

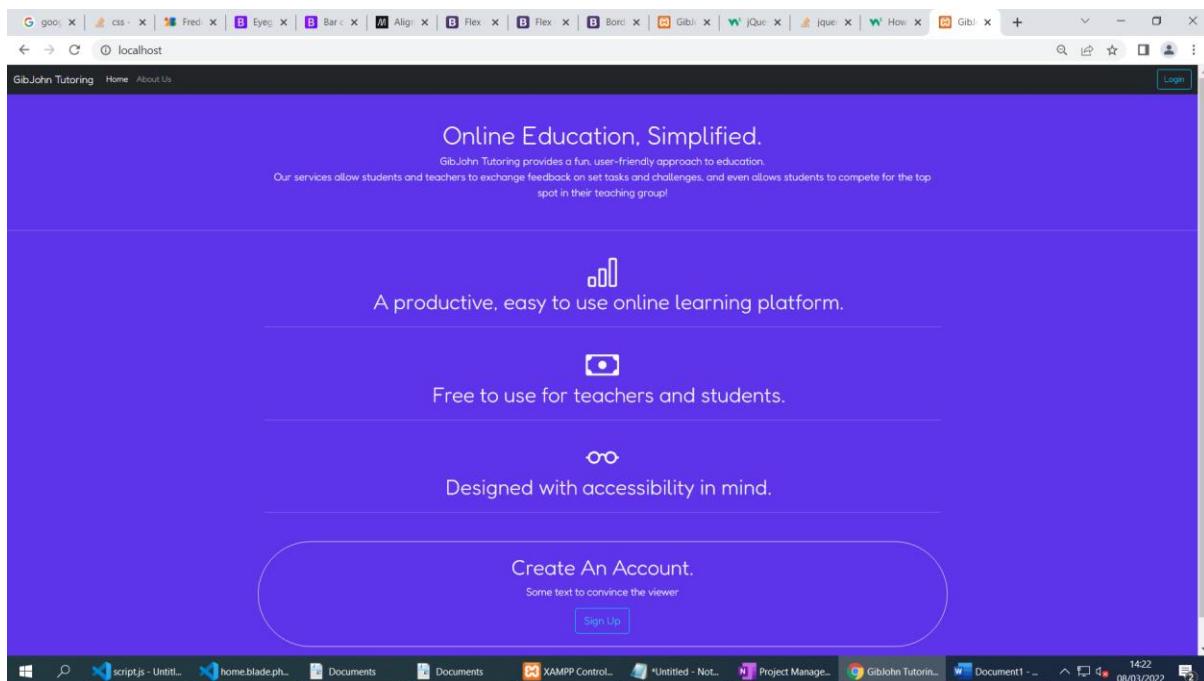
## Dev Log draft

### Day 1

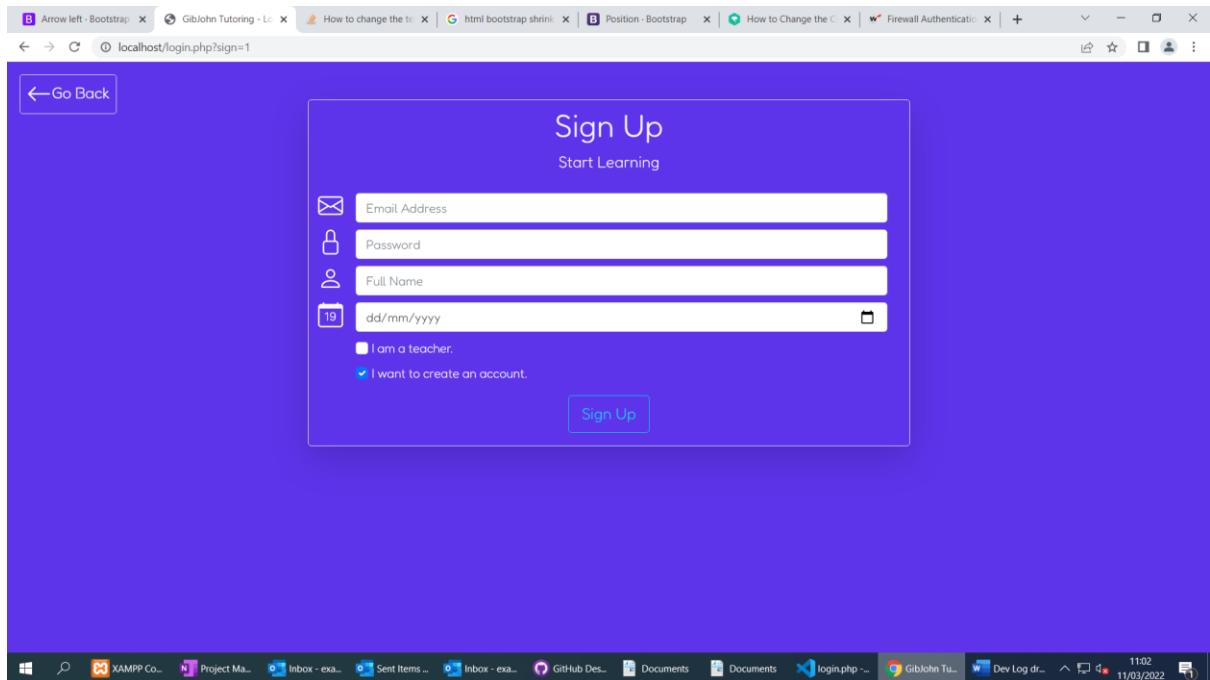
- Set up workspace
- Created a GitHub repository
- Created a base for the home page
- Created a base for the about page
- Implemented bootstrap

### Day 2

- Further work on the home page



- Although the main content of the page will not be using AJAX to save time for more important features, I have added a smooth page transition using jQuery to fade between pages.
- Changed to a different font
- Implemented flex containers to make the page usable on different device types
- Created a basic To-Do list in OneNote to manage my time properly.



- Finished the sign up and login UI
- Made the layout dynamic using jQuery
- Created a Go Back button for better user experience

```

File Edit Selection View Go Run Terminal Help
script.js - Untitled (Workspace) - Visual Studio Code

EXPLORER
UNTITLED (WORKSPACE)
www > JS script.js > on(pageshow) callback > oncheckboxChange
18 //Login page buttons
19
20 function oncheckboxchange(e){
21     if($(e)[0].checked){
22         //If signing up
23         $("#loginFullNameBox").attr("class",boxclasses);
24         $("#loginullNameBox").fadein(400);
25         $("#loginAgeBox").attr("class",boxclasses);
26         $("#loginImageBox").fadein(400);
27
28         //Change Header and button Text
29         $("#LoginPageHeader, #loginSignupButton").fadeOut(200,function(){
30             $(this).text("Sign Up");
31             $(this).attr("value", "Sign Up");
32             $(this).fadeIn(400);
33         });
34
35     }
36     else{
37         //If login in
38         $("#loginFullNameBox").fadeOut(400,function(){$(this).removeClass()});
39         $("#loginAgeBox").fadeOut(400,function(){$(this).removeClass()});
40
41         //Change Header and button Text
42         $("#LoginPageHeader, #loginSignupButton").fadeOut(200,function(){
43             $(this).text("Login");
44             $(this).attr("value", "Login");
45             $(this).fadeIn(400);
46         });
47     }
48 }
49
50 //https://stackoverflow.com/questions/9870512/how-to-obtain-the-query-string-from-the-current-url-with-javascript
51 let params = (new URL(document.location)).searchParams;
52
53 if(params.get("sign")=="1"){
54     $("#loginIsSigningUpCheckbox").prop("checked", true);
}

```

- Some of the JavaScript handling the login screen

## Day 3

The screenshot shows two instances of the phpMyAdmin interface. In the top instance, a SQL query is run to create a 'Students' table:

```
CREATE TABLE Students (ID int PRIMARY KEY AUTO_INCREMENT, StudentName VARCHAR(64), StudentEmail VARCHAR(255) UNIQUE, StudentPasswordHash VARCHAR(255), DateCreated DATETIME);
```

In the bottom instance, the 'Structure' tab is selected for the 'gibjohn' database. It displays the 'students' and 'teachers' tables, both of which have 4 columns. The 'students' table has columns: ID (auto-increment, primary key), StudentName (VARCHAR(64)), StudentEmail (VARCHAR(255), unique), and StudentPasswordHash (VARCHAR(255)). The 'teachers' table has columns: ID (auto-increment, primary key), TeacherName (VARCHAR(64)), TeacherEmail (VARCHAR(255), unique), and TeacherPasswordHash (VARCHAR(255)). Both tables use InnoDB storage engine and utf8mb4\_general\_ci collation.

The screenshot shows a Visual Studio Code interface with two tabs open:

- usermodel.php**: This tab contains PHP code for a user model. It includes functions for setting full name, date of birth, email, and password, as well as a sign-up function that connects to a MySQL database. The code uses MySQLi and includes validation logic for dates and teacher status.
- loginController.php**: This tab contains PHP code for a login controller. It handles user login, password hashing, and checks if the user is a student or teacher. It also includes a sign-up function and a validation function for input fields.

