) NOT NULL,  
 `year` int(11) NOT NULL,  
 `classID` varchar(20) NOT NULL,  
 PRIMARY KEY (`teacherID`,`block`,`roomID`,`courseID`,`year`,`classID`))

### Timetable

SELECT COUNT(\*) FROM blocks

SELECT COUNT(\*) FROM periodSettings

SELECT periodStart FROM periodSettings WHERE period = '$count'

SELECT periodEnd FROM periodSettings WHERE period = '$count'

SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Monday'

SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Tuesday'

SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Wednesday'

SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Thursday'

SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Friday'

SELECT block FROM StudentCourse WHERE studentID = '$studentID'

SELECT courseID FROM StudentCourse WHERE studentID = '$studentID'

SELECT roomID FROM StudentCourse WHERE studentID = '$studentID'

SELECT teacherID FROM TeacherClassRoomCourse WHERE courseID = '$courseID' && block = '$block' && roomID = '$roomID'

### Settings

TRUNCATE TABLE periodSettings

TRUNCATE TABLE BlockPeriodDay

TRUNCATE TABLE blocks

INSERT INTO periodSettings(period,periodStart,periodEnd)VALUES('$period','$startTime','$endTime')

INSERT INTO BlockPeriodDay(block,period,day)VALUES('$block','$period','$day')

INSERT INTO blocks(block)VALUES('$block')

### Teacher allocation

DELETE FROM TeacherClassRoomCourse WHERE teacherID = '$teacherID' && block = '$block' && roomID = '$roomID' && courseID = '$courseID'

SELECT roomID FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'

SELECT block FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'

SELECT block FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'

SELECT teacherID FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'

SELECT size FROM roomTBL WHERE roomID = '$roomID' && block = '$block'

SELECT COUNT(\*) FROM StudentCourse WHERE roomID = '$roomID' && block = '$block'

### teacher sign up

SELECT roomID FROM roomTBL WHERE block = '$block'

SELECT teacherID FROM TeacherBlock WHERE block = '$block'

INSERT INTO TeacherClassRoomCourse(teacherID,block,roomID,courseID,year)VALUES('$teacher','$block','$room','$subject','$year')

INSERT INTO roomTBL(roomID, size, block)VALUES('$room','$size','$block')

INSERT INTO TeacherBlock(teacherID,block)VALUES('$teacher','$block')

INSERT INTO courseTBL(courseID,courseName)VALUES('$subject','$subjectName')

### student SIGN up

SELECT COUNT(\*) FROM TeacherClassRoomCourse WHERE block = '$block'

SELECT courseID FROM TeacherClassRoomCourse WHERE block = '$block'

SELECT courseName FROM courseTBL WHERE courseID = '$subjectID'

DELETE FROM studentTBL WHERE studentID = '$studentID'

DELETE FROM StudentCourse WHERE studentID = '$studentID'

INSERT INTO studentTBL (yearGroup, studentID)VALUES('$year','$studentID')

SELECT courseID FROM courseTBL WHERE courseName = '$subject'

SELECT roomID FROM TeacherClassRoomCourse WHERE block = '$block' && courseID = '$courseID'

INSERT INTO StudentCourse (studentID,courseID,block,roomID)VALUES('$studentID','$courseID','$block','$roomID')

## PseudoCode

### teacher sign up

numberOfBlocks 🡨 INPUT

FOR n IN RANGE (0, numberOfBlocks)

roomID 🡨 INPUTBOX

teacherID 🡨 INPUTBOX

block 🡨 INPUTBOX

year 🡨 INPUTBOX

size 🡨 INPUTBOX

ENDFOR

### timetable

FOR day IN RANGE (0, 5)

FOR time IN RANGE (0, numberOfPeriods)

OUPUT subjectName

OUTPUT teacherID

OUTPUT “Room” + roomID

OUTPUT block + “ Block”

ENDFOR

ENDFOR

### student sign up

APPEND blockList(block) 🡨 subjectName  
  
FOR n RANGE (0,numberOfBlocks)

blockName = ASCII(65+n)

FOR subject IN RANGE(0,LENGTH(blockList(blockName)))

OUTPUT blockList(blockName)[subject]

# Bibliography

Anon., 2009. *Edutimer screen shots.* [Online]   
Available at: http://www.edutimer.com/screen\_shots\_of\_primary\_secondary\_school\_software.html  
[Accessed 27 February 2017].

Anon., 2016. *School timetable.* [Online]   
Available at: https://en.wikipedia.org/wiki/School\_timetable

# technical solution

## teacher Sign up

### teachersignup.php

1. <?php //teacherSignUp.php
2. $page\_title = 'Allocate Teacher';
3. **include** ('header.html');
4. $subject='';
5. $numblocks='';
6. $roomList=[];
7. $blockList=[];
8. $yearList=[];
9. $sizeList=[];
10. $teacherList=[];
11. $subjectName = '';
12. **require** ('dbconnect.php');
13. **if**(isset($\_POST['submit'])){
15. $subject=$\_POST['subject'];
16. $numblocks=$\_POST['numblocks'];
17. $roomList=$\_POST['room'];
18. $teacherList=$\_POST['teacher'];
19. $blockList=$\_POST['block'];
20. $yearList=$\_POST['year'];
21. $sizeList=$\_POST['size'];
22. $subjectName = $\_POST['subjectName'];
23. **if** (empty**empty**($subject)){
24. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
25. echo "<p><strong>Fill in number of subject</strong></p>";
26. }
27. **elseif** (empty**empty**($numblocks)){
28. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
29. echo "<p><strong>Fill in number of blocks</strong></p>";
30. }
31. **elseif** (**empty**empty($roomList)){
32. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
33. echo "<p><strong>Fill in room</strong></p>";
34. }
35. **elseif** (empty**empty**($teacherList)){
36. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
37. echo "<p><strong>Fill in teacher</strong></p>";
38. }
39. **elseif** (**empty**empty($blockList)){
40. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
41. echo "<p><strong>Fill in block</strong></p>";
42. }
43. **elseif** (empty**empty**($yearList)){
44. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
45. echo "<p><strong>Fill in year</strong></p>";
46. }
47. **elseif** (empty**empty**($sizeList)){
48. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
49. echo "<p><strong>Fill in size</strong></p>";
50. }
51. **elseif** (empty**empty**($subjectName)){
52. echo"<p><strong>Error - you have not filled in all of the required fields</strong></p>";
53. echo "<p><strong>Fill in subject name</strong></p>";
55. } **else** {
56. **for** ($count = 0; $count < $numblocks; $count++) {
57. $room = $roomList[$count];
58. $teacher = $teacherList[$count];
59. $block = $blockList[$count];
60. $year = $yearList[$count];
61. $size = $sizeList[$count];
63. //Check if room is being used during block
64. $roomCheck = mysqli\_query($dbc,"SELECT roomID FROM roomTBL WHERE block = '$block'");
65. $roomCheckAnswer = True;
66. **while**($roomID = mysqli\_fetch\_array($roomCheck,MYSQLI\_ASSOC)) {
67. **if** ($room == $roomID['roomID']){
68. $roomCheckAnswer = False;
69. echo "<p><strong>Error - Room has already been allocated to this block</strong></p>";
70. }
71. }
73. //Check if teacher is teaching during block
74. $teacherCheck = mysqli\_query($dbc,"SELECT teacherID FROM TeacherClassRoomCourse WHERE block = '$block'");
75. $teacherCheckAnswer = True;
76. **while**($teacherID = mysqli\_fetch\_array($teacherCheck,MYSQLI\_ASSOC)) {
77. **if** ($teacher == $teacherID['teacherID']){
78. $teacherCheckAnswer = False;
79. echo "<p><strong>Error - Teacher has already been allocated to this block</strong></p>";
80. }
81. }
82. **if** ($teacherCheckAnswer && $roomCheckAnswer == True) {
83. $classID = $teacherID+$subject+$block;
84. $TeacherClassRoomCourse = "INSERT INTO TeacherClassRoomCourse(teacherID,block,roomID,courseID,year,classID)VALUES('$teacher','$block','$room','$subject','$year','$classID')";
85. $roomTBL = "INSERT INTO roomTBL(roomID, size, block)VALUES('$room','$size','$block')";
86. $TeacherBlock = "INSERT INTO TeacherBlock(teacherID,block)VALUES('$teacher','$block')";
87. $subjectTBL = "INSERT INTO courseTBL(courseID,courseName)VALUES('$subject','$subjectName')";
88. mysqli\_query($dbc, $TeacherClassRoomCourse);
89. mysqli\_query($dbc, $roomTBL);
90. mysqli\_query($dbc, $TeacherBlock);
91. mysqli\_query($dbc, $subjectTBL);
92. }
94. }
95. mysqli\_close($dbc);
96. }
97. }
99. ?>
101. <html>
102. <script>

105. **function** createTable() {
107. document.getElementById('dtable').innerHTML = '';
108. **var** Rows = document.getElementById("numblocks").value;
110. **var** table = document.getElementById("dtable");
112. **var** header = table.createTHead(0);
114. **var** row = header.insertRow(0);
116. **var** cell1 = row.insertCell(0);
117. **var** cell2 = row.insertCell(1);
118. **var** cell3 = row.insertCell(2);
119. **var** cell4 = row.insertCell(3);
120. **var** cell5 = row.insertCell(4);
122. cell1.innerHTML = "<b>RoomID</b>";
123. cell2.innerHTML = "<b>TeacherID</b>";Â Â
124. cell3.innerHTML = "<b>Block</b>";
125. cell4.innerHTML = "<b>Year</b>";
126. cell5.innerHTML =Â "<b>Size</b>";

129. //Table loop
131. **for** (**var** count = 0; count < Rows; count++) {
132. **var** row = table.insertRow(1);
134. **var** cell1 = row.insertCell(0);
135. **var** cell2 = row.insertCell(1);
136. **var** cell3 = row.insertCell(2);
137. **var** cell4 = row.insertCell(3);
138. **var** cell5 = row.insertCell(4);
140. //room,teacher,block,year,size input
141. cell1.innerHTML = '<input id = "room" type = "text"  name="room[]" placeholder="" style="width: 60px;"/>';
142. cell2.innerHTML ='<input id = "teacher" type = "text"  name="teacher[]" placeholder="" style="width: 60px;"/>';
143. cell3.innerHTML ='<input id = "block" type = "text"  name="block[]" placeholder="" style="width: 60px;"/>';
144. cell4.innerHTML ='<input id = "year" type = "text"  name="year[]" placeholder=""  style="width: 60px;"/>';
145. cell5.innerHTML ='<input id = "size" type = "number"  name="size[]" placeholder="" style="width: 60px;"/><br />';
146. }
147. }
149. </script>


153. <body>
155. <!--Subject **and** number of block inputs -->
156. <div id='div1'>
157. <form action="" method="post" **class**="basic-grey"/>
158. <h2>Teacher Sign Up   </h2>

161. <label>
162. <label>
163. <span>SubjectID: </span>
164. <input id = "subject" type = "text"  name="subject" placeholder="" />
166. </label>
167. <label>
168. <span>SubjectName: </span>
169. <input id = "subjectName" type = "text"  name="subjectName" placeholder=""/>
171. </label>
172. </label>
173. <label>
174. <span>Number of blocks: </span>
175. <input id = "numblocks" type = "number"  name="numblocks" placeholder="" value="<?php echo $numblocks; ?>" onchange='createTable()'/>
177. </label>
178. <!--Creates room, teacher, block, year **and** size input table -->
179. <div id='div3'><table id="dtable"></table></div>
181. <!-- Submit button -->
182. <div id='div2'>
183. <label>
184. <span> </span>
185. <input type="submit" **class**="button" value="Submit" name = "submit" />
186. </label>
187. </div>
188. </form>
189. </div>




195. </html>
196. </body>
197. <?php
198. **include** ('footer.html');
199. ?>

## teacher allocation

### teacherAllocation.php

1. <?php //teacherAllocation.php
2. $page\_title = 'Teacher Allocation';

5. **include** ('header.html');
6. **require** ('dbconnect.php');
7. $subjectID='';
8. $valuesList=[];
9. $teacherIDList = [];
10. $roomIDList = [];
11. $classIDList = [];
12. //Get courseID for courseID drop down
13. $courseIDs = mysqli\_query($dbc,'SELECT courseID FROM courseTBL');
14. $courseIDList = [];
15. **while**($courseID = mysqli\_fetch\_array($courseIDs,MYSQLI\_ASSOC)){
16. array\_push($courseIDList, $courseID['courseID']);
17. }
19. **if**(isset($\_GET['submit'])){
20. $deleteList=$\_REQUEST['DELETE'];
21. $courseID = $\_REQUEST['subject'];
22. **for** ($count = 0; $count < count($deleteList); $count++) {
23. $delete = $deleteList[$count];
24. $deleteSplit = explode('--',$delete);
25. $teacherID = $deleteSplit[0];
26. $block = $deleteSplit[1];
27. $roomID = $deleteSplit[2];
28. $deleteTeacherClassRoomCourse = "DELETE FROM TeacherClassRoomCourse WHERE teacherID = '$teacherID' && block = '$block' && roomID = '$roomID' && courseID = '$courseID'";
29. mysqli\_query($dbc,$deleteTeacherClassRoomCourse);
30. $deleteRoomTBL = "DELETE FROM roomTBL WHERE block = '$block' && roomID = '$roomID'";
31. mysqli\_query($dbc,$deleteRoomTBL);
32. }
33. }
34. mysqli\_close($dbc);
36. ?>


40. <html>
41. <head>
43. <script>



48. //getTeacherIDList
49. **function** getTeacherIDList(subjectID){
50. **function** doXHRPromise() {
51. **return** **new** Promise(**function**(resolve, reject) {
52. **var** xmlhttp = **new** XMLHttpRequest();
53. xmlhttp.onreadystatechange = **function**() {
54. **if** (this.readyState == 4 && this.status == 200) {
55. resolve(this.responseText);
56. }
57. }
58. xmlhttp.open("GET", "getTeacherIDTeacherAllocation.php?q=" + subjectID, true);
59. xmlhttp.send();
60. });
61. }
63. doXHRPromise().then(**function**(responseText) {
64. **var** teacherIDList = JSON.parse(responseText);
65. getBlockList(subjectID,teacherIDList);
67. });
68. }
70. //getBlockList
71. **function** getBlockList(subjectID,teacherIDList){
72. **function** doXHRPromise() {
73. **return** **new** Promise(**function**(resolve, reject) {
74. **var** xmlhttp = **new** XMLHttpRequest();
75. xmlhttp.onreadystatechange = **function**() {
76. **if** (this.readyState == 4 && this.status == 200) {
77. resolve(this.responseText);
78. }
79. }
80. xmlhttp.open("GET", "getBlockTeacherAllocation.php?q=" + subjectID, true);
81. xmlhttp.send();
82. });
83. }
85. doXHRPromise().then(**function**(responseText) {
86. **var** blockList = JSON.parse(responseText);
87. getRoomIDList(subjectID,teacherIDList,blockList);
89. });
90. }
92. //getRoomIDList
93. **function** getRoomIDList(subjectID,teacherIDList,blockList){
94. **function** doXHRPromise() {
95. **return** **new** Promise(**function**(resolve, reject) {
96. **var** xmlhttp = **new** XMLHttpRequest();
97. xmlhttp.onreadystatechange = **function**() {
98. **if** (this.readyState == 4 && this.status == 200) {
99. resolve(this.responseText);
100. }
101. }
102. xmlhttp.open("GET", "getRoomIDTeacherAllocation.php?q=" + subjectID, true);
103. xmlhttp.send();
104. });
105. }
107. doXHRPromise().then(**function**(responseText) {
108. **var** roomIDList = JSON.parse(responseText);
109. isRoomFull(subjectID,teacherIDList,blockList,roomIDList);
111. });
112. }
114. //isRoomFull
115. **function** isRoomFull(subjectID,teacherIDList,blockList,roomIDList){
116. **function** doXHRPromise() {
117. **return** **new** Promise(**function**(resolve, reject) {
118. **var** xmlhttp = **new** XMLHttpRequest();
119. xmlhttp.onreadystatechange = **function**() {
120. **if** (this.readyState == 4 && this.status == 200) {
121. resolve(this.responseText);
122. }
123. }
124. **var** everythingList = roomIDList.concat(blockList);
125. xmlhttp.open("GET", "isRoomFullTeacherAllocation.php?q=" + JSON.stringify(everythingList), true);
126. xmlhttp.send();
127. });
128. }
130. doXHRPromise().then(**function**(responseText) {
131. **var** roomFullList = JSON.parse(responseText);
132. createTable(subjectID,teacherIDList,blockList,roomIDList,roomFullList);
134. });
135. }


