CODESET

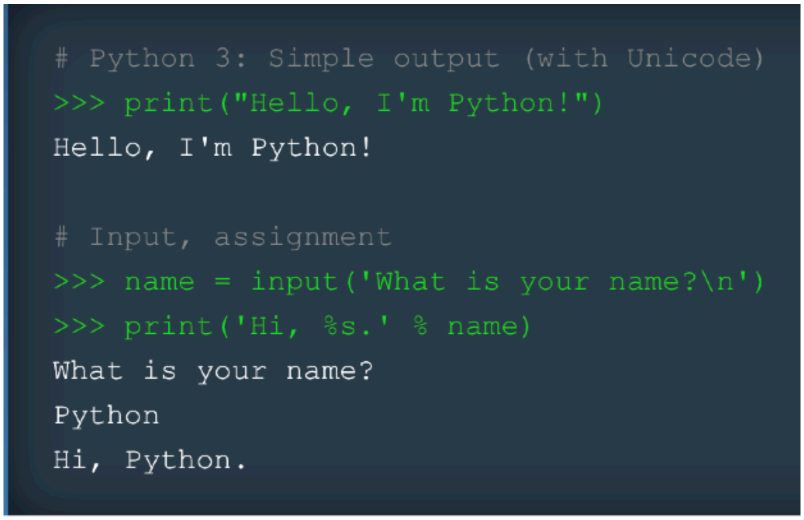
A2 COMPUTING PROJECT

Ben Mechen Centre Number: 55217 Candidate Number: 4374

CONTENTS

ANALYSIS

Introduction



```
# Python 3: Simple output (with Unicode)
>>> print("Hello, I'm Python!")
Hello, I'm Python!

# Input, assignment
>>> name = input('What is your name?\n')
>>> print('Hi, %s.' % name)
What is your name?
Python
Hi, Python.
```

Schools now teach computing in Year 7 instead of ICT. This means that the students need to learn to program in class - in many schools, Python is the language of choice for students. Python is an easy to use language for beginners, but can also be very powerful for more experienced programmers. This means that it is not too difficult for students to pick up with no programming knowledge, but still allows them to use it to create more complex programs in later years. Python is a very versatile language, and can be used for anything from web and internet development, to scientific models, to desktop software development. It is the language of choice for many big companies, such as YouTube and Google. Because of this, Python is a very useful language to learn. Also, the computing teachers that I spoke to told me that GCSE and A Level course is becoming more and more popular with students every year, as there is a national shortage of skilled developers - this means that more people will need to learn to be taught to code, so a system to make this task simple would be very useful for teachers.

However, as students joining school can have a very varied amount of computing skill – as some students do not come across programming until computing lessons in Year 7, while others already have computing experience – teaching students to code in lessons can be very difficult, as the students progress at different rates. This means some easily grasp the concepts behind programming and must wait for other students who need more time. It is also difficult for students to program while at home, as many people do not know how to install and run Python on their computer. Also, while not in computing lessons they cannot quickly get help from their teacher if they get stuck on code.

Currently, students are taught the basics of Python by the teacher talking to the whole class, as well as using a whiteboard and computer to demonstrate certain aspects, and worksheets for students to fill out using a pen. This is a normal method of teaching, and works well in other subjects, however computing is a new and unconventional subject, requiring a new and unconventional method of teaching. My client, Mr Scott-Brown - a computing teacher at a local school teaching students in years 7 to 11, hopes that the new system will allow the students to be more interested in computing, as they will be more hands on and can compete with other students to get more points by doing more programming. He also believes that the new system will allow the students to progress at their own pace, and he will be in a position to offer one-to-one support to students struggling.

Possible Solutions

1. Manual System:

    i. The teacher standing at the front of the class writing code on a whiteboard. Students then are given a task and go and code it on their computers. If they get stuck, they must put up their hand and wait for the teacher to help them, which may stop students from asking for help, and once students are done they must sit there and wait for other students. Also, when writing code on a whiteboard it is difficult to write complex code and the teacher cannot demonstrate their code.

    ii. Demonstrating code on a computer connected to a projector. This method is more interesting for the students, as they can see the code being run. They must then complete a task on their computers like the other manual solution above.

2. Interactive System - this would allow students to learn Python at their own pace, inputing code to complete tasks. They can compete against other students in the class to complete as many Python lessons as they can, and the teacher can set deadlines for the students to reach:

    i. Web based system - This would allow the system to be based on a remote web server that can be accessed from anywhere using a web browser on any device. This means their progress and details would be synched across all their devices automatically. I would use HTML, CSS and Javascript on the client side, and PHP and Python on the server side. The users would not need to install any other software than a web browser, so it would work cross-platform, as long as the user has internet access.

    ii. Non-web based system - This would still require a server to be hosted to store user's data, however each individual school would need to set up a server and install the host software, then download the client software on every computer. Also, for users to use the system at home, they would need to download and install the client software, which would use space on their computer. This system would be much more difficult to install and maintain than the other solutions, and would require the software to be rewritten in different languages for different operating systems. This solution would however work offline.

I believe the most appropriate solution for the problem would be 2i - an interactive, web based system. This is because I feel it would be the simplest system for the end users, making it more accessible to them and does not need any complicated installs to set up the system. Also, after speaking to IT teachers and network admins, the manual solutions would are not very interesting and are time consuming for teachers, and the non-web based system would be difficult to install on all school computers and get a server set up and maintained, so a remotely hosted, interactive system would be best. I can use HTML, CSS and Javascript for the client side to build the interface, and PHP and SQL to run on the server, accessing the databases to retrieve and insert information in response to user input, running processes in the background, and creating a sandboxed Python environment and passing code to it and retrieving the output.

Users

The main users of the system will be teachers and computing students. I have spoken to a number of computing teachers to find out what features would be useful to them. The teacher will see different pages to what students will see, and will be less likely to take the Python course themselves, so that will not be as important a feature on the main page once the teacher has logged in. Also, teachers will have a page not accessible to students where they can see and overview of and manage their classes and the students in them - when I observed a computing class, I found that students often will mess around in class, and like to try to access things they are not allowed to, so these pages need to be securely protected from students.

The classes usually consist of between 20 and 30 computing students from Year 7 to Year 11, and there are normally between 2 and 4 computing classes per year. Most of the users would be in Year 7 and 8, as that is now where the basics of programming is taught, however students in other year groups could take the course again to practice - from my observation of a Year 10 computing class, I saw that there is a large knowledge gap between some students in the class, so being able to work through the problems in their own time and being able to revisit the course would be beneficial to some students to stop them from falling behind others.

Because there will be lots of users from different years, with different experiences with using computers, the system needs to be easy and intuitive to use so that it does not end up slowing down teaching programming rather than making it quicker.

User Needs

Mr Scott-Brown needs a system to manage his classes and teach students Python. The students and teachers need to be able to access the system from any computer at school and at home so that they can learn Python in lessons and as homework. Their progress needs to be synced so that they do not lose any progress by changing devices, and can make changes to their account which are then reflected instantly across all the devices they use to access the system. Teachers need to be able to see students' progress updated live to be able to monitor the class's progress, from school and home.

The two types of users - students and teachers - have different needs from the system. Students need access to a high quality Python course that is easy for them to understand and complete the tasks set in the course. To carry out the tasks, they need to have access to a Python environment that can be used on any device without the student having to install Python on their computer and making sure it is kept up to date and has the correct libraries installed, as getting all users to install Python and keep it updated to the same versions would be very difficult. Teachers need to be able to have a clear overview of all their classes, to monitor their progress. They need to be able to set deadlines for the class, so that the students in that class must reach a certain point by a certain time, and they need to be able to add and remove students from a class.

General Objectives

Mr Scott-Brown needs an interactive system for students to learn how to code in Python, so that they can progress at their own rate and are kept interested in programming in Python. The system that I am going to create will allow the teacher to track the progress of students in their classes, and help students when they are stuck. The teacher needs to be able to manage their classes, adding students and setting those students deadlines.

Objectives For The Proposed Interactive System

1. The system needs to be accessible for students and teachers from school and at home, keeping their progress update across all devices the system is used on.

2. The Python environment must be secure and isolated from the rest of the system, so that unsafe code cannot cause damage if it is run.

3. The interactive course must cover the basics of Python so that the students can gain a solid understanding of the programming language, and learn at their own pace.

4. The proposed interactive course must have clear explanations and questions, with sample code and hints to help the students.

5. The system must allow students to get help from their teacher if they get stuck at home or in lesson.

6. The interactive system must allow students to be able to track their own progress, and compare their progress against other students in the class.

7. Teachers must be able to track the progress of students against others in the class and set them deadlines, monitoring their progress in reaching these deadlines.

8. The users of the system must be able to easily update their details and preferences without access to the user database.

9. The interface of the system must be easy and intuitive to use so that users do not need to be trained on how to use it. The aesthetics of the system must also be taken into account during its design.

10. Teachers must easily be able to create classes and add students to them.

Limitations

The learning system will not be able to cover every topic in Python, only the basics, as otherwise the course would go on for too long and students would lose interest, which, as I saw when I observed a computing class, lead to them messing around and distracting others. Also, currently there won't be any other programming language courses on the system - just Python, which means once students have finished the course it is unlikely they will return to the website. After speaking to teachers, I found out that the GCSE computing syllabus often changes every few years, so the course will need to be kept up to date to fit with the specification.

Client Interview

My client is Mr Scott-Brown, a computing teacher. In computing classes, he teaches Python to his students.

HOW DO YOU CURRENTLY TEACH STUDENTS PYTHON?

- *My current method is to stand at the front of the class with students sat at tables and explain a concept to them, by using the whiteboard to draw diagrams and the interactive whiteboard to display examples so they can see the code and how it works. I also occasionally hand out worksheets for them to fill out. Once I have shown them something, I usually then set them a task to do on the computers.*

HOW EFFECTIVE IS YOUR CURRENT METHOD?

- *As many of my students are new to programming they often get stuck, so will put their hand up and have to wait for me to help them. This means they do not gain confidence in debugging their programs themselves, and cannot complete programming tasks outside of lessons. Also, as my students have different skill levels some find programming easier than others, so often have to sit there and wait while I help other students. Finally, some students are not able to program outside of lessons as they do not have access to a computer with Python installed.*

DO YOU FEEL THAT AN ONLINE SYSTEM TO TEACH STUDENTS PYTHON WOULD BE?

- *Yes, if it provided examples and then allowed students to enter and run their code, with feedback given to tell them if it is correct. It would also be good because the students could access the system from anywhere.*

WHAT OTHER FEATURES WOULD YOU FIND USEFUL?

- *It would be good if I could have an interface to the system so that I can get an overview of the students in each class and monitor their progress. It would also be useful to be able to set tasks for the students to complete by a certain time. I think it would also be a good idea for students to interact with each other, possibly by competing in a leaderboard - this would make them more likely to try to complete tasks.*

HOW MANY PROGRAMMING TASKS DO YOU THINK WOULD BE NEEDED ON THE SYSTEM?

- *There needs to be enough so that the content is not delivered too quickly to the students, as the difficulty may put students off. However, there shouldn't be too many as students would then become bored.*

Data Sources and Destinations From User Inputs

After determining the objectives, I believe the following inputs from the users will be required on the system:

Data Source (Input)	Process	User	Destination
First name, email, password	Creating teacher account	Teacher	Details inserted into user database
Email, password	User log in	All	Details are checked against user details in database
Email	Password reset request	All	Email is sent to user's email
Key, new password	Resetting password	All	Key is checked, if correct user's password in database is updated
Name	Changing user name	All	Database containing user details is updated with new name
Email	Changing user email	All	Database containing user details is updated with new email
Old password, new password	Changing user password	All	Old password is checked against database, if correct password in database is updated
Class name	Creating class	Teacher	New row is added in classes database, with class name inserted
Student name, student email	Creating student account	Teacher	New user is created with details given
New class name	Changing class name	Teacher	Database containing classes is updated with new class name
Deadline point, deadline date	Creating deadline for class	Teacher	New row is added in database containing deadlines, with deadline details inserted
Python code	Running code in Python Environment from course	All	Code is checked for security issues and run in a sandboxed Python Environment, with output returned to process running the code

The main flow of data from the user to the program will be from where the user's Python code is sent to the system to be checked and run, then the output is sent back. The rest of the data in the table from the users will not be sent often, only when they create or change users or classes, and is mostly short strings, so the system won't need to process large amounts of data, and the transferring of data won't need lots of bandwidth.

Data Storage

The majority of user data will be stored in databases. Most of the user data is integers or strings, and would not be very long - therefore the databases would not become too large even if lots of users register. Some larger pieces of data, such as the user's code for each question, will be stored in text files on the server, as they will need to be accessed

On the user's side, the user's details from login will be stored on their browser in a cookie, so that the pages on the site they visit can use the data stored in the cookie to personalise the page with the user's data pulled from the database, such as their name. A cookie can hold up to 4096 bytes, however the cookie for the user's details will be no where near that amount, as it will only need to hold a unique id for the user's session, which can then be used to fetch data from the databases.

DOCUMENTED DESIGN

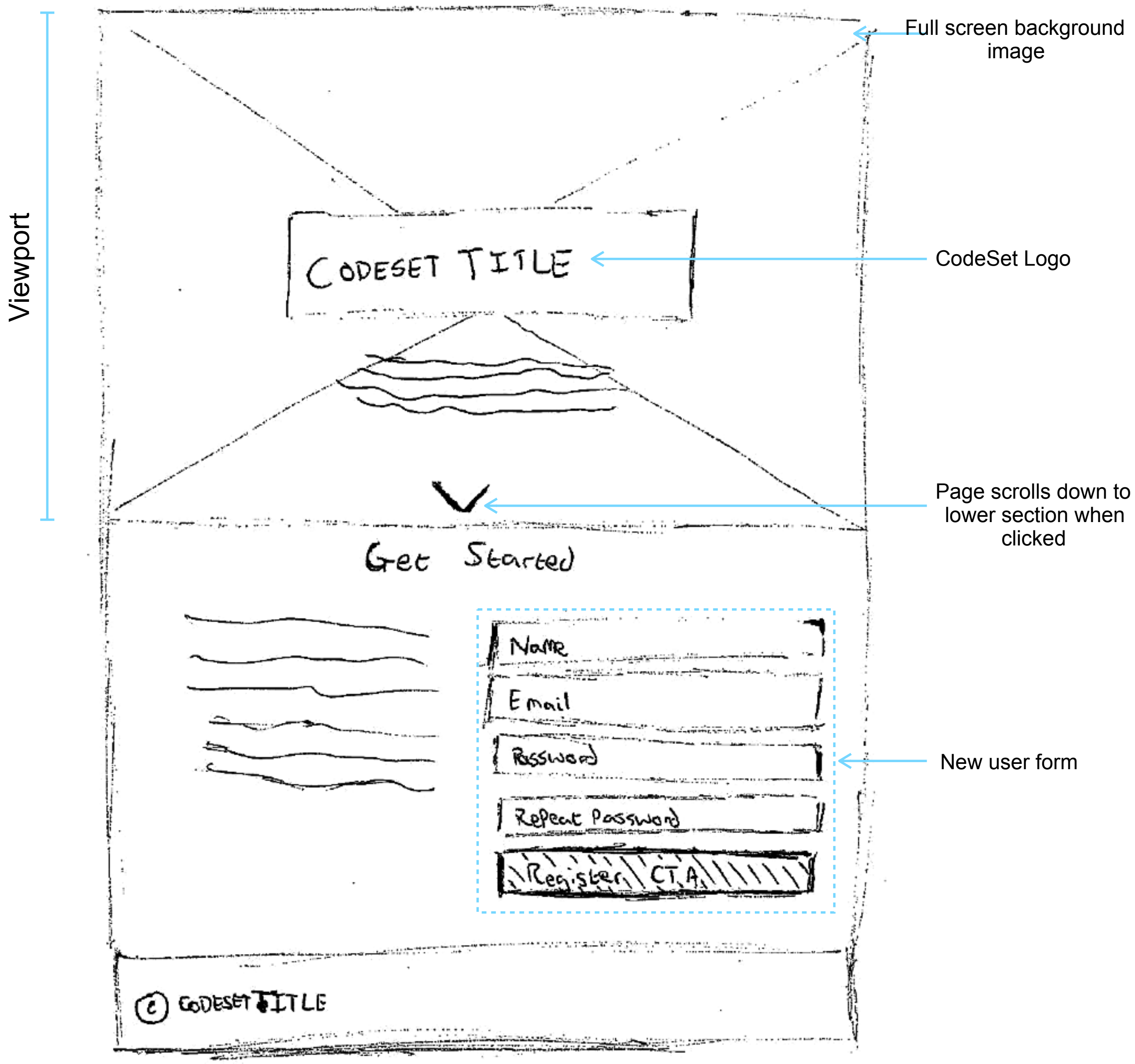
Overview

The aim of this project is to create a system to provide teachers with a new way of teaching programming to students with little to no computing experience in class. They need to be able to get an overview of the student's progress, create and edit their classes, and set deadlines for students to meet by a certain time. The students need an interactive and interesting system to learn Python, where they can complete tasks that are suitable for their experience and are not too complicated. The students should be able to compete against other students in their class, by completing tasks as quickly as possible. The students should be able to easily write their own code to complete the tasks, and should be able to run it without waiting too long - the Python environment needs to be able to handle multiple users without the performance being affected. The Python environment also needs to be restricted, so that potentially unsafe code cannot cause damage to the server. The user interface needs to be easy to use and navigate, as well as looking interesting.

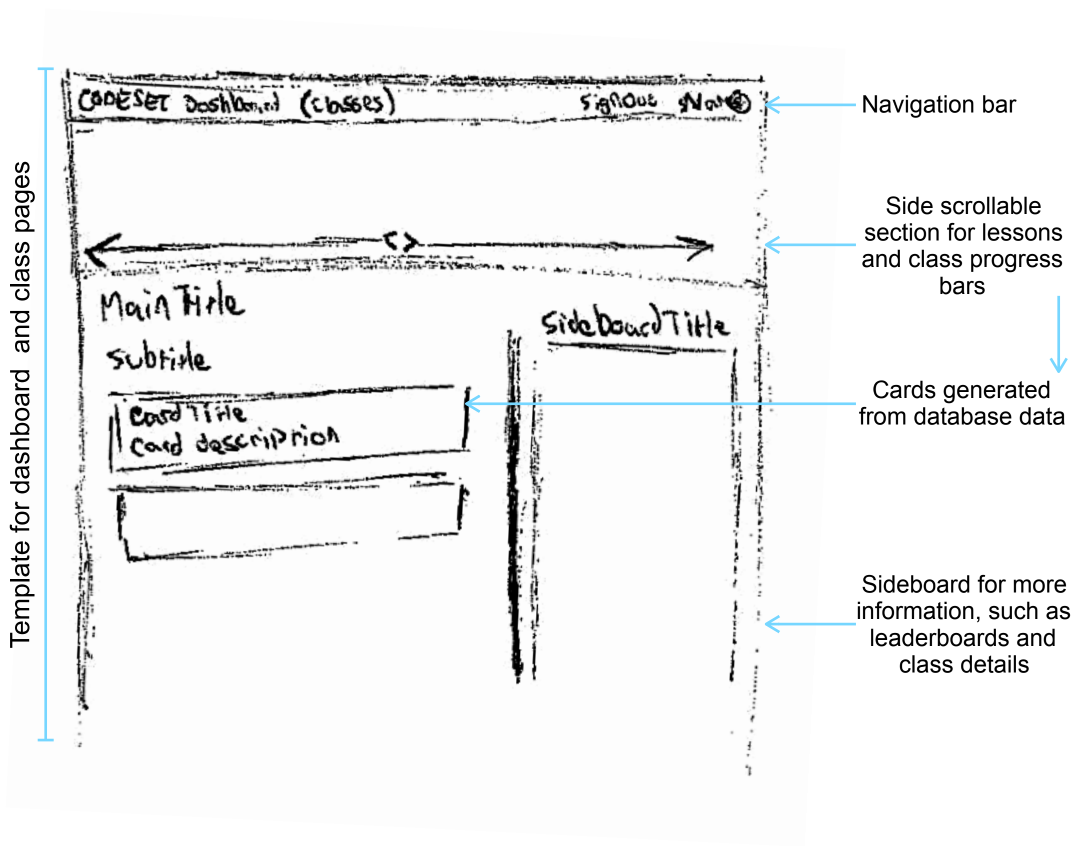
Table of Main Steps

Main Stage	Step	Description
Design	Wire framing	Create a wire frame drawing on paper of the main page layout
	Visual design	Using Adobe Photoshop, create a visual mockup of the main pages in the site
Front End Page Creation	Landing page - HTML, CSS and Javascript	Creating the page, with all text, images, and interactive elements
	Sign in and reset password pages - HTML and CSS	Creating the page containing the sign in form
	Dashboard page - HTML, CSS, and Javascript	Creating the private dashboard page, with the dynamically populated areas created ready for the backend implementation
	Classes pages - HTML, CSS and Javascript	Creating the teacher's pages to manage the classes, with the sections dynamically filled ready for the data from the database
	Account pages - HTML, CSS and Javascript	Creating the interface to change user's details, such as profile picture, name password, for example, with user's information to be filled from the database
	Python course page - HTML, CSS, Javascript	Creating the template for the lessons, with the question information split into sections and filled from the database
Backend Systems Creation	Create databases	Create the databases for authentication, classes, news and course
	Authentication system - PHP	Set up the authentication system for the site to manage users
	Create Mail class - PHP	Create the PHP class to handle all emails sent from the server
	Create Classes class - PHP	Create the PHP class to create, manage and delete classes containing students
	Create Course class	Create the PHP class to fetch question data, run code, and check code
	Create News class	Create the PHP class to get news from the database, and insert news into the database
	Create Profile Upload system	Create the system to manage changing user's profile pictures