The code is well-structured with clear comments and uses modern PHP practices like object-oriented programming and prepared statements for database interactions.

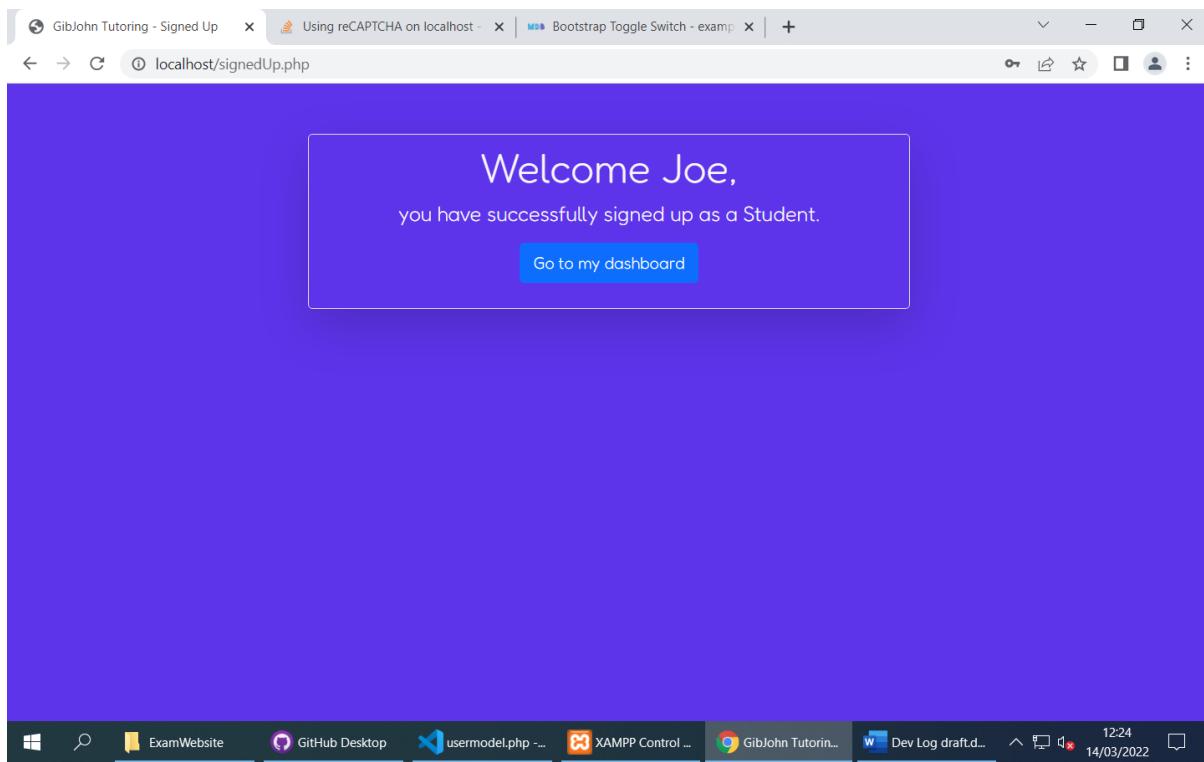
- Started working on the login system for students and teachers using Model View Controller.
  - Created a database with a Students and Teachers table
  - Login page sends POST data to PHP script
  - Script calls Login Controller
  - Login controller validates inputs and handles the login or sign-up procedure using the User Model
  - Made sure code is well commented with credit if I used something to help me.

Day 4

The screenshot shows the Visual Studio Code interface with the following details:

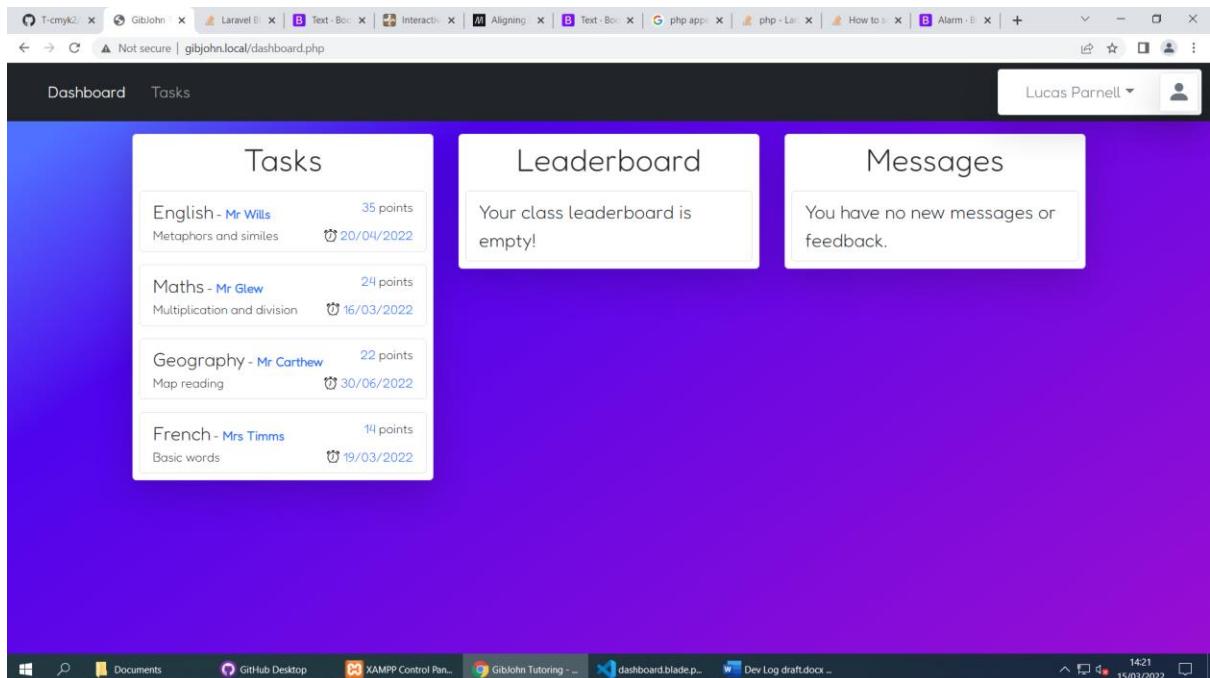
- File Explorer (Left):** Shows the project structure for "EXAMWEBSITE". The "usermodel.php" file is selected.
- Code Editor (Center):** Displays the PHP code for "usermodel.php". The code handles user sign-up, including connecting to a database, escaping strings, and inserting data into the "students" table. It also checks for existing users and handles password hashing.
- Bottom Status Bar:** Shows "Ln 38, Col 51" and "Spaces: 4" along with other standard status bar information.

```
//Clear hash as this is serialised.  
$this->passwordHash = "";  
  
return "signedUp";  
  
}  
else{  
    //Student  
    $db = $this->connectDatabase();  
  
    //Escape strings to be safe  
    $this->fullName = $db->real_escape_string($this->fullName);  
    $this->email = $db->real_escape_string($this->email);  
  
    //Current date  
    $currentDate = new DateTime();  
    $currentDate = $currentDate->format("d-m-Y");  
  
    //SQL to try and insert the user data  
    $sql = <>SQL  
    INSERT INTO students  
    (ID, StudentName, StudentDateOfBirth, StudentEmail, StudentPasswordHash, DateCreated)  
    VALUES  
    (NULL, '$this->fullName', '$this->dateOfBirth', '$this->email', '$this->passwordHash', '$currentDate')  
    $sql;  
    print("<br>");  
    print($sql);  
  
    try{  
        $db->query($sql);  
    }  
    catch(mysqli_sql_exception $e){  
        if($e->getCode() == 1062){  
            return "userExists";  
        }  
    }  
  
    //Clear hash as this is serialised.  
    $this->passwordHash = "";  
  
    return "signedUp";  
}  
}  
//-----
```