139. //create table
140. **function** createTable(subjectID,teacherIDList,blockList,roomIDList,roomFullList) {;
141. document.getElementById('teacherTable').innerHTML = '';
142. **var** table = document.getElementById("teacherTable");
143. **var** header = table.createTHead(0);
145. **var** row = header.insertRow(0);
147. **var** cell1 = row.insertCell(0);
148. **var** cell2 = row.insertCell(1);
149. **var** cell3 = row.insertCell(2);
150. **var** cell4 = row.insertCell(3);
151. **var** cell5 = row.insertCell(4);
153. cell1.innerHTML = "<b>DELETE</b>"
154. cell2.innerHTML = "<b>TeacherID</b>";
155. cell3.innerHTML = "<b>Block</b>";
156. cell4.innerHTML = "<b>RoomID</b>";
157. cell5.innerHTML = "<b>Is Full?</b>";
159. **for** (**var** count = 0; count < teacherIDList.length; count++) {
160. **var** row = table.insertRow(1);
162. **var** cell1 = row.insertCell(0);
163. **var** cell2 = row.insertCell(1);
164. **var** cell3 = row.insertCell(2);
165. **var** cell4 = row.insertCell(3);
166. **var** cell5 = row.insertCell(4);
168. cell1.innerHTML = '<input type = "checkbox" name = "DELETE[]" class = "DELETE" value = "' + teacherIDList[count] + '--' + blockList[count] + '--' + roomIDList[count]+ '">';
169. cell2.innerHTML = teacherIDList[count];
170. cell3.innerHTML = blockList[count];
171. cell4.innerHTML = roomIDList[count];
172. cell5.innerHTML = roomFullList[count];
173. }
174. }
175. **function** everything(subjectID){
176. getTeacherIDList(subjectID);
177. }
178. </script>
179. </head>
181. <form>
183. <body>
184. <label>
185. <!-- courseID drop down menu -->
186. <select name = "subject" id = "subject" onchange="everything(this.value);">
188. <option selected='true' value="" disabled selected>SubjectID</option>
189. <?php **foreach**($courseIDList **as** $courseID => $value) {
190. echo '<option value = "'.$value.'">'.$value.'</option>';
191. }
192. ?>
193. </select>
194. </label>
196. <div id='div3'><table id="teacherTable"></table></div>
198. <label>
199. <span> </span>
200. <input type="submit" **class**="button" value="Submit" name = "submit" />
201. </label>
203. <form>
205. </html>
206. <?php
207. **include** ('footer.html');
208. ?>

### getblockteacherallocation.php

1. <?php
2. //Get classID list
3. **function** getBlock($subjectID) {
4. **require** ('dbconnect.php');
5. $blockSQL = mysqli\_query($dbc,"SELECT block FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'");
6. $blockList = [];
7. **while**($block = mysqli\_fetch\_array($blockSQL,MYSQLI\_ASSOC)) {
8. array\_push($blockList, $block['block']);
9. }
10. **return** json\_encode($blockList);
11. }
12. $subjectID = $\_REQUEST["q"];
13. print\_r(getBlock($subjectID));
14. ?>

### getroomidTeacherAllocation.php

1. <?php
2. //Get roomID list
3. **function** getRoomID($subjectID) {
4. **require** ('dbconnect.php');
5. $roomIDSQL = mysqli\_query($dbc,"SELECT roomID FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'");
6. $roomIDList = [];
7. **while**($roomID = mysqli\_fetch\_array($roomIDSQL,MYSQLI\_ASSOC)) {
8. array\_push($roomIDList, $roomID['roomID']);
9. }
10. **return** json\_encode($roomIDList);
11. }
12. $subjectID = $\_REQUEST["q"];
13. print\_r (getRoomID($subjectID));
14. ?>

### getteacheridteacherallocation.php

1. <?php
2. //Get teacherID list
4. **function** getTeacherID($subjectID) {
5. **require** ('dbconnect.php');
6. $teacherIDSQL = mysqli\_query($dbc,"SELECT teacherID FROM TeacherClassRoomCourse WHERE courseID = '$subjectID'");
7. $teacherIDList = [];
8. **while**($teacherID = mysqli\_fetch\_array($teacherIDSQL,MYSQLI\_ASSOC)) {
9. array\_push($teacherIDList, $teacherID['teacherID']);
10. }
11. **return** json\_encode($teacherIDList);
12. }
13. $subjectID = $\_REQUEST["q"];
14. print\_r (getTeacherID($subjectID));
15. ?>

### isroomfullteacherallocation.php

1. <?php
2. //Get roomFull list
3. **function** isRoomFull($everythingList) {
4. **require** ('dbconnect.php');
5. $roomFullList = [];
6. **for** ($count = 0; $count < count($everythingList)/2; $count++){
7. $roomID = $everythingList[$count];
8. $block = $everythingList[$count+count($everythingList)/2];
9. $roomSize = mysqli\_fetch\_row(mysqli\_query($dbc,"SELECT size FROM roomTBL WHERE roomID = '$roomID' && block = '$block'"))[0];
10. $amountOfStudents = mysqli\_fetch\_row(mysqli\_query($dbc, "SELECT COUNT(\*) FROM StudentCourse WHERE roomID = '$roomID' && block = '$block'"))[0];
11. **if** ($roomSize > $amountOfStudents) {
12. array\_push($roomFullList, 'NOT FULL');
13. }
14. **else** **if** ($roomSize = $amountOfStudents) {
15. array\_push($roomFullList, 'FULL');
16. }
17. **else** **if** ($roomSize < $amountOfStudents) {
18. array\_push($roomFullList, 'OVER BOOKED');
19. }
20. }
21. **return** json\_encode($roomFullList);
22. }
24. $everythingList =json\_decode( $\_REQUEST["q"]);
25. print\_r (isRoomFull($everythingList));
26. ?>

## student sign up

### studentsignup.php

1. <?php //studentSignUp.php
2. $page\_title = 'Pick Subjects';
3. **include** ('header.html');
4. **require** ('dbconnect.php');
6. $studentID='';
7. $subject=[];
8. $blockList=[];
9. $year='';
11. //Get number of blocks
12. $getNumberOfBlocks = "SELECT COUNT(\*) FROM blocks";
13. $result1 = mysqli\_query($dbc,$getNumberOfBlocks);
14. $amount = mysqli\_fetch\_row($result1);
15. $numblocks = $amount[0];
17. //Creating subject dictionary
18. $subjectArray = [];
19. $roomFullList = [];
20. **for** ($count = 0; $count <= $numblocks; $count++) {
22. //Change count to ascii character
23. $block = chr(65 + $count);
25. //numSubjects used as second loop length
26. $getNumberOfSubjects = "SELECT COUNT(\*) FROM TeacherClassRoomCourse WHERE block = '$block'" ;
27. $result2 = mysqli\_query($dbc,$getNumberOfSubjects);
28. $amount = mysqli\_fetch\_row($result2);
29. $numsubject = $amount[0];
31. $getCourseID = "SELECT courseID FROM TeacherClassRoomCourse WHERE block = '$block'";
32. $result3 = mysqli\_query($dbc,$getCourseID);
34. $subjectList = [];
35. **while**($subject = mysqli\_fetch\_array($result3,MYSQLI\_ASSOC)) {
36. array\_push($subjectList, $subject['courseID']);
38. }
40. **for** ($count2 = 0;$count2 < $numsubject; $count2++) {
41. $subjectID = $subjectList[$count2];
42. $getCourseName = "SELECT courseName FROM courseTBL WHERE courseID = '$subjectID'";
43. $subjectNameResult = mysqli\_query($dbc,$getCourseName);
44. $subjectName = mysqli\_fetch\_assoc($subjectNameResult);
45. $subjectArray[$block][$count2] = $subjectName['courseName'];
46. }
47. **for** ($count3 = 0; $count3 < count($subjectList); $count3++){
48. $subjectID = $subjectList[$count3];
49. $roomID = mysqli\_fetch\_row(mysqli\_query($dbc,"SELECT roomID FROM TeacherClassRoomCourse WHERE courseID = '$subjectID' && block = '$block'"))[0];
50. $roomSize = mysqli\_fetch\_row(mysqli\_query($dbc,"SELECT size FROM roomTBL WHERE roomID = '$roomID' && block = '$block'"))[0];
51. $amountOfStudents = mysqli\_fetch\_row(mysqli\_query($dbc, "SELECT COUNT(\*) FROM StudentCourse WHERE roomID = '$roomID' && block = '$block'"))[0];
53. **if** ($roomSize > $amountOfStudents) {
54. array\_push($roomFullList, 'NOT FULL');
55. }
56. **else** **if** ($roomSize == $amountOfStudents) {
57. array\_push($roomFullList, 'FULL');
58. }
59. }
60. }
61. **if** ($numblocks)
62. {
63. echo '<script type="text/javascript">',
64. 'createTable();',
65. '</script>';
66. }
67. **else**
68. {
69. echo "<p><strong>Error</strong></p>";
70. }
72. **if**(isset($\_GET['submit'])){
74. $studentID=$\_REQUEST['studentID'];
75. $subjectBlockList = $\_REQUEST['subject'];
76. $year = $\_REQUEST['year'];
78. **if** (empty**empty**($studentID)){
79. echo "<p><strong>Error - you have not filled in all of the required fields</strong></p>";
80. echo "<p><strong>Fill in number of studentID</strong></p>";
81. } **else** {

84. $clearStudentTBL = "DELETE FROM studentTBL WHERE studentID = '$studentID'";
85. mysqli\_query($dbc,$clearStudentTBL);
86. $clearStudentCourse = "DELETE FROM StudentCourse WHERE studentID = '$studentID'";
87. mysqli\_query($dbc,$clearStudentCourse);
88. $studentTBL = "INSERT INTO studentTBL (yearGroup, studentID)VALUES('$year','$studentID')";
89. mysqli\_query($dbc,$studentTBL);
90. **for** ($count = 0; $count < count($subjectBlockList); $count++) {
91. $subjectBlock = $subjectBlockList[$count];
92. $subjectBlockSplit = explode('0',$subjectBlock);
93. $subject = $subjectBlockSplit[0];
94. $block = $subjectBlockSplit[1];
95. $getCourseID = mysqli\_query($dbc,"SELECT courseID FROM courseTBL WHERE courseName = '$subject'");
96. $courseIDResult = mysqli\_fetch\_assoc($getCourseID);
97. $courseID = $courseIDResult['courseID'];
98. $getRoomID = mysqli\_query($dbc,"SELECT roomID FROM TeacherClassRoomCourse WHERE block = '$block' && courseID = '$courseID'");
99. $RoomIDResult = mysqli\_fetch\_assoc($getRoomID);
100. $roomID = $RoomIDResult['roomID'];
101. $studentCourse = "INSERT INTO StudentCourse (studentID,courseID,block,roomID)VALUES('$studentID','$courseID','$block','$roomID')";
102. mysqli\_query($dbc,$studentCourse);
103. }
104. }
105. }
106. mysqli\_close($dbc);
107. ?>

110. <html>
112. <script>
114. **function** createTable() {
115. **var** numblocks = <?php echo $numblocks; ?>;
116. **var** Rows = parseInt(numblocks);
117. document.getElementById('dtable').innerHTML = '';
118. **var** table = document.getElementById("dtable")
119. **var** header = table.createTHead(0);
120. **var** row = header.insertRow(0);
122. //Table header
123. **for** (**var** count = 0; count < Rows; count++) {
124. **var** character = String.fromCharCode(65 + count);
125. **var** cell = row.insertCell(count);
126. cell.innerHTML =('<b>'+character+'</b>');
127. }


131. **var** subjectArray = <?php echo json\_encode($subjectArray);?>;
132. **var** numblocks = Object.keys(subjectArray).length;
133. subjectLengthArray = [];
135. **var** roomFullList = <?php echo json\_encode($roomFullList);?>;
136. //Get amount of subjects in each block
137. **for** (**var** count2 = 0; count2 < numblocks; count2++) {
138. character = String.fromCharCode (65 + count2);
139. test = subjectArray[character];
140. **if** (test != undefined) {
141. subjectLengthArray.push(subjectArray[character].length);
142. }
143. }
144. //Get largest amount of subjects in one block value
145. subjectLength = Math.max.apply(null,subjectLengthArray);
146. console.log(roomFullList);
147. //create each cell
148. **for** (**var** count2 = 0; count2< subjectLength; count2++){
149. **var** row = table.insertRow(count2+1);
150. **for** (**var** count3 = 0; count3 < numblocks; count3++){
151. character = String.fromCharCode (65 + count3);
152. test = subjectArray[character][count2];
153. **if** (test != undefined) {
154. **if** (roomFullList[count2+count3] == 'FULL') {
155. **var** cell = row.insertCell(count3);
156. cell.innerHTML = ('');
157. }
158. **else** {
159. **var** cell = row.insertCell(count3);
160. cell.innerHTML = ('<input type = "checkbox" name = "subject[]" class = "subject" value = "' + subjectArray[character][count2] + '0' + character + '">' +subjectArray[character][count2] + '<br>');
161. }
162. }
163. }
164. }
165. }

168. </script>
169. <form>
170. <h2>Pick one subject per block  </h2>
172. <body onload="createTable()">
173. <!-- Student ID input -->
174. <label>
175. <span>Student ID: </span>
176. <input id = "studentID" type = "text"  name="studentID" placeholder="" style="width: 60px;"/>
178. </label>
180. <!-- Year Group input -->
181. <label>
182. <span>Year Group: </span>
183. <input id = "year" type = "number"  name="year" placeholder="" style="width: 60px;"/>
185. </label>
187. <!-- Subject Table -->
188. <div id='div1'><table id="dtable"></table></div>
190. <!-- Submit button -->
191. <label>
192. <span> </span>
193. <input type="submit" **class**="button" value="Submit" name = "submit" />
194. </label>
195. </body>
196. </form>
197. </html>


201. <?php
202. **include** ('footer.html');
203. ?>

## timetable

### timetable.php

1. <?php //timetable.php
2. $page\_title = 'Timetable';
3. **include** ('header.html');
4. **require**('dbconnect.php');
5. $getNumberOfBlocks = "SELECT COUNT(\*) FROM blocks";
6. $numberOfBlocks =  mysqli\_fetch\_row(mysqli\_query($dbc, $getNumberOfBlocks))[0];
7. $getNumberOfPeriods = "SELECT COUNT(\*) FROM periodSettings";
8. $numberOfPeriods =  mysqli\_fetch\_row(mysqli\_query($dbc, $getNumberOfPeriods))[0];
10. //Get period start and end times
11. $periodStartList = [];
12. $periodEndList = [];
13. **for** ($count = 1; $count <= $numberOfPeriods; $count++) {
14. $getPeriodStart = "SELECT periodStart FROM periodSettings WHERE period = '$count'";
15. $periodStart = mysqli\_fetch\_row(mysqli\_query($dbc, $getPeriodStart))[0];
16. $getPeriodEnd = "SELECT periodEnd FROM periodSettings WHERE period = '$count'";
17. $periodEnd = mysqli\_fetch\_row(mysqli\_query($dbc, $getPeriodEnd))[0];
18. array\_push($periodStartList,$periodStart);
19. array\_push($periodEndList,$periodEnd);
20. }
22. //Get the blocks for each day
23. $mondayBlocks = [];
24. $tuesdayBlocks = [];
25. $wednesdayBlocks = [];
26. $thursdayBlocks = [];
27. $fridayBlocks = [];
28. **for**  ($count = 1; $count <= $numberOfPeriods; $count++){
29. $getMonday = "SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Monday'";
30. $monday = mysqli\_fetch\_row(mysqli\_query($dbc, $getMonday))[0];
31. array\_push($mondayBlocks,$monday);
33. $getTuesday = "SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Tuesday'";
34. $tuesday = mysqli\_fetch\_row(mysqli\_query($dbc, $getTuesday))[0];
35. array\_push($tuesdayBlocks,$tuesday);
37. $getWednesday = "SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Wednesday'";
38. $wednesday = mysqli\_fetch\_row(mysqli\_query($dbc, $getWednesday))[0];
39. array\_push($wednesdayBlocks,$wednesday);
41. $getThursday = "SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Thursday'";
42. $thursday = mysqli\_fetch\_row(mysqli\_query($dbc, $getThursday))[0];
43. array\_push($thursdayBlocks,$thursday);
45. $getFriday = "SELECT block FROM BlockPeriodDay WHERE period = '$count' && day = 'Friday'";
46. $friday = mysqli\_fetch\_row(mysqli\_query($dbc, $getFriday))[0];
47. array\_push($fridayBlocks,$friday);
49. }
51. ?>
53. <script>
55. **function** getCourseID(studentID){
57. //Get the courseIDs for the subjects the student does
58. **function** doXHRPromise() {
59. **return** **new** Promise(**function**(resolve, reject) {
60. **var** xmlhttp = **new** XMLHttpRequest();
61. xmlhttp.onreadystatechange = **function**() {
62. **if** (this.readyState == 4 && this.status == 200) {
63. resolve(this.responseText);
64. }
65. }
66. xmlhttp.open("GET", "getCourseIDTimetable.php?q=" + studentID, true);
67. xmlhttp.send();
68. });
69. }
70. doXHRPromise().then(**function**(responseText) {
71. **var** courseIDList = JSON.parse(responseText);
72. getBlock(studentID,courseIDList);
73. });
74. }