Main Page Template Wireframes



CTA = Call To Action



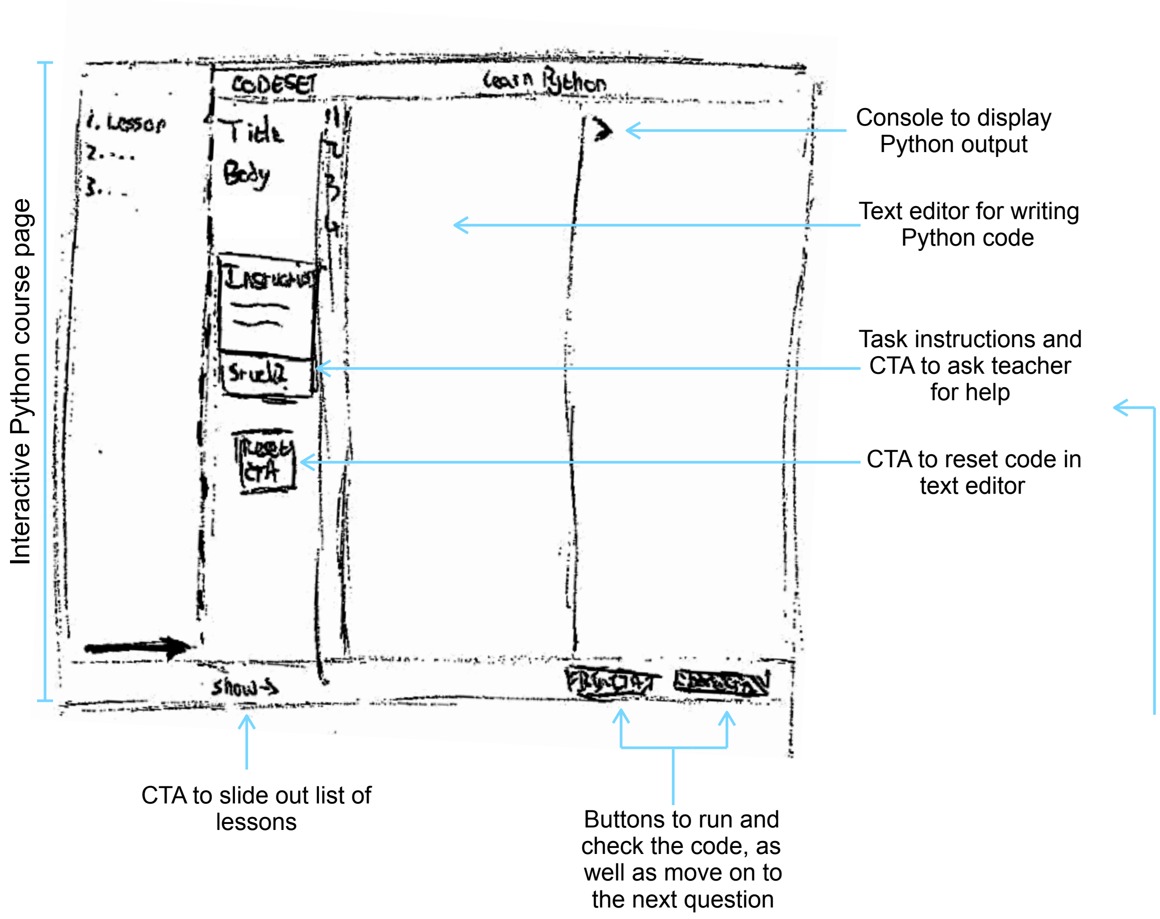
Template for dashboard and class pages

Navigation bar

Side scrollable section for lessons and class progress bars

Cards generated from database data

Sideboard for more information, such as leaderboards and class details



Interactive Python course page

Console to display Python output

Text editor for writing Python code

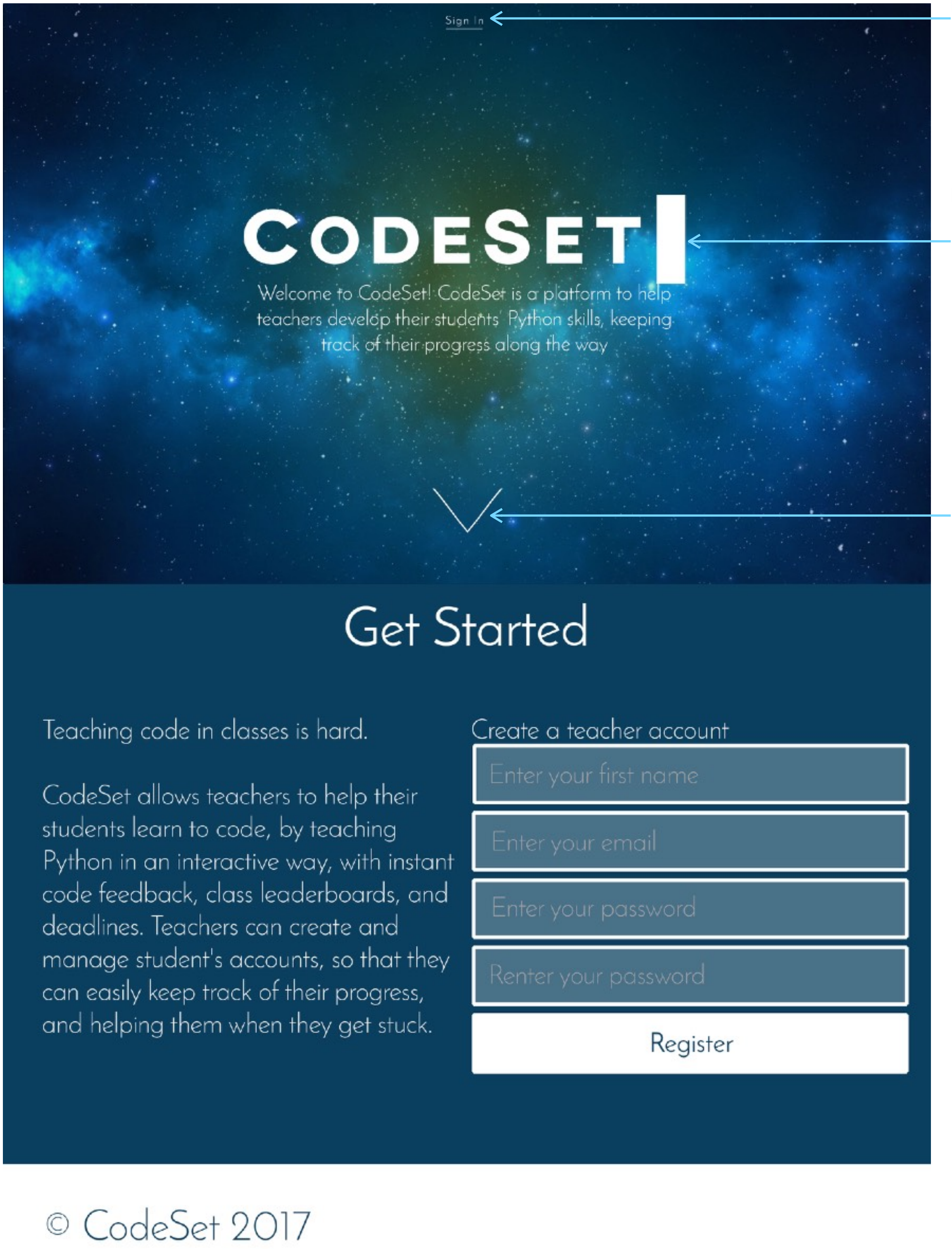
Task instructions and CTA to ask teacher for help

CTA to reset code in text editor

CTA to slide out list of lessons

Buttons to run and check the code, as well as move on to the next question

Visual Design - Photoshop



Sign In

CODESET

Welcome to CodeSet! CodeSet is a platform to help teachers develop their students' Python skills, keeping track of their progress along the way

Get Started

Teaching code in classes is hard.

CodeSet allows teachers to help their students learn to code, by teaching Python in an interactive way, with instant code feedback, class leaderboards, and deadlines. Teachers can create and manage student's accounts, so that they can easily keep track of their progress, and helping them when they get stuck.

Create a teacher account

Enter your first name

Enter your email

Enter your password

Renter your password

Register

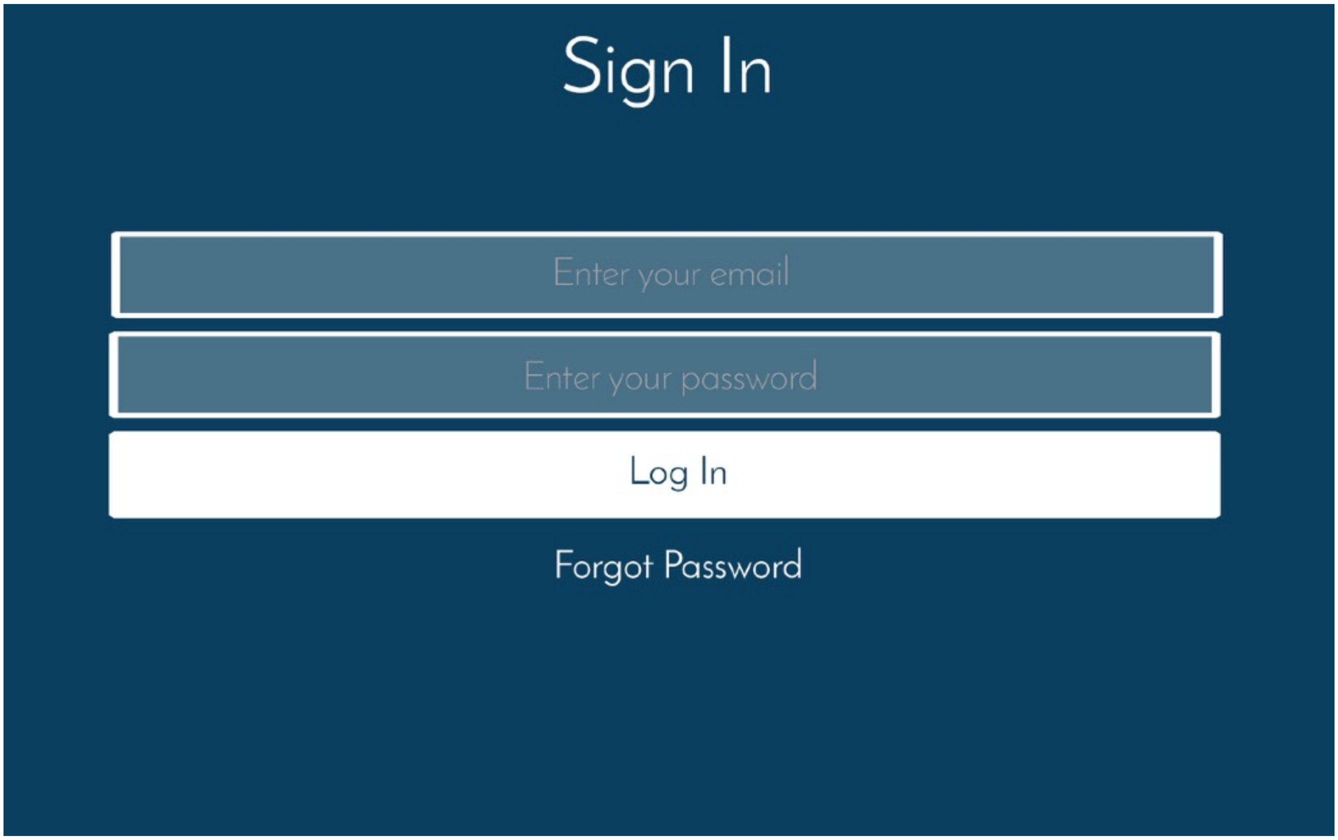
© CodeSet 2017

Link to take users to sign in page

Bar at the end of the logo will fade in and out continuously to look like the cursor from a text editor

Arrow smoothly scrolls the page down to the next section

This is the home page that users will fist land on. It is very clean and simple, with a limited amount of writing to keep users interested. It is made up of two fullscreen sections, with an arrow that when clicked scrolls the page down. The first section has a large, animated logo, introductory text and link to the sign in page. The second section is split into two columns, the left explaining CodeSet in more detail and the right containing a form for teachers to create their account.



This is the sign in page. It is very simple and easy to navigate, with large and clear text inputs and buttons. If the users forgets their password, they can click in the "*Forgot Password*" link underneath the form. This will take them to a page with the same design as the sign in page, where they enter their email. They will then have instructions on how to change their password sent to that email address. The email will have very limited styling on it so that it displays properly on all devices and is not too large to be emailed.

If the details entered are not correct or there is an error, a message will be displayed above the email box, telling the user what has happened. If the details are correct, when they click the *Log In* button they will be sent to the dashboard.

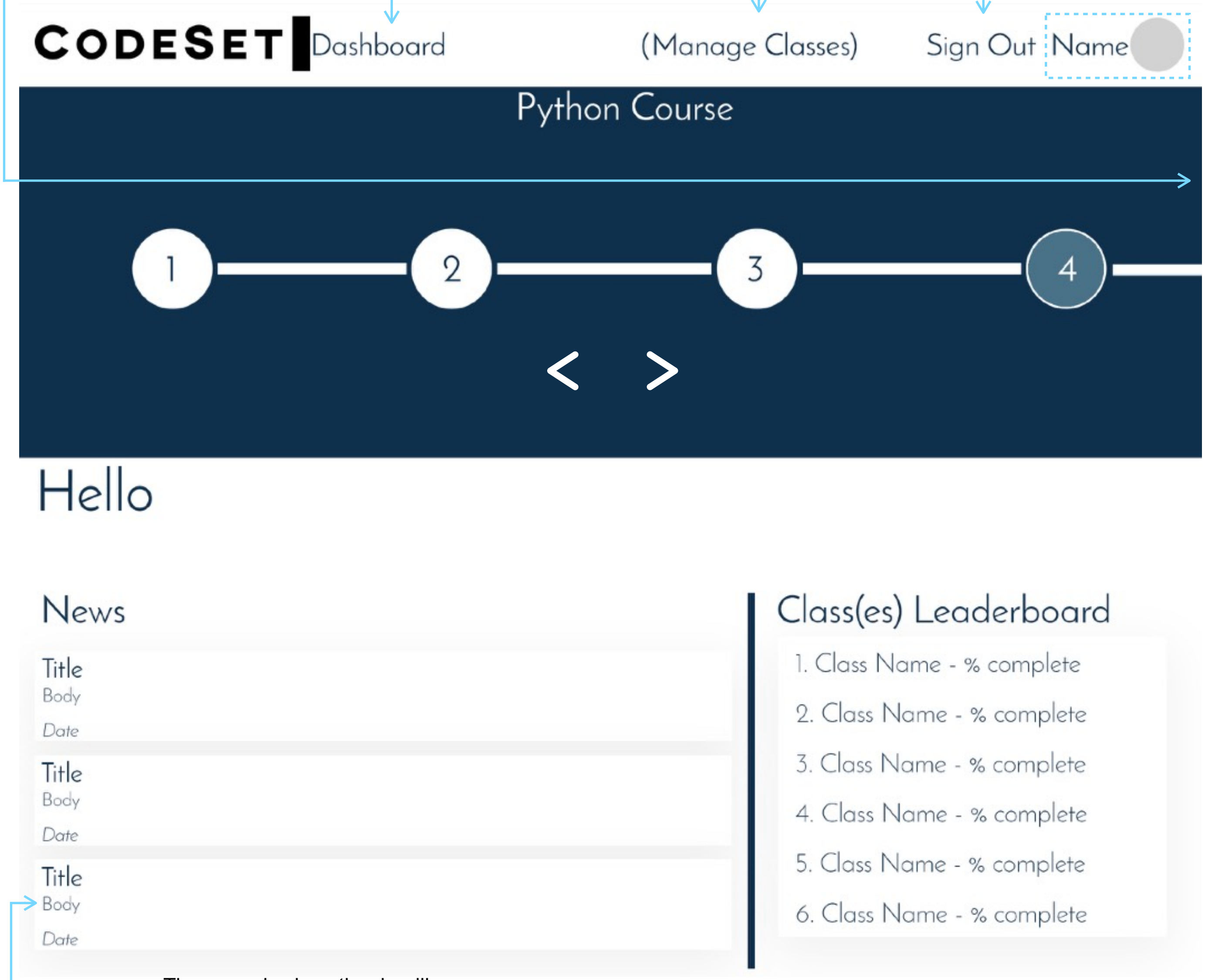
This section will scroll horizontally, showing the courses the user has and hasn't completed. When the user hovers the mouse over the area, the cursor changes to two arrows, so the user can click and drag to slide the section. The questions the user has completed are filled solid white.

Link to the current page. It is slightly darker and the font has a larger weight to show that it is the page the user is currently on

If the user is a teacher, a link will display to take them to a page to manage their classes

Sign out button - removes cookies containing user details and takes user to main page

User Name and profile picture; if the user clicks on either they will be taken to the *Account* page



Hello

News

Title
Body
Date

Title
Body
Date

Title
Body
Date

Class(es) Leaderboard

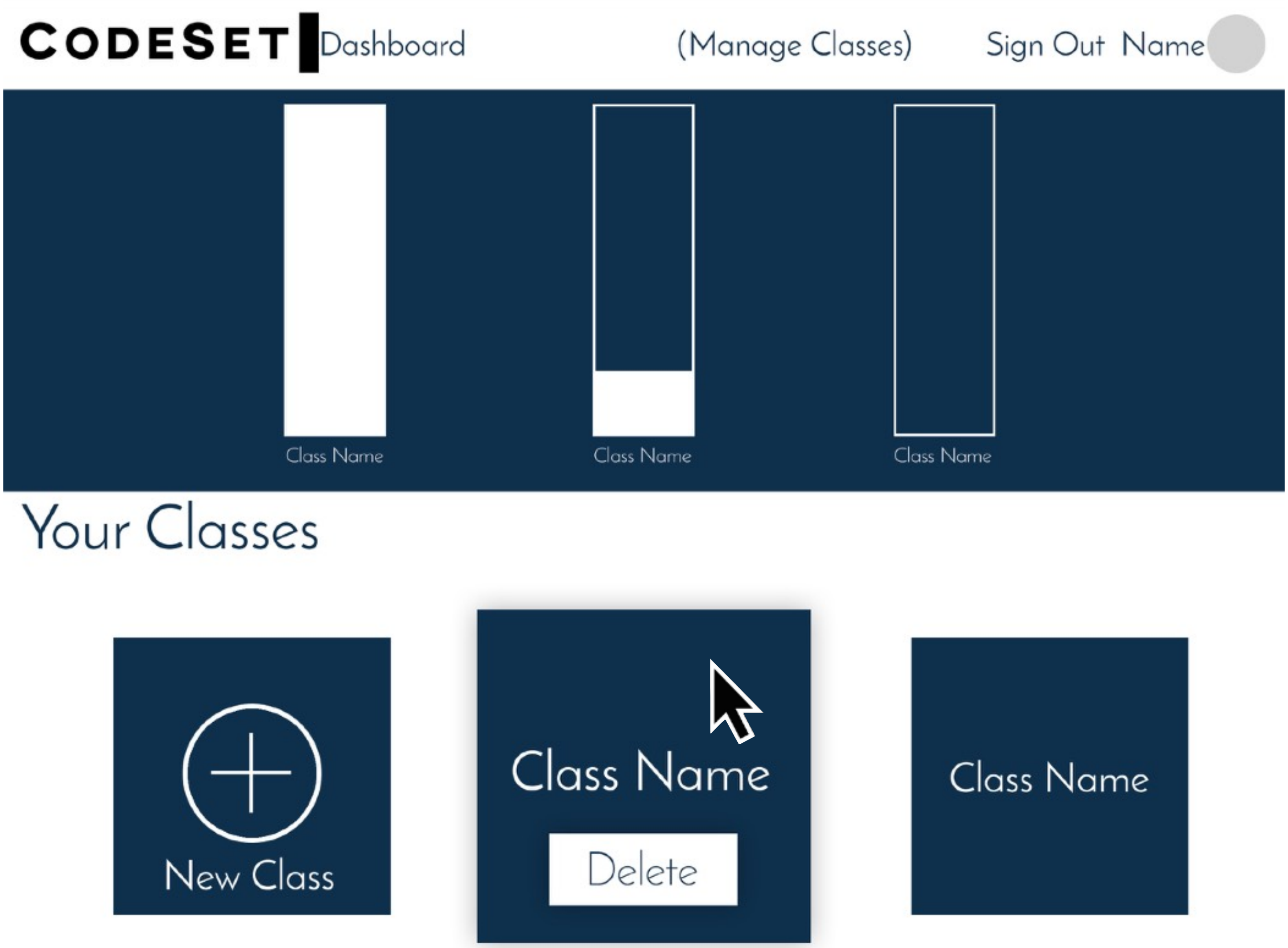
1. Class Name - % complete
2. Class Name - % complete
3. Class Name - % complete
4. Class Name - % complete
5. Class Name - % complete
6. Class Name - % complete

These cards show the deadlines and news from the class or classes the user is in, depending on whether they are a teacher or student. The information will be pulled from a database

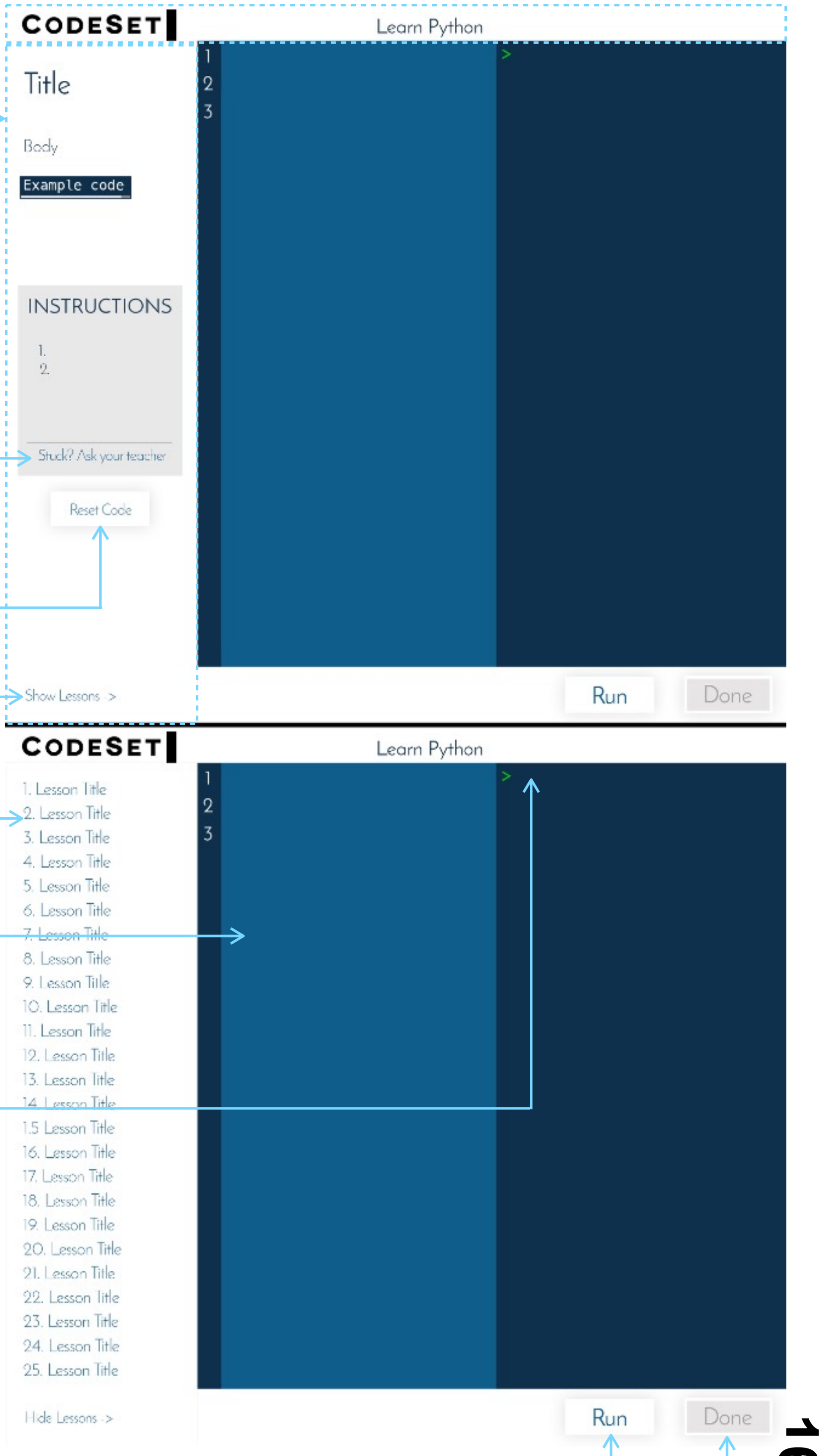
This sidebar ranks the progress of students in a users class, or a user's classes, by the percentage of the course the students have completed

This is the Dashboard page. It is the page users will see when they log on, and summarises data from the main features and shows it to the users - the information displayed changes based on whether the user is a student or teacher.

This is the *Manage Classes* page. The page is based on the main *Dashboard* page without the side section, which is the template for most of the pages once the user has logged in. It gives teachers and overview of their classes, so that they can see their progress and compare them to the other classes. The progress is displayed in vertical bars showing the percentage complete of the class calculated from the number of questions students in the class have completed. Below that section there is a grid displaying the user's classes. The top left grid item is a button to create a new class. A pop up will then show which allows the user to enter the name for the new class. The other grid places are then populated with the user's classes pulled from the database. If the user clicks on one, they will be taken to the class's page. If they hover over the class, the grid item will grow and a delete button will appear.



This is the page to teach Python. At the top, there is a header with the CodeSet logo - which is a link to get back to the main site - and the text '*Learn Python*'. On the left of the page there is a side bar containing the lesson's title, introduction including example code, the lesson's instructions, a button to request help from their teacher (which will send an email to their teacher containing their problem and code), a button to reset the code in the editor, and finally a button to show the list of lessons in the course. The middle section is a code editor, with line numbers and desktop code editor features such as syntax highlighting, code completion, indentation and bracket matching. On the right, there is a section for the terminal, which will display the output and input requests from the program. At the bottom of the screen there is a footer containing the Run button to run the program, and the Done button to move to the next question if the code is correct.



Header containing logo and page title

Sidebar containing title, introduction, instructions, example code, instructions, and reset code button

Emails teacher student's code and question information so that the teacher can help the student

When the user clicks reset, the code in the editor is updated to the default code stored in the database

When clicked, the sidebar will change to the view below, showing the lessons

Code editor, where students enter their answers to the question

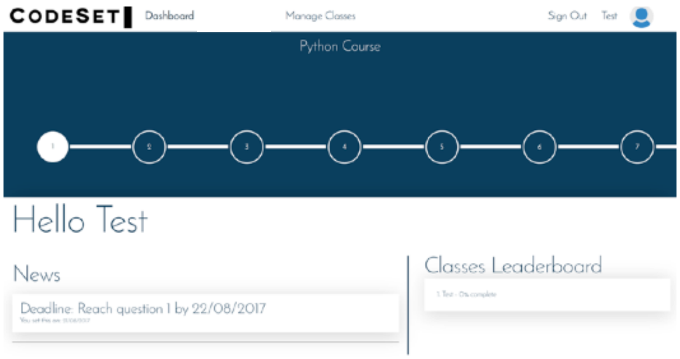
Terminal showing code input and output

Run and Done buttons. When the user clicks Run, their code is run and checked to see if it is correct. If it is correct, the Done button is changed from grey to white, and is clickable so that when the user clicks it they are taken to the next question

Front End Creation

I first made the landing page, using HTML to create the main structure, then CSS to add styling and to position the elements, then JavaScript to add the dynamic and interactive elements. This page was very simple to make, using CSS transitions to make the logo flash, and HTML IDs and javascript to make the page scroll when the arrow is clicked.