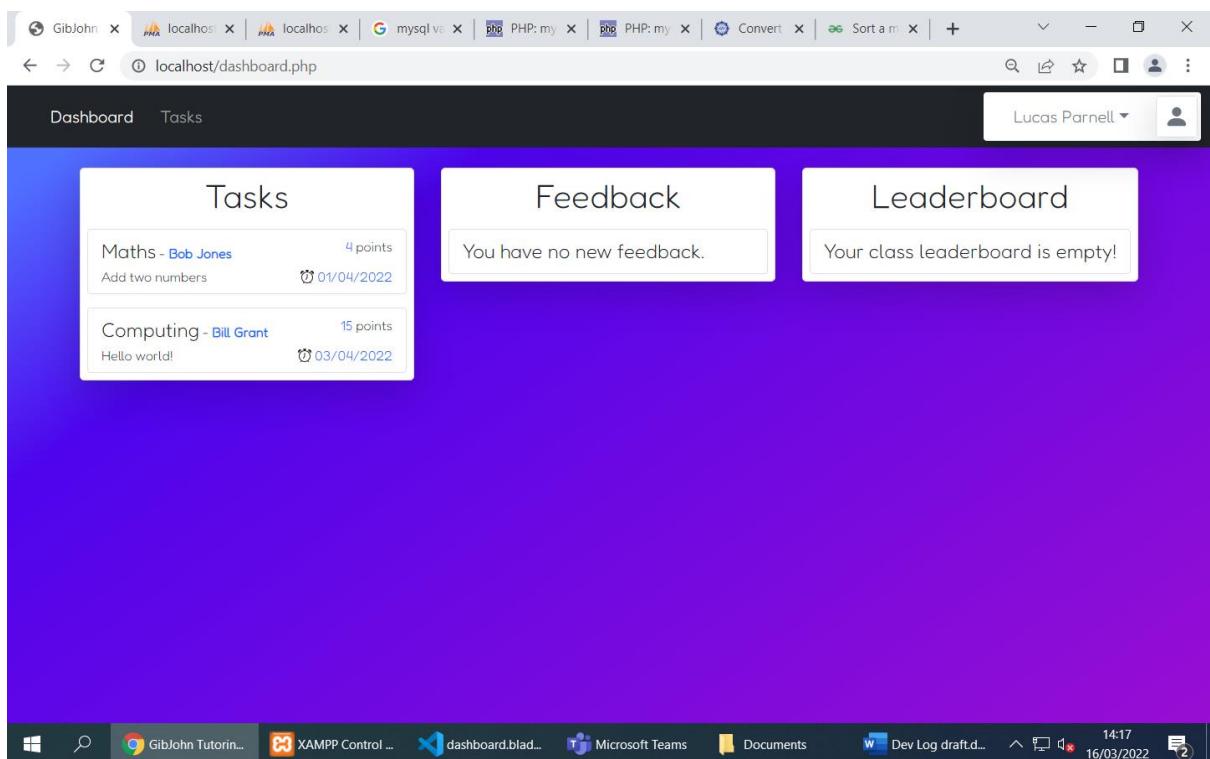


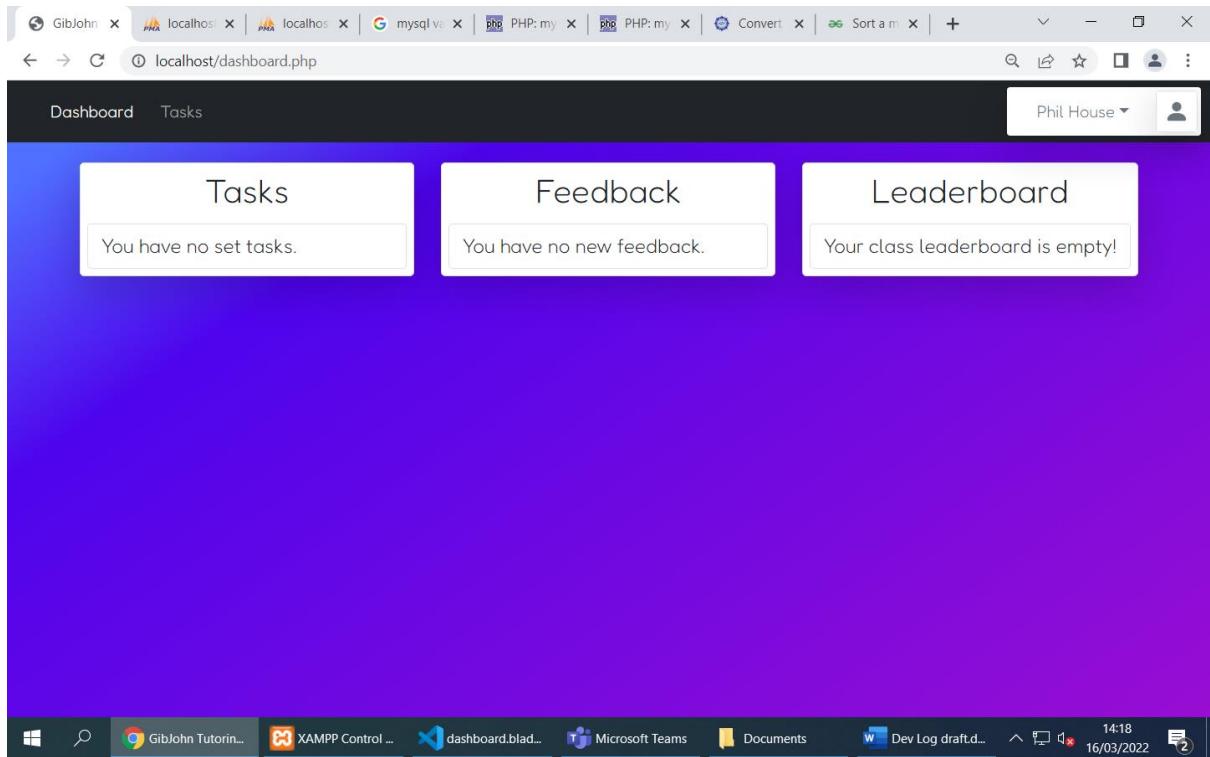
- Sign-up is now functional
  - User model gets serialised and stored in the session (Sensitive data is removed)
  - Some validation that checks age and duplicate users as well as sanitising the inputs (Removing illegal characters, Escaping the rest).
  - Will finish login next and then begin creating a personalised layout and dashboard.

## Day 4



- Login system is mostly finished, just need to add client-side validation
- Worked on the personalised student dashboard, Templates can be used to easily add Tasks to the interface.
- Started working on a profile drop down interface that displays some information about the user as well as a log out button





## (Different users have different set tasks)

```
<?php
class Task {

    function getStudentTasks($user){
        $sql = "";

        $userId = $user->getUserId();
        //First, get TeachingGroups of the user

        $sql = <<<SQL
SELECT `TeachingGroupID` FROM `TeachingGroupStudents` WHERE `StudentID` = $userId;
SQL;

$db = $this->connectDatabase();

$result = $db->query($sql);

$resultArray = $result->fetch_all();

//Final array for tasks
$tasks = [];

for($i=0; $i<count($resultArray); $i++){
    $teachingGroup = $resultArray[$i][0];

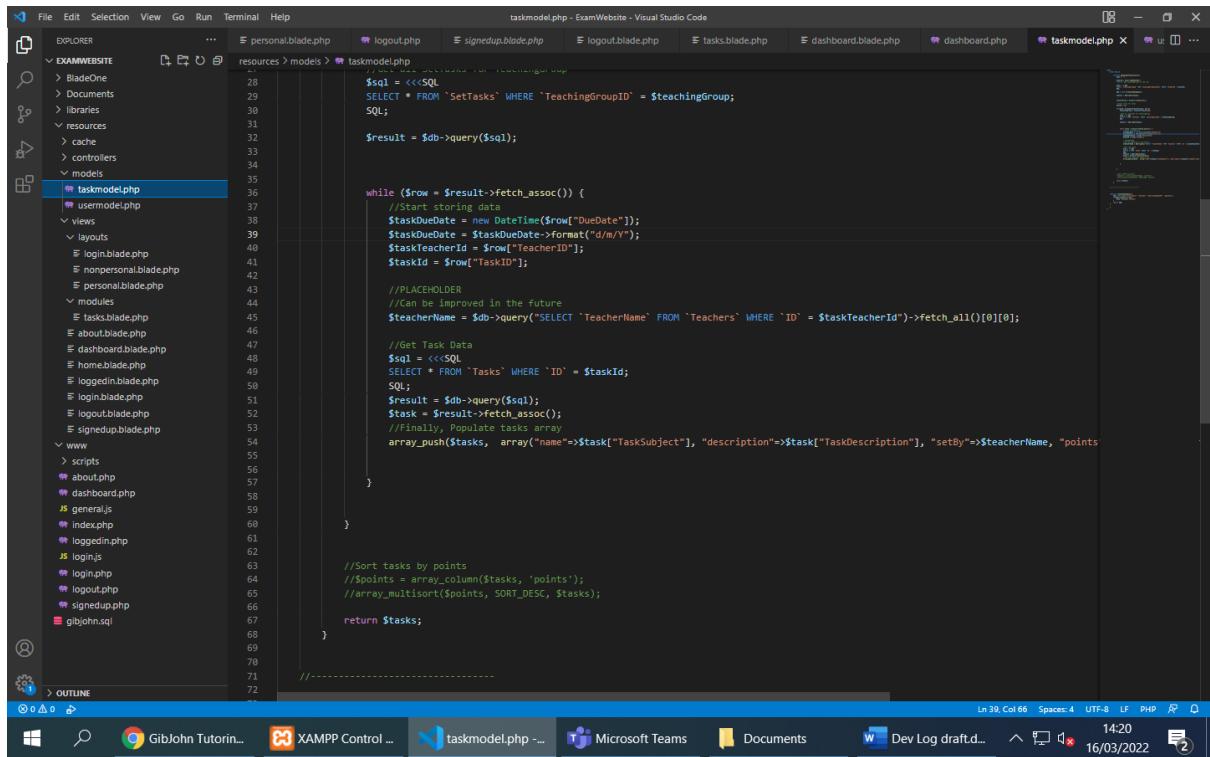
    //Get all SetTasks for TeachingGroup
    $sql = <<<SQL
SELECT * FROM `SetTasks` WHERE `TeachingGroupID` = $teachingGroup;
SQL;

$result = $db->query($sql);

while ($row = $result->fetch_assoc()) {
    //Start storing data
    $taskDueDate = new DateTime($row["DueDate"]);
    $taskDueDate = $taskDueDate->format("d/m/Y");
    $taskTeacherId = $row["TeacherID"];
    $taskId = $row["TaskID"];

    //PLACEHOLDER
    //Can be improved in the future
    $teacherName = $db->query("SELECT `TeacherName` FROM `Teachers` WHERE `ID` = $taskTeacherId")->fetch_all()[0][0];
}
```