77. **function** getBlock(studentID,courseIDList){
78. //Get the blocks for the subjects the student does
79. **function** doXHRPromise() {
80. **return** **new** Promise(**function**(resolve, reject) {
81. **var** xmlhttp = **new** XMLHttpRequest();
82. xmlhttp.onreadystatechange = **function**() {
83. **if** (this.readyState == 4 && this.status == 200) {
84. resolve(this.responseText);
85. }
86. }
87. xmlhttp.open("GET", "getBlockTimetable.php?q=" + studentID, true);
88. xmlhttp.send();
89. });
90. }
91. doXHRPromise().then(**function**(responseText) {
92. **var** blockList = JSON.parse(responseText);
93. getRoomID(studentID,courseIDList,blockList);
94. });
95. }
97. **function** getRoomID(studentID,courseIDList,blockList){
98. //Get the rooms for the subjects the student does
99. **function** doXHRPromise() {
100. **return** **new** Promise(**function**(resolve, reject) {
101. **var** xmlhttp = **new** XMLHttpRequest();
102. xmlhttp.onreadystatechange = **function**() {
103. **if** (this.readyState == 4 && this.status == 200) {
104. resolve(this.responseText);
105. }
106. }
107. xmlhttp.open("GET", "getRoomIDTimetable.php?q=" + studentID, true);
108. xmlhttp.send();
109. });
110. }
112. doXHRPromise().then(**function**(responseText) {
113. **var** roomIDList = JSON.parse(responseText);
114. getTeacherID(courseIDList,blockList,roomIDList);
115. });
116. }
118. **function** getTeacherID(courseIDList,blockList,roomIDList){
119. //Get the rooms for the subjects the student does
120. **function** doXHRPromise() {
121. **return** **new** Promise(**function**(resolve, reject) {
122. **var** xmlhttp = **new** XMLHttpRequest();
123. xmlhttp.onreadystatechange = **function**() {
124. **if** (this.readyState == 4 && this.status == 200) {
125. resolve(this.responseText);
126. }
127. }
128. **var** everythingArray = courseIDList.concat(blockList).concat(roomIDList);
129. xmlhttp.open("GET", "getTeacherIDTimetable.php?q=" + JSON.stringify(everythingArray), true);
130. xmlhttp.send();
131. });
132. }
133. doXHRPromise().then(**function**(responseText) {
134. **var** teacherIDList = JSON.parse(responseText);
135. createTimeTable(courseIDList,blockList,roomIDList,teacherIDList);
136. });
138. }
140. **function** createTimeTable(courseIDList,blockList,roomIDList,teacherIDList){
141. **var** mondayBlocks = <?php echo json\_encode($mondayBlocks);?>;
142. **var** tuesdayBlocks = <?php echo json\_encode($tuesdayBlocks);?>;
143. **var** wednesdayBlocks = <?php echo json\_encode($wednesdayBlocks);?>;
144. **var** thursdayBlocks = <?php echo json\_encode($thursdayBlocks);?>;
145. **var** fridayBlocks = <?php echo json\_encode($fridayBlocks);?>;
146. **var** periodStartList = <?php echo json\_encode($periodStartList);?>;
147. **var** periodEndList = <?php echo json\_encode($periodEndList);?>;
148. **var** numberOfPeriods = <?php echo json\_encode($numberOfPeriods);?>;
150. document.getElementById('timetable').innerHTML = '';
151. **var** table = document.getElementById("timetable");
152. **var** header = table.createTHead(0);
153. **var** row = header.insertRow(0);
155. **var** cell0 = row.insertCell(0);
156. **var** cell1 = row.insertCell(1);
157. **var** cell2 = row.insertCell(2);
158. **var** cell3 = row.insertCell(3);
159. **var** cell4 = row.insertCell(4);
160. **var** cell5 = row.insertCell(5);
162. cell0.innerHTML = "start - <br>end";
163. cell1.innerHTML = "Monday";
164. cell2.innerHTML = "Tuesday";
165. cell3.innerHTML = "Wednesday";
166. cell4.innerHTML = "Thursday";
167. cell5.innerHTML = "Friday";
169. **for** (count = 0; count < numberOfPeriods; count++){
170. **var** row = table.insertRow();
172. **var** cell0 = row.insertCell(0);
173. **var** cell1 = row.insertCell(1);
174. **var** cell2 = row.insertCell(2);
175. **var** cell3 = row.insertCell(3);
176. **var** cell4 = row.insertCell(4);
177. **var** cell5 = row.insertCell(5);
179. **var** periodTime = periodStartList[count] + ' - <br>' + periodEndList[count];
180. cell0.innerHTML = periodTime;
181. //Monday
182. **var** position = blockList.indexOf(mondayBlocks[count]);
183. **if** (position != -1) {
184. cell1.innerHTML = 'Course: ' + courseIDList[position] + '<br>Teacher: ' + teacherIDList[position] + '<br>Room: ' + roomIDList[position] + '<br>Block: ' + blockList[position];
185. }
186. //Tuesday
187. **var** position = blockList.indexOf(tuesdayBlocks[count]);
188. **if** (position != -1) {
189. cell2.innerHTML = 'Course: ' + courseIDList[position] + '<br>Teacher: ' + teacherIDList[position] + '<br>Room: ' + roomIDList[position] + '<br>Block: ' + blockList[position];
190. }
191. //Wednesday
192. **var** position = blockList.indexOf(wednesdayBlocks[count]);
193. **if** (position != -1) {
194. cell3.innerHTML = 'Course: ' + courseIDList[position] + '<br>Teacher: ' + teacherIDList[position] + '<br>Room: ' + roomIDList[position] + '<br>Block: ' + blockList[position];
195. }
196. //Thursday
197. **var** position = blockList.indexOf(thursdayBlocks[count]);
198. **if** (position != -1) {
199. cell4.innerHTML = 'Course: ' + courseIDList[position] + '<br>Teacher: ' + teacherIDList[position] + '<br>Room: ' + roomIDList[position] + '<br>Block: ' + blockList[position];
200. }
201. //Friday
202. **var** position = blockList.indexOf(fridayBlocks[count]);
203. **if** (position != -1) {
204. cell5.innerHTML = 'Course: ' + courseIDList[position] + '<br>Teacher: ' + teacherIDList[position] + '<br>Room: ' + roomIDList[position] + '<br>Block: ' + blockList[position];
205. }
206. }
207. }
208. </script>
210. <label>
211. <span>Student ID: </span>
212. <input id = "studentID" type = "text"  name="studentID" placeholder="" style="width: 60px;" onchange = 'getCourseID(this.value)'/>
214. </label>
215. <div id='div'><table id="timetable"></table></div>
216. <?php
217. **include** ('footer.html');
218. ?>

### getblocktimetable.php

1. <?php
2. //Get block list
4. **function** getBlock($studentID) {
5. **require** ('dbconnect.php');
6. $blockSQL = mysqli\_query($dbc,"SELECT block FROM StudentCourse WHERE studentID = '$studentID'");
7. $blockList = [];
8. **while**($block = mysqli\_fetch\_array($blockSQL,MYSQLI\_ASSOC)) {
9. array\_push($blockList, $block['block']);
10. }
11. **return** json\_encode($blockList);
12. }
13. $studentID = $\_REQUEST["q"];
14. print\_r (getBlock($studentID));
15. ?>

### getcourseidtimetable.php

1. <?php
2. //Get courseID list
4. **function** getCourseID($studentID) {
5. **require** ('dbconnect.php');
6. $courseIDSQL = mysqli\_query($dbc,"SELECT courseID FROM StudentCourse WHERE studentID = '$studentID'");
7. $courseIDList = [];
8. **while**($courseID = mysqli\_fetch\_array($courseIDSQL,MYSQLI\_ASSOC)) {
9. array\_push($courseIDList, $courseID['courseID']);
10. }
11. **return** json\_encode($courseIDList);
12. }
13. $studentID = $\_REQUEST["q"];
14. print\_r (getCourseID($studentID));
15. ?>

### getroomidtimetable.php

1. <?php
2. //Get roomID list
4. **function** getRoomID($studentID) {
5. **require** ('dbconnect.php');
6. $roomIDSQL = mysqli\_query($dbc,"SELECT roomID FROM StudentCourse WHERE studentID = '$studentID'");
7. $roomIDList = [];
8. **while**($roomID = mysqli\_fetch\_array($roomIDSQL,MYSQLI\_ASSOC)) {
9. array\_push($roomIDList, $roomID['roomID']);
10. }
11. **return** json\_encode($roomIDList);
12. }
13. $studentID = $\_REQUEST["q"];
14. print\_r (getRoomID($studentID));
15. ?>

### getteacheridTimetable.php

1. <?php
2. //Get teacherID
4. **function** getTeacherID($everythingArray) {
5. **require** ('dbconnect.php');
6. $length = count($everythingArray)/3;
7. $teacherIDList = [];
8. **for**  ($count = 0; $count < $length; $count++){
9. $courseID = $everythingArray[$count];
10. $block = $everythingArray[$count+$length];
11. $roomID = $everythingArray[$count+$length+$length];
12. $getTeacherID = "SELECT teacherID FROM TeacherClassRoomCourse WHERE courseID = '$courseID' && block = '$block' && roomID = '$roomID'";
13. $teacherIDSQL = mysqli\_query($dbc,$getTeacherID);
14. **while**($teacherID = mysqli\_fetch\_array($teacherIDSQL,MYSQLI\_ASSOC)) {
15. array\_push($teacherIDList, $teacherID['teacherID']);
16. }
17. }
18. **return** json\_encode($teacherIDList);
19. }
20. $everythingArray = json\_decode($\_REQUEST["q"]);
21. print\_r(getTeacherID($everythingArray));
22. ?>

## settings

### settings.php

1. <?php //settings.php
2. $page\_title = 'Settings';
3. **include** ('header.html');
4. $numberOfPeriods ='';
5. $startTimeList = [];
6. $endTimeList = [];
7. $numberOfBlocks = '';
8. $blockList = [];
10. **if**(isset($\_POST['submit'])){
11. $startTimeList=$\_POST['startTime'];
12. $endTimeList =$\_POST['endTime'];
13. $blockList =$\_POST['block'];
14. $numberOfBlocks = $\_POST['numberOfBlocks'];
15. $numberOfPeriods = $\_POST['numberOfPeriods'];
16. **require** ('dbconnect.php');
17. $clearTable = "TRUNCATE TABLE periodSettings";
18. $clear = mysqli\_query($dbc, $clearTable);
19. $clearTable = "TRUNCATE TABLE BlockPeriodDay";
20. $clear = mysqli\_query($dbc, $clearTable);
21. $clearTAble = "TRUNCATE TABLE blocks";
22. $clear = mysqli\_query($dbc, $clearTable);
23. **for** ($count = 0; $count < $numberOfPeriods; $count++) {
24. $startTime = $startTimeList[$count];
25. $endTime = $endTimeList[$count];
26. $period = $count + 1;
27. $periodSettings = "INSERT INTO periodSettings(period,periodStart,periodEnd)VALUES('$period','$startTime','$endTime')";
28. mysqli\_query($dbc, $periodSettings);
29. }
30. **for** ($count1 = 0; $count1 < $numberOfPeriods; $count1 ++){
31. **for** ($count2 = 0; $count2 < 5; $count2 ++){
32. $block = $blockList[($count1\*5)+$count2];
33. $period = $count1 + 1;
34. **if** ($count2 == 0){
35. $day = "Monday";
36. }
37. **if** ($count2 == 1){
38. $day = "Tuesday";
39. }
40. **if** ($count2 == 2){
41. $day = "Wednesday";
42. }
43. **if** ($count2 == 3){
44. $day = "Thursday";
45. }
46. **if** ($count2 == 4){
47. $day = "Friday";
48. }
49. $BlockPeriodDay = "INSERT INTO BlockPeriodDay(block,period,day)VALUES('$block','$period','$day')";
50. mysqli\_query($dbc, $BlockPeriodDay);
51. $blocksSQL = "INSERT INTO blocks(block)VALUES('$block')";
52. mysqli\_query($dbc, $blocksSQL);
53. }
54. }
55. mysqli\_close($dbc);
56. }
57. ?>

60. <html>
61. <script>

64. **function** createPeriodTable(numberOfPeriods) {
65. document.getElementById('periodTable').innerHTML = '';
66. **var** table = document.getElementById("periodTable");

69. //Header
70. **var** header = table.createTHead(0);
71. **var** row = header.insertRow(0);
72. **var** cell = row.insertCell(0);
73. cell.innerHTML = "Period: ";
74. **for** (**var** count = 0; count<numberOfPeriods; count++) {
75. **var** cell = row.insertCell(count+1);
76. cell.innerHTML = (count+1).toString();
77. }
79. //Start time input
80. **var** row = table.insertRow(1);
81. **var** cell = row.insertCell(0);
82. cell.innerHTML = "Start time: ";
83. **for** (**var** count = 0; count<numberOfPeriods; count++) {
84. **var** cell = row.insertCell(count+1);
85. cell.innerHTML = '<input id = "startTime" type = "time"  name="startTime[]" placeholder=""  style="width: 100px;"/>';
86. }
88. //End time input
89. **var** row = table.insertRow(2);
90. **var** cell = row.insertCell(0);
91. cell.innerHTML = "End time: ";
92. **for** (**var** count = 0; count<numberOfPeriods; count++) {
93. **var** cell = row.insertCell(count+1);
94. cell.innerHTML = '<input id = "endTime" type = "time"  name="endTime[]" placeholder=""  style="width: 100px;"/>';
95. }
96. }
98. **function** createBlockTable(numberOfBlocks,numberOfPeriods){
100. document.getElementById('blockTable').innerHTML = '';
101. **var** table = document.getElementById("blockTable");
103. **var** header = table.createTHead(0);
104. //Title Row
105. **var** row = header.insertRow(0);
107. **var** cell0 = row.insertCell(0);
108. **var** cell1 = row.insertCell(1);
109. **var** cell2 = row.insertCell(2);
110. **var** cell3 = row.insertCell(3);
111. **var** cell4 = row.insertCell(4);
112. **var** cell5 = row.insertCell(5);
114. cell0.innerHTML = "Period";
115. cell1.innerHTML = "Mon";
116. cell2.innerHTML = "Tue";
117. cell3.innerHTML = "Wed";
118. cell4.innerHTML = "Thu";
119. cell5.innerHTML = "Fri";
121. **for** (**var** count = 0; count<numberOfPeriods; count++) {
122. **var** row = table.insertRow(count+1);
124. //period column
125. **var** cell0 = row.insertCell(0);
126. cell0.innerHTML = count+1;
128. //loops for number of days
129. **for** (**var** count2 = 1; count2<6; count2++) {
130. blockList = [];
131. **var** cell = row.insertCell(count2);
132. **for** (**var** count3 = 0; count3<numberOfBlocks; count3++) {
133. **var** block = String.fromCharCode(65 + count3);
134. blockList.push(block);

137. }
138. selectBox = ['<select  name = "block[]"><option selected="true" value="" disabled selected>Block</option>'];
139. **for** (**var** count4 = 0; count4<numberOfBlocks; count4++){
140. selectBox.push('<option value = "');
141. selectBox.push(blockList[count4]);
142. selectBox.push('">');
143. selectBox.push(blockList[count4]);
144. selectBox.push('</option>');
145. }
146. selectBox.push('<option value = "0">NONE</option>');
147. cell.innerHTML = selectBox.join("");
148. }
149. }
151. }
152. </script>

155. <body>
156. <form action="" method="post" **class**="basic-grey"/>
157. <h2>Settings</h2>
158. <label>
159. <span>Number Of Periods: </span>
160. <input id = "numberOfPeriods" type = "number"  name="numberOfPeriods" id = "numberOfPeriods" placeholder="" value="<?php echo $numberOfPeriods; ?>" style="width: 60px;" onchange = 'createPeriodTable(this.value)'/>
161. </label
162. <!-- Period Time Table -->
163. <div id='div1'><table id="periodTable"></table></div>
165. <span>Number Of Blocks: </span>
166. <input id = "numberOfBlocks" type = "number"  name="numberOfBlocks" placeholder="" value="<?php echo $numberOfBlocks; ?>" style="width: 60px;" onchange = 'createBlockTable(this.value,document.getElementById("numberOfPeriods").value)'/>
168. <div id='div1'><table id="blockTable"></table></div>
170. <div id='div2'>
171. <label>
172. <span> </span>
173. <input type="submit" **class**="button" value="Submit" name = "submit" />
174. </label>
175. </div>
177. </body>
178. </html>
180. <?php
181. **include** ('footer.html');
182. ?>

## header.html

1. **<head>**
2. **<title><?php** echo $page\_title; **?></title>**
3. **<link** rel='stylesheet' href='style.css' type='text/css' media='screen' **/>**
4. **<meta** http-equiv='content=type' content='text/html; charset=ust=8'**/>**
5. **</head>**
6. **<body>**
7. **<div** id="header"**>**
8. **<h1>**My Collyers**</h1>**
9. **</div>**
10. **<div** id="navigation"**>**
11. **<ul>**
12. **<li><a** href="teacherSignUp.php"**>**Teacher Sign Up**</a></li>**
13. **<li><a** href="teacherAllocation.php"**>**Teacher Allocation**</a></li>**
14. **<li><a** href="studentSignUp.php"**>**Student Sign Up**</a></li>**
15. **<li><a** href="timetable.php"**>**Time Table**</a></li>**
16. **<li><a** href="settings.php"**>**Settings**</a></li>**
17. **</ul>**
18. **</div>**
19. **<div** id='content'**>**