The next pages I created were the sign in and reset password pages. As the Authentication system was not set up yet, the form did not work at that stage however the page was ready for the system to be included to make the page functional.



Hello Test

Deadline: Reach question 1 by 22/08/2017

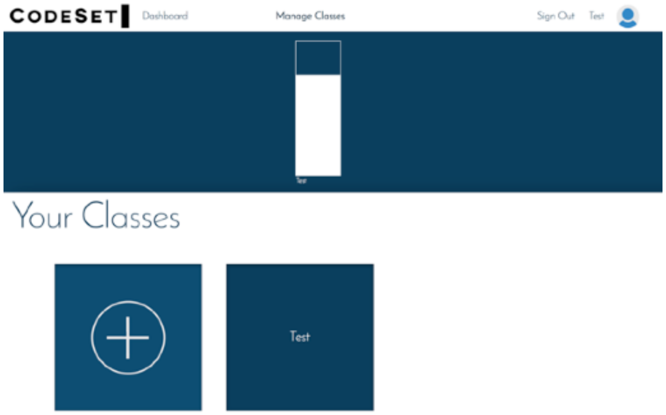
1. Test - 0% complete

After this, I made the dashboard page. As almost all of the text and graphics on the page are generated from databases, I made the page with placeholder data. The dashboard only shows the "*Manage Classes*" link if the user is a teacher, and the Leaderboard section displays differently for students and teachers - for students the leaderboard is for all the students in their class as students can only be in one class, but teachers can have multiple classes so the leaderboard shows the overall progress of each class. The News section displays approaching class deadlines at the top, then other pieces of news, such as students' progress, underneath. Each item of news is displayed in it's own card, containing the title, body and date of each piece of news. The dark blue section of the page contains the section containing the links to the Python course. When the user has not yet started the Python course, a start button is displayed, otherwise the course is shown on a horizontal line, with circles containing numbers on that line - the circles represent the lessons on the course, if the circle has a white background the lesson has been completed, otherwise it is yet to be completed. This section of the page scrolls horizontally, and originally the main method of navigating it was by clicking on the mouse and dragging left or right, and when the user hovered their mouse over the section the cursor would change to two arrows pointing left and right. However, this was not very intuitive or easy to navigate, and I thought that many users would only need to find the next lesson in the course rather than choosing random lessons from different points in the course. Because of this I made it so that the section automatically scrolls to the last completed lesson, and I changed the cursor back to default. n my first attempt to make the section scroll automatically to the last completed lesson, I tried to break the course up into seven sections and determine what section the lesson required was in, then scroll to that section. However, this method did not work well and would not work at all on a different sized screen:

```
var lastBarrier = 0;
var barriers = [];
var n = 1;
for (var i = 1; i <= <?= $codesetCourse->getLastQuestionID() + 2 ?>; i++) {
      if (i % 7 == 0 ) {
            barriers.push([lastBarrier + 1,  7 * n]);
        lastBarrier += 7;
        n++;
    }
}
var section = null;
for (var i = 0; i < barriers.length; i++) {
      if (<?= $user['progress'] ?> >= barriers[i][0] && <?=
$user['progress'] ?> <= barriers[i][1]) {
            section = i;
      }
}
$('#dashboard-progress').scrollLeft($( document ).width() * section);
```

This method was very overcomplicated, so in my second attempt I narrowed it down to one line of code, by using the jQuery library's `scrollTo` function to scroll to the element on the page with the `'dashboard-progress-circle-current'` id:

```
$('.dashboard-progress').scrollTo('#dashboard-progress-circle-current');
```



The next page was the Manage Classes page. This uses the same layout as the Dashboard page, with header, horizontal scrolling section, and main section - however I removed the sidebar on this page. In the horizontal section I replaced the circles with upright bars representing the total of each class's student's progress filling up the bar. Below that, there is a grid containing the user's classes and a button to create classes. When the hovers their mouse over one of the class tiles, the tile grows and adelete button is displayed. When they click on the create class button, the background darkens and a pop up appears with a form asking for the new class's name.

Code to display bars:

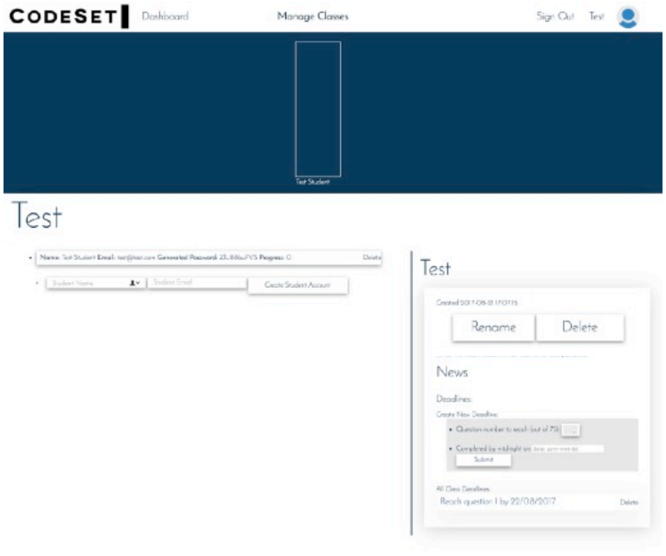
```
<div class="classes-progress-class">
      <div class="classes-progress-bar" style="height: calc(35vh - <?= 100-
$studentProgress ?>%);"></div>
      <span class="classes-progress-name"><?= $studentName ?></span>
</div>
```

To create the popups, I used a `div` element to create a dark overlay for the background, with another `div` element inside that to create the box. When the user clicked on the button, it triggers the JavaScript code to show the overlay and popup. If the user clicks the cross, clicks outside the popup, or presses escape another piece of JavaScript code will be triggered, hiding the popup elements.
HTML Code:

```
<div class="popup-overlay" id="create-class">
      <div class="popup" id="create-class-popup">
                  <span class="popup-close" id="create-class-close"
            style="cursor: pointer;">x</span>
                  <form method="post" action="">
                        <input type="text" name="create-class-name"
                  placeholder="Class Name" autofocus>
                        <br>
                  <input type="submit" name="create-class-submit">
            </form>
      </div>
</div>
<div id="create-class-button" class="dashboard-class-tile" style="background:
url('/public_html/assets/images/plus.png') no-repeat center #0c4e73;
background-size: 50%; cursor: pointer;">
      <div class="class-tile-content">
            <div class="class-tile-content-table-hover">
                  <h4 style="margin-top: 80%;">Create New Class</h4>
            </div>
      </div>
</div>
```

JavaScript Code:

```
<script type="text/javascript">
    // Show new class pop up
    $("#create-class-button").click(function() {
        $("#create-class").fadeIn();
    });
    // Close new class pop up
    $("#create-class-close").click(function() {
        $("#create-class").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#create-class").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#create-class-popup");
        if (!$('#create-class-button').is(e.target) && !popup.is(e.target) &&
popup.has(e.target).length == 0) {
            $("#create-class").fadeOut();
        }
    });
</script>
```



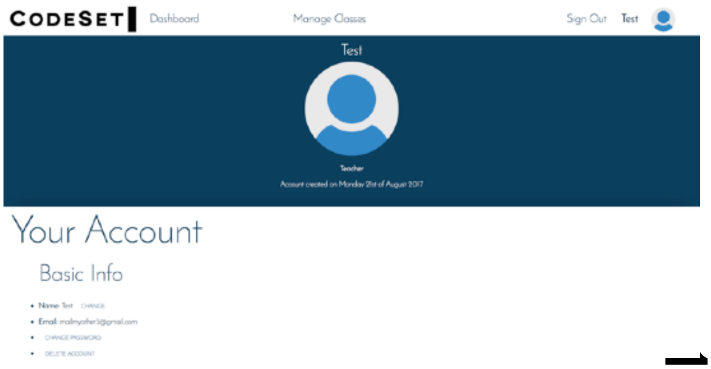
Name: Test Student Email: test@test.com Generated Password: 23L86buFVS Progress: 0

Created 2017-08-21 17:07:15

- Question number to reach (out of 75):
- Completed by midnight on: [illegible]

Reach question 1 by 22/08/2017

I then created the Class page, which displays information about a class. The page is also set out like the Dashboard page, with the horizontal section displaying individual student progress. In the main section, there is a list of the students in the class, showing their name, email, generated password, progress and a Delete button. At the bottom of the list there is a form to create new student accounts with the student's name and email - their password is automatically generated. At the right is the sidebar, giving more information and options. This shows when the class was created, Rename and Delete buttons, a button to email all students in the class their login details, a form to create new deadlines, and a list of the active and passed deadlines. When the user clicks on rename, a form slides down asking the user for the new name, and when they click on a deadline they are taken to a page where they can easily see which students have reached the deadline.



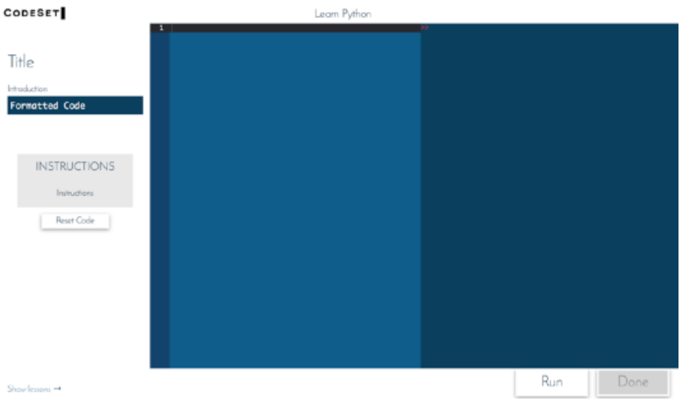
Teacher

Account created on Monday 21st of August 2017

- Name: Test
- Email: mailmyother5@gmail.com

The next page I created was the Account page. This page allows users to see an overview of the information stored about them and allows them to change it. The layout is again the same as the Dashboard, but without the sidebar and the horizontal section does not scroll. In the horizontal section, there is a big circle displaying the user's profile picture. If the user hovers their mouse over the image, a button displays allowing them to change the picture. If they click the button, the browser will allow them to choose a picture to upload from their computer. The file will then upload in the background and the page will refresh once it has finished. Below the profile picture the account's type is displayed and the date the account was created. In the main section, the user's name and email is displayed, along with the option to change their name. Underneath these, there are two other options; change password and delete account. If the user clicks on the buttons to change either their name or email a popup is displayed containing forms to allow them to do so. If they click the delete button, a popup will display asking the user to confirm they want to delete their account.

I then created the template for the lesson page. The data on this page will be populated from question data pulled from the database, however that system has not yet been created so I have used placeholders to show where it will go. The page has a header at the top, with the CodeSet logo as a link back to the Dashboard and the page's title. Below that the main section is split into three columns: sidebar explaining the lesson, code editor, and terminal. The sidebar is made up of a lesson title, introduction and instructions, and can display formatted example code using the html `<pre></pre>` and `<code></code>` tags. Below the instructions is a button to reset the code in the editor to the default code for the question. The code editor uses a code editor called *Ace*. I originally tried to make my own code editor, and I managed to create an editor with



```
var editor = ace.edit(“python-editor");
editor.setTheme("ace/theme/tomorrow_night");
editor.getSession().setMode("ace/mode/python");
editor.getSession().setUseWrapMode(true);
editor.focus();
var n = editor.getSession().getValue().split("\n").length;
editor.gotoLine(n);
```

line numbers, however I was not able to add features such as syntax highlighting and code completion. I wrote the following JavaScript code to setup the editor with the correct theme, programming language, focus the cursor on it, and go to the last line:
In the left section, the terminal, is where the output of the program being run will be displayed. It is also where programs with `input()` commands will get their input from the user. At the bottom of the page is the footer. On the left of the footer is a *Show Lessons* button, which, when clicked, triggers the sidebar containing the list of lessons to slide out, using the following code:

```
<a id="python-sidebar-arrow-out" onclick="openNav()">Show lessons &rarr;</a>

<script>
      function openNav() {
            document.getElementById("python-sidebar-arrow-out").style.zIndex
= "0";
            document.getElementById("python-sidebar-lessons").style.width =
"20%";
            document.getElementById("python-sidebar-lessons").style.padding =
      "1%";
      }
      // Set the width of the side navigation to 0 and the left margin of the
page content to 0
      function closeNav() {
            document.getElementById(“python-sidebar-arrow-out").style.zIndex
= "1";
```

```
            document.getElementById("python-sidebar-lessons").style.width =
"0";
            document.getElementById("python-sidebar-lessons").style.padding =
      "0";
      }
</script>
```

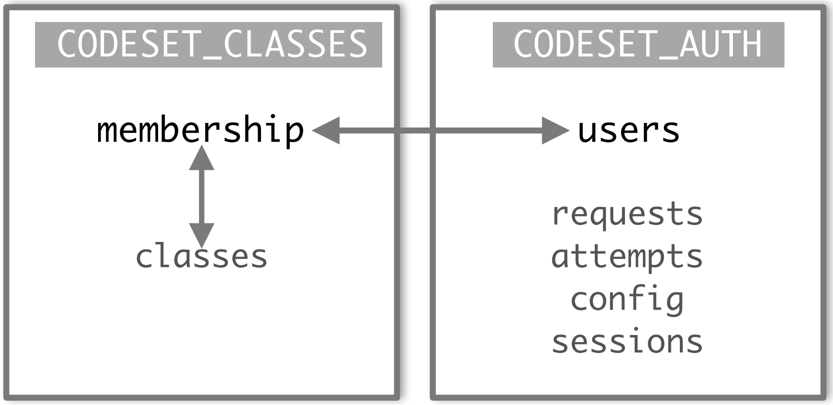
On the right of the footer is the Run and Done buttons. When the user clicks the Run button, the code in the editor is run and checked to see if it is correct. If it is, the Done button is made clickable, and when pressed saves the code and moves the user onto the next lesson.

Backend Systems Creation

The backend is made up of 5 main systems - Authentication, Classes, Course, Mail and News - which process data, manage the Python course, and interact with the databases. Because each system is used in multiple different places throughout the site, I decided to use Object Orientation, using a class for each system that can be constructed in config.php and then accessed from any page that includes the config.php file. This means that for most pages the imports can be managed from one file, so can be kept up to date easily.

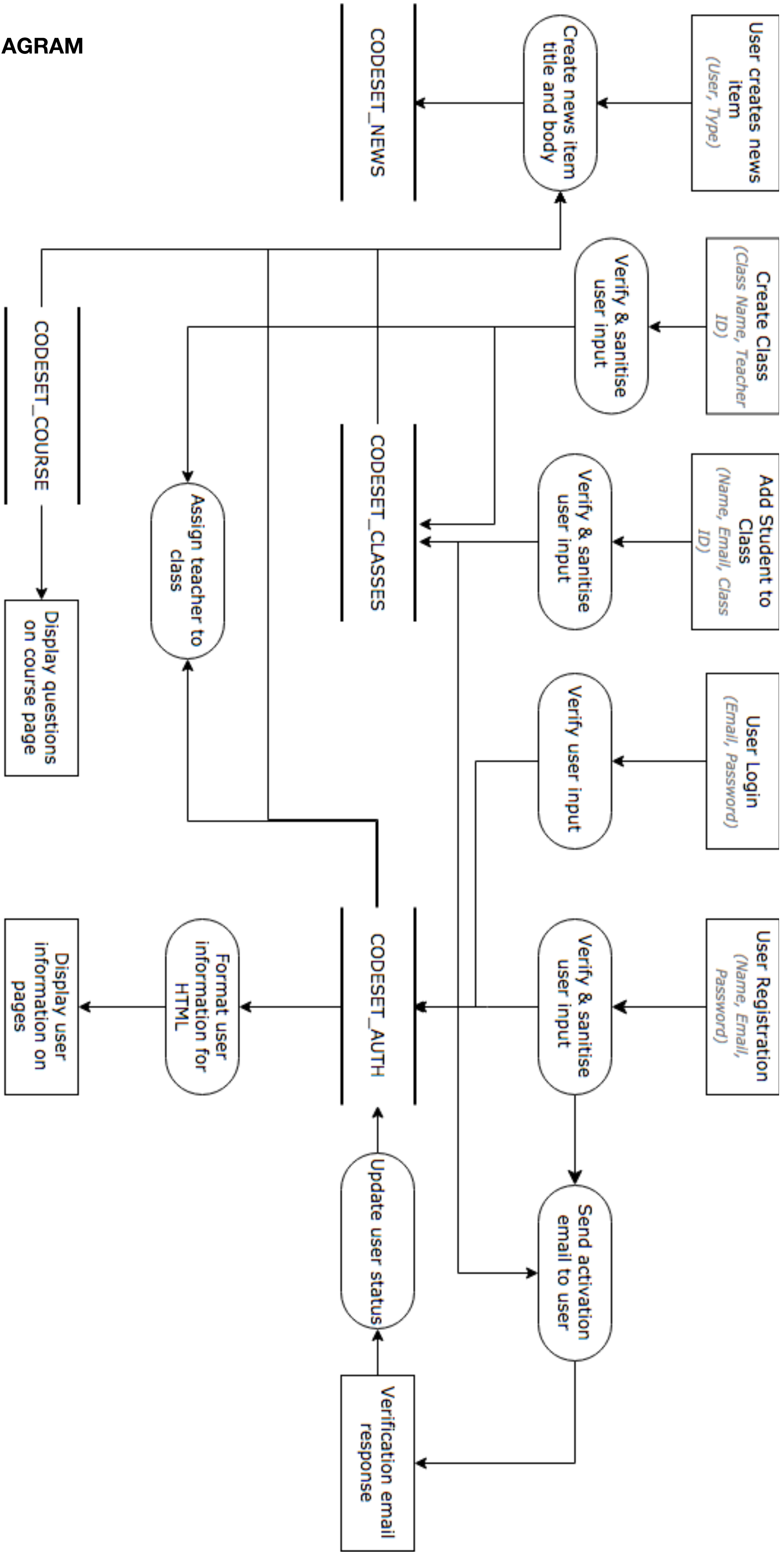
DATABASES
The first step of creating the backend systems was to setup the databases used to store information for the website. As the server the site is hosted on will only be used for this site, I can use a database for each system instead of using one database for the whole site.
The databases I created were:

- `CODESET_AUTH` - For the Authentication system - storing user details
- `CODESET_NEWS` - For the Deadlines and News systems - storing pieces of news
- `CODESET_CLASSES` - For the Classes system - storing class information
- `CODESET_COURSE` - For the Python Course - storing questions



The `CODESET_CLASSES` database is a relational database, containing two tables - `classes` and `membership`. The table `classes` contains rows of all the classes on the site, storing their id, name, date created and author's id. The table `membership` contains the relationship between the users and the classes, which each row storing the class id and user id, as well as the user's permissions in the class. The `membership` table is used as a way to communicate between the Classes and Authentication systems.

DATA FLOW DIAGRAM



AUTHENTICATION SYSTEM

The authentication system needs to be secure, so that user's passwords and details cannot easily be stolen, as that could result in any other accounts on other websites where they use the same password being compromised too. The system needs to use security practices such as sending authentication emails, not storing users' passwords in plain text, logging users out after a certain amount of time, be protected from SQL injection attacks and block suspected attackers. For the user, the system needs to be fast, they need to be able to change their password and other details, and be able to delete their account easily.

First, I attempted to create the system myself, by using the MD5 hashing algorithm to encrypt passwords in the database, however after running a number of tests I quickly realised the system was not secure enough, as the MD5 hash suffers from an extensive array of vulnerabilities and the passwords were sent in plain text so could easily be intercepted. Because of this and the requirements for the system above, I decided to use an open source system called PHPAuth (*https://github.com/PHPAuth/PHPAuth*), created by a number of GitHub users. The system is very secure, and is easy to modify - an important requirement for my site. The system includes the following features:

- Authentication by email and password combination
- Uses bcrypt to hash passwords, a secure algorithm that uses an extensive key setup phase
- Uses an individual 128 bit salt for each user, pulled from /dev/urandom, making rainbow tables useless
- Uses PHP's PDO database interface and uses prepared statements meaning an efficient system, resilient against SQL injection
- Blocks or verifies attackers by IP for any defined time after any amount of failed actions
- No plain text passwords are sent or stored by the system
- Blocks disposable email addresses from registration
- Allows users to login or register
- Accounts are activated by email
- Allows users to reset password and change email address
- Allows users to log out
- Allows users to delete their account

The system uses a *MySQL* database system to store the user's details, current user sessions, login attempts, and blocked users. It has extensive configuration settings, so it can be made to work with my site. The system is installed and kept up to date using the composer system, which is installed on the server. This means that if an important security update is released, the update will automatically be installed.

The system is very simple to set up, and is easy to use. To make a page only available to authenticated users, the following code is used:

```php
<?php

include("Config.php");
include("Auth.php");

$dbh = new PDO("mysql:host=localhost;dbname=phpauth", "username", "password");

$config = new PHPAuth\Config($dbh);
$auth   = new PHPAuth\Auth($dbh, $config);

if (!$auth->isLogged()) {
    header('HTTP/1.0 403 Forbidden');
    echo "Forbidden";

    exit();
}

?>
```

When creating other systems for the site, I modified the PHPAuth `Auth` class to work with them, adding my own methods and changing the ones already created. One method I added was the `registerStudent()` function, which used the `register()` function already in the class to create a student account with a generated password, generated using the `generatePassword()` function:

```
public function registerStudent($name, $email) {
        $password = $this->getRandomKey(10);

        $register = $this->register(htmlspecialchars($name),
htmlspecialchars($email), $password, $password);

        if ($register['error'] == true) {
            return $register;
        } else {
            $query = $this->dbh->prepare("UPDATE users SET permission = 0,
generated_password = ? WHERE email = ?");

            if (!$query->execute(array($password, $email))) {
                $this->deleteUser($getUID($email));
                return "Could not edit user in table. Please try again";
            }

            return true;
        }
    }

public function getRandomKey($length) {
        $alphabet =
             ‘abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
         $pass = array();
        $alphaLength = strlen($alphabet) - 1;

        for ($i = 0; $i < $length; $i++) {
            $n = rand(0, $alphaLength);
            $pass[] = $alphabet[$n];
        }

        return implode($pass);
    }
```

I also modified the `Auth` class to use my mail system rather than it's default system, which did not work on my server and did not meet my requirements.

In the MySQL `CODESET_AUTH` database, there are five tables:

- `attempts` - logs login attempts to prevent brute force attacks
- `config` - stores site information used by the Auth library
- `requests` - stores requests such as account activation and password reset
- `sessions` - stores sessions currently stored in cookies on user browsers
- `users` - stores user information

MAIL SYSTEM

A system to send emails to users was needed for the site's authentication system - to email users about account activation, password resets for example - and emailing teachers and students about their classes.

The class takes the server's email address' username, password and port when the class is first called. The class only has one function other than `__construct`, `Send()`. This function takes the email subject and body, who the email is from and who to send the email to. The formatting of the email body is done by the program calling the `Send()` function, rather than the function doing it itself. This allows each system sending emails to format them differently to suit the content of the email, rather than all emails sent being displayed the same way.

When I first made the mail system, I found that all emails sent from the server went straight into the receiver’s spam mailbox. I eventually discovered that if the email’s header did not say who the email was from in the format “Site Name <site email address>”, the email would be classified as spam. Also, as the class had `namespace CodeSet;` at the top, I could not call the built in PHP `Mail` class unless I used a forward slash before the class name, like this: `$smtp = \Mail`. Finally, to use the built in `Mail` class, I had to include the class in the function, using `require_once “Mail.php”;`.

CLASSES SYSTEM
The site has two types of users; teachers and students. Teachers create classes, then create student accounts that then belong to that class. To create a student account, the teacher must fill in a form asking for the student’s name and email address - the system then generates a password for the account. Once the account has been created, the teacher can then give the student their password, allowing them to log on and change the password. I created a button to allow teachers to email all the students in a class their detail, using a for loop to iterate through the users, and the CodeSet Mail system to send the email.

A teacher can create a class by clicking on the “*Create Class*” button and typing in a name. The system then adds a row to the classes table in the database - containing class ids, names, author and date created - and adds a row into the membership table - containing the details of which users belong to which classes, and their permissions in those classes.