Some of the prototype code for getting a list of tasks for a given student



```

File Edit Selection View Go Run Terminal Help taskmodel.php - ExamWebsite - Visual Studio Code
EXPLORER resources > models > taskmodel.php
resources > models > taskmodel.php
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

$SQL = <<<SQL
SELECT * FROM `SetTasks` WHERE `TeachingGroupID` = $teachingGroup;
SQL;

$result = $db->query($SQL);

while ($row = $result->fetch_assoc()) {
    //Start storing data
    $taskDueDate = new DateTime($row["DueDate"]);
    $taskDueDate = $taskDueDate->format("d/m/Y");
    $taskTeacherId = $row["TeacherID"];
    $taskId = $row["TaskID"];

    //PLACEHOLDER
    //Can be improved in the future
    $teacherName = $db->query("SELECT `TeacherName` FROM `Teachers` WHERE `ID` = $taskTeacherId")->fetch_all()[0][0];

    //Get Task Data
    $SQL = <<<SQL
SELECT * FROM `Tasks` WHERE `ID` = $taskId;
SQL;
    $result = $db->query($SQL);
    $task = $result->fetch_assoc();
    //Finally, Populate tasks array
    array_push($tasks, array("name"=>$task["TaskSubject"], "description"=>$task["TaskDescription"], "setBy"=>$teacherName, "points"
array_push($tasks, array("name"=>$task["TaskSubject"], "description"=>$task["TaskDescription"], "setBy"=>$teacherName, "points"

return $tasks;
}

//-----
//Sort tasks by points
//points = array_column($tasks, 'points');
//array_multisort($points, SORT_DESC, $tasks);

return $tasks;
}

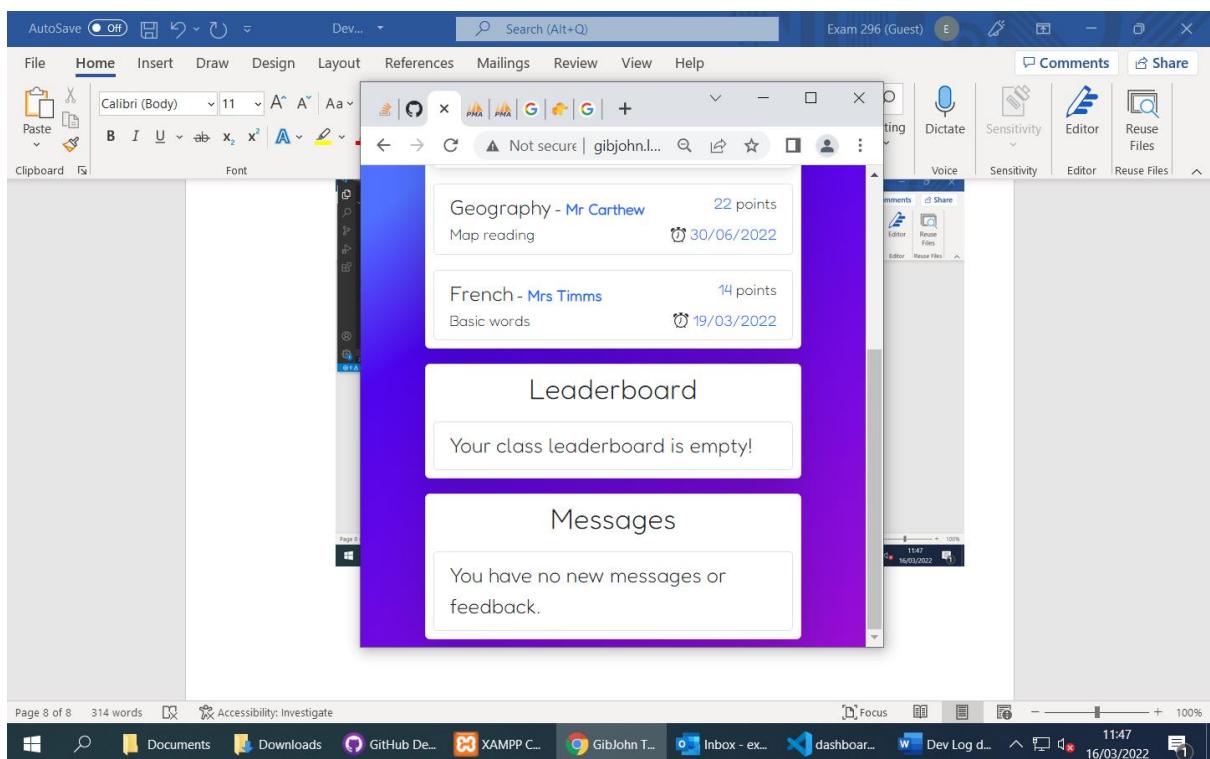
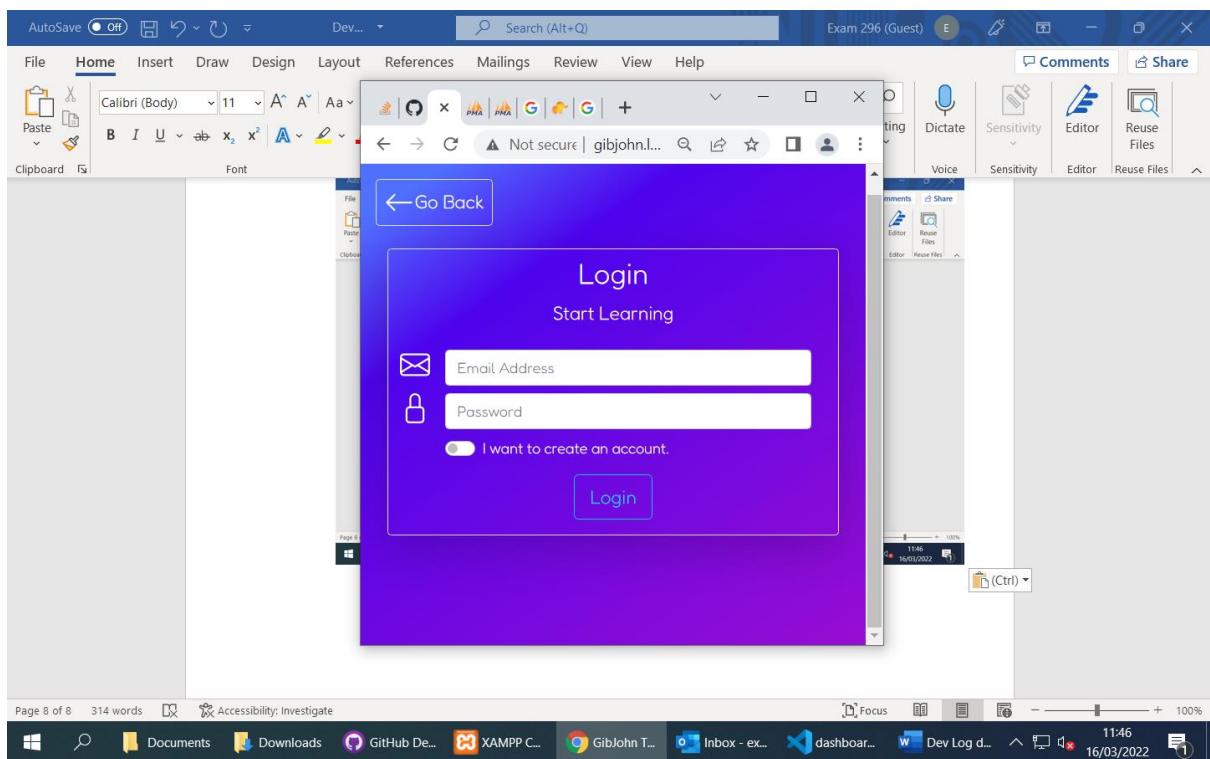
```