## footer.html

1. **</div>**
3. **<div** id='footer'**>**
4. **<p>**Copyright © Dylan Ritchings | 2016**</p>**
5. **</div>**
6. **</body>**
7. **</html>**

## dbconnect.php

1. <?php
3. DEFINE ('DB\_SERVER','collycomp.uk');
4. DEFINE ('DB\_USER','15ritchingd03');
5. DEFINE ('DB\_PASSWORD','SY156675');
6. DEFINE ('DB\_NAME', '15ritchingd03\_3');
8. $dbc=@mysqli\_connect (DB\_SERVER,DB\_USER,DB\_PASSWORD,DB\_NAME) OR **die** ('Connection failed: '.mysqli\_connect\_error());

11. mysqli\_set\_charset($dbc, 'utf8');
12. ?>

## style.css

1. \* {
2. **border**:0;
3. **margin**:0;
4. **padding**:0;
5. }
7. a {
8. **color**:#777;
9. **text-decoration**:none;
10. }
12. a:hover {
13. **color**:#333;
14. **text-decoration**:none;
15. }
17. table,th,td{**border**:1px solid black;}

20. body {
21. **background**:#ffffff;
22. **color**:#555;
23. **font**:0.8em Arial, Helvetica, sans-serif;
24. }

27. #header {
28. **border-bottom**:1px solid #999;
29. **height**:80px;
30. **margin**:0 auto;
31. **width**:751px;
32. }
33. #header h1 {
34. **color**:#888;
35. **font-size**:300%;
36. **letter-spacing**:-3px;
37. **text-align**:right;
38. **padding**:5px;
39. **margin-bottom**:-20px;
40. }
41. #header h2 {
42. **color**:#CCC;
43. **font-size**:200%;
44. **letter-spacing**:-2px;
45. **text-align**:left;
46. }
48. #content {
49. **height**:auto;
50. **margin**:0 auto;
51. **padding**:0 0 20px;
52. **width**:751px;
53. }
54. #content h1 {
55. **border-bottom**:1px dashed #999;
56. **font-size**:1.8em;
57. **padding**:20px 0 0;
58. }
59. #content p {
60. **padding**:20px 20px 0;
61. }

64. #navigation {
65. **background**:#fafafa;
66. **border-right**:1px solid #999;
67. **margin**:0 auto;
68. **width**:750px;
69. **height**:40px;
70. **list-style**:none;
71. }
72. #navigation li {
73. **border-left**:1px solid #999;
74. **float**:left;
75. **width**:149px;
76. **list-style**:none;
77. }
78. #navigation a {
79. **color**:#555;
80. **display**:block;
81. **line-height**:40px;
82. **text-align**:center;
83. }
84. #navigation a:hover {
85. **background**:#e3e3e3;
86. **color**:#555;
87. }
88. #navigation .active {
89. **background**:#e3e3e3;
90. **color**:#777;
91. }

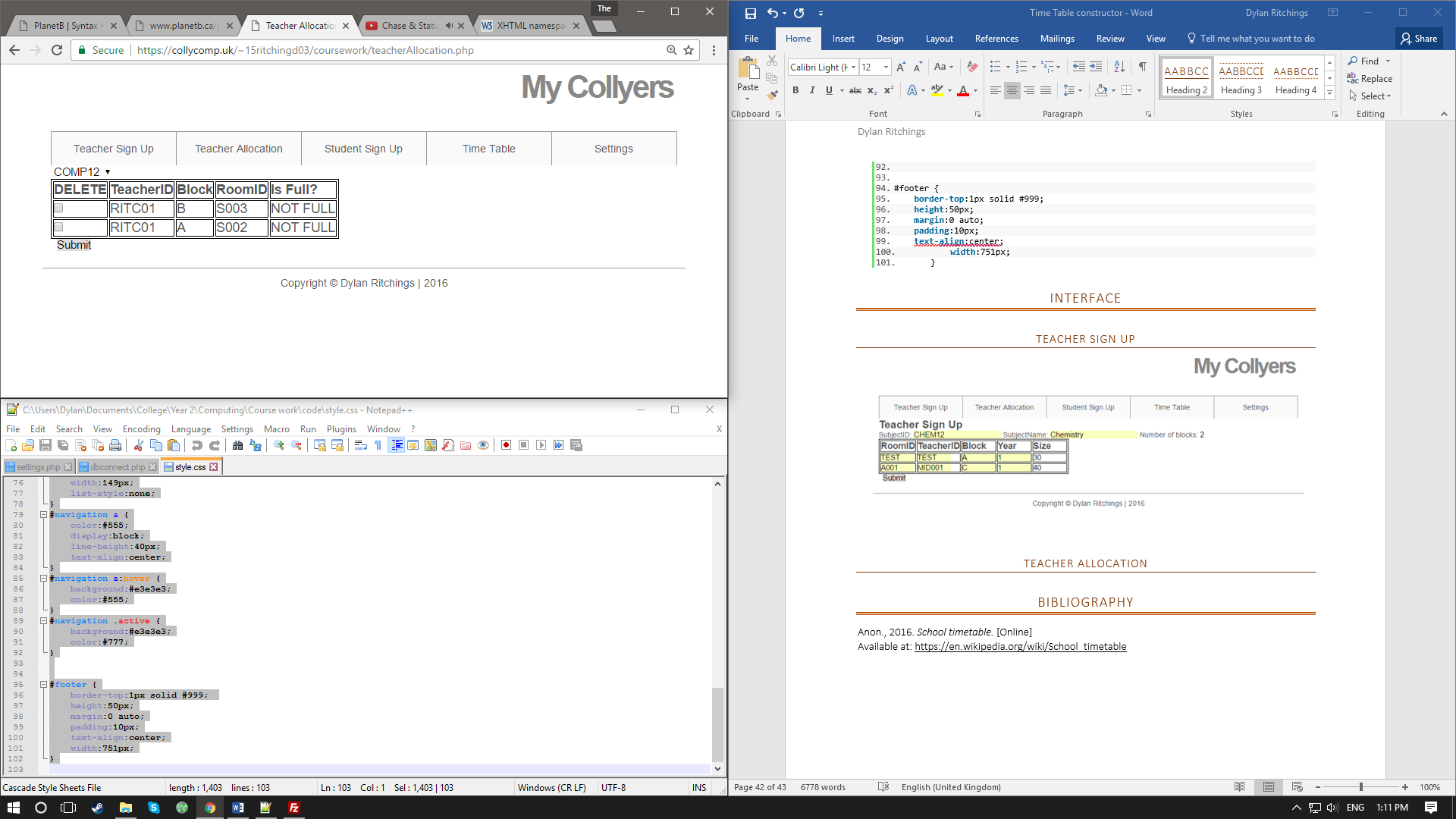
94. #footer {
95. **border-top**:1px solid #999;
96. **height**:50px;
97. **margin**:0 auto;
98. **padding**:10px;
99. **text-align**:center;
100. **width**:751px;
101. }