One the Manage Classes page, there is a grid made up of tiles - each class will have a tile that is a link to the class page. A user’s classes are selected from the classes table using the `getClasses()` function, which uses the SQL command:

```
SELECT * FROM classes WHERE author_id = $userID
```

The function then returns an array containing the classes the user owns, which the page can then loop through using a `foreach` loop, creating the tiles:

```
    <? $classes = $codesetClasses->getClasses($uid);
    foreach ($classes as $class) { ?>
        <div class="dashboard-class-tile">
            <div class="class-tile-content">
                <a href="class.php?class=<?= $class['id'] ?>"><div class="class-
                tile-content-table">
                    <h4><?= $class['name'] ?></h4>
                </div></a>
                <div class="class-tile-content-table-hover">
                    <form style="width: 45%; margin: 0 auto; margin-top: -30%;"
                    method="post" action="">
                        <input type="hidden" name="delete-class-id" value="<?=
                           $class['id'] ?>">
                        <input type="submit" name="delete-class-submit"
                           value="Delete" style="height: 2em; font-size: 1.5em;">
                    </form>
                </div>
            </div>
        </div>
<? } ?>
```

The progress bars at the top of the Manage Classes page and the Classes Leaderboard on the teacher's dashboard use the same backend function - `getClassesLeaderboard($userID, $auth, $questions)`, which selects all of a user's classes from the database, then loops through each class's students selecting their progress from the authentication database and adding it to the class total. The percentage is calculated using following function, which takes an array containing student's scores , `$scores`, and the total number of questions multiplied by the number of students in the class, `$max`:

```
private function percentScore($scores, $max) {
    $scoreTotal = 0;
    foreach ($scores as $score) {
        $scoreTotal += $score;
    }
    return round(($scoreTotal/$max)*100);
}
```

The total percentages for the classes are then returned and used to set the height of the bars and order the leaderboard. For the student's dashboard and individual class pages, the class leaderboard is created using the `getClassLeaderboard($userID, $auth, $questions)` function, which is very similar to `getClassesLeaderboard`, however it does not loop through all the user's classes. When a teacher clicks on a class tile, they are taken to class.php, and the class id is passed to the page using the url and a `GET` request. I decided to use `GET` instead of `POST` because the page can then be reloaded without losing the class id. However, using `GET` does have security problems, as users can change the value passed in the url. To prevent this, each class has a unique 20 character long id so that users cannot guess other classes' ids, and the user is checked to make sure they have permission to access the class when they load the page. Once on the class page, the class information is pulled from the classes table in the database, as well as student names, emails, generated passwords and progress from the authentication database.

COURSE SYSTEM

The Python Course system comprises of the `Course` class, `CODESET_COURSE` database, and the files contained in the folder `public_html/pages/private/Course/` - index.php and run.php. index.php is the front end of the course system, displaying questions, receiving user code and displaying it's output. run.php handles communication between the frontend of the system and the backend - the Course class and other intermediate tasks, receiving the user's code passed from index.php and passing it to the Course class to be run and checked, saving the code, updating the user's progress, creating news based on the user's progress, and preparing the output from running to code ready for index.php to display it. I also made an Editor page, which is the same as the Course page without the questions. This allows users to experiment with their own code.

The user inputs code into the code editor, then clicks Run to run the code. When the button is clicked, a jQuery `.on(‘click’, function(e))` is triggered, running a function to get the code from the editor using `editor.getValue()` and the code is processed. The code is then passed to run.php using JavaScript's AJAX system through jQuery's methods which simplifies using AJAX with different browsers with different standards. AJAX is a web technology where where data is exchanged with the server and parts of the web page are updated without reloading the page. The code, user's ID, the question ID and the inputs are passed to run.php using POST. I decided to use AJAX to pass the data to run.php instead of just using POST because the page does not then need to be reloaded, creating a better user experience. If the AJAX POST request is successful, the output from the code is inserted into the terminal, and if the code was correct, the Done button is enabled.

```
$.ajax({
  url: 'run.php',
  type: 'post',
  dataType: 'json',
  data: {'code': editor.getValue(), 'userID': <?= $uid ?>, 'questionID': <?=
$_GET['id'] ?>, 'inputs': inputs},
  success: function(data) {
   $(‘#course-terminal').html('<p><span class="course-terminal-red">>></
      span>'+data[0]['message']+'</p>');
    if (data[1]['correct'] == true) {
        $("#course-bar-done").removeAttr("class");
        $("#course-bar-done").removeAttr("disabled");
    } else {
        $("#course-bar-done").attr("class", “disabled");
        $("#course-bar-done").attr("disabled", true);
    }
    $("body").css({"cursor": "default"});
  },
  error: function(data) {
    $('#course-terminal').html('<p><span class="course-terminal-red">>></
span>Error running code</p>');
    $("body").css({"cursor": "default"});
  }
});
```

Once run.php receives the data, the code is then run in Python. I went through lots of different versions of two different methods to test running code in Python before settling on the current version. The first method I used centred around PHP's `exec()`, `shell_exec()` and `system()` functions. The first two functions are very similar, and call a system command then return the output to be dealt with. The `system()` function executes a system command and then immediately displaying the output. These functions are run in the following way:

```
$command = escapeshellcmd('test.py');
$output = exec($command);
echo $output;
```

The system function runs python the following way, which allows arguments to be passed to Python, then the output is returned in the variable `$retval`:

```
$mystring = system('python test.py $arguments', $retval);
```

To protect the server, an important factor in developing the method for running Python was security. One requirement I had was that users cannot import built in Python libraries that could be used to damage the server, such as `os`, `system`, `shutil` and `subprocess` which give the user access to the filesystem, however I still needed the user to have access to other built in libraries, such as `random` and `math`. To do this, I created a Python file called python-guard.py, which was passed the user's Python code in a file when executed, using the following PHP command:

```
$execute = shell_exec("/kunden/homepages/8/d657042007/htdocs/python34/
Python-3.4.2/python python-guard.py 2>&1 $usercodefile");
```

Inside python-guard.php, the following code is used to secure the execution of the user's code. The four dangerous Python libraries are set to None in the custom namespace the user's code is run in to sandbox it, and there are various checks for exceptions, to enforce other security features - such as memory limits, and to handle errors gracefully so that the server details are shown to the user and so that the errors are easier for beginners to understand. The user's Python file containing their code is retrieved by the program as an input:

```
import sys
import os

def compileThenExec(file):
    ns = {}
    sys.modules['os'] = None
    sys.modules['sys'] = None
    sys.modules['shutil'] = None
    sys.modules['subprocess'] = None
    try:
        exec(open(file).read(), ns)
    except MemoryError:
        print("Error: Your program exceeds the memory limit")
    except EOFError:
        print("Error: Input commands disabled on this interpreter")
    except ImportError:
        print("Error: Python library blocked")
    except Exception as inst:
        print('Error:', str(inst.args[0]).capitalize())

file = input()
os.chdir(os.path.split(file+"/")
compileThenExec(file)
```

This method worked well at executing the code and allowed the blocking of dangerous libraries, however inputs could not be passed from the user to the user's Python program. To overcome this, I decided to use a different method, using subprocesses to run the user's code in a separate process, then opening a pipe between them to allow data to be passed between the two processes continuously. To do this, I used PHP's `proc_open()` function to open a new process, and used a file called error.txt in each folder containing the user's Python program as the pipe. This meant that while the process is open, the file is used as a stream for data, then if there is an error in running the process, the message is stored in error.txt for debugging. I attempted to get the user's Python program to send a request back to the page the user is on when it needed an input, however I could not work out how to get a Python input request to get back through the multiple files, functions and processes to the user, display a popup asking for input, then send the input back to the Python program. There was also no way to keep the AJAX request open to send multiple requests forwards and backwards between the programs, so I looked at creating a pipe between the programs to send data two ways, however there was no way to do this with my current setup - the only way would be to use Node.js, an asynchronous event driven JavaScript runtime, which does not work on my server. Because of this, the only way to get the user's inputs is to search through the user's code for Python `input()` functions and prompting the user for an input for each one. This array of user inputs is then passed through to the `execute()` function in the Course class, which then passes the inputs one at a time through the read pipe to the subprocess running the user’s Python program. When I first tried this, I could only pass one input at a time through - any more would not be passed to Python. I found out that this was because after each input had been written to the pipe file, there needed to be a new line character inserted to separate each input. Once the read pipe has finished being written to, the pipe must be closed before moving onto the write pipe, which returns outputs from the Python program. The following code is used to create the pipes, open a new process, pass inputs through the pipes, and get an output back. With this method, I could not use python-guard.py to run the user’s Python program, as the way to pass inputs to Python programs is to use an `input()` function to accept each input. Because the number of inputs the user has put in their code is unknown to python-guard.py and there is no way to dynamically write the correct number of `input()` functions in Python, I decided to run the user’s code directly from the Course class’ `execute()` function as a separate process so that inputs could be passed directly to the user’s Python program. However, with python-guard.py is no longer in use, the user would be able to import any built in Python library they would like, so I needed to block them again. I looked at using the same method as before, with a custom namespace, but custom namespaces can only be passed to a Python program using Python’s `exec()` function, which I could not use as I needed to pass inputs directly to the user’s code. To overcome this I took advantage of Python’s strict import syntax - all imports must be lowercase, must not contain any quotation marks or symbols for built in libraries, and must have one space between the word `import` and the library name - by searching through the user’s code before running it for the importation of dangerous libraries and stopping the code from being run if any are found. The final `execute()` function is below:

```
private function execute($file, $inputs) {
    chdir(dirname($file));
    $code = file_get_contents($file);

    if (strpos($code, 'import os') !== False) {
        $output = "Module 'os' is blocked";
        return $output;
    }
    if (strpos($code, 'import sys') !== False) {
        $output = "Module 'sys' is blocked";
        return $output;
    }
    if (strpos($code, 'import shutil') !== False) {
        $output = "Module 'shutil' is blocked";
        return $output;
    }
    if (strpos($code, 'import subprocess') !== False) {
        $output = "Module 'subprocess' is blocked";
        return $output;
    }

    $output = "";
    $descriptorspec = array(
        0 => array("pipe", "r"),
        1 => array("pipe", "w"),
        2 => array("file", "error.txt", "w"),
    );

    $process = proc_open("python3 $file", $descriptorspec, $pipes);

    if (is_resource($process)) {
        $count = substr_count(file_get_contents($file), “input(");
        for ($i=0; $i < $count; $i++) {
            fwrite($pipes[0], $inputs[$i]);
            fwrite($pipes[0], "\n");
        }
        fclose($pipes[0]);
        $output = stream_get_contents(
            $pipes[1]);
        fclose($pipes[1]);
        $return_value = proc_close($process);
        if ($return_value != 0){
            $output = file_get_contents('error.txt');
            $output = str_replace('File "'.$file.'", ', "", $output);
        }
    }
    return $output;
}
```

The user's Python code is stored in a file inside a cell for each user, then inside a folder for each question. Each of these folders is given access permissions (0700) so that the programs run inside these folders can only execute code - they cannot read, write or destroy anything.

FOLDER STRUCTURE

```
.
├── guard
│   └── CELL-{user-id}
│       └── Q-{question-id}
│           └── {user-id}-{question-id}.py
```

Once the code has been executed, the output is returned back to run.php, which then calls the Course class' `check()` function on the user's code and its output. Each question is stored in a row in the `CODESET_COURSE` database, and contains the question ID, title, introduction, instructions, default code, output, expected code, and percent. The `check()` function makes use of the output, expected code and percent for each question when checking whether the user's code is correct. The output is what the user's code should output if it is correct and it has four different possible types - `EMPTY`, `NULL`, `INT` and else. Else is used for most questions, and is usually the output of a `print()` function or expected error. `EMPTY` tells the check() function not to check the code, while `NULL` means the output should be nothing - if anything is output, the code is marked as incorrect. While `INT` isn't used often, some questions require the user to choose their own numbers to perform mathematical operations on, so the output should be a number. If the output is correct, the code moves onto the next check, however if it is not correct the user's code is marked as incorrect and the check is stopped. The expected code is a PHP array encoded into JSON format for storing in the database - I originally used PHP's `serialize()` function to convert the array into a string so that it could be stored in the database, however this function did not work when there was whitespace in the code. The arrays needed to be converted to strings as arrays cannot be stored directly in a database. The arrays contain the different ways to get the correct output for each question, so that the users have a greater chance of getting the questions correct. The percent column contains how similar the user's code needs to be to one of the variations of the correct code as a percentage. To calculate how similar the code is as a percentage, the user's code is first stripped of all comments and newlines. This means only the code being run is checked, which gives the user a better chance of getting a higher percentage. The expected code is then decoded and each variation is looped through and compared to the user's code using the PHP function similar_text(), which calculates the similarity between two strings, returning a percentage. The highest percentage out of all the variations is chosen, and if the percentage is equal to or greater than the value stored in the percent column for the question the question is marked as correct.

Once the `check()` function has finished, if the user's code is correct the user's progress in the Authentication database is updated and the progress is checked to see if the user is a quarter, half, three quarters or four quarters through the course - the user's previous question percentage, current question percentage and next question percentage are calculated. If the one of the percentages for each quarter is between the user's previous percentage and their next percentage, a news item is created so that other students in their class can see that they have reached a milestone. Then, the user's code's output is formatted using PHP's `htmlspecialchars()` to protect from cross site scripting attacks, and the newline characters are converted to HTML `<br>` tags so that the output is displayed correctly. Finally, the user's code and the output of the `check()` function are put into an array, which is then encoded into JSON format and echoed so that it can be retrieved by the AJAX method, in index.php. Once received by index.php, the code is displayed in the terminal and if the code was determined to be correct, the Done button is enabled.

NEWS SYSTEM

The News system manages the News and Deadlines items on the site. The News items are set when the user reaches certain points in the course and displayed on the Dashboard for other students in their class to see. Deadlines are set by teachers on a class' individual page and are show to the students on the Dashboard. Teachers can see more information on a Deadline by clicking on them, which takes them to a page showing their student's progress and allows them to delete the Deadline. The News items are selected from the database with a limit of 10, because they cannot be deleted so if there was a large number of them the list of them would be very long. All of a class' active deadlines are shown, and are ordered using SQL's `ORDER_BY` command on the expiration column, which has an SQL column data type of `datetime`. The Deadlines that have expired are greyed out on the individual classes page however the teacher can still click on them to see which students have completed it. On the Dashboard, only the active deadlines are shown to users. The expiration date needed to be formatted correctly for it to be inserted into a column with type `datetime`, otherwise it could not be sorted. To convert the text date inputed by the user to a `datetime` object for storing in the database, the date is first converted to a PHP DateTime object. When I first tried this, I put the format and date in the DateTime `__construct()` function, however that didn't work with the database, so I separated it out into the separate functions:

```php
$date = explode('-', $expiration);
$expiration = new \DateTime();
$expiration->setDate($date[0], $date[1], $date[2]);
$expiration->setTime(23, 59, 59);
```

PROFILE PICTURE UPLOAD SYSTEM

On the Account page, the user's profile picture is displayed at the top of the page - when the user hovers over the picture, a button is displayed. When the user clicks on the button, the browser's built in file manager is displayed allowing the user to choose a picture to upload. When the user chooses a picture, the picture will then upload automatically in the background. Once the picture has uploaded, the page will reload to display the changes. This is done using HTML's `onchange` attribute on the `input` tag, which calls `this.form.submit()` to force the form to submit. The image is passed to handler.php, which checks that the file uploaded is an image, deletes the user's old profile picture, moves the new picture to the correct directory, crops the image to a square so that it fits to the circular profile picture, then changes the profile in the database. The images are renamed to user-{id}, however because the user could upload images of any file format, the image name is stored in the user's row in the Authentication database. When I then tested this, I found that some images uploaded successfully but others were rotated. I found out that if the camera is rotated when taking the picture, the image is rotated incorrectly when uploaded. These rotated images contain an exif value called orientation, which is a number correlating to which way round the camera was rotated - 8 and 6 for sideways pictures, and 3 for upside down pictures. This allows me to rotate the image to the correct orientation using PHP's built in `imagerotate()` function. Finally, once this processing of the image has been completed, the page is refreshed using PHP's `header()` function.

Improvements

Based on feedback from users, I identified the following changes needed to be made:

1. Fix "Show Lessons" button in Google Chrome
2. Allow teachers to choose lessons if they haven't yet completed previous questions
3. Add link to bottom of the email containing student information
4. Add student's email address at bottom of ask teacher for help email
5. Change generated passwords to dictionary words
6. Add a larger space between sign out and user's name
7. Show question number on Course page
8. Fix some questions
9. One user was able to get the answers to the questions
10. Show numbers on progress bars
11. Students found a way to get around method to stop users skipping questions
12. Increase font size on Account page
13. Stop user from being able to resubmit forms on some pages
14. Redirect user to first uncompleted question if they change the question id in the address bar
15. Show error messages on Account page
16. Change "class" to "student" in error message on class page

I fixed the "Show Lessons" button by hiding the "Hide Lessons" button when the sidebar was not displayed, by setting its CSS display attribute to `none` to hide it and `inherit` to show it.

I allowed teachers to choose questions that they had not yet completed because they need to be able to view the tasks so that they knew what their students are doing.

I added a link to the website to the bottom of the email sent to students containing their details so that they can easily get to the site.

I also added the student's email to the bottom of the help teacher email, using a HTML `<a>` tag and `mailto:student email` as the href so that the teacher can click on the link and a compose email window will open.

I then changed the password generator function to take two random words from a dictionary stored in a txt file.

To add more space between the sign out button and name by giving the last element in the `nav` a negative CSS `margin-right` value.

To add the numbers to the Course page, I used the same elements and styling as the lesson circles on the Dashboard, and inverted the background and text colours.

When testing my site with users, I found that people found other ways to complete the question, but their solution was not marked as correct because their solutions was very different to those I came up with. The main issue was with users not putting spaces between variable names and equal signs, as all of my solutions contained spaces. To fix this, I stripped as much whitespace as possible from the user's code so that if they put a large number of new lines in it did not affect their percentage correct, and added solutions for each question with no spaces.

During testing with the Year 10, one user had all of the solutions already in the editor when they went onto a question. I believe this was because when I was testing the site I created some test users. Once I was finished with these users, I deleted them directly from the database and reset the table's ID auto increment, which meant their code was left behind in their folders. When the student's account was created it had the same ID as the deleted test user, so had access to the test user's code.

When getting feedback, my teacher suggested that showing the student's progress on the progress bars would be useful, as while the graphical representation is useful to get an overview of the whole class, it would be helpful to see in more detail the student's progress.

The original code on the Course page, shown below, allowed users to change the redirect attribute in the URL, then change the id attribute to go to any question. The original solution was much more complicated than it needed to be:

```
if ($_GET['id'] > $user['progress']+1 && !isset($_GET['redirect'])) {
    $question = $user['progress']+1;
    header("Location: ../Course/index.php?id=$question&redirect=True");
}
```

The new method is simpler, and does not allow users to change the question id to one they have not yet completed (this code also shows the solution to improvement number 2):

```
if ($_GET['id'] > $user['progress']+1 && $user['permission'] != 1) {
    $question = $user['progress']+1;
    header("Location: ../Course/index.php?id=$question");
}
```

The font size for the Change buttons on the Account page was very small, so on some monitors the text was very hard to read.

After a user submits a form the page is reloaded and the PHP uses POST to get the data and process it. Once the data has been processed, the rest of the page is then loaded. This means that if the user then refreshes the page the browser asks them to either cancel or resubmit the data, which could mean data in the database is duplicated or deleted. To stop the user from being able to resubmit forms, after the data had been processed I used PHP's `header()` function or JavaScript's `window.location.reload()` function if an alert had to be shown. However on some pages the output of the data processing needed to be displayed, so I had to place the data in the URL and fetch it using GET once the page had reloaded.

When the user goes onto the Course page, if they change the id to anything other than a number between 1 and 75, before the page would display an alert telling the user the page could not get the question, then display a blank Course page. This meant the user was still able to use functions such as emailing their teacher, even though there was no question which resulted in the teacher receiving a blank email. To stop this, instead of giving the user an alert telling them the question could not be fetched, the user is sent to the first uncompleted question.

The only other changes left were to do with improving the user experience, such as showing error messages on the Account page and making sure the error messages gave the user the right information.

TECHNICAL SOLUTION

File List and Description

File Name	Description
index.php	Main page where users visiting the site will first land
signin.php	Page for users to sign into their account
requestreset.php	Allows users to enter their email address to have an email sent to them allowing them reset their password
reset.php	The user can enter the key from the email sent to them when they requested a new password and their new password to change their account password
activate.php	Allows the user to enter the key sent to them in an email when they first created their account to activate their account
signout.php	When the user lands on this page, they are automatically logged out of the site by removing the session, then sends the user to index.php
private/index.php	Dashboard page the user is taken to once they have logged in. Shows the user's course progress and allows them to select a lesson, shows the news and deadlines from their class/classes, and shows the leaderboard for their class/classes
account.php	Displays the user's information and allows them to make changes to their account
classes/index.php	Shows the user's classes as tiles
class.php	Individual class page, displaying the class's information held in the database, shows the class' students and their progress, allows the user to create and access deadlines for the class, and allows the teacher to rename or delete the class
deadline.php	Shows individual deadlines and the progress the students in the class have made towards completing it
course/index.php	Displays the course lesson title, introduction and instructions, as well as a code editor for users to enter their lesson code and a terminal to display the output of their code
run.php	Runs code passed through AJAX and then checks code. Also checks if the user has reached a milestone. Output is then returned back to index.php by echoing it
complete.php	Page the user is taken to once they have completed the course
editor/index.php	Page containing a code editor to allow the user to enter their own code, and a terminal to display the output of their code
main.css	Contains most of the CSS styling rules for the site's HTML
config.php	Included in most pages, contains general site settings, database information, and variables storing the PHP objects used throughout the site
Auth.php	Contains the main code for the PHPAuth library - I have made changes to the code and added my own functions
class.mail.php	Contains the class used to send emails from the site
class.classes.php	Contains the class used to manage the classes containing students on the site
class.course.php	Contains the class for the Python Course system, handling questions and running user code
class.news.php	Contains the news class, which handles the site's news and deadline systems
handler.php	When the user is uploading a new profile picture, the image is sent to this file to be resized and stored in the correct directory

CODESET.CO.UK/INDEX.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-16 14:40:38
 * @Project: CodeSet
 * @File Name: index.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-30 19:43:01
*/
if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
include "config.php";
$codesetLogs->log('landed_home');
//Set registered to true so the form shows
$register = array('error' => TRUE, 'message' => "Create a teacher account");
//If sign up form is submitted, register account
if(isset($_POST['register-submit'])) {
    if(strlen($_POST['register-name'])) {
        $register = $auth->register($_POST['register-name'], $_POST['register-email'],
$_POST['register-password'], $_POST['register-repassword'], NULL, NULL, TRUE);
        if ($register['error'] == FALSE) {
            $codesetLogs->log('user_signup_teacher', $_POST['register-email']);
        } else {
            $codesetLogs->log('user_signup_teacher_error', $register['message']);
        }
    } else {
        $register = array('error' => True);
        $register['message'] = "Please enter a name";
    }
    header("Location: ../?".http_build_query(array('register' => $register)));
}
?>


<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet | The platform for teaching code in school</title>
    <link rel="stylesheet" type="text/css" href="public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
    <script type="text/javascript" src="public_html/assets/js/scroll.js"></script>
</head>
<body>
    <header>
        <a href="public_html/pages/public/signin.php" class="front-link">Sign In</a>
        <div class="expanding-line" data-ix="expanding-line-starts-hidden" style="width:
       100%; transition: width 350ms ease-in-out; -webkit-transition: width 350ms ease-in-
       out;"></div>
    </header>
    <main>
        <div class="front-main">
            <div class="front-logo">
                <img src="public_html/assets/images/front-logo/text.png" id="front-logo-
                text">
                <img src="public_html/assets/images/front-logo/block.png" id="front-logo-
                block" class="flicker">
            </div>
            <div class="front-about">
                <p>Welcome to CodeSet! CodeSet is a platform to help teachers develop their
                students’ Python skills, keeping track of their progress along the way</p>
            </div>
            <div class="front-arrow">
                <a href="#login"></a>
            </div>
        </div>
        <div class="front-login" id="login">
            <h1>Get Started With CodeSet</h1>
            <div class="front-login-left">
                <p>
                    Teaching code in classes is hard.<br><br>
                    CodeSet allows teachers to help their students learn to code, by teaching
                    Python in an interactive way, with instant code feedback, class
                    leaderboards and milestones. Teachers can create and manage student's
                    accounts, so that they can easily keep track of their progress and help
                    them when they get stuck.
                </p>
            </div>
            <div class="front-login-right">
                <?php
                if (isset($_GET['register'])) {
                    $register = $_GET['register'];
                }
                if($register['error'] == 0) {
                ?>
                    <h4 style="max-width: 40vw;">Account created. Check your email to verify
                    your account.</h4>