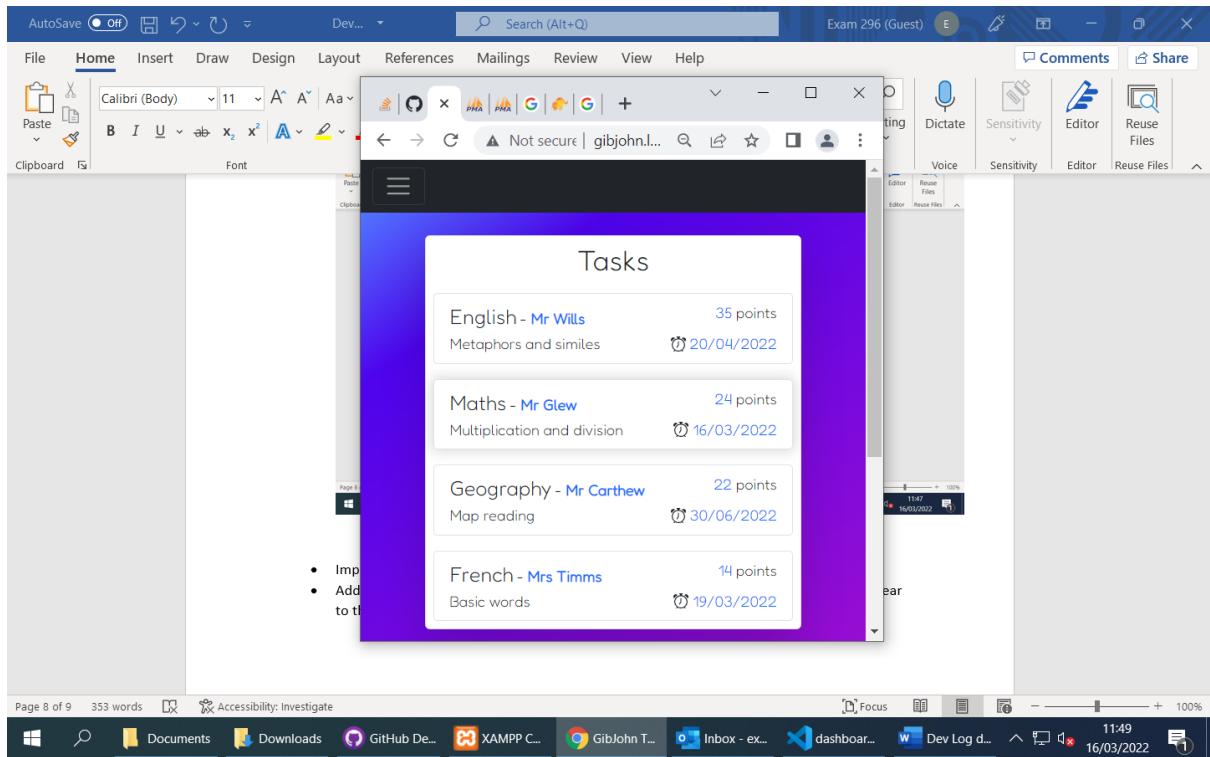
- Programmed the task model to pull a student's tasks from the database by first finding their teaching group and then getting their set tasks. (Set tasks are different from tasks)
- Fixed a few issues with dates not being stored correctly
- Added some entries into the database
- Next, I will program some of the feedback model and start on the student tasks page.

## Day 5

The screenshot shows two instances of the phpMyAdmin interface. The top instance displays the 'Structure' tab for the 'gibjohn' database, showing a list of 9 tables: leaderboard, settasks, students, taskfeedback, tasks, teachers, teachersteachinggroups, teachinggroups, and teachinggroupstudents. The bottom instance shows the 'Designer' tab, where the database schema is visualized as an Entity-Relationship (ER) diagram. The entities are represented as boxes with their primary keys in bold: gibjohn.students, gibjohn.teachers, gibjohn.teachinggroupstudents, gibjohn.leaderboard, gibjohn.taskfeedback, gibjohn.teachersteachinggroups, gibjohn.teachinggroups, and gibjohn.settasks. Relationships are shown as lines connecting the entities, with arrows indicating the direction of the relationship. For example, there are relationships between students and leaderboard, students and taskfeedback, students and teachersteachinggroups, and students and teachinggroups.

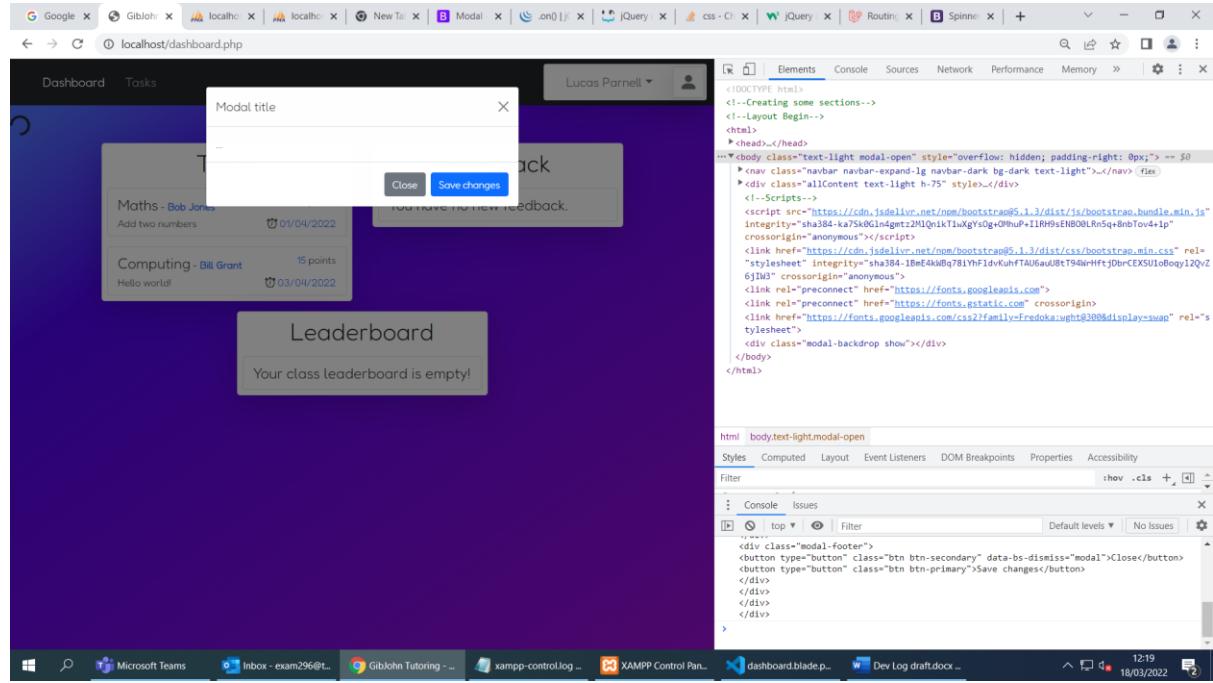
- Implemented the database from the design documentation
- Created all relationships after setting indexes
- Made the database design similar to my original documentation



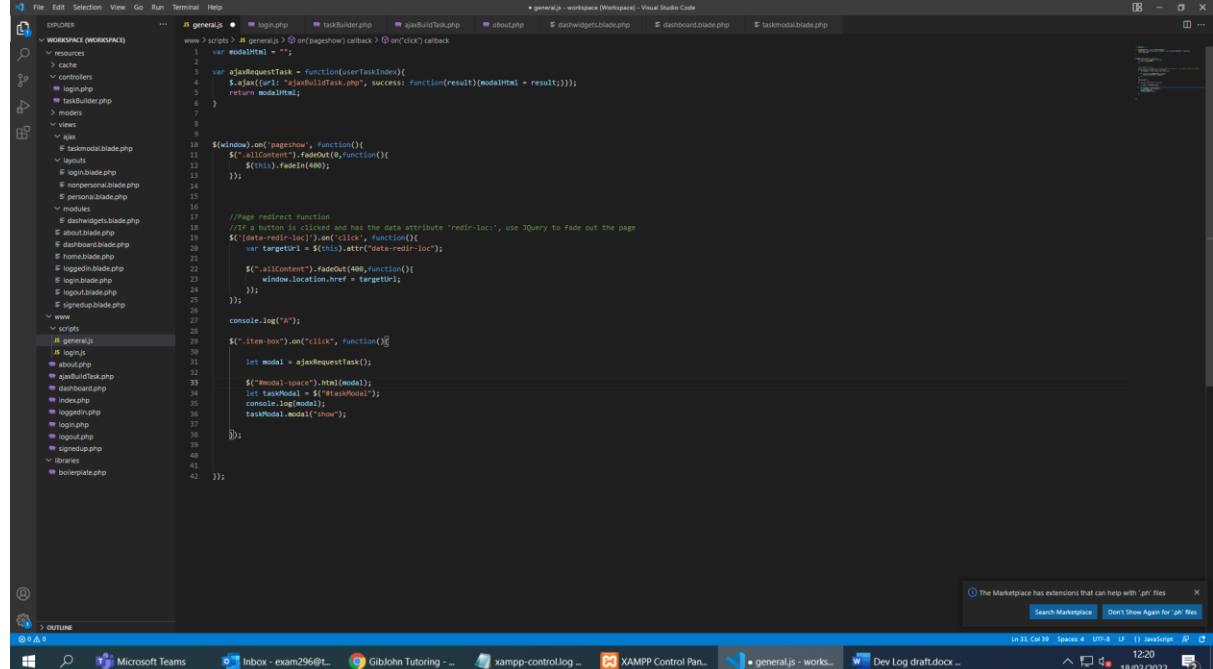


- Improved scaling of the flexboxes, website is now usable on mobile devices
- Added a shadow effect to the task items when the mouse is over them, this makes it clear to the user that the item is being selected.
- Next will start on AJAX task system

## Day 6



The screenshot shows a web browser window with multiple tabs open. The active tab is 'localhost/dashboard.php'. A modal dialog is displayed in the center of the page with the title 'Modal title'. The modal content includes a text area with placeholder text '...', a 'Close' button, and a 'Save changes' button. Below the modal, a message says 'You have NO new feedback.' The browser's developer tools are open, showing the HTML structure of the modal. The modal has a class of 'text-light modal-open' and contains a close button and a save changes button. The background page shows a dashboard with tasks like 'Maths - Bob Jones' and 'Computing - Bill Grant'.