# interface

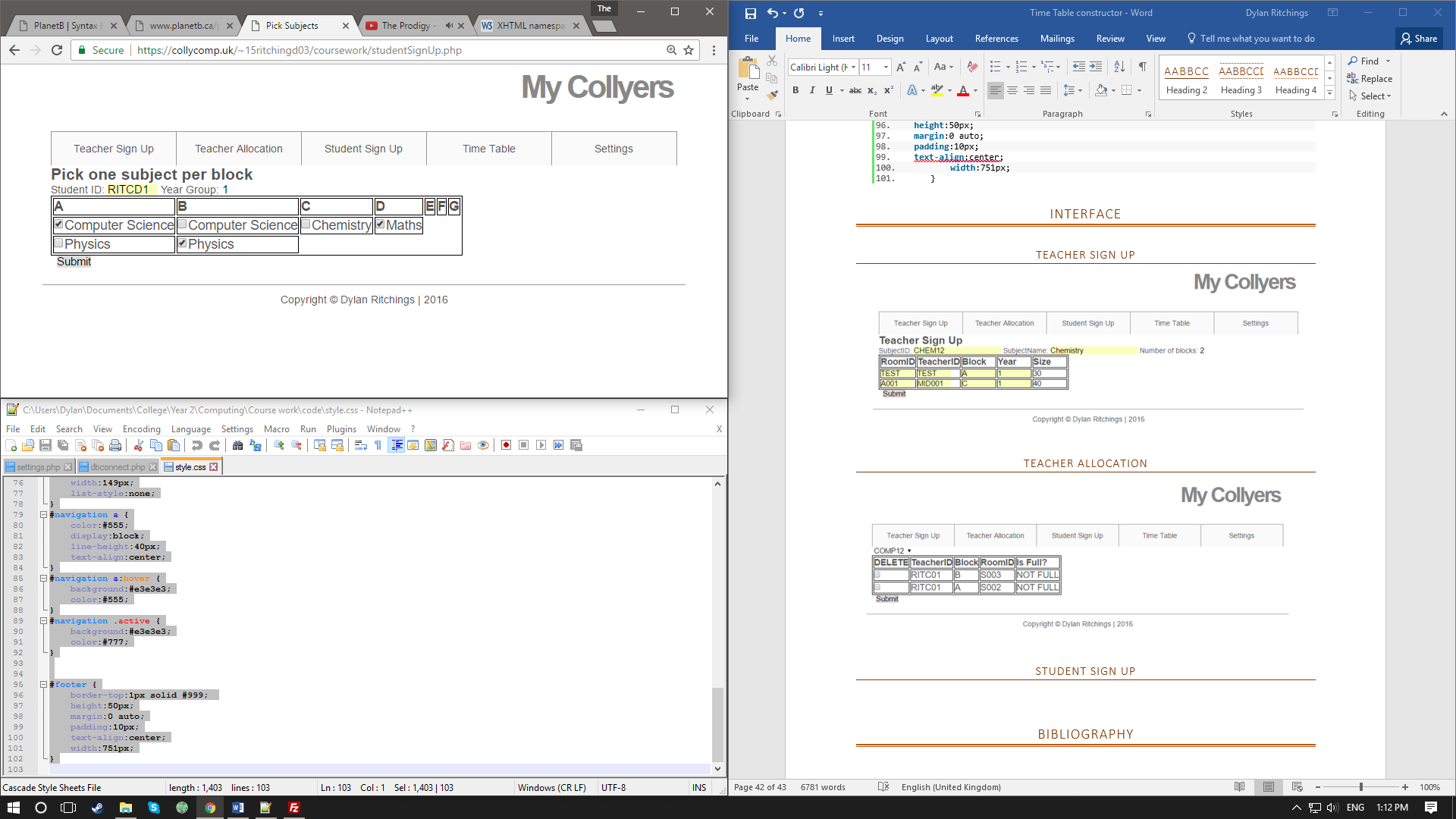
## teacher sign up



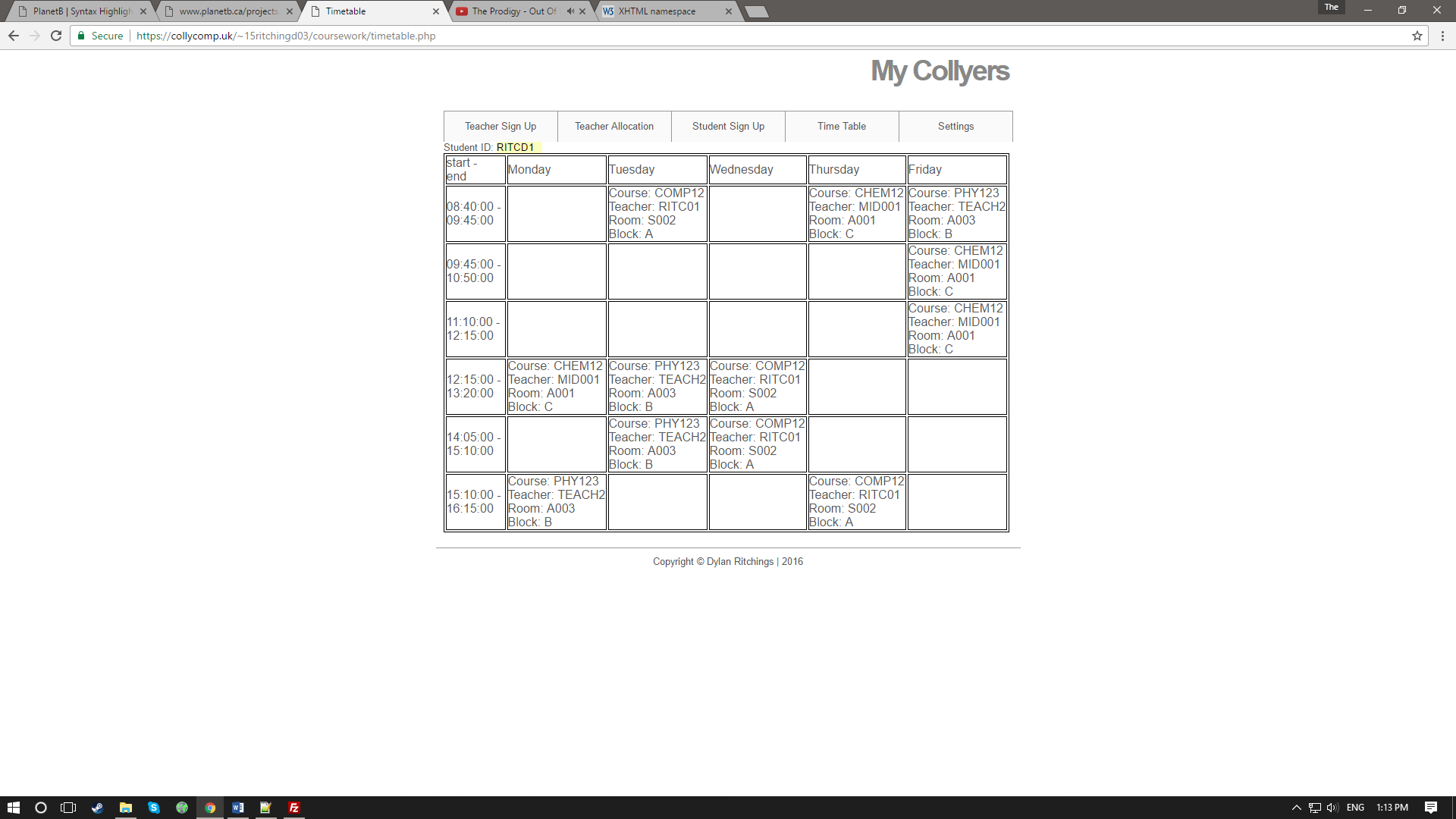
## teacher allocation



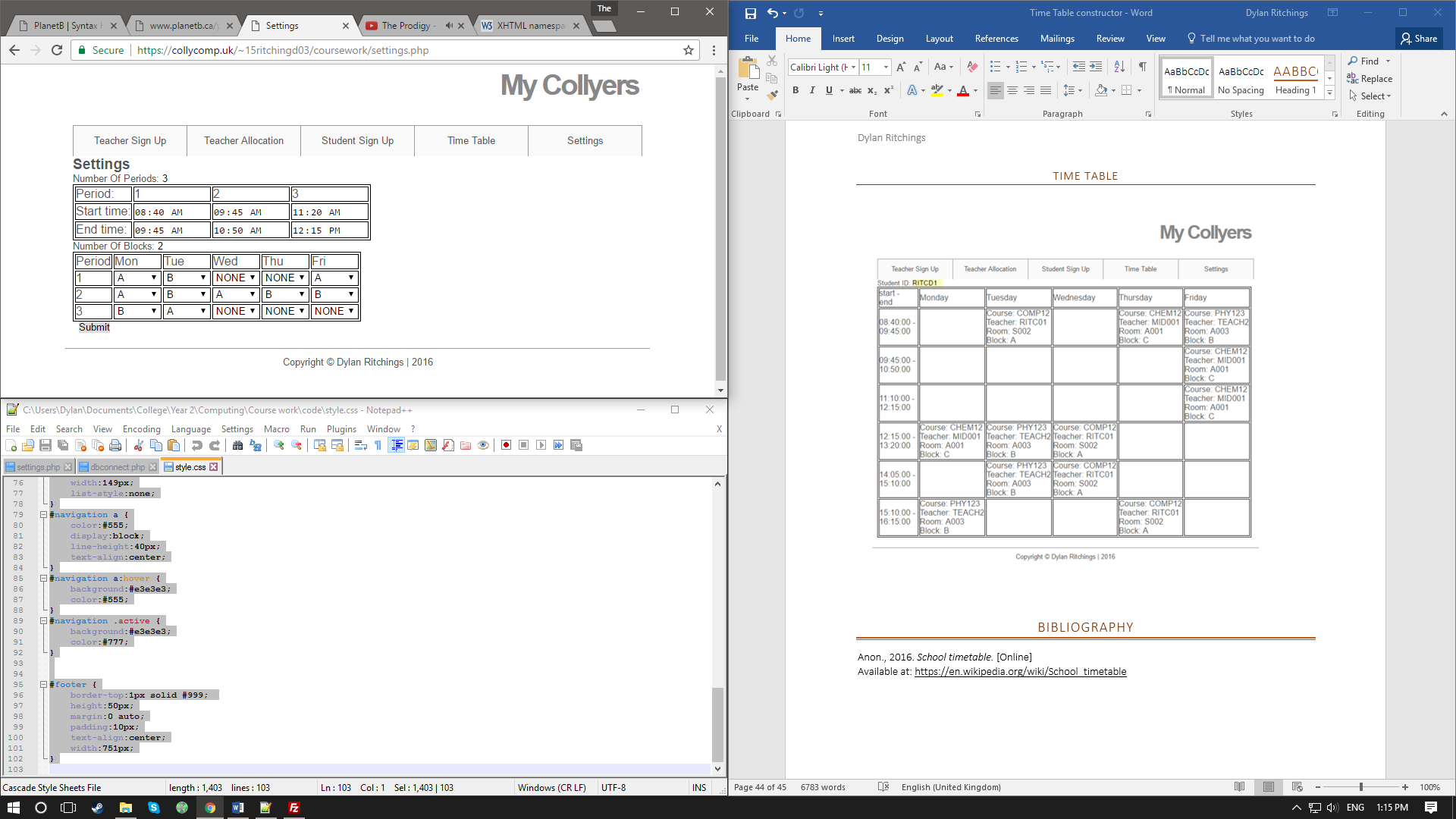
## student sign up



## time table



## settings

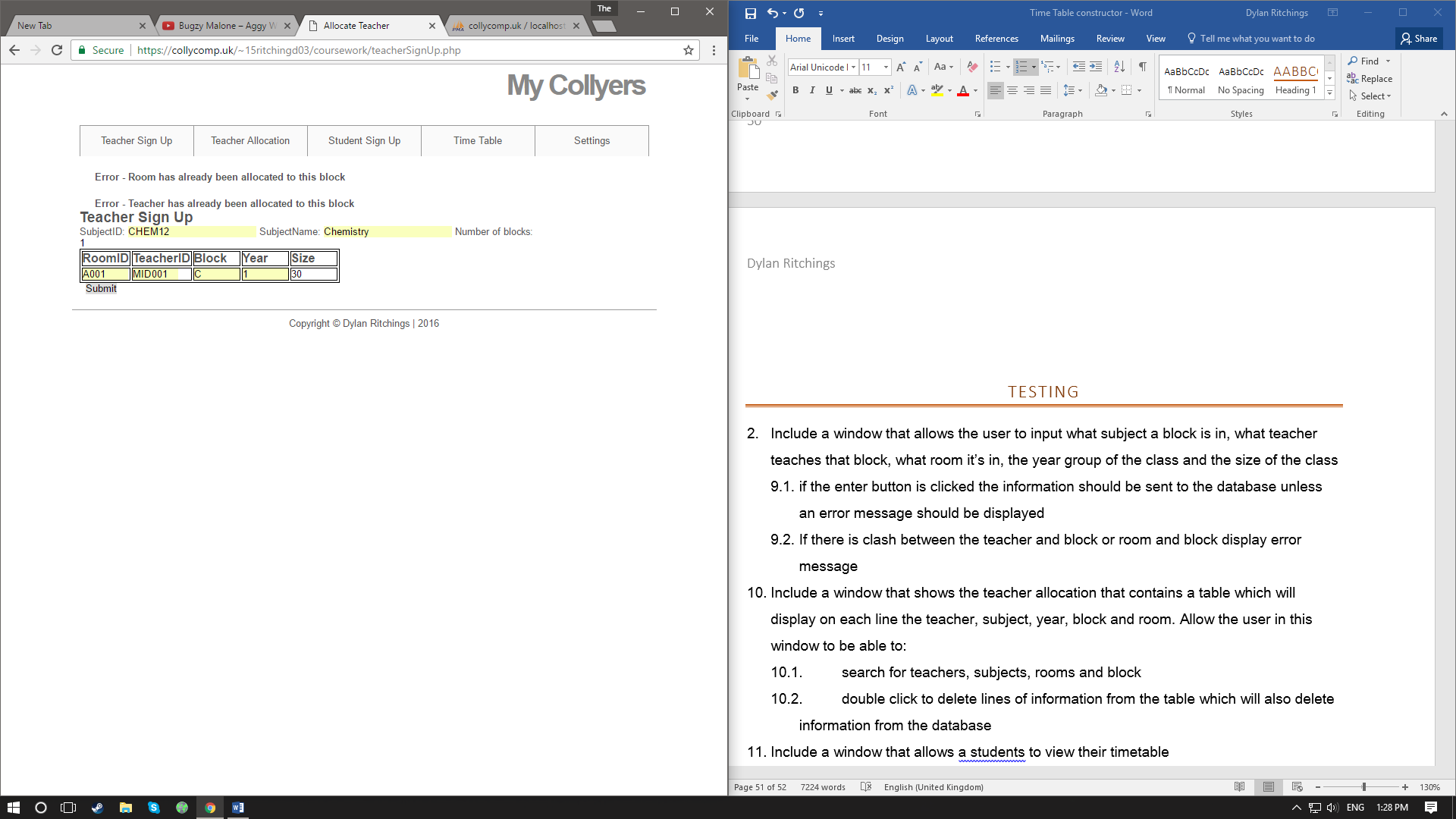


# Testing

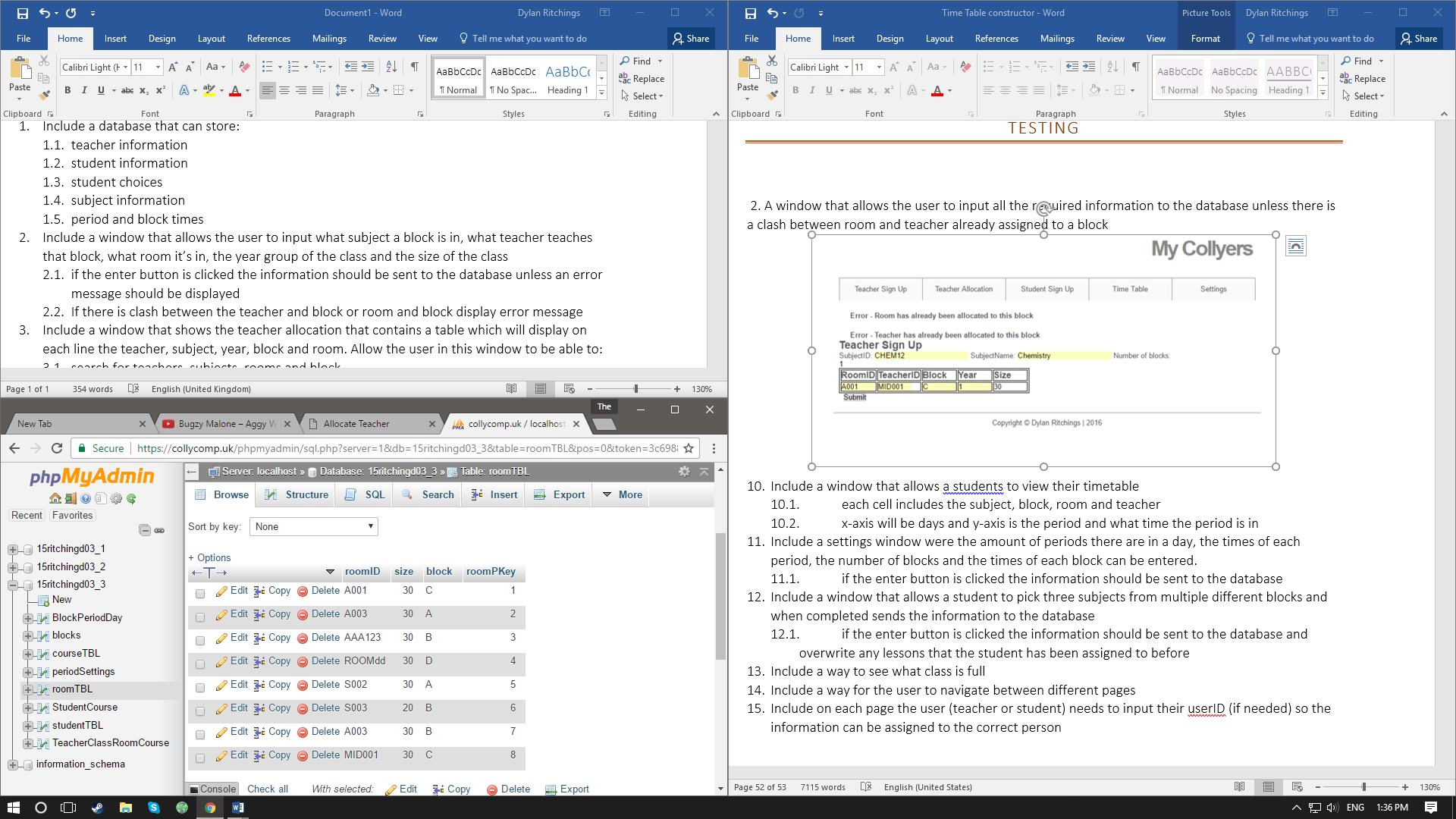
## objective 2.

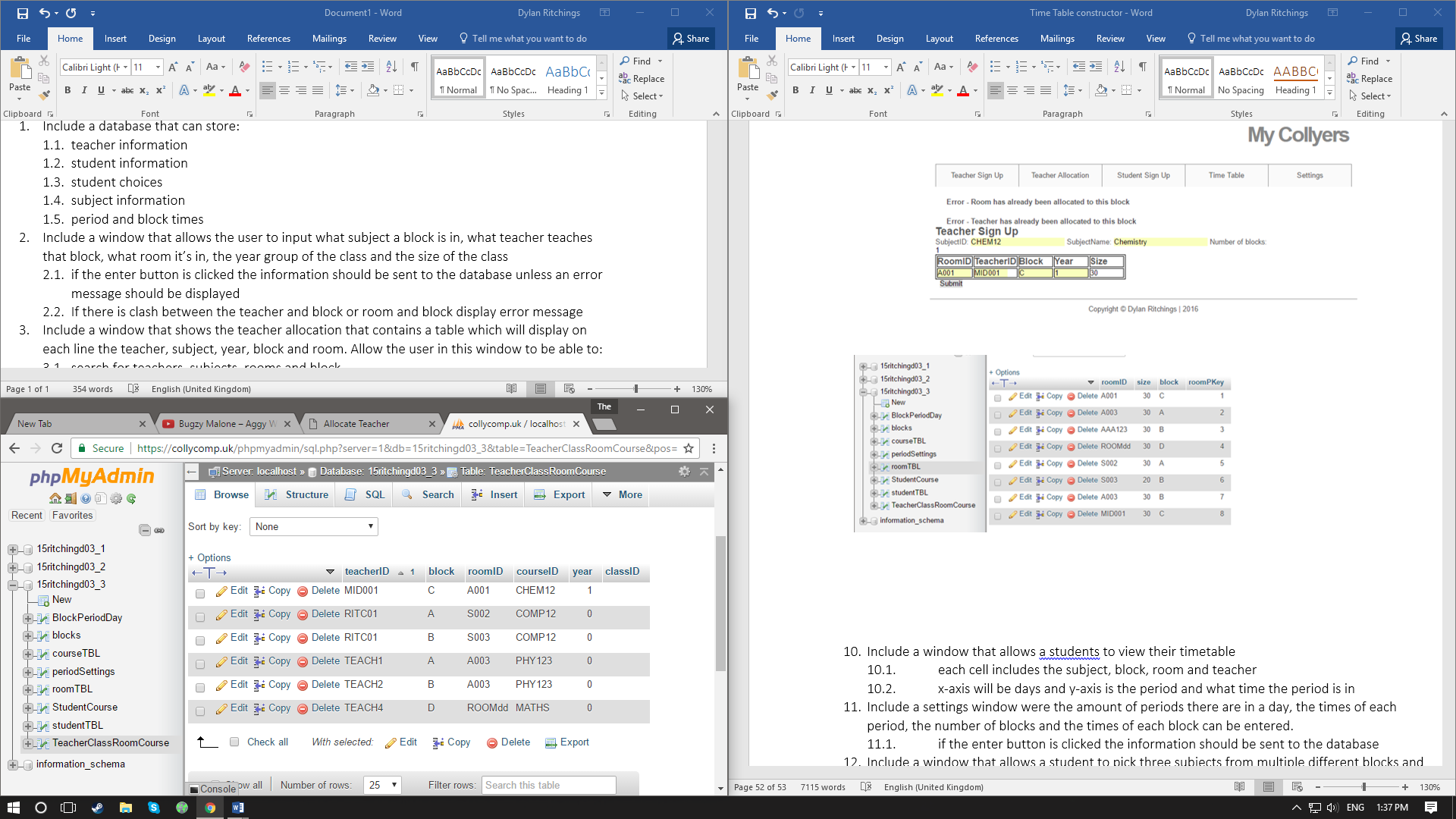
A window that allows the user to input what subject a block is in, what teacher teaches that block, what room it’s in, the year group of the class and the size of the class. When a teacher or room is already assigned to a block an error message appears.

### website



### database

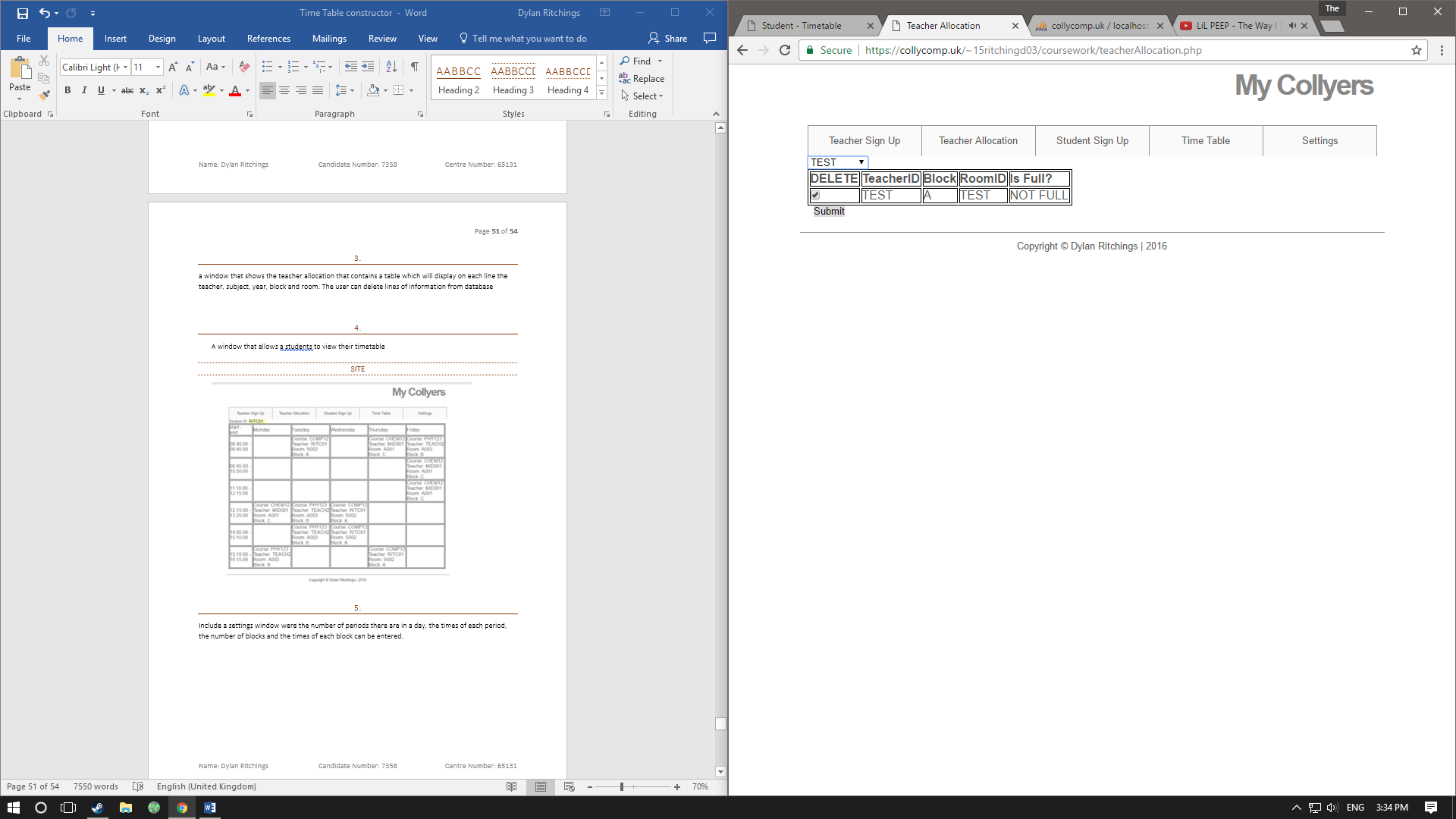
roomTBL TeacherClassRoomCourse



## objective 3.

a window that shows the teacher allocation that contains a table which will display on each line the teacher, subject, year, block and room. The user can delete lines of information from database