                <?php
                } else {
                ?>
                    <h4><?= $register['message'] ?></h4>
                    <form action="#login" method="POST">
                        <input type="text" name="register-name" placeholder="Enter your first
                          name"><br>
                        <input type="email" name="register-email" placeholder="Enter your
                          email"><br>
                        <input type="password" name="register-password" placeholder="Enter
                          your password"><br>
                        <input type="password" name="register-repassword" placeholder="Renter
                          your password"><br>
                        <input type="submit" name="register-submit" value="Register">
                    </form>
                <?php
                }
                ?>
            </div>
        </div>
    </main>
    <footer>
        <h5>&copy; CodeSet 2017</h5>
    </footer>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PUBLIC/SIGNIN.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-19 22:25:05
 * @Project: codeset.co.uk
 * @File Name: signin.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-03-19 13:34:05
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
```

```
include "../../../config.php";

$signin = array('error' => TRUE, 'message' => “”);

if(isset($_POST['signin-submit'])) {
    $signin = $auth->login($_POST['signin-email'], $_POST['signin-password']);
    if ($signin['error'] == FALSE) {
        setcookie($auth->config->cookie_name, $signin['hash'], $signin['expire'], $auth-
       >config->cookie_path, $auth->config->cookie_domain, $auth->config->cookie_secure,
       $auth->config->cookie_http);
        header("Location: ../private/index.php");
    }
}

?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Sign In</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body>
    <header class="signin-header">
        <h2 class="signin-header-title">Sign into your CodeSet account</h1>
        <a href="<?= $PUBLIC_DIR ?>">Or go back to register an account</a>
    </header>
    <main class="signin-main">
        <h4><?= $signin['message'] ?></h4>
        <form action="" method="POST">
            <input type="email" name="signin-email" placeholder="Enter your email"><br>
            <input type="password" name="signin-password" placeholder="Enter your
      password"><br>
            <input type="submit" name="signin-submit" value="Log In" style="width: 103.5%;">
        </form>
        <a href="requestreset.php"><h4>Forgot Password?</h4></a>
    </main>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PUBLIC/REQUESTRESET.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-20 10:04:55
 * @Project: codeset.co.uk
 * @File Name: requestreset.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-03-19 13:33:58
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include "../../../config.php";

$reset = array('error' => TRUE, 'message' => "");

if(isset($_POST['reset-submit'])) {
    $reset = $auth->requestReset($_POST['reset-email']);
}
```

```
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Reset Password</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body>
    <header class="signin-header">
        <h2 class="signin-header-title">Reset your CodeSet account password</h1>
        <a href="<?= $PUBLIC_DIR ?>">Or go back to log into your account</a>
    </header>
    <main class="signin-main">
        <h4><?= $reset['message'] ?></h4>
        <? if($reset['error'] == TRUE) { ?>
            <form action="" method="POST">
                <input type="email" name="reset-email" placeholder="Enter your email"><br>
                <input type="submit" name="reset-submit" value="Send Email" style="width:
          103.5%;">
            </form>
        <? } ?>
    </main>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PUBLIC/RESET.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-20 10:04:55
 * @Project: codeset.co.uk
 * @File Name: reset.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-03-19 13:34:01
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include "../../../config.php";

$reset = array('error' => TRUE, 'message' => "");

if(isset($_POST['reset-submit'])) {
    $reset = $auth->resetPass($_POST['reset-key'], $_POST['reset-password'], $_POST['reset-
repeatpassword']);
    if($reset['error'] == FALSE) {
        $signin = $auth->login($reset['user']['email'], $_POST['reset-password']);
        if ($signin['error'] == FALSE) {
            setcookie($auth->config->cookie_name, $signin['hash'], $signin['expire'], $auth-
          >config->cookie_path, $auth->config->cookie_domain, $auth->config-
>cookie_secure, $auth->config->cookie_http);
            header("Location: ../private/index.php");
        }
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Reset Password</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body>
    <header class="signin-header">
        <h2 class="signin-header-title">Reset your CodeSet account password</h1>
        <a href="<?= $PUBLIC_DIR ?>">Or go back to log into your account</a>
    </header>
    <main class="signin-main">
        <h4><?= $reset['message'] ?></h4>
        <form action="" method="POST" autocomplete="off">
            <input type="text" name="reset-key" placeholder="Enter your reset key"
      autocomplete="off"><br>
            <input type="password" name="reset-password" placeholder="Enter your new
      password"><br>
            <input type="password" name="reset-repeatpassword" placeholder="Enter your
      password again"><br>
            <input type="submit" name="reset-submit" value="Change Password" style="width:
      103.5%;">
        </form>
    </main>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PUBLIC/ACTIVATE.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-19 22:25:05
 * @Project: codeset.co.uk
 * @File Name: activate.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-03-19 13:33:54
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include "../../../config.php";

$activate = array('error' => TRUE, 'message' => "");

if(isset($_POST['activate-submit'])) {
    $activate = $auth->activate($_POST['activate-key']);
}

?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Activate Account</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body>
    <header class="signin-header">
        <h2 class="signin-header-title">Activate your CodeSet account</h1>
        <a href="<?= $PUBLIC_DIR ?>">Or go back to register an account</a>
    </header>
    <main class="signin-main">
        <h4><?= $activate['message'] ?></h4>
        <? if($activate['error'] == True) { ?>
```

```
            <form action="" method="POST">
                <input type="text" name="activate-key" placeholder="Enter your activation
          key"><br>
                <input type="submit" name="activate-submit" value="Activate" style="width:
          103.5%;">
            </form>
            <? } else { ?>
                <a href="signin.php">Log in to your account</a>
            <? } ?>
    </main>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/SIGNOUT.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-25 22:15:34
 * @Project: codeset.co.uk
 * @File Name: signout.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-03-19 13:34:17
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include '../../../config.php';

$auth->logout($_COOKIE[$auth->config->cookie_name]);

header("Location: /");
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/INDEX.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-19 23:11:21
 * @Project: codeset.co.uk
 * @File Name: index.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-21 17:02:42
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include '../../../config.php';

if (!$auth->isLogged()) {
    header("Location: ../public/signin.php");
}

$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];

$user = $auth->getUser($uid);

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - <?= $user['name'] ?></title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
    <script src="https://cdn.rawgit.com/asvd/dragscroll/master/dragscroll.js"></script>
    <script src="//cdn.jsdelivr.net/jquery.scrollto/2.1.2/jquery.scrollTo.min.js"></script>
</head>
<body>
    <header class="dashboard-header">
        <nav>
            <a href=""><img src="/public_html/assets/images/logo-black.png"></a>
            <ul>
                <a href="../private/"><li class="active">Dashboard</li></a>
                <a href=“Editor/“><li>Python Editor</li></a>
                <? if ($user['permission'] == 1){?><a href="classes/"><li>Manage Classes</
          li></a><? } ?>
                <a href="account.php"><img src="/public_html/assets/images/profiles/<?=
          $user['profile'] ?>" class="profile"></a>
                <a href="account.php" id="dashboard-header-name"><li><?= $user['name'] ?></
          li></a>
                <a href="signout.php" <? if ($user['permission'] == 0){?> style="margin-
          right: -2.5vw;" <?} ?>>Sign Out</a>
            </ul>
        </nav>
        <div <? if ($user['progress'] == 0) { ?> class="dashboard-progress" style="cursor:
             default;"<? } else { ?> class="dashboard-progress dragscroll" id="dashboard-
             progress"> <? } ?>
            <br>
            <h4>Python Course</h4>
            <? if ($user['progress'] == 0) { ?>
                <button class="dashboard-progress-start button"
                onclick="location.href='Course/?id=1';">Start</button>
            <? } else {
                for ($i = 1; $i <= $codesetCourse->getLastQuestionID(); $i++) {
                    if ($i > $user['progress']) { ?>
                        <a href="Course/?id=<?= $i ?>"><span class="dashboard-progress-
                 circle dashboard-progress-circle-uncomplete"><?= $i ?></span></a>
                        <? if ($i < $codesetCourse->getLastQuestionID()) { ?>
                            <span class="dashboard-progress-line"></span>
                        <? } ?>
                    <? } elseif ($i == $user['progress']-3) { ?>
                        <a href="Course/?id=<?= $i ?>"><span class="dashboard-progress-
                 circle" id="dashboard-progress-circle-current"><?= $i ?></span></a>
                        <? if ($i < $codesetCourse->getLastQuestionID()) { ?>
                            <span class="dashboard-progress-line"></span>
                        <? } ?>
                    <? } else { ?>
                        <a href="Course/?id=<?= $i ?>"><span class="dashboard-progress-
                 circle"><?= $i ?></span></a>
                        <? if ($i < $codesetCourse->getLastQuestionID()) { ?>
                            <span class="dashboard-progress-line"></span>
                        <? }
                    }
                }
            } ?>
        </div>
    </header>
    <main class="dashboard-main">
        <h1>Hello <?= $user['name'] ?></h1>
        <br>
        <div class="dashboard-news">
            <h3>News</h3>
            <br>
            <!-- Active Deadlines -->
            <?
                if ($user['permission'] != 1) {
                    // Students
                    $activeDeadlines = $codesetNews->getActiveDeadlines($codesetClasses-
                    >getUsersClass($uid));
                    foreach ($activeDeadlines as $activeDeadline) {
                        if ($user['progress'] >= $activeDeadline['point']) {
                            $expired = 'dashboard-news-card-expired';
                        } else {
                            $expired = "";
                        }
                        $date = new DateTime($activeDeadline['expiration']);
                        $dateSet = new DateTime($activeDeadline['date']);
                        ?>
                        <div class="dashboard-news-card dashboard-news-card-active <?=
                        $expired ?>">
                            <h4 class="dashboard-news-card-title">Deadline: Reach question <?
                             = $activeDeadline['point'] ?> by <?= $date->format('d/m/Y') ?></
                            h4>
                            <? if ($user['progress'] >= $activeDeadline['point']) { ?>
                                <p>You have reached this deadline</p>
                            <? } else { ?>
                                <p>You are on question <?= $user['progress']?> out of <?=
                                 $activeDeadline['point']?></p>
                            <? } ?>
                            <i style="font-size: 0.75em;"><?= $dateSet->format('d/m/Y') ?></i
                             >
                        </div>
                    <? }
                } else {
                    // Teachers
                    foreach ($codesetClasses->getClasses($uid) as $classID) {
                        $classID = $classID["id"];
                        $activeDeadlines = $codesetNews->getActiveDeadlines($classID);
                        foreach ($activeDeadlines as $activeDeadline) {
                            $date = new DateTime($activeDeadline['expiration']);
                            $dateSet = new DateTime($activeDeadline['date']);
                            ?>
                            <div class="dashboard-news-card dashboard-news-card-active">
                                <h4 class="dashboard-news-card-title"><a href="classes/
                                 deadline.php?id=<?= $activeDeadline['id'] ?>" style="color:
                                 #0a3f5e;">Deadline: Reach question <?=
                                 $activeDeadline['point'] ?> by <?= $date->format('d/m/Y') ?
                                 ></a></h4>
                                You set this on:
                                <i style="font-size: 0.75em;"><?= $dateSet->format('d/m/Y') ?
                                 ></i>
                            </div>
                        <? }
                    }
                }?>
            <hr>
            <?  $classIDs = $codesetClasses->getUsersClass($uid, True);
                foreach ($classIDs as $classID) {
                    $news = $codesetNews->getClassNews($classID);
                    foreach ($news as $new) {
                        $date = new DateTime($new['date']);
                    ?>
                        <div class="dashboard-news-card">
                            <h4 class="dashboard-news-card-title"><?= $new['title'] ?></h4>
                            <p><?= $new['body'] ?></p>
                            <i style="font-size: 0.75em;"><?= $date->format("d/m/Y") ?></i>
                        </div>
                <?  }
                } ?>
        </div>
        <div class="dashboard-leaderboard">
            <? if ($user['permission'] == 0) { ?>
                <h3>Class Leaderboard</h3>
                <? $leaderboard = $codesetClasses->getClassLeaderboard($codesetClasses-
                >getUsersClass($uid), $authDBH, $codesetCourse->getLastQuestionID());
                ?>
                <ol>
```

```
                    <? foreach ($leaderboard as $student) {
                    ?>
                        <li style="margin-top: 5%;"><img style="border-radius: 100%; width:
                        10%; margin-bottom: -3%; padding-right: 1%;" src="/public_html/
                        assets/images/profiles/<?= $student[2] ?>"><?= $student[1] ?> - <?=
                        $student[0] ?>% complete</li>
                    <? } ?>
                </ol>
            <? } elseif ($user['permission'] == 1) { ?>
                <h3>Classes Leaderboard</h3>
                <? $leaderboard = $codesetClasses->getClassesLeaderboard($uid, $authDBH,
                    $codesetCourse->getLastQuestionID()); ?>
                <ol>
                    <? foreach ($leaderboard as $class) {
                        ?> <li><?= $class[1] ?> - <?= $class[0] ?>% complete</li>
                    <? } ?>
                </ol>
            <? } ?>
        </div>
    </main>
    <footer>
        <h5>&copy; CodeSet 2016</h5>
    </footer>
</body>
<script type="text/javascript">
    (function($) {
        $.fn.goTo = function() {
            $('html, body').animate({
                scrollTop: $(this).offset().top + 'px'
            }, 'fast');
            return this;
        }
    })(jQuery);
    $('.dashboard-progress').scrollTo('#dashboard-progress-circle-current');
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/ACCOUNT.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-19 23:11:21
 * @Project: codeset.co.uk
 * @File Name: account.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-30 20:59:40
**/
if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
if (isset($_GET['redirect'])) {
    header('Location: account.php');
}
include '../../../config.php';
$codesetLogs->log('landed_private_account');
if (!$auth->isLogged()) {
    $codesetLogs->log('user_wrong_permission');
    header("Location: ../public/signin.php");
}
$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];
$user = $auth->getUser($uid);
if (isset($_POST['change-name-submit'])) {
    if (!empty($_POST['change-name-new'])) {
        if ($auth->changeName($uid, $_POST['change-name-new'])) {
            $codesetLogs->log('user_account_change_name');
            header('Location: account.php');
        } else {
            $codesetLogs->log('user_account_change_name_error');
            echo "<script>alert('Could not change name. Please try again');
             window.location.reload();</script>";
        }
    } else{
        echo "<script>alert('Please enter a name'); window.location.reload();</script>";
    }
}
$passwordMessage = "";
if (isset($_POST['change-password-submit'])) {
    if (!empty($_POST['change-password-old']) && !empty($_POST['change-password-new'] &&
       $_POST['change-password-newrepeat'])) {
        $password = $auth->changePassword($uid, $_POST['change-password-old'],
       $_POST['change-password-new'], $_POST['change-password-newrepeat']);
        if ($password['error'] == False) {
            $passwordMessage = "Password changed";
            $codesetLogs->log('user_password_reset');
            header('Location: account.php');
        } else {
            $codesetLogs->log('user_password_reset_error');
            echo "<script>alert('".$password['message']."'); window.location.reload();</
             script>";
        }
    } else {
        echo "<script>alert('Could not change password. Please try again');
         window.location.reload();</script>";
    }
}
if (isset($_POST['delete-account-submit'])) {
    if ($user['profile'] != "user-default.png" && file_exists('../../assets/images/profiles/'
       . $user['profile'])) {
        unlink('../../assets/images/profiles/' . $user['profile']);
    }
    if ($codesetClasses->deleteUser($codesetClasses->getUsersClass($uid), $uid)) {
        if ($auth->deleteUser($uid)) {
            $codesetLogs->log('user_delete_account', $user['id']);
            header('Location: ../../../');
        } else {
            $codesetLogs->log('user_delete_account_error');
            echo "<script>alert('Could not delete account. Please try again');
             window.location.reload();</script>";
        }
    }
    echo "<script>alert('Could not delete account. Please try again');
       window.location.reload();</script>";
}
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Account</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
    <script src="https://cdn.rawgit.com/asvd/dragscroll/master/dragscroll.js"></script>
</head>
<body>
    <header class="dashboard-header">
        <nav>
            <a href="../private/"><img src="/public_html/assets/images/logo-black.png"></a>
            <ul>
                <a href="../private/"><li>Dashboard</li></a>
                <a href="Editor/"><li>Python Editor</li></a>
                <? if ($user['permission'] == 1){?><a href="classes/"><li>Manage Classes</
                li></a><? } ?>
                <a href="account.php"><img src="/public_html/assets/images/profiles/<?=
                $user['profile'] ?>" class="profile"></a>
                <a href="account.php" id="dashboard-header-name" class="active"><li><?=
                $user['name'] ?></li></a>
                <a href="signout.php" <? if ($user['permission'] == 0){?> style="margin-
                right: -2.5vw;" <?} ?>>Sign Out</a>
            </ul>
        </nav>
        <div class="dashboard-progress" style="cursor: auto;">
            <br>
            <h4 style="position: absolute;"><?= $user['name'] ?></h4>
            <div class="account-profile-container">
                <img src="/public_html/assets/images/profiles/<?= $user['profile'] ?>"
                class="account-profile">
                <form action="/private_server/includes/CodeSetProfileUpload/handler.php"
                method="post" enctype="multipart/form-data" id="account-profile-upload">
                    <input type="file" id="account-profile-file" name="account-profile-file"
                    onchange="this.form.submit()">
                    <input type="button" value="Change" id="account-profile-change"
                    name="account-profile-change-submit">
                </form>
            </div>
            <br>
            <? if ($user['permission'] == 0) {
                echo "<b>Student</b><br><br>";
            } elseif ($user['permission'] == 1) {
                echo "<b>Teacher</b><br><br>";
            } ?>
            Account created on <?= date('l jS \of F Y', strtotime($user['dt'])) ?>
        </div>
    </header>
    <div class="popup-overlay" id="change-name-overlay">
        <div class="popup" id="change-name-popup">
                <span class="popup-close" id="change-name-close" style="cursor: pointer;">x</
                span>
                <form method="post" action="">
                    <input type="text" name="change-name-new" placeholder="Name" autofocus>
                    <br>
                    <input type="submit" name="change-name-submit">
                </form>
        </div>
    </div>
    <div class="popup-overlay" id="change-password-overlay">
        <div class="popup" id="change-password-popup">
                <span class="popup-close" id="change-password-close" style="cursor:
                pointer;">x</span>
                <form method="post" action="" autocomplete="off">
                    <input type="password" name="change-password-old" placeholder="Old
                    Password" autofocus>
                    <input type="password" name="change-password-new" placeholder="New
                    Password">
                    <input type="password" name="change-password-newrepeat"
                    placeholder="Repeat New Password">
                    <br>
                    <input type="submit" name="change-password-submit">
                </form>
        </div>
    </div>
    <div class="popup-overlay" id="delete-account-overlay">
        <div class="popup" id="delete-account-popup">
                <span class="popup-close" id="delete-account-close" style="cursor:
                pointer;">x</span>
                <form method="post" action="" autocomplete="off">
                    <h4 style="text-align: center;">Are you sure you want to delete your
                    account?</h4>
                    <br>
```

```
                    <input type="submit" name="delete-account-submit" value="Delete Account">
                </form>
        </div>
    </div>
    <main class="dashboard-main account-main">
        <h1>Your Account</h1>
        <br>
        <div class="account-info">
            <h3>Basic Info</h3>
            <br>
            <ul>
                <li><b>Name:</b> <?= $user['name'] ?> <span class="account-info-change"
                id="change-name">CHANGE</span></li>
                <li><b>Email:</b> <?= $user['email'] ?></li>
                <span style="color: red;"><?= $passwordMessage ?></span>
                <li><a id="change-password"><span class="account-info-change">CHANGE
                PASSWORD</span></li></a></li>
                <li><a id="delete-account"><span class="account-info-change">DELETE ACCOUNT</
                span></li></a></li>
            </ul>
            <br>
            <br>
            <!-- <h3>Mail Info</h3> -->
        </div>
    </main>
    <footer>
        <h5>&copy; CodeSet 2016</h5>
    </footer>
</body>
<script>
    var form = document.getElementById('account-profile-upload');
    var fileSelect = document.getElementById('account-profile-file');
    var uploadButton = document.getElementById('account-profile-change');
    $(form).bind('submit', function (event) {
        event.preventDefault();
    });
    $('#account-profile-change').click(function(){
        $('#account-profile-file').click();
    });
    // Show change name pop up
    $("#change-name").click(function() {
        $("#change-name-overlay").fadeIn();
    });
    // Close change name pop up
    $("#change-name-close").click(function() {
        $("#change-name-overlay").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#change-name-overlay").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#change-name-popup");
        if (!$('#change-name').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#change-name-overlay").fadeOut();
        }
    });
    // Show change password pop up
    $("#change-password").click(function() {
        $("#change-password-overlay").fadeIn();
    });
    // Close change password pop up
    $("#change-password-close").click(function() {
        $("#change-password-overlay").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#change-password-overlay").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#change-password-popup");
        if (!$('#change-password').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#change-password-overlay").fadeOut();
        }
    });
    // Show delete account pop up
    $("#delete-account").click(function() {
        $("#delete-account-overlay").fadeIn();
    });
    // Close delete account pop up
    $("#delete-account-close").click(function() {
        $("#delete-account-overlay").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#delete-account-overlay").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#delete-account-popup");
        if (!$('#delete-account').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#delete-account-overlay").fadeOut();
        }
    });
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/CLASSES/INDEX.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-26 13:37:06
 * @Project: codeset.co.uk
 * @File Name: index.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-25 19:05:31
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include '../../../../config.php';

if (!$auth->isLogged()) {
    header("Location: ../../public/signin.php");
}

$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];

$user = $auth->getUser($uid);

if ($user['permission'] < 1) {
    header('Location: ../../private/');
}

// Create new class
if (isset($_POST['create-class-submit'])) {
    if (strlen($_POST['create-class-name']) > 0) {
        if($codesetClasses->create($_POST['create-class-name'], $uid)){
            // Class created
        } else {
            echo "<script type='text/javascript'>alert('Could not create class. Please try
again');</script>";
        }
    } else {
        echo "<script type='text/javascript'>alert('Please enter a name for the class');</
script>";
    }
}

// Delete a class
if (isset($_POST['delete-class-submit'])) {
    $delete = $codesetClasses->delete($_POST['delete-class-id'], $uid);
    if ($delete) {
        // Class deleted
        foreach ($codesetNews->getClassDeadlines($_POST['delete-class-id']) as $deadline) {
            $codesetNews->deleteClassDeadline($deadline['id']);
        }
        foreach ($codesetNews->getClassNews($_POST['delete-class-id']) as $news) {
            $codesetNews->deleteClassNews($news['id']);
        }
    } else {
        echo "<script type='text/javascript'>alert('$delete');</script>";
    }
}

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Manage Classes</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>

<body>
    <header class="dashboard-header">
        <nav>
            <a href="../"><img src="/public_html/assets/images/logo-black.png"></a>
            <ul>
                <a href="../"><li>Dashboard</li></a>
                <a href="../Editor/"><li>Python Editor</li></a>
                <? if ($user['permission'] == 1){?><a href="../classes/"><li
          class="active">Manage Classes</li></a><? } ?>
                <a href="account.php"><img src="/public_html/assets/images/profiles/<?=
          $user['profile'] ?>" class="profile"></a>
                <a href="../account.php" id="dashboard-header-name"><li><?= $user['name'] ?
          ></li></a>
                <a href="../signout.php">Sign Out</a>
            </ul>
        </nav>
        <div class="dashboard-progress classes-progress" style="cursor: auto;">
            <?
                $classesProgress = $codesetClasses->getClassesLeaderboard($uid, $authDBH,
                $codesetCourse->getLastQuestionID());

                foreach ($classesProgress as $classProgress) { ?>
                    <div class="classes-progress-class">
                        <div class="classes-progress-bar" style="height: calc(35vh - <?= 100-
                 $classProgress[0] ?>%);"></div>
                        <span class="classes-progress-name"><?= $classProgress[1] ?></span>
                    </div>
            <?   } ?>
        </div>
    </header>
    <main class="dashboard-main">
```