The screenshot shows the Visual Studio Code interface with the 'general.js' file open in the editor. The code is written in JavaScript and handles modal requests. It defines a function 'ajaxRequestTask' that takes a task ID and returns a modal HTML string. It then uses jQuery to fade out the current content and fade in the new modal. The code also includes a redirect function for pages with the 'data-redirect-loc' attribute. The 'general.js' file is part of a larger workspace containing other files like 'login.php', 'taskBuilder.php', 'ajaxBuildTask.php', 'logout.php', 'dashboard.blade.php', 'index.blade.php', 'about.blade.php', 'tasks.blade.php', 'home.blade.php', 'logedit.blade.php', 'logedit.php', 'login.blade.php', 'login.php', 'signup.blade.php', 'signup.php', and 'boilerplate.php'.

- Started on AJAX code to build and load task content using JSON files on the server and templates for the HTML, JSON files will contain information and the actual content of the task, Files are referenced in the database entries for each task under the column "taskFileIdentifier"
- Set up basic AJAX request, client can request a modal from the server, it then places it into an empty div and renders it using `\$( "[modalSelector]" ).modal("show");`
- Added a loading icon while the content is being created as depending on the load of the website it may take some time.
- Server can now read from JSON file and push the data into arrays, these can be sent to the template which can be processed and returned to the client.

C:\Users\User\Documents\GitHub\ExamWork\ExamWebsite\taskEntry\maths0001.json - Notepad++

```

1  [
2      "title": "Adding Two Numbers",
3      "description": "Try and add as many numbers as you can.",
4      "questions": [
5          {
6              "id": 1,
7              "type": "text",
8              "question": "15+3?",
9              "answer": "18"
10         }
11     ]

```

length : 197 lines : 11 Ln : 5 Col : 12 Pos : 119 Windows (CR LF) UTF-8 INS

JSON File Microsoft Te... Inbox - exam... GibJohn Tuto... xampp-contr... XAMPP Contr... ajaxBuildTask... Dev Log draf... GitHub Deskt... taskEntry C:\Users\User... 14:35 18/03/2022

localhost/dashboard.php

Dashboard Task Questions Cor Mat Add

Close Save changes

Elements > |

```

<!DOCTYPE html>
<!--Creating some sections-->
<!--Layout Begin-->
<html>
  <head></head>
  <body class="text-light modal-open" style="overflow: hidden; padding-right: 0px;">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark text-light">
      <!--Content-->
      <div class="allContent text-light h-100" style="min-width: 25rem; flex: 1; position: relative; z-index: 1; background-color: #fff; border-radius: 10px; border: 1px solid #ccc; padding: 10px; margin: 0 auto; width: fit-content; height: 100%;>
        <div class="content mt-3 h-100">
          <div class="d-flex justify-content-center align-items-start flex-wrap">
            <div class="bg-body text-dark d-inline-flex flex-column justify-content-between border rounded-3 shadow shadow-lg mx-3 my-2" style="min-width: 25rem; flex: 1; position: relative; z-index: 1; background-color: #fff; border-radius: 10px; border: 1px solid #ccc; padding: 10px; margin: 0 auto; width: fit-content; height: 100%;>
              <span class="display-6 text-center px-4 py-2">Tasks</span>
            </div>
          </div>
        </div>
      </div>
    </nav>
  </body>

```

Styles Computed Layout >

Filter i hover .cls +

Console Issues

Issues top

Filter Default levels

No Issues

- Questions passed to template

## Day 7

The screenshot shows a web browser window with multiple tabs open. The active tab displays a dashboard titled "Tasks". It lists two items: "Computing - Bill Grant" (Hello world!) and "Maths - Bob Jones" (Add two numbers). Below the list is a modal window titled "Task" containing a math problem: "1+1=2". The modal has four input fields labeled "Question 1", "Question 2", "Question 3", and "Question 4", each with a placeholder value. At the bottom of the modal are "Close" and "Finish" buttons. To the right of the dashboard, there is a "Leaderboard" section stating "Your class leaderboard is empty!". The browser's developer tools are open, showing the HTML code for the "Task" modal.

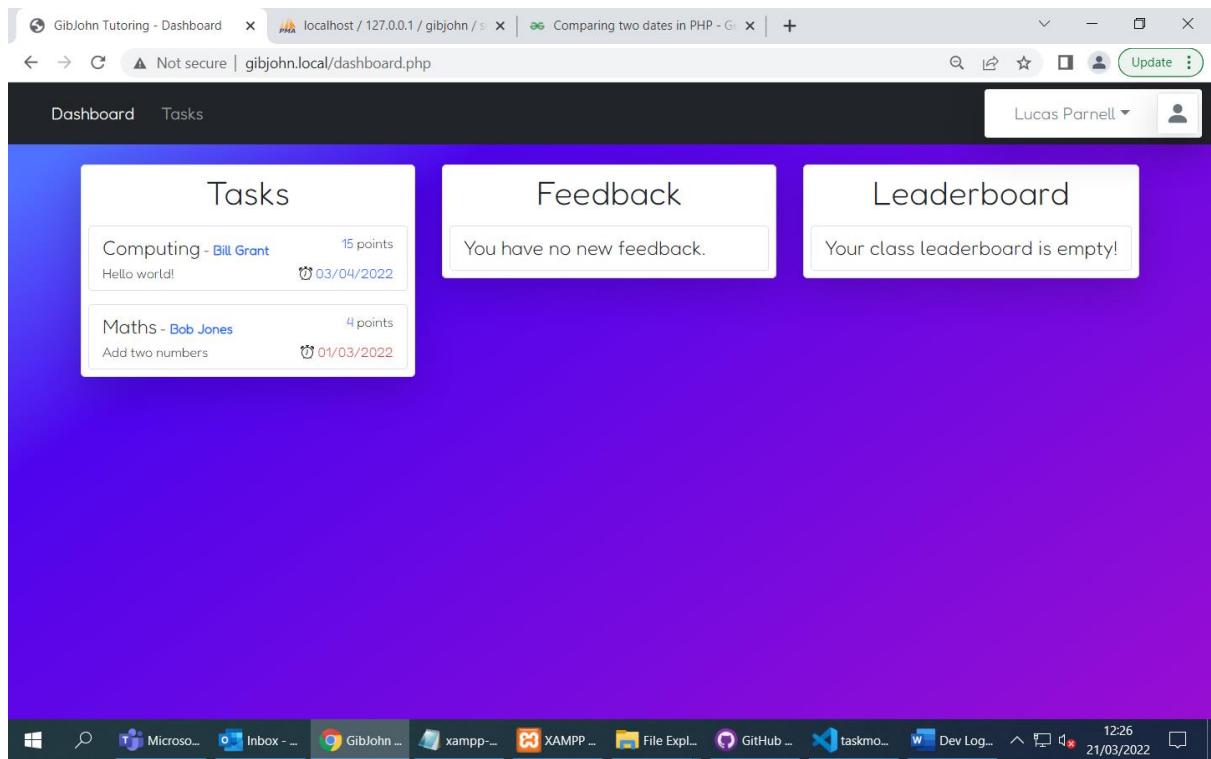
The screenshot shows the Visual Studio Code interface with a JSON file named "math0001.json" open in the editor. The file contains the following JSON configuration:

```

{
    "title": "Adding Two Numbers",
    "description": "Try and add as many numbers as you can.",
    "entries": [
        {
            "type": "text",
            "entry": "1+3?"
        },
        {
            "type": "image",
            "location": "res/img/math0001_1.webp"
        }
    ],
    "questions": [
        {
            "1": {
                "type": "textinput",
                "question": "1+3?",
                "answer": "18"
            },
            "2": {
                "type": "textinput",
                "question": "59+25?",
                "answer": "84"
            },
            "3": {
                "type": "textinput",
                "question": "43+67?",
                "answer": "110"
            }
        },
        {
            "4": {
                "type": "openinput",
                "question": "Describe and explain the process of vector multiplication"
            }
        }
    ]
}

```

- Almost finished task feature
- An alias set in the vhosts file is used to access task images without introducing security issues
- Can easily add questions, descriptions, textboxes, images and textareas in the JSON file
- Blade takes these variables and sorts them into a presentable template which is sent to the client.