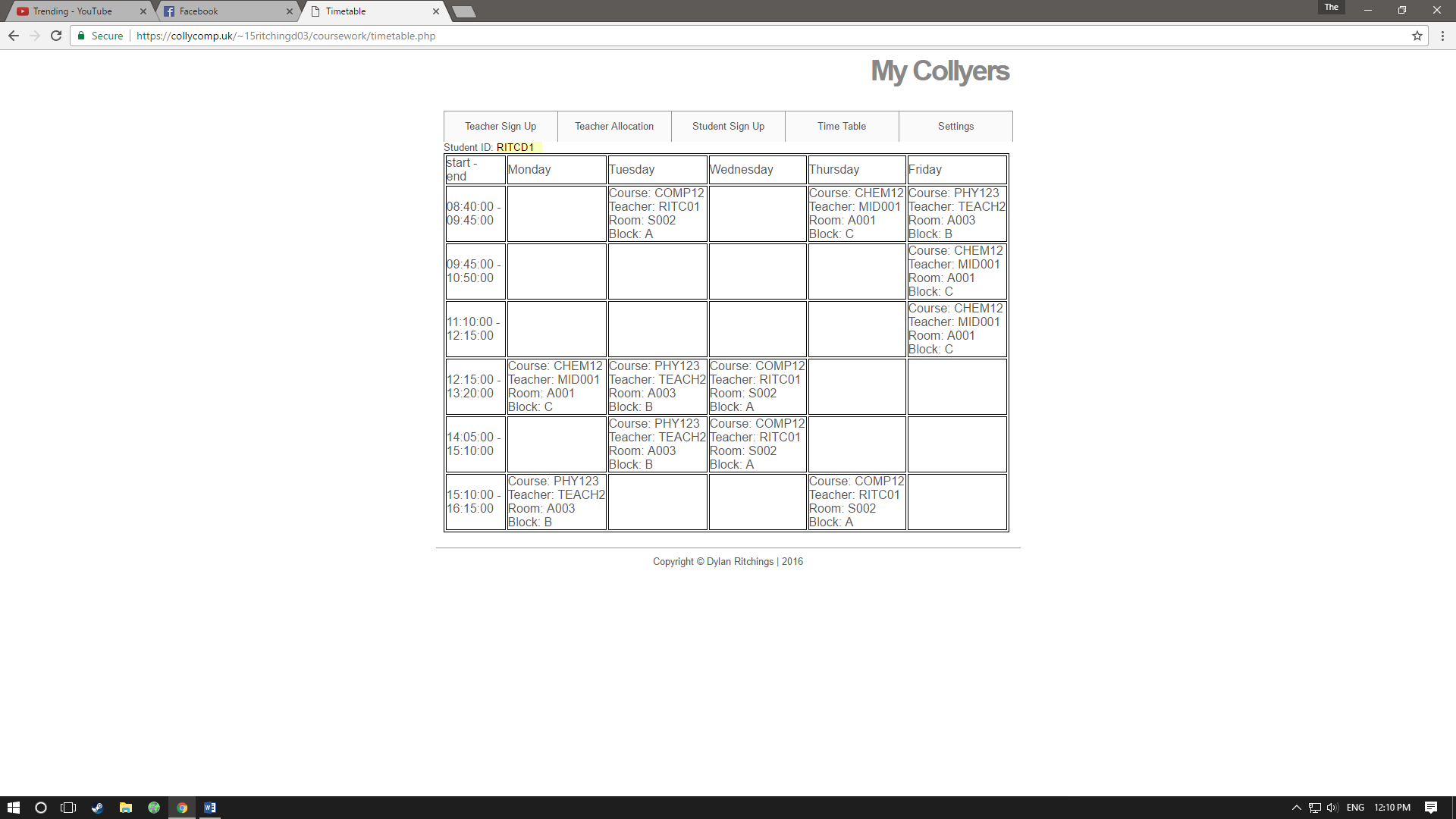
### website



## objective 4.

A window that allows a students to view their timetable

### website

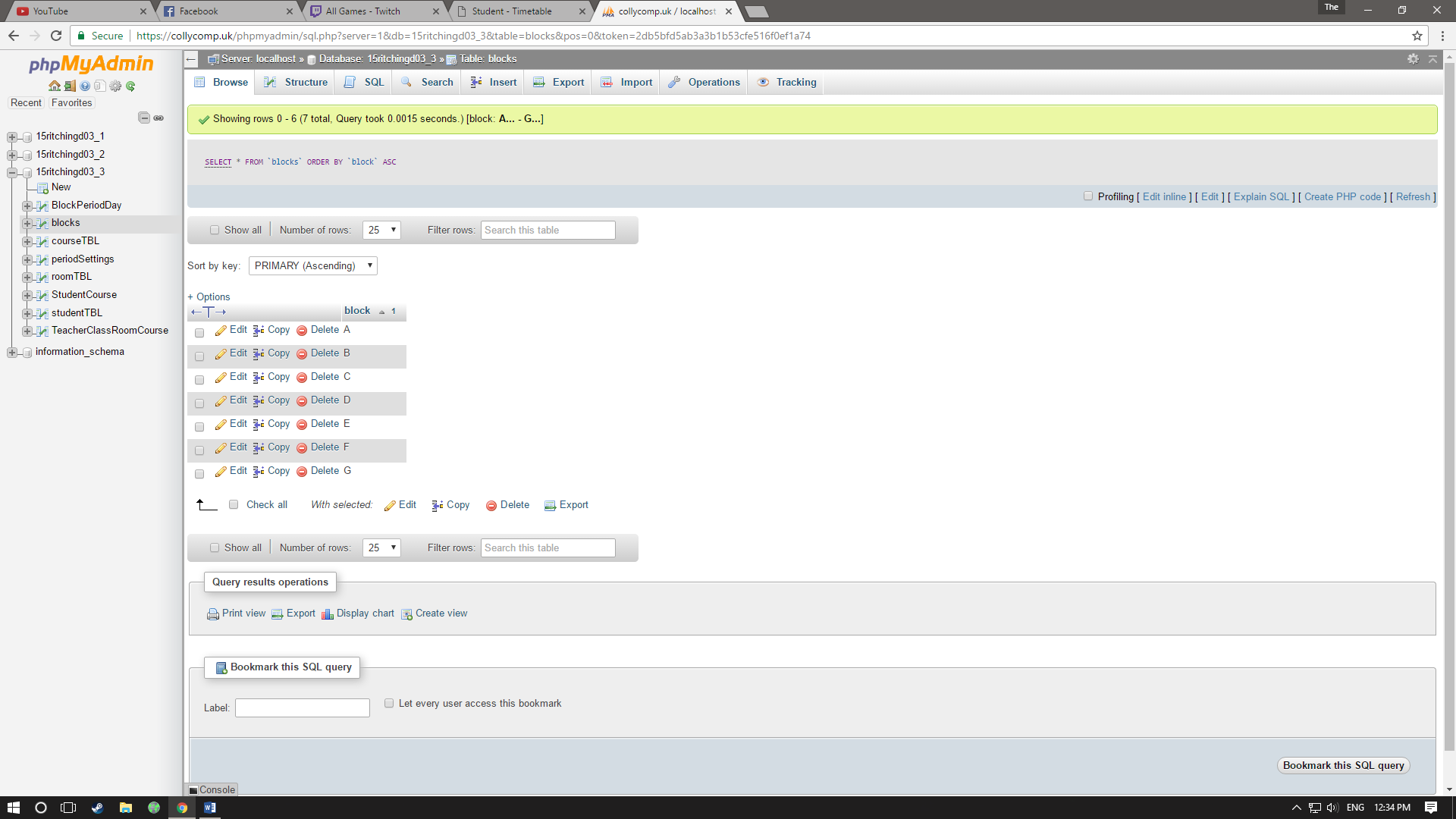
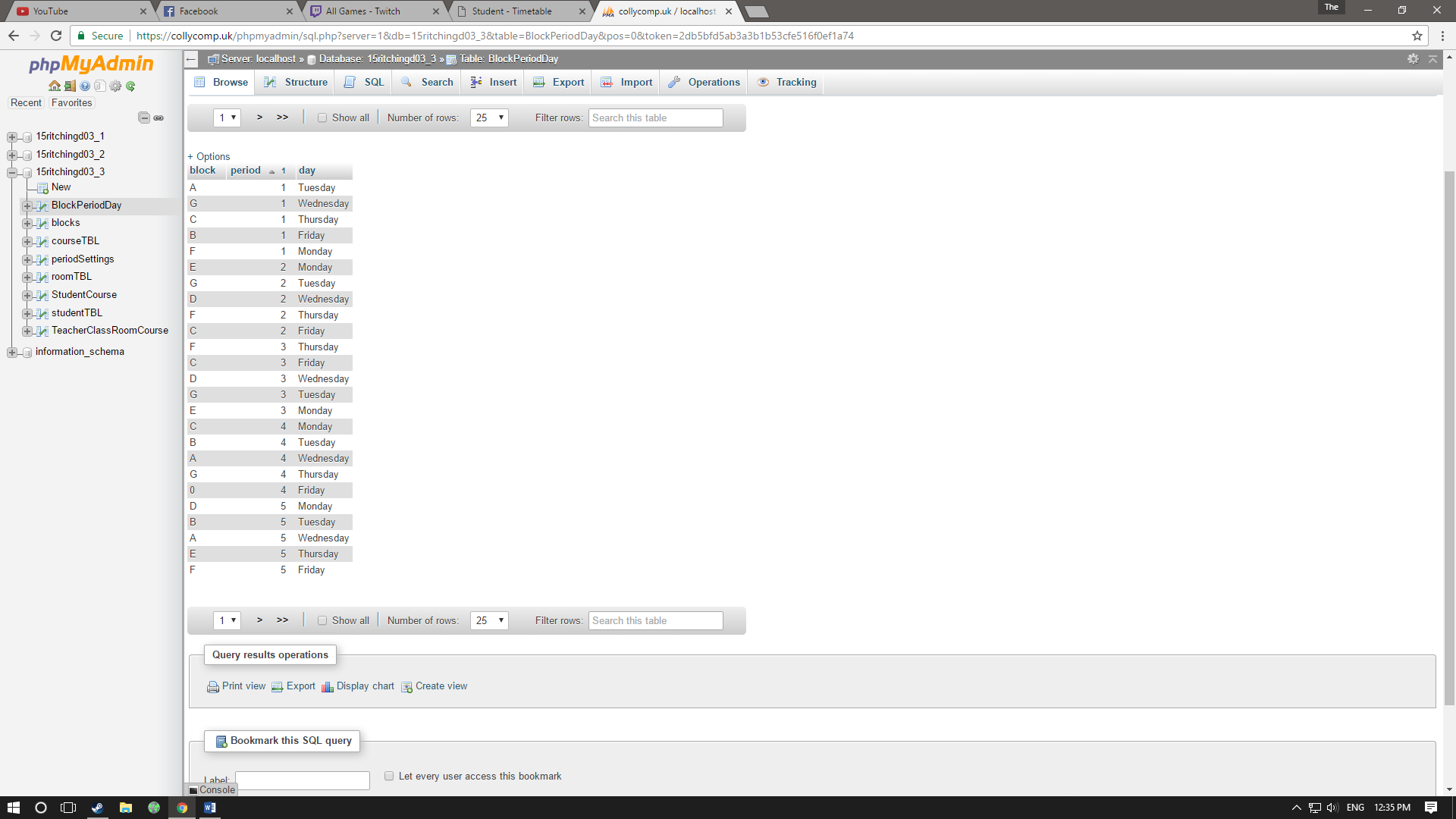


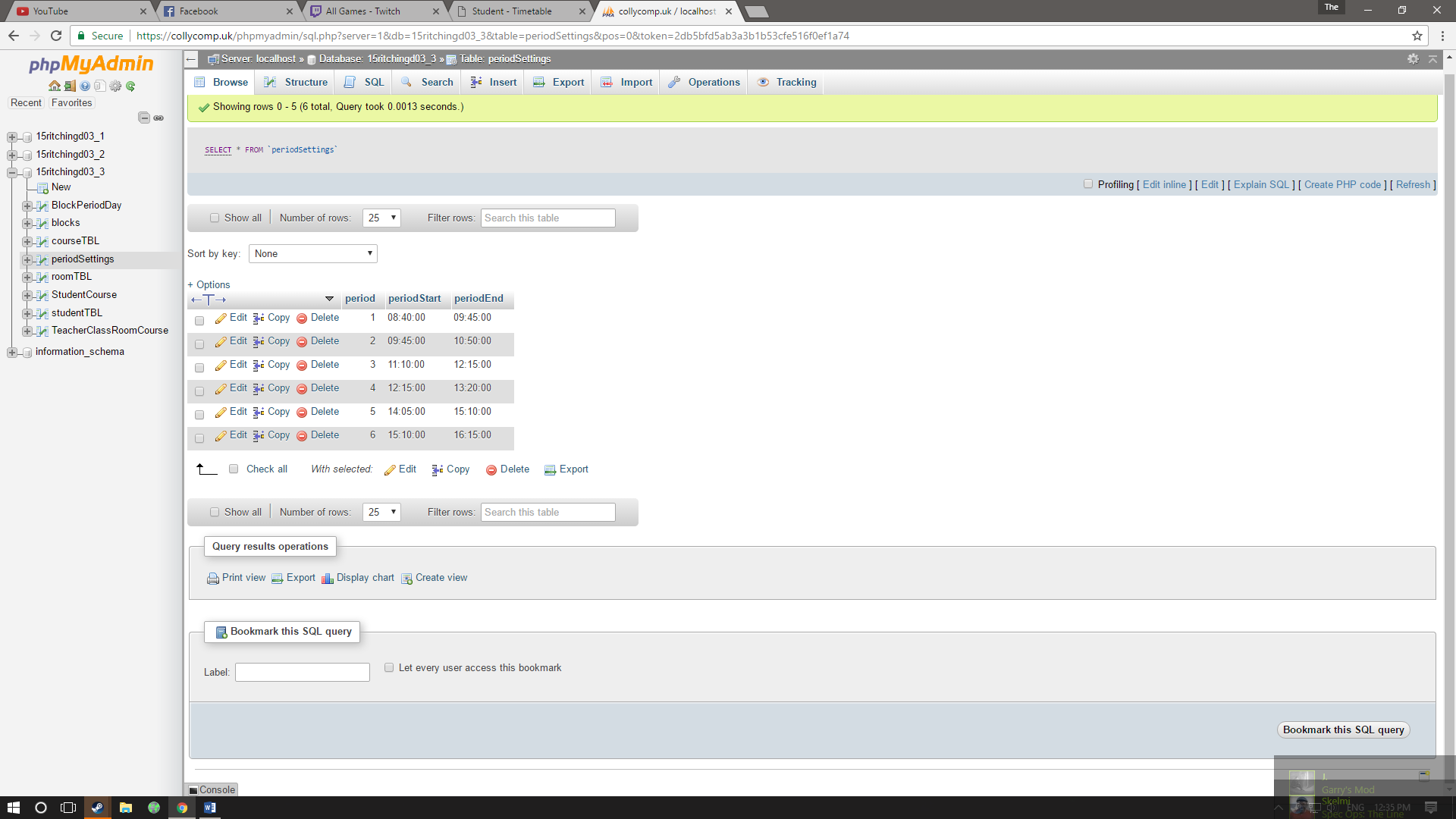
## objective 5.

Include a settings window were the number of periods there are in a day, the times of each period, the number of blocks and the times of each block can be entered.