```
        <h1>Your Classes</h1>
        <br>
        <div class="dashboard-classes">
            <div class="popup-overlay" id="create-class">
                <div class="popup" id="create-class-popup">
                    <span class="popup-close" id="create-class-close" style="cursor:
             pointer;">x</span>
                    <form method="post" action="">
                        <input type="text" name="create-class-name" placeholder="Class Name"
                 autofocus>
                        <br>
                        <input type="submit" name="create-class-submit">
                    </form>
                </div>
            </div>
            <div id="create-class-button" class="dashboard-class-tile" style="background:
             url('/public_html/assets/images/plus.png') no-repeat center #0c4e73;
             background-size: 50%; cursor: pointer;">
                <div class="class-tile-content">
                    <div class="class-tile-content-table-hover">
                        <h4 style="margin-top: 80%;">Create New Class</h4>
                    </div>
                </div>
            </div>
            <?
                $classes = $codesetClasses->getClasses($uid);
                foreach ($classes as $class) {
            ?>
                    <div class="dashboard-class-tile">
                        <div class="class-tile-content">
                            <a href="class.php?class=<?= $class['id'] ?>"><div class="class-
                      tile-content-table">
                                <h4><?= $class['name'] ?></h4>
                            </div></a>
                            <div class="class-tile-content-table-hover">
                                <form style="width: 45%; margin: 0 auto; margin-top: -30%;"
                          method="post" action="">
                                    <input type="hidden" name="delete-class-id" value="<?=
                              $class['id'] ?>">
                                    <input type="submit" name="delete-class-submit"
                              value="Delete" style="height: 2em; font-size: 1.5em;">
                                </form>
                            </div>
                        </div>
                    </div>
            <?
                }
            ?>
        </div>
    </main>
</body>
<script type="text/javascript">
    // Show new class pop up
    $("#create-class-button").click(function() {
        $("#create-class").fadeIn();
    });
    // Close new class pop up
    $("#create-class-close").click(function() {
        $("#create-class").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#create-class").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#create-class-popup");
        if (!$('#create-class-button').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#create-class").fadeOut();
        }
    });
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/CLASSES/CLASS.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-26 13:37:06
 * @Project: codeset.co.uk
 * @File Name: class.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-09-07 19:14:59
**/
if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
include '../../../../config.php';
$codesetLogs->log('landed_private_class', $_GET['class']);
if (!$auth->isLogged()) {
    $codesetLogs->log('user_wrong_permission');
    header("Location: ../../public/signin.php");
}
$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];
$user = $auth->getUser($uid);
$class = $codesetClasses->getClass($_GET['class']);
if ($class['author_id'] != $uid) {
    $codesetLogs->log('user_wrong_permission');
    header("Location: ../classes/");
}
// Delete a class
if (isset($_POST['delete-class-submit'])) {
    $delete = $codesetClasses->delete($_POST['delete-class-id'], $uid);
    if ($delete) {
        foreach ($codesetNews->getClassDeadlines($_POST['delete-class-id']) as $deadline) {
            $codesetNews->deleteClassDeadline($deadline['id']);
        }
        foreach ($codesetNews->getClassNews($_POST['delete-class-id']) as $news) {
            $codesetNews->deleteClassNews($news['id']);
        }
        $codesetLogs->log('class_deleted', $_POST['delete-class-id']);
        header("Location: ../classes/");
    } else {
        $codesetLogs->log('class_deleted_error');
        echo "<script type='text/javascript'>alert('$delete'); window.location.reload();</
        script>";
    }
}
// Rename a class
if (isset($_POST['rename-class-submit'])) {
    if (strlen($_POST['rename-class-newname']) > 0) {
        $rename = $codesetClasses->rename($_POST['rename-class-newname'], $_POST['rename-
         class-id'], $uid);
        if ($rename) {
            $codesetLogs->log('class_renamed', $_POST['rename-class-id']);
            header('Location: '.$_SERVER['REQUEST_URI']);
        } else {
            $codesetLogs->log('class_renamed_error');
            echo "<script type='text/javascript'>alert('$rename');
             window.location.reload();</script>";
        }
    } else {
```

```php
        echo "<script type='text/javascript'>alert('Please enter a name for the class');
         window.location.reload();</script>";
    }
}
// Add a user
if (isset($_POST['new-student-submit'])) {
    if (strlen($_POST['new-student-name']) > 0 && strlen($_POST['new-student-email']) > 0) {
        $newstudent = $auth->registerStudent($_POST['new-student-name'], $_POST['new-student-
         email']);
        if ($newstudent['error'] == false) {
            $studentUID = $auth->getUID($_POST['new-student-email']);
            if ($codesetClasses->addUser($class['id'], $studentUID, 0)) {
                header('Location: class.php?class='.$_GET['class']);
            } else {
                $password = $auth->getUser($studentUID);
                $auth->deleteUser($studentUID, $password);
                echo "<script type='text/javascript'>alert('Could not add user to class.
                 Please try again' window.location.reload(););</script>";
            }
        } else {
            $message = $newstudent['message'];
            echo "<script type='text/javascript'>alert('$message');
             window.location.reload();</script>";
        }
    } else {
        echo "<script type='text/javascript'>alert('Please enter a name and email for the
         student'); window.location.reload();</script>";
    }
}
// Delete a student
if (isset($_POST['delete-student-submit'])) {
    $delete = $auth->deleteStudent($_POST['delete-student-id']);
    if ($delete) {
        $codesetClasses->deleteUser($class['id'], $_POST['delete-student-id']);
        header('Location: class.php?class='.$_GET['class']);
    } else {
        echo "<script type='text/javascript'>alert('$delete'); window.location.reload();</
         script>";
    }
}
// Set a deadline
if (isset($_POST['new-deadline-submit'])) {
    if (!empty($_POST['new-deadline-question']) && !empty($_POST['new-deadline-date'])) {
        $deadline = $codesetNews->setClassDeadline($class['id'], $_POST['new-deadline-
         question'], $_POST['new-deadline-date'], $codesetCourse->getLastQuestionID());
        if ($deadline['error'] == false) {
            header('Location: class.php?class='.$_GET['class']);
        } else {
            $message = $deadline['message'];
            echo "<script type='text/javascript'>alert('$message');
             window.location.reload();</script>";
        }
    } else {
        echo "<script type='text/javascript'>alert('Question to reach and date cannot be
         empty'); window.location.reload();</script>";
    }
}
// Delete a deadline
if (isset($_POST['delete-deadline-submit'])) {
    if ($codesetNews->deleteClassDeadline($_POST['delete-deadline-id'])) {
        header('Location: class.php?class='.$_GET['class']);
    } else {
        echo "<script type='text/javascript'>alert('Could not delete deadline. Please try
         again'); window.location.reload();</script>";
    }
}
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - <?= $class['name'] ?></title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>


<body>
    <header class="dashboard-header">
        <nav>
            <a href="../"><img src="/public_html/assets/images/logo-black.png"></a>
            <ul>
                <a href="../"><li>Dashboard</li></a>
                <a href="../Editor/"><li>Python Editor</li></a>
                <? if ($user['permission'] == 1){?><a href="../classes/"><li
                class="active">Manage Classes</li></a><? } ?>
                <a href="account.php"><img src="/public_html/assets/images/profiles/<?=
                $user['profile'] ?>" class="profile"></a>
                <a href="../account.php" id="dashboard-header-name"><li><?= $user['name'] ?
                ></li></a>
                <a href="../signout.php">Sign Out</a>
            </ul>
        </nav>
        <div class="dashboard-progress classes-progress dragscroll" style="cursor: auto;">
            <?
                $studentsProgress = $codesetClasses->getClassLeaderboard($class['id'],
                $authDBH, $codesetCourse->getLastQuestionID());
                foreach ($studentsProgress as $studentProgress) {
                    ?>
                    <div class="classes-progress-class" style="word-wrap: break-word;">
                        <div class="classes-progress-bar" style="height: calc(35vh - <?= 100-
                        $studentProgress[0] ?>%);"></div>
                       <span class="classes-progress-name" style="width: 5vw; word-wrap:
                       break-word;"><?= $studentProgress[1] ?> (<?= $studentProgress[3] ?>)</
                       span>
                    </div>
                <?}
            ?>
        </div>
    </header>
    <div class="popup-overlay" id="email-students-sent-overlay">
        <div class="popup" id="email-students-sent-popup">
                <span class="popup-close" id="email-students-sent-close" style="cursor:
                pointer;">x</span>
                <h4 style="text-align: center;">An email containing their CodeSet username
                and password has been sent to all the students of this class</h4>
        </div>
    </div>
    <div class="popup-overlay" id="email-students-overlay">
        <div class="popup" id="email-students-popup">
                <span class="popup-close" id="email-students-close" style="cursor:
                    pointer;">x</span>
                <h4 style="text-align: center;">Click below to send an email to the students
                    in the class their username and password:</h4>
                <div style="width: 50%; margin: 0 auto;">
                    <input type="submit" name="email-students-send" id="email-students-send"
                    value="Send">
                </div>
        </div>
    </div>
    <main class="dashboard-main">
        <h1><?= $class['name'] ?></h1>
        <br>
        <div class="dashboard-classes">
```

```
<div class="dashboard-sidebar" style="border-left: 2.5px solid #0a3f5e; padding:
 1.25%;">
    <h3><?= $class['name'] ?></h3>
    <br>
    <div class="class-sidebar">
        Created <?= $class['date'] ?>
        <br><br>
        <div class="sidebar-class-edit">
            <form method="post" class="class-edit-button">
                <input type="hidden" name="delete-class-id" value="<?=
                 $class['id'] ?>">
                <input type="submit" name="delete-class-submit" value="Delete">
            </form>
            <input type="button" id="rename-class-button" value="Rename"
              class="class-edit-button">
            <br><br>
            <form method="post" id="rename-class-form">
                <input type="hidden" name="rename-class-id" value="<?=
                 $class['id'] ?>">
                <input type="text" name="rename-class-newname" placeholder="New
                 Name" style="color: #0a3f5e; width: 95%;">
                <input type="submit" name="rename-class-submit" value="Rename"
                 style="width: 108.5%;">
            </form>
        </div>
        <br>
        <br>
        <br>
        <a style="color: #0f5d8a; cursor: pointer;" id="email-students">Email the
        class's students their username and password</a>
        <br>
        <br>
        <div class="sidebar-class-news">
            <h4>News</h4>
            <br>
            <br>
            <h5>Deadlines:</h5>
            <br>
            <!-- Current Deadlines displayed as tiles -->
            Create New Deadline:
            <form method="post">
                <ul class="new-deadline-list">
                    <li>Question number to reach (out of <?= $codesetCourse-
                     >getLastQuestionID() ?>):<input type="number" name="new-
                     deadline-question" style="margin-left: 1%; width: 10%;
                     height: 100%; font-size: 1em; display: inline; color:
                     #0a3f5e;" min="1" max="<?= $codesetCourse-
                     >getLastQuestionID() ?>"></li><br>
                    <li>Completed by midnight on:<input type="date" name="new-
                     deadline-date" placeholder="Date: yyyy-mm-dd" style="width:
                     40%; display: inline; color: #0a3f5e;" min="<?= date('Y-m-
                     d') ?>"></li>
                    <input type="submit" name="new-deadline-submit">
                </ul>
            </form>
            <br>
            <br>
            All Class Deadlines:
            <br>
            <?
                $deadlines = $codesetNews->getClassDeadlines($class['id']);
                if ($deadlines != False) {
                    foreach ($deadlines as $deadline) {
                        $date = new DateTime($deadline['expiration']);
                        if ($date < new DateTime()) {
                            $expired = "deadline-card-expired";
                        } else {
                            $expired = "";
```

```
                                    }
                                    ?>
                                    <div class="deadline-card <?= $expired ?>">
                                        <a class="deadline-card-text" href="deadline.php?
                                        id=<?= $deadline['id'] ?>"><h5>Reach question <?=
                                        $deadline['point'] ?> by <?= $date->format("d/m/Y") ?
                                        ></h5></a>
                                        <form action="" method="post">
                                            <input type="hidden" name="delete-deadline-id"
                                              value="<?= $deadline['id'] ?>">
                                            <button id="delete-deadline-submit" name="delete-
                                              deadline-submit">Delete</button>
                                        </form>
                                    </div>
                            <?  }
                            } else {
                                echo "<span style='margin-left: 1em; font-size: 75%;'>Nothing
                                 here</span>";
                            } ?>
                    </div>
                </div>
            </div>
            <div>
                <ul class="class-students">
                <?
                    $students = $codesetClasses->getStudents($class['id']);
                    foreach ($students as $student) {
                ?>
                        <li style="list-style-type: disc;">
                            <span style="font-weight: 500;">Name:</span> <?= $auth-
                             >getUser($student['user_id'])['name'] ?> <span style="font-
                             weight: 500;">Email:</span> <?= $auth-
                             >getUser($student['user_id'])['email'] ?> <span style="font-
                             weight: 500;">Generated Password:</span> <?= $auth-
                             >getUser($student['user_id'])['generated_password'] ?>
                            <span style="font-weight: 500;">Progress:</span> <?= $auth-
                             >getUser($student['user_id'])['progress'] ?>
                            <span class="space"> </span>
                            <span class="space"> </span>
                            <form action="" method="post">
                                <input type="hidden" name="delete-student-id" value="<?=
                                 $student['user_id'] ?>">
                                <button id="delete-student-submit" name="delete-student-
                                 submit">Delete</button>
                            </form>
                        </li>
                <?
                    }
                ?>  <br>
                    <li id="new-student-li">
                        <form method="post" id="new-student-form">
                            <input type="text" name="new-student-name" placeholder="Student
                             Name" autofocus>
                            <input type="email" name="new-student-email" placeholder="Student
                             Email">
                            <input type="submit" name="new-student-submit" value="Create
                             Student Account">
                        </form>
                    </li>
                </ul>
            </div>
        </div>
    </main>
</body>
<script type="text/javascript">
    // Show and hide rename class form{}
    var clicked = false;
    $("#rename-class-button").click(function() {
        if(clicked) {
            $("#rename-class-button").css("background-color", "white");
            $("#rename-class-button").css("color", "#0a3f5e");
            $("#rename-class-form").slideUp("fast");
            clicked = false;
        } else {
            $("#rename-class-button").css("background-color", "#0a3f5e");
            $("#rename-class-button").css("color", "white");
            $("#rename-class-form").slideDown("fast");
            clicked = true;
        }
    });
    // Show email students pop up
    $("#email-students").click(function() {
        $("#email-students-overlay").fadeIn();
    });
    $("#email-students-send").click(function() {
        $("#email-students-overlay").fadeOut();
        var classID = "<?= $class['id'] ?>";
        $.ajax({
              url: 'email_students.php',
              type: 'post',
              data: {'classID': classID},
              success: function(data) {
                $("#email-students-sent-overlay").fadeIn();
              },
              error: function(data) {
                alert("Could not email students. Please try again.");
              }
            });
    });
    // Close email students pop up
    $("#email-students-close").click(function() {
        $("#email-students-overlay").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#email-students-overlay").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#email-students-popup");
        if (!$('#email-students').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#email-students-overlay").fadeOut();
        }
    });
    // Close email students sent pop up
    $("#email-students-sent-close").click(function() {
        $("#email-students-sent-overlay").fadeOut();
    });
    $(document).keyup(function(e) {
        if (e.keyCode == 27) {
            $("#email-students-sent-overlay").fadeOut();
        }
    });
    $(document).mouseup(function (e) {
        var popup = $("#email-students-sent-popup");
        if (!$('#email-students-sent').is(e.target) && !popup.is(e.target) &&
         popup.has(e.target).length == 0) {
            $("#email-students-sent-overlay").fadeOut();
        }
    });
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/CLASSES/DEADLINE.PHP

```
<?php
```

```
/**
 * @Author: Ben
 * @Date: 2016-11-26 13:37:06
 * @Project: codeset.co.uk
 * @File Name: deadline.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-21 14:24:59
**/
if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
include '../../../../config.php';
$codesetLogs->log('landed_private_class', $_GET['class']);
if (!$auth->isLogged()) {
    $codesetLogs->log('user_wrong_permission');
    header("Location: ../../public/signin.php");
}
$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];
$user = $auth->getUser($uid);
$class = $codesetClasses->getClass($codesetNews->getClassIdFromDeadlineId($_GET['id']));
$deadline = $codesetNews->getDeadlineInfo($_GET['id']);
if ($class['author_id'] != $uid) {
    $codesetLogs->log('user_wrong_permission');
    header("Location: ../classes/");
}
// Delete a deadline
if (isset($_POST['delete-deadline-submit'])) {
    if ($codesetNews->deleteClassDeadline($_POST['delete-deadline-id'])) {
        header('Location: class.php?class='.$class["id"]);
    } else {
        echo "<script type='text/javascript'>alert('Could not delete deadline. Please try
         again'); window.location.reload();</script>";
    }
}
?>


<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - <?= $class['name'] ?> Deadline</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>


<body>
    <header class="dashboard-header">
        <nav>
            <a href="../"><img src="/public_html/assets/images/logo-black.png"></a>
            <ul>
                <a href="../"><li>Dashboard</li></a>
                <a href="../Editor/"><li>Python Editor</li></a>
                <? if ($user['permission'] == 1){?><a href="../classes/"><li
                class="active">Manage Classes</li></a><? } ?>
                <a href="account.php"><img src="/public_html/assets/images/profiles/<?=
                $user['profile'] ?>" class="profile"></a>
                <a href="../account.php" id="dashboard-header-name"><li><?= $user['name'] ?
                ></li></a>
                <a href="../signout.php">Sign Out</a>
            </ul>
        </nav>
        <div class="dashboard-progress classes-progress" style="cursor: auto;">
            <?
                $studentsProgress = $codesetClasses->getClassLeaderboard($class['id'],
                $authDBH, $deadline['point']);
                foreach ($studentsProgress as $studentProgress) {
                    if ($studentProgress[0] > 100) {
                        $studentProgress[0] = 100;
                    }
                ?>
                    <div class="classes-progress-class">
                        <div class="classes-progress-bar" style="height: calc(35vh - <?= 100-
                        $studentProgress[0] ?>%);"></div>
                        <span class="classes-progress-name"><?= $studentProgress[1] ?></span>
                    </div>
                <?}
            ?>
        </div>
    </header>
    <main class="dashboard-main">
        <h1><a href="class.php?class=<?= $class['id'] ?>" style="color: #0f5d8a">&larr;<?=
         $class['name'] ?></a></h1>
        <br>
        <div class="dashboard-classes">
            <div class="dashboard-sidebar" style="border-left: 2.5px solid #0a3f5e; padding:
             1.25%;">   <? $date = new DateTime($deadline['expiration']); ?>
                <h3>Deadline: Reach question <?= $deadline['point'] ?> by <?= $date-
                >format("d/m/Y") ?></h3>
                <br>
                <br>
                <?  $countComplete = 0;
                    $students = $codesetClasses->getStudents($class['id']);
                    foreach ($students as $student) {
                        if ($auth->getUser($student['user_id'])['progress'] >=
                           $deadline['point']) {
                            $countComplete++;
                        }
                    }
                ?>
                <? if ($countComplete == 1) { ?>
                    <?= $countComplete ?> student out of <?= sizeof($students) ?> have
                    reached the deadline.
                <? } else { ?>
                    <?= $countComplete ?> students out of <?= sizeof($students) ?> have
                    reached the deadline.
                <? } ?>
                <br>
                <br>
                <form method="post" class="class-edit-button" style="width: 90%;">
                    <input type="hidden" name="delete-deadline-id" value="<?= $deadline['id']
                    ?>">
                    <input type="submit" name="delete-deadline-submit" value="Delete">
                </form>
            </div>
            <div>
                <ul class="class-students">
                <?
                    foreach ($students as $student) {
                ?>
                        <li style="list-style-type: disc;">
                            <span style="font-weight: 500;">Name:</span> <?= $auth-
                            >getUser($student['user_id'])['name'] ?>
                            <span style="margin-left: 1em; font-weight: 500;">Deadline
                             Progress:</span> <?= $auth->getUser($student['user_id'])
                             ['progress'] ?> / <?= $deadline['point'] ?>
                            <? if ($auth->getUser($student['user_id'])['progress'] >=
                             $deadline['point']) { ?>
                                <span style="font-size: 0.8em; float: right; margin-top:
                                 0.4%;">COMPLETE</span>
                            <? } ?>
                       </li>
                <?
                    }
                ?>
                </ul>
            </div>
        </div>
    </main>
</body>
<script type="text/javascript">
    // Show and hide rename class form{}
    var clicked = false;
    $("#rename-class-button").click(function() {
        if(clicked) {
            $("#rename-class-button").css("background-color", "white");
            $("#rename-class-button").css("color", "#0a3f5e");
            $("#rename-class-form").slideUp("fast");
            clicked = false;
        } else {
            $("#rename-class-button").css("background-color", "#0a3f5e");
            $("#rename-class-button").css("color", "white");
            $("#rename-class-form").slideDown("fast");
            clicked = true;
        }
    });
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/COURSE/INDEX.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2017-01-24 16:50:19
 * @Project: codeset.co.uk
 * @File Name: index.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-07-17 10:49:33
**/
if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}
include '../../../../config.php';
if (!$auth->isLogged()) {
    header("Location: ../../public/signin.php");
}
if ($_GET['id'] == -1) {
    header('Location: complete.php');
}
$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];
$user = $auth->getUser($uid);
if ($_GET['id'] > $user['progress']+1 && $user['permission'] != 1) {
    $question = $user['progress']+1;
    header("Location: ../Course/index.php?id=$question");
}
$class = $codesetClasses->getClass($codesetClasses->getUsersClass($uid));

$question = $codesetCourse->getQuestion(htmlspecialchars($_GET['id']));
$allQuestions = $codesetCourse->getAllQuestions();
$savedCode = $codesetCourse->getSavedCode($uid, htmlspecialchars($_GET['id']));
if (!$question) {
    header("Location: ../Course/index.php?id=".$user['progress']);
}
if (isset($_POST['ask-teacher-submit'])) {
    $teacherEmail = $auth->getUser($class['author_id'])['email'];
    $email = $codesetClasses->composeStudentToTeacherEmail($teacherEmail, $user['name'],
      $user['email'], $class['name'], $question['id'], $_POST['ask-teacher-code'],
      $_POST['ask-teacher-instructions'], $_POST['ask-teacher-comments']);
    $codesetMail->Send("CodeSet <server@codeset.co.uk>", $email['email'], $email['subject'],
      $email['body']);
}
```

```
?>


<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>CodeSet - Learn Python</title>
        <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></
         script>
    </head>
    <body style="max-height: 100vh; max-width: 100%; overflow: hidden;">
        <header class="course-header">
            <nav>
                <a href="../" alt="Back"><img src="/public_html/assets/images/logo-
                black.png"></a>
                <ul>
                    <li>Learn Python</li>
                    <li style="margin-top: -4vh; margin-right: -70vw;"><span class="course-
             progress-circle <? if (intval($_GET['id']) >= intval($user['progress']))
             {echo "course-progress-circle-uncomplete";} ?>"><?= $_GET['id'] ?></
             span></li>
                </ul>
            </nav>
        </header>
        <div class="popup-overlay" id="ask-teacher-overlay">
            <div class="popup" id="ask-teacher-popup">
                    <span class="popup-close" id="ask-teacher-close" style="cursor:
                    pointer;">x</span>
                    <form method="post" action="">
                        <p style="color: white; font-size: 1.5em;">Your code will be emailed
                           to your teacher. You can add any additional information in the box
                           below.</p>
                        <textarea placeholder="Enter comments" cols="10" rows="4" name="ask-
                           teacher-comments"></textarea>
                        <input type="hidden" name="ask-teacher-instructions" value="<?=
                           htmlspecialchars($question['instructions']) ?>">
                        <input type="hidden" id="ask-teacher-code" name="ask-teacher-code"
                           value="">
                        <br>
                        <input type="submit" name="ask-teacher-submit">
                    </form>
            </div>
        </div>
        <main>
            <div class="course-sidebar">
                <div class="course-sidebar-lessons" id="course-sidebar-lessons">
                    <div>
                        <?
                            $i = 1;
                            foreach ($allQuestions as $x) {
                                if ($_GET['id'] == $i) { ?>
                                    <a href="?id=<?= $x['id'] ?>" class="course-sidebar-
                                    lessons-active"><?= $i ?>. <?= $x['title'] ?></a>
                                <?} else { ?>
                                    <a href="?id=<?= $x['id'] ?>"><?= $i ?>. <?=
                                    $x['title'] ?></a>
                            <?    }
                                $i++;
                            } ?>
                    </div>
                    <a id="course-sidebar-arrow-in" onclick="closeNav()">Hide lessons
             &larr;</a>
                </div>
                <div class="course-sidebar-normal">
                    <br><br>
                    <h4><?= $question['title'] ?></h4>
                    <br>
                    <div class="course-sidebar-intro"><?= $question['intro'] ?></div>
                    <div class="course-sidebar-instructions">
                        <h5>INSTRUCTIONS</h5>
                        <br>
                        <?= $question['instructions'] ?>
                        <? if ($user['permission'] != 1) { ?>
                            <hr>
                            <a id="ask-teacher">Stuck? Ask your teacher</a>
                        <? } ?>
                    </div>
                    <input type="button" id="course-reset-code" value="Reset Code">
                    <a id="course-sidebar-arrow-out" onclick="openNav()">Show lessons
             &rarr;</a>
                </div>
            </div>
            <div class="course-editor" id="course-editor"><? if ($savedCode !== False) {
                echo $savedCode;
            } else {
                echo $question['default'];
            } ?></div>
            <div class="course-terminal" id="course-terminal"><span class="course-
       terminal-red">>>></span></div>
            <div class="course-bar">
                <form class="course-bar-done" action="" method="get">
                    <? if ($_GET['id'] == $codesetCourse->getLastQuestionID()) { ?>
                        <input type="hidden" name="id" value="-1">
                    <? } else { ?>
                        <input type="hidden" name="id" value="<?= $_GET['id']+1 ?>">
                    <? } ?>
                    <input type="submit" name="" value="Done" id="course-bar-done"
                    class="disabled" disabled>
                </form>
                <div class="course-bar-run">
                    <input type="button" id="course-bar-run" value="Run">
                </div>
            </div>
        </main>
    </body>
    <script>
        function openNav() {
            document.getElementById("course-sidebar-arrow-out").style.zIndex = "0";
            document.getElementById("course-sidebar-arrow-in").style.display = "inherit";
            document.getElementById("course-sidebar-lessons").style.width = "20%";
            document.getElementById("course-sidebar-lessons").style.padding = "1%";
        }
        /* Set the width of the side navigation to 0 and the left margin of the page content
        to 0 */
        function closeNav() {
            document.getElementById("course-sidebar-arrow-out").style.zIndex = "1";
            document.getElementById("course-sidebar-arrow-in").style.display = "none";
            document.getElementById("course-sidebar-lessons").style.width = "0";
            document.getElementById("course-sidebar-lessons").style.padding = "0";
        }
    </script>
    <script src="/public_html/libs/ace/ace.js" type="text/javascript" charset="utf-8"></
       script>
    <script>
        var editor = ace.edit("course-editor");
        editor.setTheme("ace/theme/tomorrow_night");
        editor.getSession().setMode("ace/mode/python");
        editor.getSession().setUseWrapMode(true);
        document.getElementById('course-editor').style.fontSize='0.9em';
        editor.focus(); //To focus the ace editor
        var n = editor.getSession().getValue().split("\n").length; // To count total no. of
         lines
        editor.gotoLine(n); //Go to end of document
        $(function(){
          $('#course-bar-run').on('click', function(e){
            $('#course-terminal').html('<p><span class="course-terminal-
       red">>>Running...</span></p>');
            var inputs = []
            var lines = editor.getValue().match(/[^\r\n]+/g);
            if (lines.length > 0) {
                for (var i = 0; i <= lines.length - 1; i++) {
                    if (lines[i].length > 0) {
                        if (lines[i].includes('input(')) {
                            if (lines[i].includes('#') == false || lines[i].indexOf('#') >
                             lines[i].indexOf('input(')) {
                                for (var x = 0; x < lines[i].split("input(").length - 1; x++)
                                 {
                                    inputs.push(prompt("Enter the values for each of the
                                    inputs in your program:", " "));
                                }
                            }
                        }
                    }
                }
            }
            $("body").css({"cursor": "progress"});
            e.preventDefault();
            $.ajax({
              url: 'run.php',
              type: 'post',
              dataType: 'json',
              data: {'code': editor.getValue(), 'userID': <?= $uid ?>, 'questionID': <?=
               $_GET['id'] ?>, 'inputs': inputs},
              success: function(data) {
                console.log(data);
                $('#course-terminal').html('<p><span class="course-terminal-red">>></
          span>'+data[0]['message']+'</p>');
                if (data[1]['correct'] == true) {
                    $("#course-bar-done").removeAttr("class");
                    $("#course-bar-done").removeAttr("disabled");
                } else {
                    $("#course-bar-done").attr("class", "disabled");
                    $("#course-bar-done").attr("disabled", true);
                }
                $("body").css({"cursor": "default"});
              },
              error: function(data) {
                $('#course-terminal').html('<p><span class="course-terminal-red">>></
          span>Error running code</p>');
                $("body").css({"cursor": "default"});
              }
            });
          });
        });
    </script>
    <script>
        $(function(){
          $('#course-reset-code').on('click', function(e){
            e.preventDefault();
            $.ajax({
              url: 'run.php',
              type: 'post',
              data: {'userID': <?= $uid ?>, 'questionID': <?= $_GET['id'] ?>},
              success: function(data) {
                console.log(data);
                if (data) {
                    location.reload();
                }
              },
              error: function(data) {
                alert("Could not reset code. Please try again.");
              }
```

```
            });
          });
        });
        document.getElementById('ask-teacher-code').value = editor.getValue();
        // Show ask teacher pop up
        $("#ask-teacher").click(function() {
            document.getElementById('ask-teacher-code').value = editor.getValue();
            $("#ask-teacher-overlay").fadeIn();
        });
        // Close ask teacher pop up
        $("#ask-teacher-close").click(function() {
            $("#ask-teacher-overlay").fadeOut();
        });
        $(document).keyup(function(e) {
            if (e.keyCode == 27) {
                $("#ask-teacher-overlay").fadeOut();
            }
        });
        $(document).mouseup(function (e) {
            var popup = $("#ask-teacher-popup");
            if (!$('#ask-teacher').is(e.target) && !popup.is(e.target) &&
             popup.has(e.target).length == 0) {
                $("#ask-teacher-overlay").fadeOut();
            }
        });
    </script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/COURSE/RUN.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2017-02-01 18:28:41
 * @Project: codeset.co.uk
 * @File Name: run.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-09-07 18:04:25
**/
ini_set('memory_limit','500M');
include $_SERVER['DOCUMENT_ROOT'].'/config.php';
$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];
$user = $auth->getUser($uid);
if (isset($_POST['code']) && isset($_POST['userID']) && isset($_POST['questionID'])) {
    if (isset($_POST['inputs'])) {
        $output = $codesetCourse->run($_POST['code'], $_POST['userID'],
         $_POST['questionID'], $_SERVER['DOCUMENT_ROOT']."/private_server/guard/",
         $_POST[‘inputs']);
    } else {
        $output = $codesetCourse->run($_POST['code'], $_POST['userID'],
        $_POST['questionID'], $_SERVER['DOCUMENT_ROOT']."/private_server/guard/");
    }
    $check = $codesetCourse->check($_POST['questionID'], $output['message'],
       $_POST['code']);

    if ($check['correct']) {
        if ($auth->getProgress($uid) < $_POST['questionID']) {
            $update = $auth->setProgress($uid, $_POST['questionID']);
        }
        $percentage = round(($_POST['questionID'] / $codesetCourse->getLastQuestionID()) *
         100);
        $previousPercentage = round((($_POST['questionID']-1) / $codesetCourse-
         >getLastQuestionID()) * 100);
        $nextPercentage = round((($_POST['questionID']+1) / $codesetCourse-
         >getLastQuestionID()) * 100);
        $output['user_code'] = $check['CODE1'];
        $output['question_code'] = $check['CODE2'];
        if ($previousPercentage < 25 && $nextPercentage > 25) {
            $codesetNews->addClassNews($codesetClasses->getUsersClass($uid), $user['name']."
             has reached a milestone in the course", $user['name']." is 25% of the way
             through the course.");
        } elseif ($previousPercentage < 50 && $nextPercentage > 50) {
            $codesetNews->addClassNews($codesetClasses->getUsersClass($uid), $user['name']."
             has reached a milestone in the course", $user['name']." is 50% of the way
             through the course.");
        } elseif ($previousPercentage < 75 && $nextPercentage > 75) {
            $codesetNews->addClassNews($codesetClasses->getUsersClass($uid), $user['name']."
             has reached a milestone in the course", $user['name']." is 75% of the way
             through the course.");
        } elseif ($percentage == 100) {
            $codesetNews->addClassNews($codesetClasses->getUsersClass($uid), $user['name']."
             has reached a milestone in the course", $user['name']." has completed the
             course!");
        }
    }
    $output['message'] = nl2br(htmlspecialchars($output['message']));
    $json = array($output, $check);
    echo json_encode($json);
} elseif (isset($_POST['userID']) && isset($_POST['questionID'])) {
    echo $codesetCourse->deleteSavedCode($_POST['userID'], $_POST['questionID']);
}
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/COURSE/COMPLETE.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-19 22:25:05
 * @Project: codeset.co.uk
 * @File Name: complete.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-07-18 21:04:33
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include "../../../../config.php";

$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];

$user = $auth->getUser($uid);

?>

<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Course Complete!</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body>
    <header class="signin-header">
        <h2 class="signin-header-title">Course Complete!</h1>
    </header>
    <main class="signin-main">
        Well done, you've completed all <?= $codesetCourse->getLastQuestionID() ?>
questions - you're officially a Python wizard!
        <br>
        <br>
        <br>
        <img src="/public_html/assets/images/wizard.gif">
        <br>
        <br>
        <br>
        <br>
        <a href="../">Go back to the Dashboard</a>
    </main>
</body>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/PAGES/PRIVATE/EDITOR/INDEX.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2017-01-24 16:50:19
 * @Project: codeset.co.uk
 * @File Name: index.php
 * @Last Modified by: Ben
 * @Last Modified time: 2017-04-25 22:15:18
**/

if (empty($_SERVER["HTTPS"]) || $_SERVER["HTTPS"] !== "on") {
    header("Location: https://" . $_SERVER["HTTP_HOST"] . $_SERVER["REQUEST_URI"]);
    exit();
}

include '../../../../config.php';

if (!$auth->isLogged()) {
    header("Location: ../../public/signin.php");
}

$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];

$user = $auth->getUser($uid);

$savedCode = $codesetCourse->getSavedCode($uid, "INT");

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>CodeSet - Python Editor</title>
    <link rel="stylesheet" type="text/css" href="/public_html/assets/css/main.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
</head>
<body style="max-height: 100vh; max-width: 100%; overflow: hidden;">
    <header class="course-header">
        <nav>
            <a href="../" alt="Back"><img src="/public_html/assets/images/logo-black.png"></
      a>
            <ul>
                <li>Python Editor</li>
            </ul>
        </nav>
    </header>
    <main>
        <div class="python-editor" id="python-editor"><? if ($savedCode !== False) {
            echo $savedCode;
        } else {
            echo $question['default'];
        } ?></div>
        <div class="python-terminal" id="python-terminal"><span class="course-terminal-
        red">>>></span></div>
        <div class="course-bar">
            <div class="course-bar-run">
```

```
                <input type="button" id="course-bar-run" value="Run">
            </div>
        </div>
    </main>
</body>
<script src="/public_html/libs/ace/ace.js" type="text/javascript" charset="utf-8"></script>
<script>
    function openNav() {
        document.getElementById("course-sidebar-arrow-out").style.zIndex = "0";
        document.getElementById("course-sidebar-lessons").style.width = "20%";
        document.getElementById("course-sidebar-lessons").style.padding = "1%";
    }

    /* Set the width of the side navigation to 0 and the left margin of the page content to 0
      */
    function closeNav() {
        document.getElementById("course-sidebar-arrow-out").style.zIndex = "1";
        document.getElementById("course-sidebar-lessons").style.width = "0";
        document.getElementById("course-sidebar-lessons").style.padding = "0";
    }

    var editor = ace.edit("python-editor");
    editor.setTheme("ace/theme/tomorrow_night");
    editor.getSession().setMode("ace/mode/python");
    editor.getSession().setUseWrapMode(true);
    document.getElementById('python-editor').style.fontSize='0.9em';
    editor.focus(); //To focus the ace editor
    var n = editor.getSession().getValue().split("\n").length; // To count total no. of lines
    editor.gotoLine(n); //Go to end of document
</script>
<script>
    $(function(){
      $('#course-bar-run').on('click', function(e){
        $('#python-terminal').html('<p><span class="course-terminal-red">>>Running...</
        span></p>');
        var inputs = []
        var lines = editor.getValue().match(/[^\r\n]+/g);
        if (lines.length > 0) {
            for (var i = 0; i <= lines.length - 1; i++) {
                if (lines[i].length > 0) {
                    if (lines[i].includes('#') == false) {
                        for (var x = 0; x < lines[i].split("input(").length - 1; x++) {
                            inputs.push(prompt("Enter the values for each of the inputs in
                      your program:", " "));
                        }
                    }
                }
            }
        }

        $("body").css({"cursor": "progress"});
        e.preventDefault();

        $.ajax({
          url: '../Course/run.php',
          type: 'post',
          dataType: 'json',
          data: {'code': editor.getValue(), 'userID': <?= $uid ?>, 'questionID': 'INT',
          'inputs': inputs},
          success: function(data) {
            $('#python-terminal').html('<p><span class="course-terminal-red">>></
             span>'+data[0]['message']+'</p>');
            if (data[1]['correct'] == true) {
                $("#course-bar-done").removeAttr("class");
                $("#course-bar-done").removeAttr("disabled");
            } else {
                $("#course-bar-done").attr("class", "disabled");
                $("#course-bar-done").attr("disabled", true);
            }
            $("body").css({"cursor": "default"});
          },
          error: function(data) {
            $('#python-terminal').html('<p><span class="course-terminal-red">>></
             span>Error running code</p>');
            $("body").css({"cursor": "default"});
          }
        });
      });
    });
</script>
</html>
```