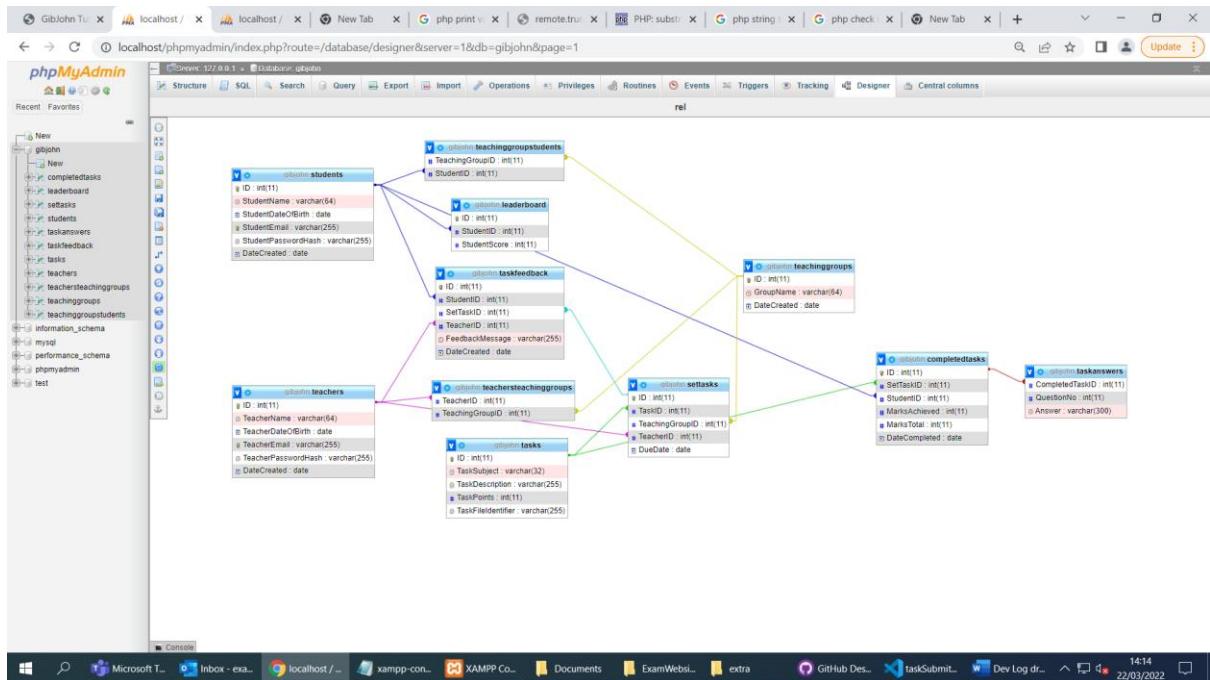


- Added an indicator for whether a task is overdue
- Started working on task submit process on the server-side
- Started work on some of the teacher side features

## Day 8

The screenshot shows a browser window with two tabs open. The top tab is 'localhost/phpmyadmin/index.php?route=/database/sql&db=gibjohn', which displays the phpMyAdmin interface. It shows the creation of two MySQL tables: 'CompletedTasks' and 'TaskAnswers'. The 'CompletedTasks' table has columns ID, TaskID, StudentID, MarksAchieved, MarksTotal, and DateCompleted. The 'TaskAnswers' table has columns StudentID, TaskID, QuestionNo, and Answer. The bottom tab is 'localhost:127.0.0.1', which shows a user interface for managing tasks and feedback. It includes a 'Tasks' section with a card for 'Computing - Bill Grant' and another for 'Maths - Bob Jones'. A 'Feedback' section says 'You have no new feedback.' Below is a 'Leaderboard' section stating 'Your class leaderboard is empty!'. The browser's developer tools are open, showing the HTML and CSS for the dashboard page.

- Answers are shown in the console after being returned by the AJAX PHP script



- Student side is functional, task answers are stored in the database
- Points are stored in leader board and displayed
- Since the database is relational, if a completed task is deleted, all of its associated values in other tables are too

```

File Edit Selection View Go Run Terminal Help taskSubmitter.php - workspace (Workspace) - Visual Studio Code
resources> controllers > taskSubmitter.php
UserAnswerable = [];

//Iterate through each user answer
//Get the id from the same in JSON
//Check if the question in the JSON file has an answer
//Compare answers

for($i=0; $i<count($_POST['formdata']); $i++){
    $currentFormId = $_POST['formdata'][$i]['name'];
    $currentFormValue = $_POST['formdata'][$i]['value'];

    array_push($userAnswers, $currentFormValue);

    $idPos = strpos($currentFormId, "_");
    //Since $idPos can be 0 but not false, is_numeric is used to check.
    if(is_numeric($idPos) && strlen($currentFormId)>0){
        $id = substr($currentFormId, $idPos);
        $id = intval($id);
        //We now have integer id from post.
        //Get the question id from the question in the JSON file
        if(isset($fileQuestions[$id])){
            //array_key_exists("answer", $fileQuestions[$id]));
            //This is answerable
            array_push($userAnswerable, $id);
        }
    }
}

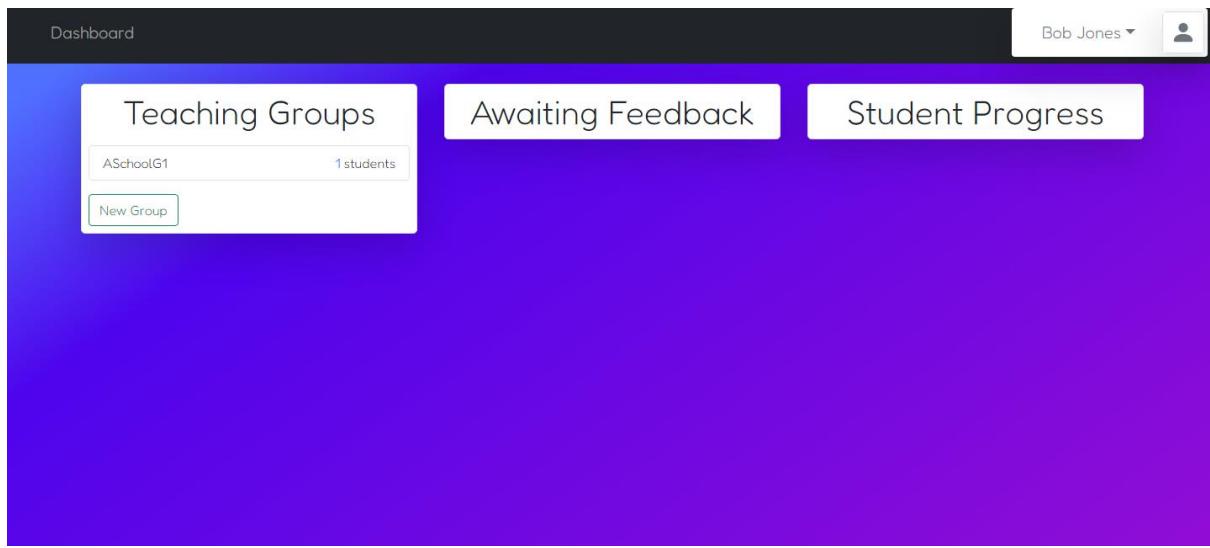
//Now we have array of user answers
//Compare answers against JSON id
for($id=0; $id<count($userAnswerable); $id++){
    $fileAnswer = $fileQuestions[$userAnswerable[$id]]['answer'];
    $fileMarks = $fileQuestions[$userAnswerable[$id]]['marks'];
    $userAnswer = $userAnswers[$userAnswerable[$id]];
    $userAnswer = str_replace("\r\n", "\n", $userAnswer);
    $userAnswer = trim($userAnswer);
    $userAnswerCorrect = ($fileAnswer===$userAnswer);

    $userAnswerCorrectStr = $userAnswerCorrect ? "correct" : "incorrect";
    $userAnswerCorrectStr .= "\n";
    print($fileAnswer . " = " . $userAnswer . " : " . $userAnswerCorrectStr . "\n");

    $result = new UserResult();
    $result->questionNo = $userAnswerable[$id];
    $result->marksObtained = $fileMarks;
}