### website

### database

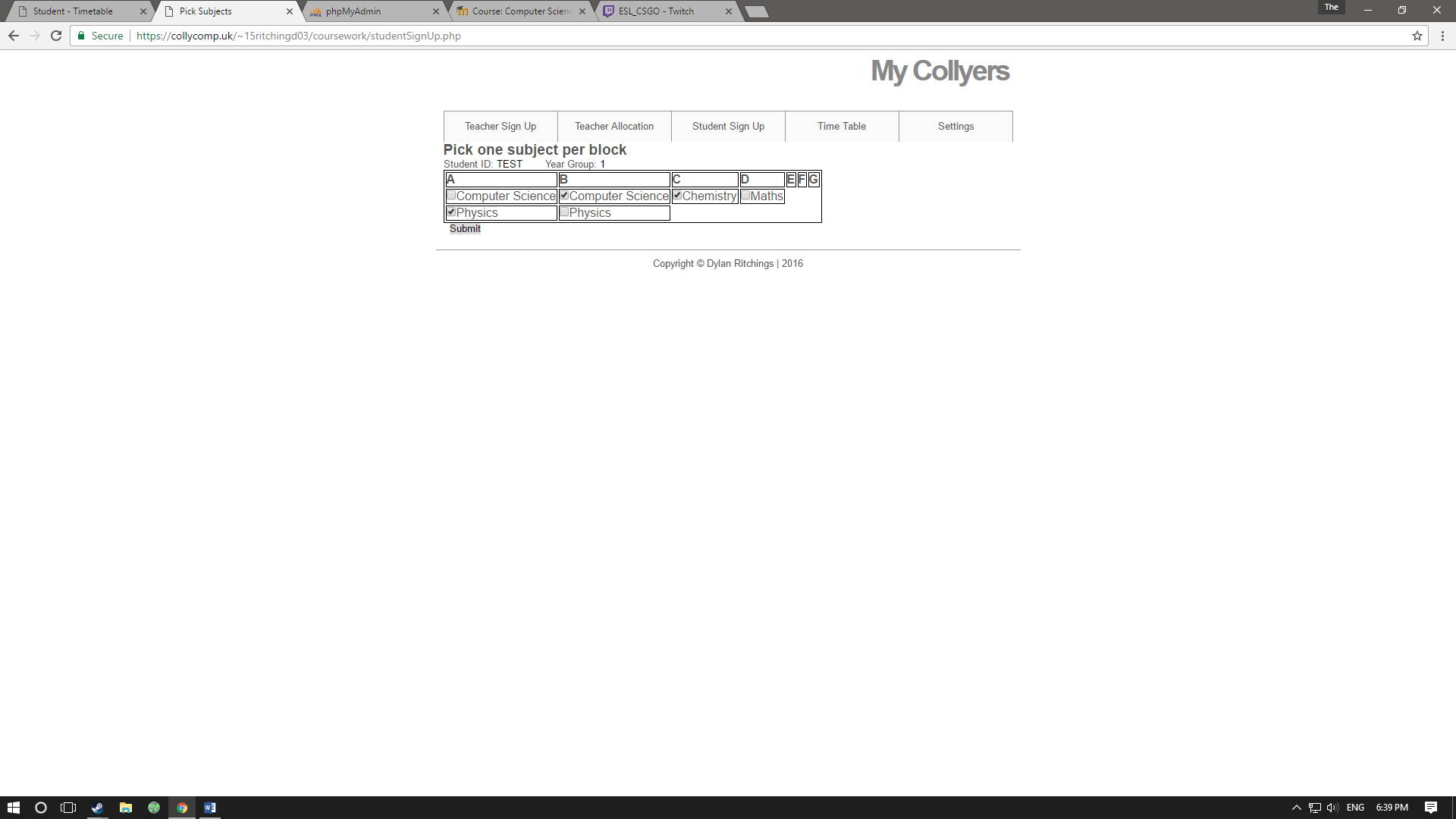
blocks BlockPeriodDay periodSettings



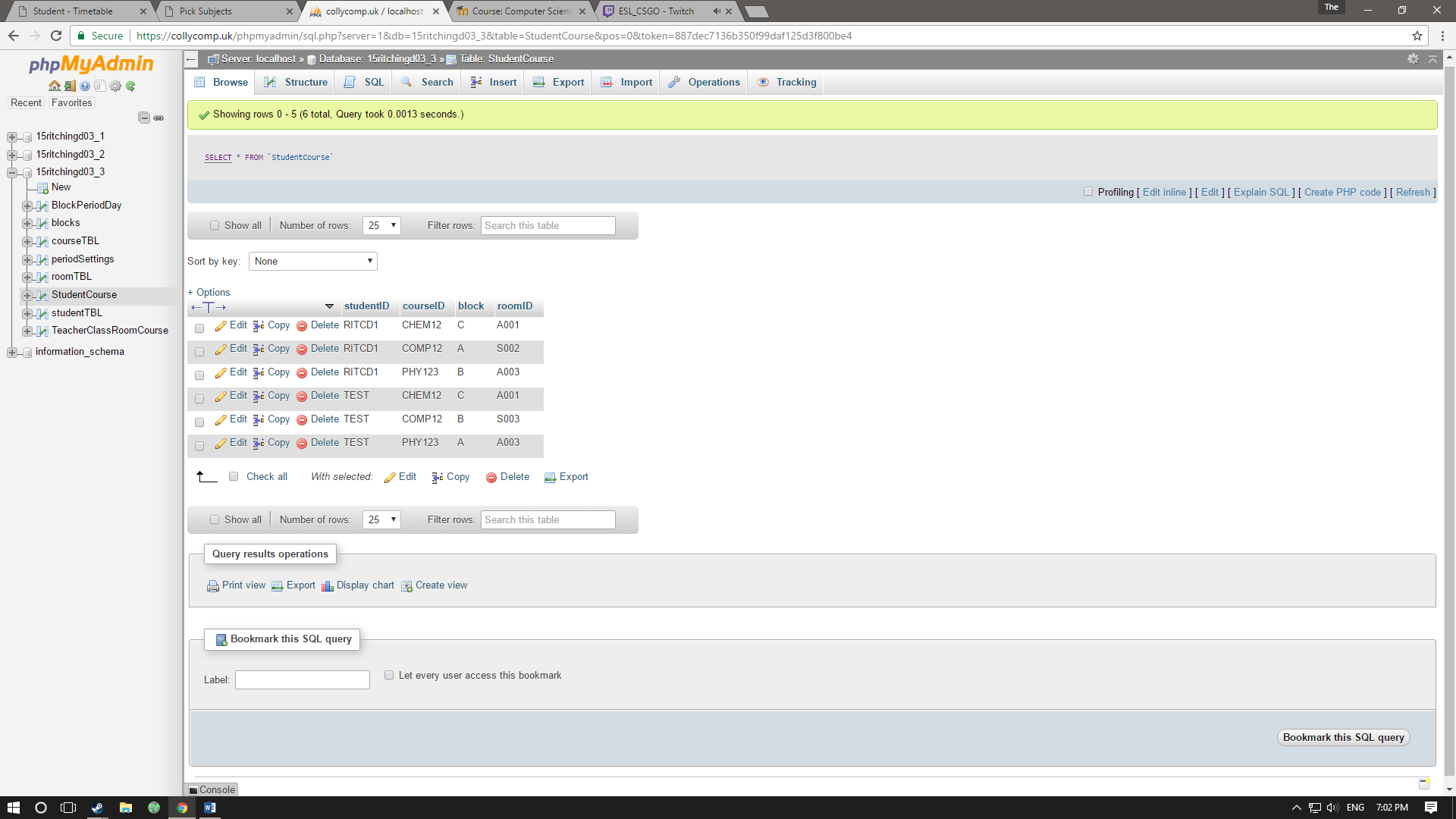
## objective 6.

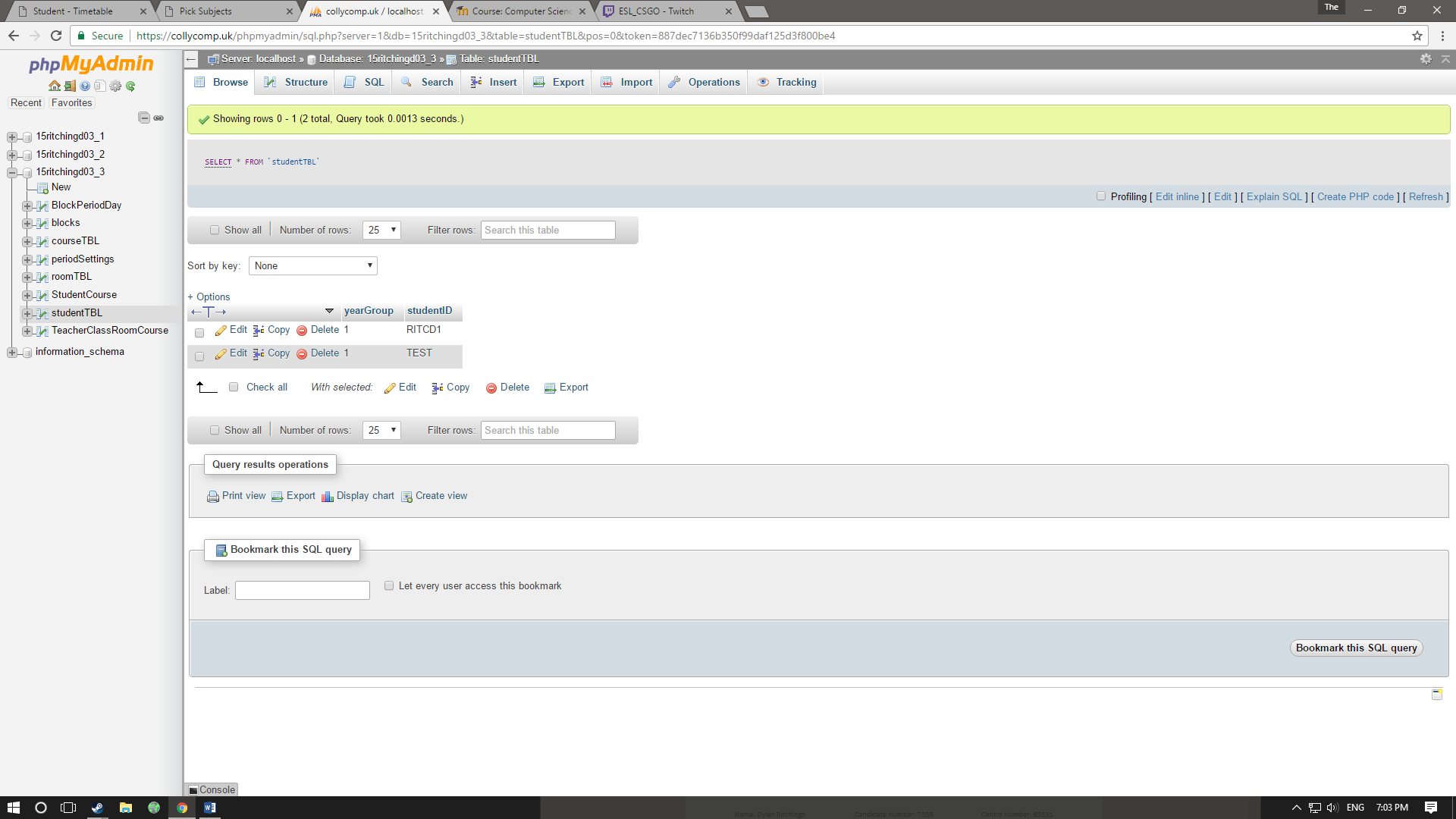
a window that allows a student to pick three subjects from multiple different blocks and when completed stores the information