CODESET.CO.UK/PUBLIC_HTML/ASSETS/CSS/MAIN.CSS

```
/**
 * @Author: Ben
 * @Date: 2016-11-16 14:14:46
 * @Project: CodeSet
 * @File Name: main.css
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-09-07 19:21:13
**/
@import url('https://fonts.googleapis.com/css?family=Josefin+Sans:200,300,400');
@import url('https://fonts.googleapis.com/css?family=Source+Code+Pro:300,400');


/*RESET*/
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
body {
    line-height: 1;
}
/*ul {
    list-style: none;
}*/
ol {
    list-style-position: inside;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}



/*STYLES*/


    /**Basics**/
    ::selection {
        background: #4f4f4f;
        color: #fff;
    }
    ::-moz-selection {
        background: #4f4f4f;
        color: #fff;
    }


    h1 {
        font-size: 5em;
    }


    h2 {
        font-size: 4em;
    }


    h3 {
        font-size: 3em;
    }


    h4 {
        font-size: 2em;
    }


    h5 {
        font-size: 1.5em;
    }


    h5 {
        font-size: 1.25em;
    }


    b {
        font-weight: 400;
    }


    i {
        font-style: italic;
    }


    form {
        position: relative;
        width: 100%;
    }
    input[type="text"],input[type="email"],input[type="password"],input[type="submit"],
input[type="button"], input[type="number"], textarea {
        position: relative;
        background: rgba(255,255,255,0.20);
        display: block;
        height: 6vh;
        width: 100%;
        padding: 0.10em 0.5em;
        border-radius: none;
        border: 4px solid white;
        font-family: 'Josefin Sans', sans-serif;
        font-weight: 300;
        font-size: 2em;
        box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.5);
        color: #fff;
    }


    textarea {
        width: 95%;
        font-size: 1em;
    }


    input[type="submit"], input[type="button"]{
        background-color: #fff;
        color: #0a3f5e;
        width: 107%;
        height: 7vh;
    }


    .button {
        position: relative;
        display: block;
        padding: 0.10em 0.5em;
        border-radius: none;
        border: 4px solid white;
        font-family: 'Josefin Sans', sans-serif;
        font-weight: 300;
        font-size: 2em;
        box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.5);
        background-color: #fff;
        color: #0a3f5e;
        width: 10vw;
        height: 7vh;
    }


    input[type="submit"]:active, input[type="button"]:active, button:active {
        background-color: rgba(255,255,255,0.25);
        color: #fff;
    }


    input[type="text"]:focus,input[type="email"]:focus,input[type="password"]:focus {
        outline: none;
    }


    a {
        text-decoration: none;
        color: white;
    }


    a:hover {
        color: #e8e8e8;
    }


    ol li {
        list-style-type: decimal;
        padding: 0;
        margin: 0;
    }


    /**Front Page**/
    html, body {
        background: #0a3f5e;
        font-family: 'Josefin Sans', sans-serif;
        font-weight: 300;
        max-width: 100vw;
        color: white;
    }


    header {
        position: absolute;
        width: 100vw;
    }


    header a {
        margin: 0 auto;
        text-align: center;
        float: center;
        color: white;
    }


    header .front-link {
        position: relative;
        top: 1vh;
        float: center;
        text-align: center;
        text-decoration: none;
        font-size: 1.25em;
        margin-left: 48vw;
    }


    header .front-link:before {
        content: "";
        position: absolute;
        top: 110%;
        width: 100%;
        height: 1px;
        bottom: 0;
        left: 0;
        background-color: #fff;
        visibility: hidden;
        -webkit-transform: scaleX(0);
        transform: scaleX(0);
        -webkit-transition: all 0.3s ease-in-out 0s;
        transition: all 0.3s ease-in-out 0s;
    }


    header .front-link:hover {
        color: #fff;
    }
```

```
header .front-link:hover:before {
    visibility: visible;
    -webkit-transform: scaleX(1);
    transform: scaleX(1);
}


/*Front Main*/
main .front-main {
    height: 100vh;
    background-image: url("../images/background.jpg");
    background-position: center center;
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: cover;
}


main .front-main .front-logo {
    width: 100%;
    display: block;
    transform: translate(0, 45vh);
}


main .front-main .front-logo #front-logo-text {
    width: 50%;
    margin-left: 22.5%;
    display: inline;
}


main .front-main .front-logo #front-logo-block {
    width: 3%;
    margin-left: 75%;
    margin-top: -8%;
    display: inline;
    -webkit-animation: flickerAnimation 3s infinite;
    -moz-animation: flickerAnimation 3s infinite;
    -o-animation: flickerAnimation 3s infinite;
    animation: flickerAnimation 3s infinite;
}


@keyframes flickerAnimation {
    0%   { opacity:1; }
    50%  { opacity:0; }
    100% { opacity:1; }
}
@-o-keyframes flickerAnimation {
    0%   { opacity:1; }
    50%  { opacity:0; }
    100% { opacity:1; }
}
@-moz-keyframes flickerAnimation {
    0%   { opacity:1; }
    50%  { opacity:0; }
    100% { opacity:1; }
}
@-webkit-keyframes flickerAnimation{
    0%   { opacity:1; }
    50%  { opacity:0; }
    100% { opacity:1; }
}


main .front-main .front-about {
    color: #fff;
    text-align: center;
    margin: 0 auto;
    transform: translate(0, 50vh);
    font-size: 1.5em;
    max-width: 50vw;
}


main .front-main .front-arrow {
    text-align: center;
    transform: translate(0, 75vh);
}


main .front-main .front-arrow a{
    -moz-transition: all .2s ease-in;
    -o-transition: all .2s ease-in;
    -webkit-transition: all .2s ease-in;
    transition: all .2s ease-in;
    width: 0;
    height: 0;
    border-style: solid;
    border-width: 50px 50px 0 50px;
    border-color: #e8e8e8 transparent transparent transparent;
    text-decoration: none;
}


main .front-main .front-arrow a:hover {
    width: 0;
    height: 0;
    border-style: solid;
    border-width: 50px 50px 0 50px;
    border-color: #fff transparent transparent transparent;
    text-decoration: none;
}


/*Front Login*/
main .front-login {
    height: 100vh;
}


main .front-login h1 {
    color: #fff;
    margin: 0 auto;
    text-align: center;
    padding-top: 3%;
    font-size: 5em;
}


main .front-login div {
    margin-top: 10%;
}


main .front-login .front-login-left {
    float: left;
    margin-left: 5%;
    line-height: 2em;
    max-width: 45%;
    color: #fff;
    font-size: 1.35em;
}
```

```
main .front-login .front-login-right {
    float: right;
    margin-right: 5%;
}


main .front-login .front-login-right form {
    position: relative;
    width: 40vw;
}


/*Front Footer*/
footer {
    padding: 5%;
    background-color: #fff;
    color: #0a3f5e;
    font-size: 2em;
}


/**Sign In Page**/
.signin-header {
    text-align: center;
    margin-top: 2em;
}


.signin-main {
    padding-top: 15em;
    width: 80vw;
    margin: 0 auto;
    text-align: center;
}


.signin-main form {
    padding-bottom: 2em;
}


/**Dashboard**/
/*Header*/
.dashboard-header, .course-header {
    width: 100vw;
    margin: 0 auto;
    text-align: center;
    display: block;
    position: absolute;
}


.dashboard-header nav, .course-header nav {
    background: white;
    -webkit-box-shadow: 0px 12px 39px -25px rgba(0,0,0,0.6);
    -moz-box-shadow: 0px 12px 39px -25px rgba(0,0,0,0.6);
    box-shadow: 0px 12px 39px -25px rgba(0,0,0,0.6);
    overflow: hidden;
    width: 100vw;
}


.dashboard-header nav img {
    width: 20%;
    display: inline;
    float: left;
    padding: 0.5% 1% 0.5% 0;
}
```

```
.dashboard-header nav ul {
    list-style-type: none;
    width: 99%;
    margin-top: 1.5%;
    font-size: 1.5em;
}


.dashboard-header nav ul a {
    text-align: left;
    float: left;
    color: #0a3f5e;
    cursor: pointer;
    padding-right: 2.5%;
}


.dashboard-header nav ul .active {
    color: #082e45;
    font-weight: 500;
}


.dashboard-header nav ul a:nth-last-child(-n+3) {
    float: right;
}


.dashboard-header nav ul a:nth-last-child() {
    margin-left: 15vw;
}


.dashboard-header nav ul a:nth-child(1) {
    margin-left: 0;
}


.dashboard-header nav ul a:nth-last-child(-n+2) {
    margin-right: -15vw;
}


.dashboard-header nav ul a:nth-last-child(-n+1) {
    margin-right: -10vw;
    width: 10%;
}


.dashboard-header nav ul a:hover, .dashboard-header nav ul .active:hover {
    color: #0f5d8a;
}


.dashboard-header nav a .profile {
    max-width: 250px;
    max-height: 250px;
    margin-top: -0.55em;
    width: 20%;
    border-radius: 100%;
    float: right;
    padding: 0;
}


.dashboard-header .dashboard-progress {
    background-color: #0a3f5e;
    height: 45vh;
    clear: both;
    overflow: scroll;
    white-space: nowrap;
    padding-left: 5%;
    padding-right: 5%;
    /*cursor: url('/public_html/assets/images/cursor.gif'), all-scroll;*/
}


.dashboard-header .dashboard-progress .dashboard-progress-circle {
    background-color: #fff;
    border-radius: 100%;
    padding: 25px;
    /*padding-left: 30px;*/
    /*padding-right: 30px;*/
    width: 1.25%;
    color: #0a3f5e;
    display: inline-block;
    margin-top: 15%;
}


.dashboard-header .dashboard-progress .dashboard-progress-circle-uncomplete {
    background-color: rgba(0,0,0,0);
    color: #fff;
    border: 2.5px solid #fff;
}


.dashboard-header .dashboard-progress .dashboard-progress-start, .dashboard-progress-circle, .dashboard-progress-line {
    margin: 0 auto;
    margin-top: 8%;
}


.dashboard-header .dashboard-progress .dashboard-progress-line {
    display: inline-block;
    width: 10%;
    height: 2%;
    background-color: #fff;
}


.dashboard-header .dashboard-progress h4 {
    position: fixed;
    left: 0;
    right: 0;
    margin-right: auto;
    margin-left: auto;
}


#dashboard-header-name {
    padding-right: 3%;
}


/*Dashboard Main*/
.dashboard-main {
    position: absolute;
    top: 50vh;
    display: block;
    width: 100%;
    background-color: white;
    color: #0a3f5e;
    min-height: 100%;
    -webkit-box-shadow: 0 -12px 39px -25px  rgba(0,0,0,0.6);
    -moz-box-shadow: 0 -12px 39px -25px  rgba(0,0,0,0.6);
    box-shadow: 0 -12px 39px -25px  rgba(0,0,0,0.6);
}


.dashboard-main a {
    color: white;
}


.dashboard-main a:hover {
    color: white;
}


.dashboard-main h1 {
    padding: 1%;
}


.dashboard-main .dashboard-news {
    width: 57.5%;
    float: left;
    border-right: 2.5px solid #0a3f5e;
    height: 100%;
    /*padding-right: 1.25%;*/
    padding: 1.25%;
}


.dashboard-main .dashboard-leaderboard, .dashboard-sidebar {
    width: 37.5%;
    float: right;
}


/* Dashboard Main News Section */
.dashboard-main .dashboard-news .dashboard-news-card{
    -webkit-box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
    -moz-box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
    box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
    padding: 2%;
    margin-bottom: 2.5%;
}


.dashboard-main .dashboard-news .dashboard-news-card-active{
    -webkit-box-shadow: 0px 8px 39px 0 rgba(0,0,0,0.2);
    -moz-box-shadow: 0px 8px 39px 0 rgba(0,0,0,0.2);
    box-shadow: 0px 8px 39px 0 rgba(0,0,0,0.2);
}


.dashboard-main .dashboard-news .dashboard-news-card-expired{
    opacity: 0.5;
}


/* Dashboard Main Leaderboard Section */
.dashboard-main .dashboard-leaderboard ol {
    padding: 5%;
    list-style-type: decimal;
    margin-right: 2.5%;
    -webkit-box-shadow: 0px 5px 39px 0 rgba(0,0,0,0.2);
    -moz-box-shadow: 0px 5px 39px 0 rgba(0,0,0,0.2);
    box-shadow: 0px 5px 39px 0 rgba(0,0,0,0.2);
```

```
}


/**Classes**/
/*User's classes*/
.classes-progress .classes-progress-class {
    max-height: 35vh;
    height: 35vh;
    display: inline-block;
    width: 7.5%;
    bottom: -2.5vh;
    position: relative;
    border: 1px solid white;
    margin-right: 5%;
}


.student-progress-name {
    bottom: -4.6vh!important;
}


.classes-progress .classes-progress-class .classes-progress-name {
    position: absolute;
    bottom: -2.5vh;
    left: 0;
}


.classes-progress .classes-progress-class .classes-progress-bar {
    position: absolute;
    width: 100%;
    background-color: white;
    bottom: 0;
}


.dashboard-main .dashboard-classes {
    margin-left: 5%;
}


.dashboard-main .dashboard-classes .dashboard-class-tile {
    float:left;
    position: relative;
    width: 20%;
    padding-bottom : 20%; /* = width for a 1:1 aspect ratio */
    margin:1.66%;
    background-color: #0a3f5e;
    overflow:hidden;
    transition: .2s ease-in-out;
    box-shadow: inset 0px 3px 8px rgba(0, 0, 0, 0.5);
}


.dashboard-main .dashboard-classes .dashboard-class-tile .class-tile-content {
    position:absolute;
    height:90%; /* = 100% - 2*5% padding */
    width:90%; /* = 100% - 2*5% padding */
    padding: 5%;
    color: white;
    text-align: center;
}


.dashboard-main .dashboard-classes .dashboard-class-tile .class-tile-content-table {
    display: table;
    width: 100%;
```

```
        height: 100%;
    }


    .dashboard-main .dashboard-classes .dashboard-class-tile .class-tile-content-table * {
        display:table-cell;
        vertical-align:middle;
    }


    .dashboard-main .dashboard-classes .dashboard-class-tile:hover {
        -webkit-transform: scale(1.1);
        -ms-transform: scale(1.1);
        transform: scale(1.1);
        box-shadow: inset 0px 3px 8px rgba(0, 0, 0, 0);
    }


    .class-tile-content-table-hover {
        opacity: 0;
        transition: .2s ease-in-out;
    }


    .dashboard-classes .dashboard-class-tile:hover .class-tile-content-table-hover {
        opacity: 100;
    }


    .popup-overlay {
        display: none;
        position: fixed;
        background-color: rgba(0,0,0,0.6);
        width: 100%;
        height: 100%;
        top: 0;
        left: 0;
        z-index: 10;
    }


    .popup {
        position: absolute;
        background: #0a3f5e;
        margin: 0 auto;
        left: 25%;
        top: 37.5%;
        width: 50%;
        height: 25%;
        z-index: 10;
        border-radius: 10px;
        border: 4px solid white;
    }


    .popup-overlay .popup form {
        position: relative;
        margin: 0 auto;
        width: 75%;
        left: 0;
        top: -4%;
    }


    .popup-overlay .popup form input {
        width: 100%;
    }
    .popup-overlay .popup form input[type="text"], .popup-overlay .popup form
input[type="password"], .popup-overlay .popup form input[type="email"] {
        width: 94%;
        margin-left: -1%;
    }


    .popup-overlay .popup .popup-close {
        color: white;
        font-size: 3em;
        font-weight: bold;
        position: relative;
        top: -0.25em;
        left: 96%;
        padding: 0;
        margin: 0;
    }


    /*Individual Class*/
    .class-sidebar {
        padding: 5%;
        margin-right: 2%;
        margin-left: 2%;
        -webkit-box-shadow: 0px 5px 39px 0 rgba(0, 0, 0, 0.2);
        -moz-box-shadow: 0px 5px 39px 0 rgba(0, 0, 0, 0.2);
        box-shadow: 0px 5px 39px 0 rgba(0, 0, 0, 0.2);
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-edit {
        width: 90%;
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-edit .class-edit-button {
        width: 45%;
        float: right;
        margin-left: 1%;
    }


    #rename-class-form {
        display: none;
        width: 86.5%;
        margin: 0 auto;
        margin: 10% 10% 10% 9%;
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-news form .new-deadline-list {
        list-style: disc;
        margin-left: 5%;
        background-color: #e8e8e8;
        padding-left: 5%;
        padding-top: 2.5%;
        padding-bottom: 2.5%;
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-news form .new-deadline-list
input[type="submit"] {
        width: 30%;
        height: 10%;
        font-size: 1em;
    }
    .dashboard-classes .dashboard-sidebar .sidebar-class-news .deadline-card {
        -webkit-box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
        -moz-box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
        box-shadow: 0px 12px 39px 0 rgba(0,0,0,0.1);
        padding: 2%;
        margin-bottom: 2.5%;
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-news .deadline-card-expired {
        opacity: 0.5;
    }


    .dashboard-classes .dashboard-sidebar .sidebar-class-news .deadline-card .deadline-card-text {
        color: #0f5d8a;
    }


    .dashboard-classes .class-students {
        list-style: none;
        width: 55%;
    }


    .dashboard-classes .class-students li {
        padding: 1%;
        margin-bottom: 1%;
        box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.5);
    }


    #new-student-li {
        box-shadow: none;
        margin-left: 2%;
    }


    #new-student-li:before {
        content: '+';
        margin-left: -25px;
    }


    #new-student-form {
        margin-top: -2.75%;
        /*width: 90%;*/
    }


    #new-student-form input {
        display: inline;
        float: left;
        width: calc(90% / 3);
        height: 1.5em;
        font-size: 100%;
        color: #0a3f5e;
        margin-right: 0.5%;
    }


    #new-student-form input[type="submit"] {
        height: 2.2em;
    }
```

```
#delete-student-submit, #delete-deadline-submit {
    color: #0a3f5e;
    float: right;
    background: none!important;
    border: none;
    padding: 0!important;
    font: inherit;
    cursor: pointer;
    margin-top: -1em;
    margin-right: -0.5em;
}


#delete-student-submit:hover, #delete-deadline-submit:hover, #email-students:hover {
    color: #0f5d8a;
    border-bottom: 1px solid #0f5d8a;
}


.space {
    padding: 1%;
}


#email-students-popup {
    height: 25%;
}


/**Account**/
.account-profile {
    border-radius: 100%;
    width: 100%;
    /*width: 100%;*/
    margin-top: 3%;
}


.account-profile-container {
    position: relative;
    width: 15%;
    margin: 0 auto;
    margin-top: 2.5%;
    /*padding-bottom: 5%;*/
    overflow: hidden;
}


#account-profile-file {
    visibility: hidden;
}


.account-profile-container input[type="button"] {
    position: relative;
    margin: 0 auto;
    margin-top: -77.5%;
    width: 75%;
    opacity: 0;
    transition: opacity .35s ease;
}


.account-profile-container:hover input[type="button"] {
    opacity: 1;
}
```

```
    .account-main a {
        color: #0a3f5e;
        cursor: pointer;
    }


    .account-main a:hover {
        color: #0f5d8a;
    }


    .account-main .account-info {
        padding-left: 5%;
        line-height: 2em;
    }


    .account-main .account-info .account-info-change {
        font-size: 0.75em;
        margin-left: 1%;
        cursor: pointer;
        color: #0f5d8a;
    }


    .account-main .account-info .account-info-change:hover {
        color: #0a3f5e;
    }


    #change-email-popup {
        margin-top: -5%;
        padding-bottom: 5%;
    }


    #change-password-popup {
        margin-top: -10%;
        padding-bottom: 10%;
    }


    /**Course**/
    /*Header*/
    .course-header {
        width: 100%;
    }


    .course-header nav {
        height: 5vh;
    }


    .course-header nav img {
        width: 10%;
        display: inline;
        float: left;
        padding: 0.5% 1% 0.5% 0;
    }


    .course-header nav ul {
        list-style-type: none;
        width: 100%;
        margin-top: 0.75%;
        font-size: 1.5em;
    }
```

```
.course-header nav ul li {
    text-align: center;
    float: center;
    color: #0a3f5e;
}


.course-progress-circle {
    background-color: #0a3f5e;
    border-radius: 100%;
    padding: 3px;
    width: 22.5px;
    /*padding-left: 30px;*/
    /*padding-right: 30px;*/
    /*width: 1.25%;*/
    color: #fff;
    display: inline-block;
    /*margin-top: 15%;*/
    margin-right: -15vw;
}


.course-progress-circle-uncomplete {
    background-color: #fff;
    color: #0a3f5e;
    border: 2.5px solid #0a3f5e;
}


.course-progress-circle .course-progress-circle-uncomplete {
    float: right;
    margin-left: 100%;
    margin-top: -50%;
    /*margin-top: 8%;*/
}


#ask-teacher-popup {
    margin-top: -5%;
    padding-bottom: 5%;
}


/*Sidebar*/
.course-sidebar {
    color: #0a3f5e;
    width: 20%;
    float: left;
    margin-top: 5vh;
    height: calc(91.5vh - 3em);
    background-color: #fff;
    padding: 1%;
    line-height: 1.5em;
    font-size: 1.2em;
    overflow: scroll;
    /*padding-bottom: 50vh;*/
}


.course-sidebar a {
    color: #0f5d8a;
    cursor: pointer;
}


.course-sidebar a:hover {
    color: #0a3f5e;
}


.course-sidebar .course-sidebar-normal .course-sidebar-instructions {
    position: relative;
    margin: 0 auto;
    padding: 5%;
    text-align: center;
    background-color: #e8e8e8;
    width: 75%;
}


.course-sidebar .course-sidebar-normal #course-sidebar-arrow-out, #course-sidebar-arrow-
in {
    position: fixed;
    bottom: 0.5vh;
    z-index: 1;
}


#course-sidebar-arrow-in {
    display: none;
}


.course-sidebar .course-sidebar-lessons {
    margin-top: 5vh;
    height: 91.5vh;
    max-height: 91.5vh;
    width: 0;
    position: fixed;
    z-index: 1;
    top: 0;
    left: 0;
    background-color: #fff;
    overflow-x: hidden;
    overflow-y: hidden;
    transition: 0.5s;
}


.course-sidebar .course-sidebar-lessons * {
    display: block;
}


.course-sidebar .course-sidebar-lessons div {
    overflow-y: scroll;
    max-height: 90%;
    margin: 0 auto;
}


.course-sidebar .course-sidebar-lessons div a {
    border-bottom: 1px solid #0a3f5e;
    padding-top: 1%;
    padding-bottom: 1%;
}


.course-sidebar .course-sidebar-lessons div a:first-child {
    border-top: 1px solid #0a3f5e;
}


.course-sidebar .course-sidebar-lessons .course-sidebar-lessons-active {
    background: rgba(0,0,0,0.025);
}


#course-reset-code {
    font-size: 1em;
    width: 50%;
    height: auto;
    margin: 0 auto;
    margin-top: 5%;
}


pre {
    background-color: #0a3f5e;
    padding: 2%;
    font-family: 'Monaco', monospace;
    font-weight: 300;
    color: #fff;
    overflow-y: scroll;
}


code {
    display: inline;
    padding: 0;
    background-color: #0a3f5e;
    font-family: 'Monaco', monospace;
    font-weight: 300;
    color: #fff;
}


/*Bar*/
.course-bar {
    position: absolute;
    width: 100%;
    background-color: white;
    height: 7.5vh;
    top: 92.5vh;
}


.course-bar .course-bar-done, .course-bar-run {
    float: right;
    margin-right: 2%;
    width: 10vw;
    height: 5vh;
}


.disabled {
    background: lightgrey!important;
    color: grey!important;
    border-color: white!important;
}


/*Editor*/
.course-editor {
    width: 40%;
    height: 87.5vh;
    margin-top: 5vh;
    float: left;
    background-color: #0f5d8a;
}
/*Terminal*/
.course-terminal {
    width: 38%;
    height: 87.5vh;
    margin-top: 5vh;
    float: right;
    background-color: #0a3f5e;
    overflow: scroll;
}


.course-terminal-red {
    color: red;
}


/**Course**/
/*Editor*/
.python-editor {
    width: 50%;
    height: 87.5vh;
    margin-top: 5vh;
    float: left;
    background-color: #0f5d8a;
}


.python-terminal {
    width: 50%;
    height: 87.5vh;
    margin-top: 5vh;
    float: right;
    background-color: #0a3f5e;
    overflow: scroll;
}
```