```

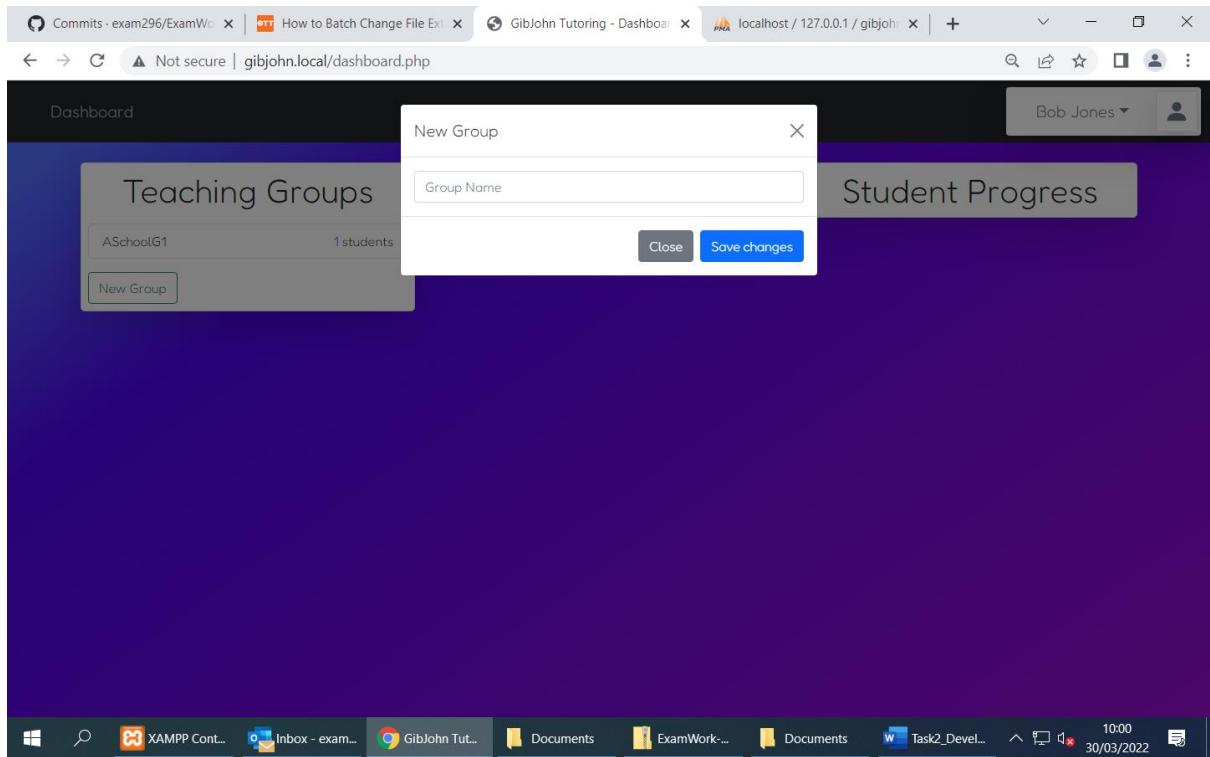
- Code to check user answers, open-ended answers will be added to task answers and teachers will be able to give feedback and award marks for them.



- Created layout for teaching groups on the teacher side.
- Will finish this next.

## Day 9

A screenshot of a web browser window. The address bar shows the URL "gibjohn.local/dashboard.php". The main content area displays a "Manage Students" modal. The modal has a title bar "Manage Students" with a close button. Inside, it says "Displaying 1 students". There is a table with one row containing a "Manage" button, a "Student Email" column with the value "Lucas Parnell", and a red "X" button. At the bottom of the modal are "Finalize" and "Close" buttons. In the background, the teacher dashboard is visible with its "Teaching Groups" card showing "ASchoolG1" and "1 student". The browser taskbar at the bottom shows various open tabs and windows, including "XAMPP Control", "Inbox - exam...", "GibJohn Tut...", "Documents", "ExamWork...", "Documents", "Task2\_Devel...", and system icons for date and time.



- Teachers can now create new teaching groups
- Teachers can now assign and remove students to groups given their email address
- Separated the student script file and the teacher script file
- Started writing ajaxManageGroups.php to handle teaching group changes

general\_teacher.js - workspace (Workspace) - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help general_teacher.js - workspace (Workspace) - Visual Studio Code

EXPLORER WORKSPACE (WORKSPACE) ...
models ...
views ...
layouts ...
modules ...
www ...
scripts ...
general_student.js ...
general_teacher.js ...
personal.blade.php ...
boilerplate.php ...
ajaxSubmitTask.php ...
taskSubmitter.php ...
taskanswer.php ...
completedtask.php ...
feedback.php ...
leaderboard.php ...
result.php ...
tasks.php ...
tasks.php ...
teachingGroups.php ...
user.php ...
views ...
ajax ...
layouts ...
login.blade.php ...
nonpersonal.blade.php ...
personal.blade.php ...
modules ...
studentdashwidgets.blade.php ...
teacherdashwidgets.blade.php ...
about.blade.php ...
home.blade.php ...
loggedin.blade.php ...
login.blade.php ...
logout.blade.php ...
signup.blade.php ...
studentdashboard.blade.php ...
teacherdashboard.blade.php ...
www ...
scripts ...
general_student.js ...
general_teacher.js ...
login.js ...
about.php ...
ajaxSubmitTask.php ...
ajaxManageGroups.php ...
ajaxSubmitTask.php ...
dashboard.php ...
index.php ...
loggedin.php ...
general_teacher.js

general_teacher.js
31 //Async stuff
32
33 $(function(){
34   $(".group-item-box").on("click", function(){
35     var teachingGroupId = $(this).attr("data-teachingGroup-id");
36     var thisIdString = "#teachingGroupModal[data-teachingGroup-id='"+teachingGroupId+"']";
37     let teachingGroupModal = $(thisIdString);
38     teachingGroupModal.modal("show");
39
40     $("#"+thisIdString+" .btn-manageGroups").off("click").on("click", function(){
41       //When group manage button is pressed-
42       //Display add student by email field
43       //Display cross icon next to entries
44       $(thisIdString+" .group-manage").fadeToggle(200);
45
46       //Handle adding and removal of students
47       studentsToAdd = [];
48       studentsToRemove = [];
49
50       //Handle adding students
51       $(".btn-addStudentToGroup").off("click").on("click",function(){
52         studentEmail = $(thisIdString+" .group-studentBox").val();
53         //Verify email
54         if(studentEmail.length<4 || !studentEmail.includes("@")){
55           return;
56         }
57
58         //Make sure student is not already in studentsToAdd
59         if(!studentsToAdd.includes(studentEmail)){
60           //If in studentsToRemove, just delete it from there
61           if(studentsToRemove.splice(studentsEmail).length<1){
62             studentsToAdd.push(studentEmail);
63             $(".manage-group-list").append($("#group-list-item").html());
64             newItem = null;
65             let groupItems = $(".manage-group-list").children();
66             groupItems.each(function( index ) {
67
68               let item = $(this).find(".group-item-student-name");
69               console.log(item.html());
70               if(item.html() === "Name"){
71                 newIndex = index;
72                 $(this).attr("data-temp-index", index);
73                 newItem = item;
74
75               }
76
77             });
78
79             let item = $(this).find(".group-item-student-name");
80             console.log(item.html());
81             if(item.html() === "Name"){
82               newIndex = index;
83               $(this).attr("data-temp-index", index);
84               newItem = item;
85
86             }
87
88           }
89         }
90       });
91     });
92   });
93
94 
```

```
File Edit Selection View Go Run Terminal Help
WORKSPACE (WORKSPACE)
resources > models > teachingGroups.php
11 $db = Utils::connectDatabase();
12
13 $teachingGroups = [];
14
15 //Get the teaching groups that this teacher is part of
16 $sql = <<<SQL
17     SELECT * FROM `TeachersTeachingGroups` WHERE `TeacherID` = {$this->user->getUserId()};
18     SQL;
19
20     $result = $db->query($sql);
21
22     while($row = $result->fetch_assoc()) {
23         //Return each teaching group
24         $sqlTeachingGroup = <<<SQL
25             SELECT * FROM `TeachingGroups` WHERE `ID` = {$row["TeachingGroupID"]};
26             SQL;
27
28             $groupResult = $db->query($sqlTeachingGroup);
29             $teachingGroup = $groupResult->fetch_assoc();
30
31             array_push($teachingGroups, $teachingGroup);
32     }
33
34
35     return $teachingGroups;
36
37 function newTeachingGroup($teachingGroupName){
38     $db = Utils::connectDatabase();
39     $today = Utils::today_dbFormatted();
40
41     $sql = <<<SQL
42         INSERT INTO `TeachingGroups` VALUES (NULL, '$teachingGroupName', '$today')
43         SQL;
44     //create teaching group
45     $db->query($sql);
46
47     //$/db->insert_id is a thing?
48
49     //Assign teacher to teaching group
50     $sql = <<<SQL
51         INSERT INTO `TeachersTeachingGroups` VALUES ('{$this->user->getUserId()}', '{$db->insert_id}')
52         SQL;
53
54     $db->query($sql);
55 }
```

Line 22, Col 45 Spaces: 4 UTF-8 LF PHP 10:03 30/03/2022

## Some of the TeachingGroup class