### website



### database

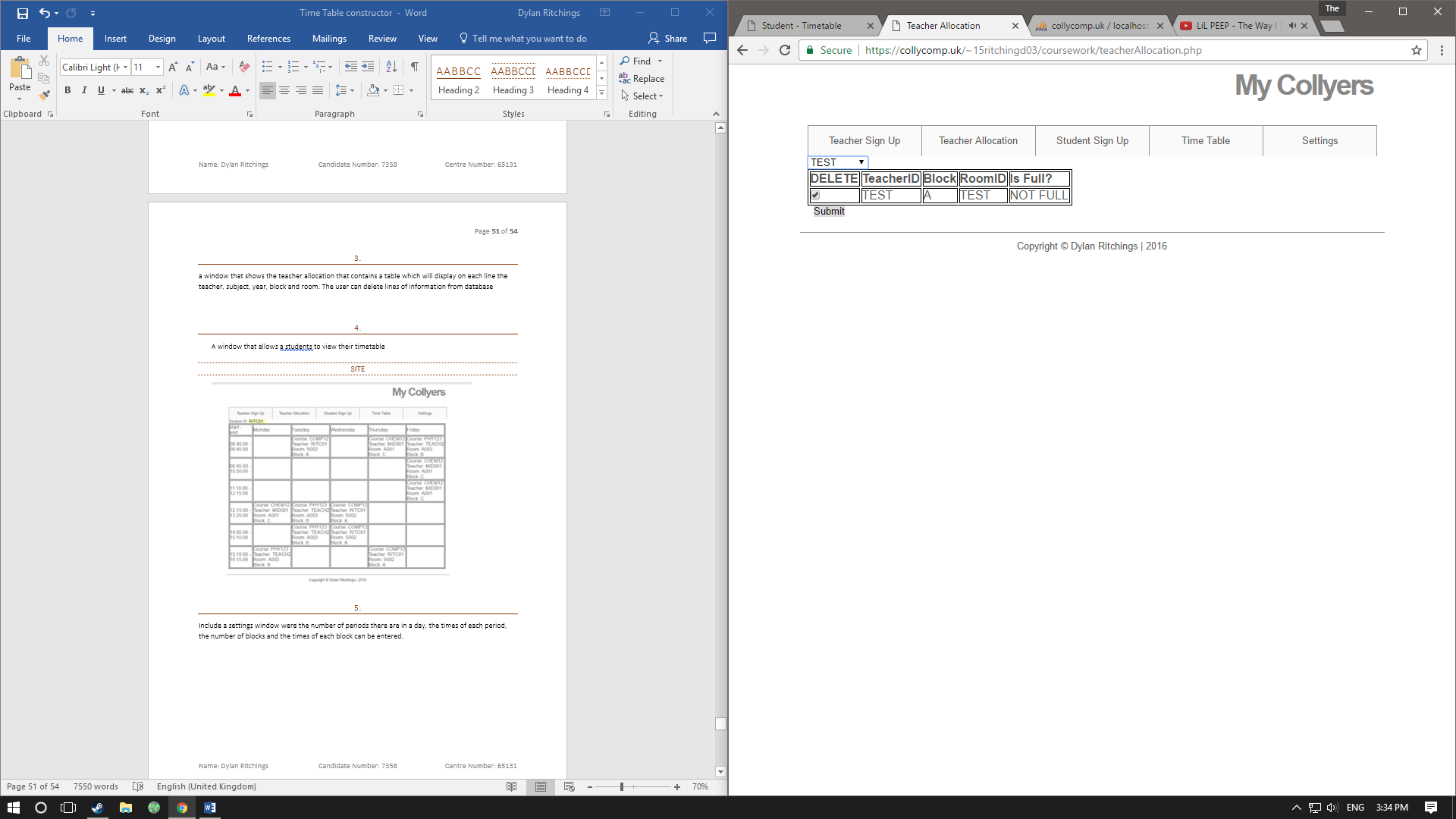
StudentCourse studentTBL



## objective 7.

A way to see what class is full

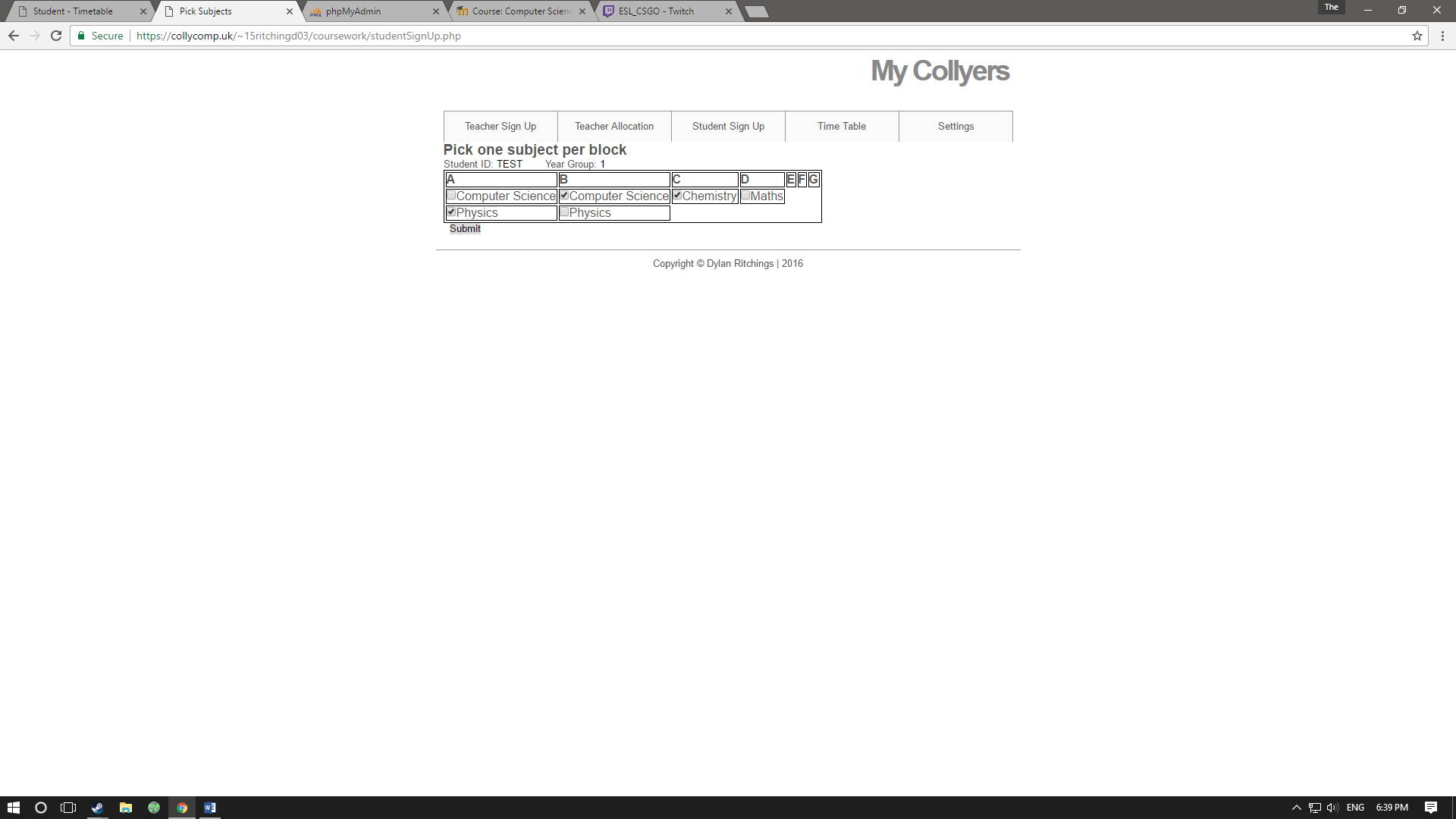
### website



## objective 8

A way for the user to navigate between different pages

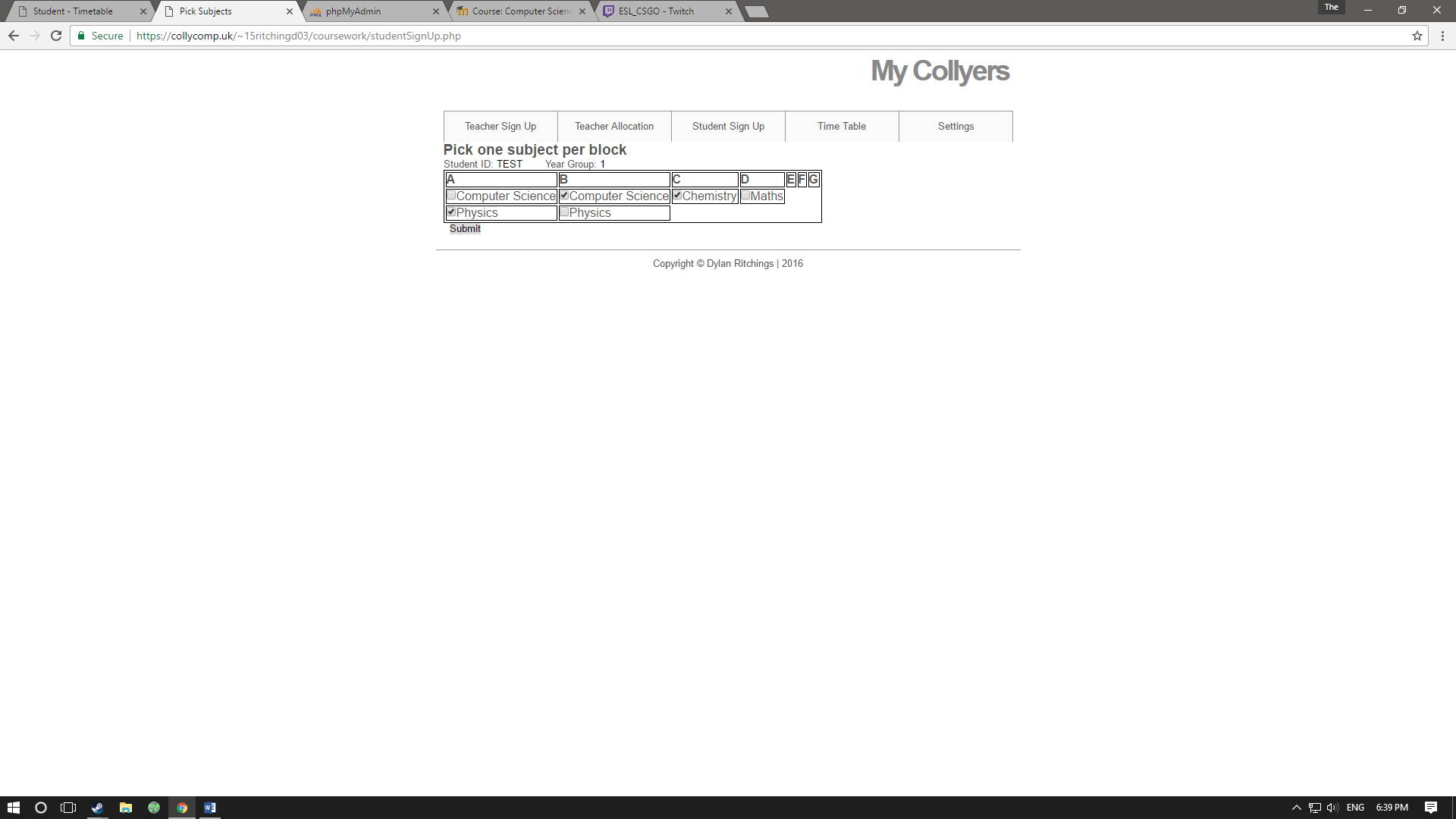
### website



## objective 9.

On each page the user (teacher or student) needs to input their userID (if needed) so the information can be assigned to the correct person.

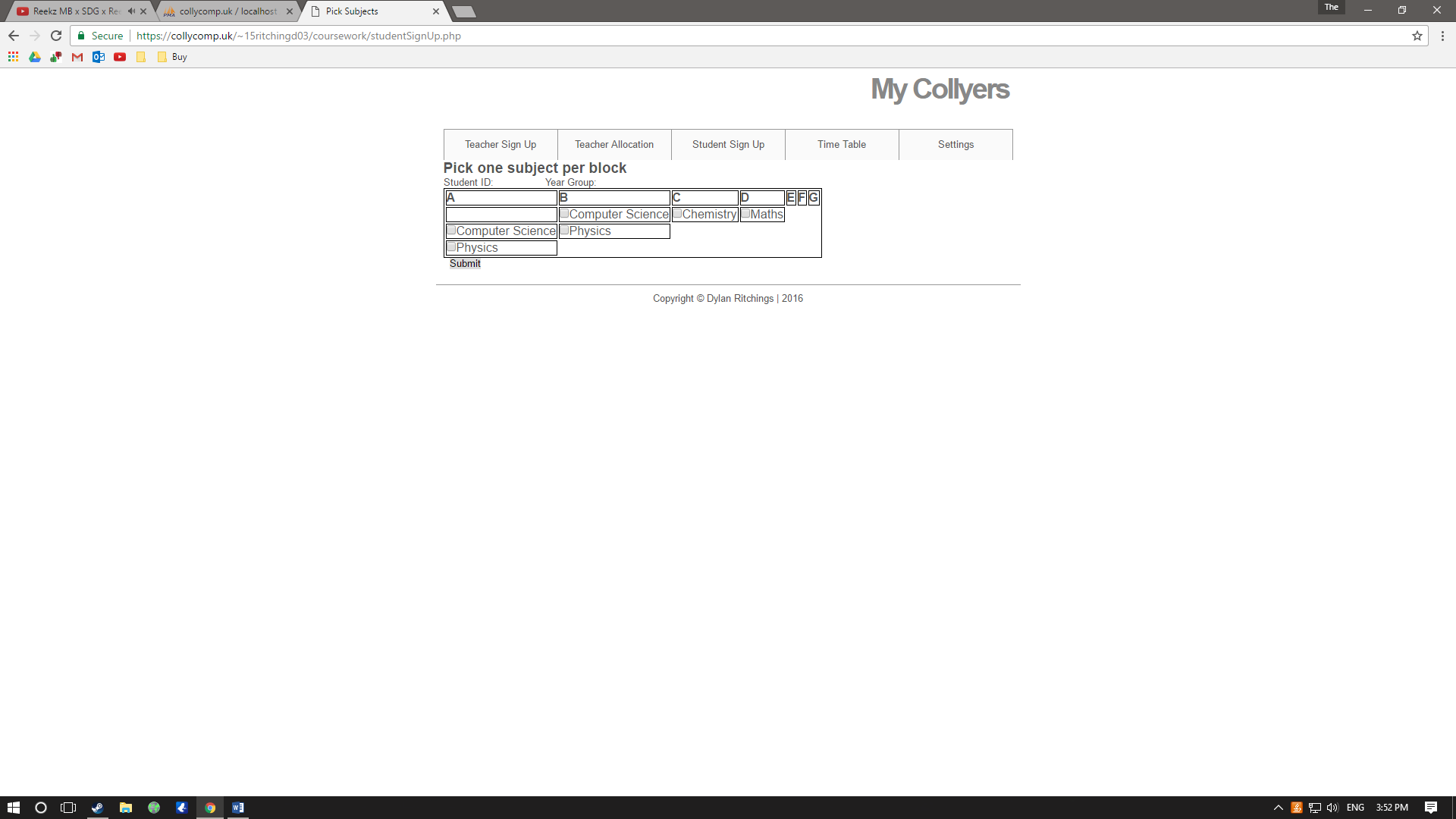
### website



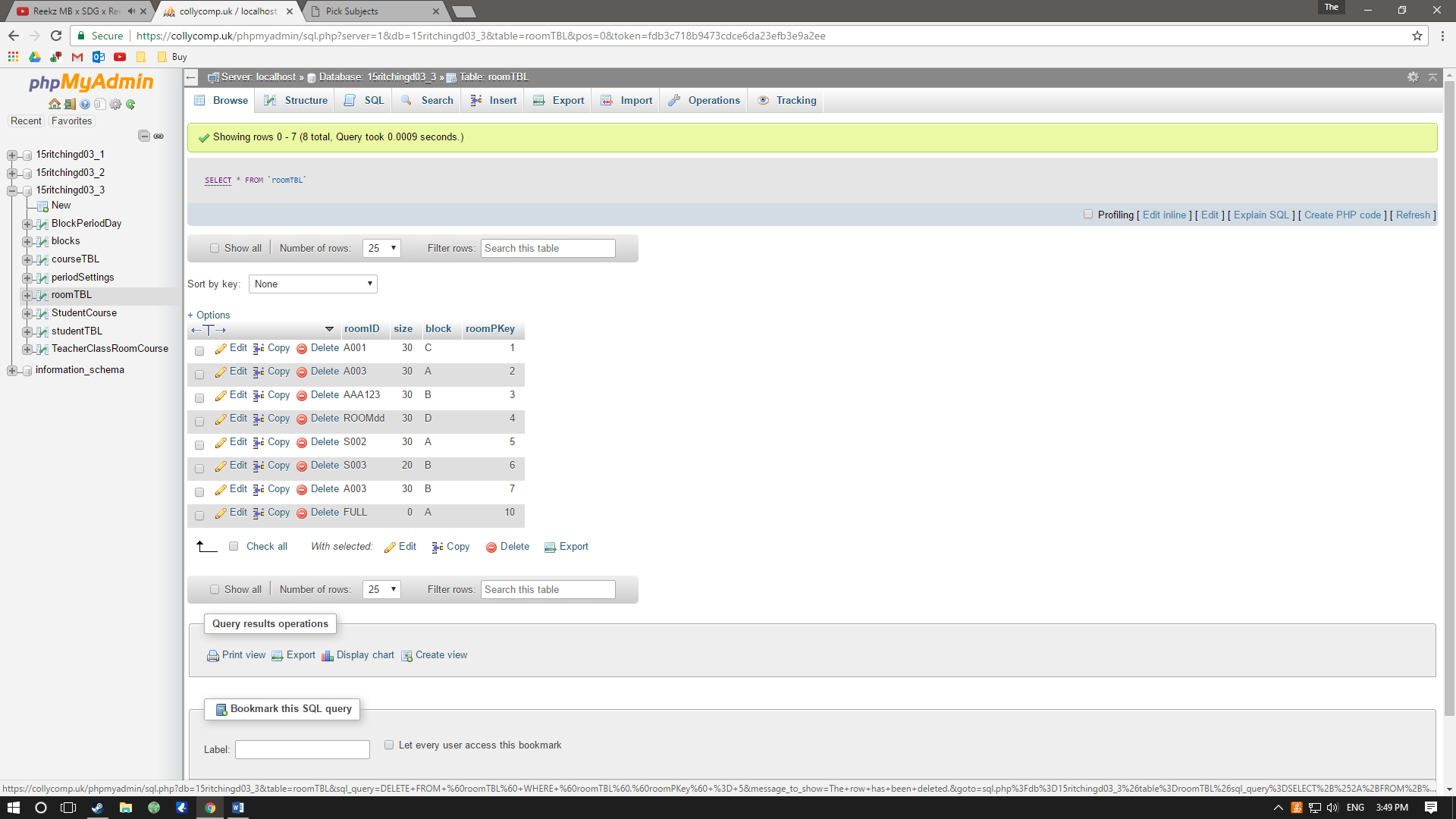
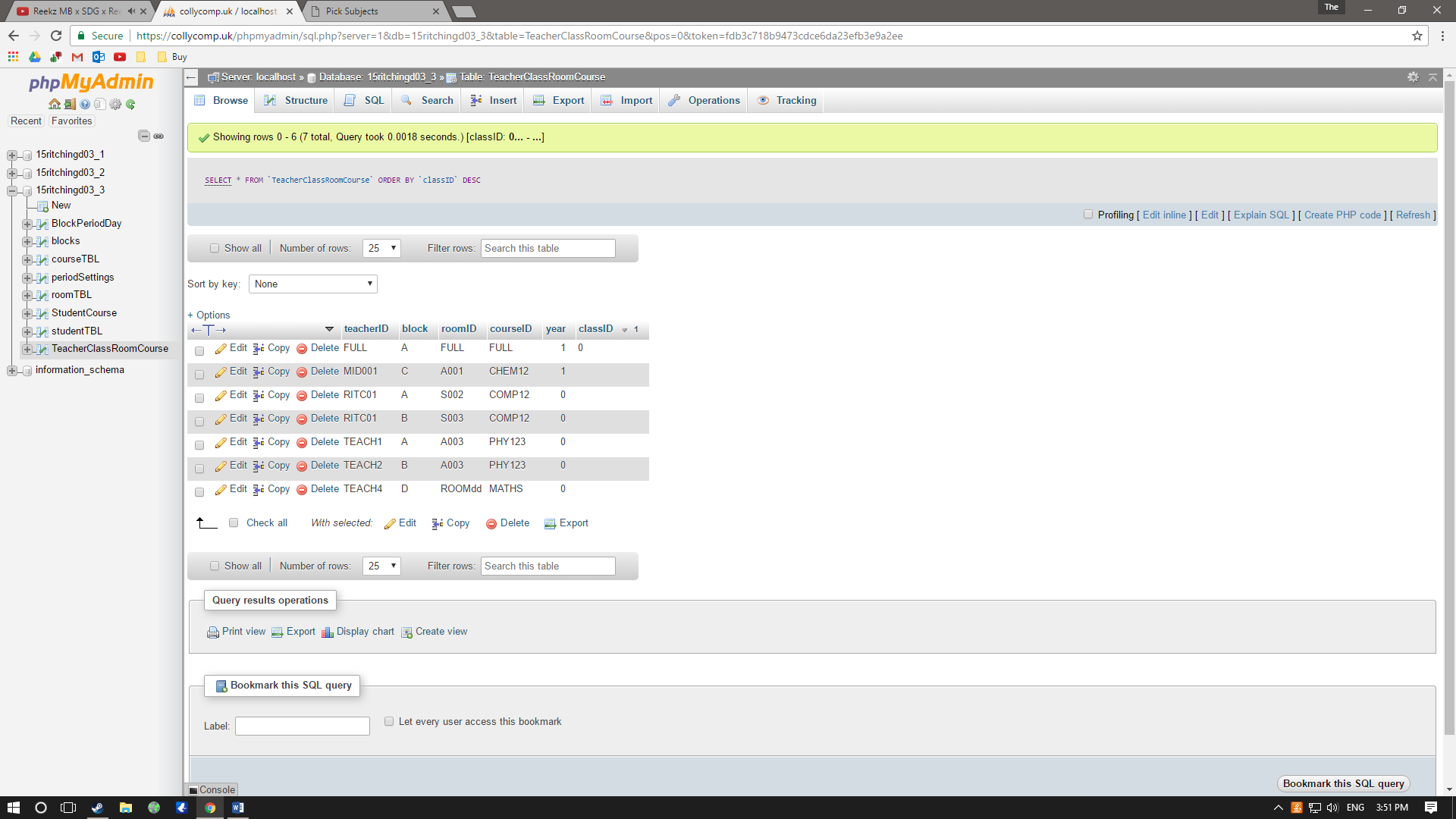
## objective 10

When a class is full don’t allow any more students to sign up to that class.

### website



### database



# evaluation

## third party feedback

I got feedback from a teacher who would use this software.

Feedback: “There should be a way you can view what the subject ID is for each subject.”

Analysis: This is a good idea which can be done externally but would make the software be more complete

Feedback: “Needs a login for security/data protection.”

Analysis: If a school/college where to buy/use my software I would add a login system but because that is not the purpose of the software I didn’t include it.

Feedback: “User interface needs to be laid out in a clearer way.”

Analysis: My focus of the project was not to make the website look good but again if It was going to be used I would make the lay it out better