CODESET.CO.UK/CONFIG.PHP

```
<?php
/** * @Author: Ben
 * @Date: 2016-11-16 14:42:55
 * @Project: CodeSet
 * @File Name: config.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-24 11:52:00
**/

/*SERVER SETUP*/

    $DIR = $_SERVER['DOCUMENT_ROOT'] . '/';
    $PUBLIC_DIR = "/";

    //Define the database details
    $dbs = array(
        "CODESET_AUTH" => array(
            "dbname" => "db657046199",
            "username" => "xxxxxxxxxxxx",
            "password" => "xxxxxxxxxxxx",
            "host" => "xxxxxxxxxxxxxxxxxxxxxxx"
        ),
        "CODESET_CLASSES" => array(
            "dbname" => "db662374861",
            "username" => "xxxxxxxxxxxx",
            "password" => "xxxxxxxxxxxxx",
            "host" => "xxxxxxxxxxxxxxxxxxxxxxx"
        ),
        "CODESET_COURSE" => array(
            "dbname" => "db668273832",
            "username" => "xxxxxxxxxxxx",
            "password" => "xxxxxxxxxxxxxxx",
            "host" => "xxxxxxxxxxxxxxxxxxxxxxx"
        ),
        "CODESET_NEWS" => array(
            "dbname" => "db658087996",
            "username" => "xxxxxxxxxxxx",
            "password" => "xxxxxxxxxxx",
            "host" => "xxxxxxxxxxxxxxxxxxxxxxx"
        ),
        "CODESET_CONTACT" => array(
            "dbname" => "db686001594",
            "username" => "xxxxxxxxxxxx",
            "password" => "xxxxxxxxxxxxxx",
            "host" => "xxxxxxxxxxxxxxxxxxxxxxx"
        )
    );

    //Set default timezone for all date/time functions
    date_default_timezone_set('Europe/London');

/*INCLUDES SETUP*/
    //Set up composer
    require_once $DIR.'vendor/autoload.php';

    //Set up CodeSet Mail
    require_once $DIR.'private_server/includes/CodeSetMail/class.mail.php';
    $codesetMail = new CodeSet\SendMail('587', 'server@codeset.co.uk', 'CodeSetServer1');

    //Set up PHPAuth system
    $authDBH = new PDO("mysql:host=".$dbs['CODESET_AUTH']['host'].";dbname=".
       $dbs['CODESET_AUTH']['dbname'], $dbs['CODESET_AUTH']['username'], $dbs['CODESET_AUTH']
       ['password']);
    $config = new PHPAuth\Config($authDBH);
    $auth = new PHPAuth\Auth($authDBH, $config, "en_GB", $codesetMail);

    //Set up CodeSet Classes
    require_once $DIR.'private_server/includes/CodeSetClasses/class.classes.php';
    $classesDBH = new PDO("mysql:host=".$dbs['CODESET_CLASSES']['host'].";dbname=".
       $dbs['CODESET_CLASSES']['dbname'], $dbs['CODESET_CLASSES']['username'],
       $dbs['CODESET_CLASSES']['password']);

    $codesetClasses = new CodeSet\Classes($classesDBH);

    //Set up CodeSet Course
    require_once $DIR.'private_server/includes/CodeSetCourse/class.course.php';
    $courseDBH = new PDO("mysql:host=".$dbs['CODESET_COURSE']['host'].";dbname=".
       $dbs['CODESET_COURSE']['dbname'], $dbs['CODESET_COURSE']['username'],
       $dbs['CODESET_COURSE']['password']);
    $codesetCourse = new CodeSet\Course($courseDBH);
    //Set up CodeSet News
    require_once $DIR.'private_server/includes/CodeSetNews/class.news.php';
    $newsDBH = new PDO("mysql:host=".$dbs['CODESET_NEWS']['host'].";dbname=".
       $dbs['CODESET_NEWS']['dbname'], $dbs['CODESET_NEWS']['username'], $dbs['CODESET_NEWS']
       ['password']);
    $codesetNews = new CodeSet\News($newsDBH);
```

CODESET.CO.UK/VENDOR/PHPAUTH/PHPAUTH/AUTH.PHP

```
<?php
namespace PHPAuth;
include $_SERVER['DOCUMENT_ROOT']."/config.php";
include $_SERVER['DOCUMENT_ROOT']."/vendor/autoload.php";
/**
 * Auth class
 * Required PHP 5.4 and above.
 */
class Auth
{
```

```php
    protected $dbh;
    public $config;
    public $lang;
    private $codesetMail;
    /**
     * Initiates database connection
     */
    public function __construct(\PDO $dbh, $config, $language = "en_GB", $mail)
    {
        $this->dbh = $dbh;
        $this->config = $config;
        $this->codesetMail = $mail;
        if (version_compare(phpversion(), '5.4.0', '<')) {
            die('PHP 5.4.0 required for PHPAuth engine!');
        }
        if (version_compare(phpversion(), '5.5.0', '<')) {
            require("files/password.php");
        }
        // Load language
        require "languages/{$language}.php";
        $this->lang = $lang;
        date_default_timezone_set($this->config->site_timezone);
    }
    /**
     * Logs a user in
     * @param string $email
     * @param string $password
     * @param int $remember
     * @param string $captcha = NULL
     * @return array $return
     */
    public function login($email, $password, $remember = 1, $captcha = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        $validateEmail = $this->validateEmail($email);
        $validatePassword = $this->validatePassword($password);
        if ($validateEmail['error'] == 1) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_password_invalid"];
            return $return;
        } elseif ($validatePassword['error'] == 1) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_password_invalid"];
            return $return;
        } elseif ($remember != 0 && $remember != 1) {
            $this->addAttempt();
            $return['message'] = $this->lang["remember_me_invalid"];
            return $return;
        }
        $uid = $this->getUID(strtolower($email));
        if (!$uid) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_password_incorrect"];
            return $return;
        }
        $user = $this->getBaseUser($uid);
        if (!password_verify($password, $user['password'])) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_password_incorrect"];
            return $return;
        }
        if ($user['isactive'] != 1) {
            $this->addAttempt();
            $return['message'] = $this->lang["account_inactive"];
            return $return;
        }
        $sessiondata = $this->addSession($user['uid'], $remember);
        if ($sessiondata == false) {
            $return['message'] = $this->lang["system_error"] . " #01";
            return $return;
        }
        $return['error'] = false;
        $return['message'] = $this->lang["logged_in"];
        $return['hash'] = $sessiondata['hash'];
        $return['expire'] = $sessiondata['expiretime'];
        return $return;
    }
    /**
    * Generates random password, creates user
    * @param string $name
    * @param string $email
    * @return array $return
    */
    public function registerStudent($name, $email)
    {
        $password = $this->getRandomPassword(2);
        $register = $this->register(htmlspecialchars($name), htmlspecialchars($email),
         $password, $password);
        if ($register['error'] == true) {
            // echo "string";
            return $register;
        } else {
            $query = $this->dbh->prepare("UPDATE users SET permission = 0, generated_password
             = ? WHERE email = ?");
            if (!$query->execute(array($password, $email))) {
                $this->deleteUser($getUID($email));
                return "Could not edit user in table. Please try again";
            }
            return true;
        }
    }
    /**
    * Creates a new user, adds them to database
    * @param string $email
    * @param string $password
    * @param string $repeatpassword
    * @param array  $params
    * @param string $captcha = NULL
    * @param bool $sendmail = NULL
    * @return array $return
    */
    public function register($name, $email, $password, $repeatpassword, $params = array(),
       $captcha = null, $sendmail = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        if ($password !== $repeatpassword) {
            $return['message'] = $this->lang["password_nomatch"];
            return $return;
        }
        // Validate email
        $validateEmail = $this->validateEmail($email);
        if ($validateEmail['error'] == 1) {
            $return['message'] = $validateEmail['message'];
            return $return;
        }
        // Validate password
        $validatePassword = $this->validatePassword($password);
        if ($validatePassword['error'] == 1) {
            $return['message'] = $validatePassword['message'];
            return $return;
        }

        if ($this->isEmailTaken($email)) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_taken"];
            return $return;
        }
        $addUser = $this->addUser(htmlspecialchars($name), $email, $password, $params,
         $sendmail);
        if ($addUser['error'] != 0) {
            $return['message'] = $addUser['message'];
            return $return;
        }
        $return['error'] = false;
        $return['message'] = ($sendmail == true ? $this->lang["register_success"] : $this-
         >lang['register_success_emailmessage_suppressed'] );
        return $return;
    }
    /**
    * Activates a user's account
    * @param string $key
    * @return array $return
    */
    public function activate($key)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        if (strlen($key) !== 20) {
            $this->addAttempt();
            $return['message'] = $this->lang["activationkey_invalid"];
            return $return;
        }
        $getRequest = $this->getRequest($key, "activation");
        if ($getRequest['error'] == 1) {
            $return['message'] = $getRequest['message'];
            return $return;
        }
        if ($this->getBaseUser($getRequest['uid'])['isactive'] == 1) {
            $this->addAttempt();
            $this->deleteRequest($getRequest['id']);
            $return['message'] = $this->lang["system_error"] . " #02";
            return $return;
        }
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET isactive = ?
         WHERE id = ?");
        $query->execute(array(1, $getRequest['uid']));
        $this->deleteRequest($getRequest['id']);
        $return['error'] = false;
        $return['message'] = $this->lang["account_activated"];
        return $return;
    }
    /**
    * Creates a reset key for an email address and sends email
    * @param string $email
    * @return array $return
    */
    public function requestReset($email, $sendmail = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        $validateEmail = $this->validateEmail($email);
        if ($validateEmail['error'] == 1) {
            $return['message'] = $this->lang["email_invalid"];
            return $return;
        }
        $query = $this->dbh->prepare("SELECT id FROM {$this->config->table_users} WHERE email
         = ?");
        $query->execute(array($email));
        if ($query->rowCount() == 0) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_incorrect"];
            return $return;
        }
        $addRequest = $this->addRequest($query->fetch(\PDO::FETCH_ASSOC)['id'], $email,
         "reset", $sendmail);
        if ($addRequest['error'] == 1) {
            $this->addAttempt();
            $return['message'] = $addRequest['message'];
            return $return;
        }
        $return['error'] = false;
        $return['message'] = ($sendmail == true ? $this->lang["reset_requested"] : $this-
        >lang['reset_requested_emailmessage_suppressed']);
        return $return;
    }
    /**
    * Logs out the session, identified by hash
    * @param string $hash
    * @return boolean
    */
    public function logout($hash)
    {
        if (strlen($hash) != 40) {
            return false;
        }
        return $this->deleteSession($hash);
    }
    /**
    * Gets user's progress
    * @param int $uid
    * @return boolean
    */
    public function getProgress($uid)
    {
        $query = $this->dbh->prepare("SELECT progress FROM users WHERE id = ?");
        $query->execute(array($uid));
        if ($query->rowCount() == 0) {
            return false;
        }
        return $query->fetch(\PDO::FETCH_ASSOC)['progress'];
    }
    /**
    * Sets user's progress
    * @param int $uid
    * @return boolean
    */
    public function setProgress($uid, $progress)
    {
        $query = $this->dbh->prepare("UPDATE users SET progress = ? WHERE id = ?");
        if (!$query->execute(array($progress, $uid))) {
            return false;
        }
        return true;
    }
    /**
    * Hashes provided password with Bcrypt
    * @param string $password
    * @param string $password
    * @return string
    */
    public function getHash($password)
    {
        return password_hash($password, PASSWORD_BCRYPT, ['cost' => $this->config-
        >bcrypt_cost]);
    }
    /**
    * Gets UID for a given email address and returns an array
    * @param string $email
    * @return array $uid
    */
    public function getUID($email)
    {
        $query = $this->dbh->prepare("SELECT id FROM {$this->config->table_users} WHERE email
         = ?");
        $query->execute(array($email));
        if ($query->rowCount() == 0) {
            return false;
        }
        return $query->fetch(\PDO::FETCH_ASSOC)['id'];
    }
    /**
    * Creates a session for a specified user id
    * @param int $uid
    * @param boolean $remember
    * @return array $data
    */
    protected function addSession($uid, $remember)
    {
        $ip = $this->getIp();
        $user = $this->getBaseUser($uid);
        if (!$user) {
            return false;
        }
        $data['hash'] = sha1($this->config->site_key . microtime());
        $agent = $_SERVER['HTTP_USER_AGENT'];
        $this->deleteExistingSessions($uid);
        if ($remember == true) {
            $data['expire'] = date("Y-m-d H:i:s", strtotime($this->config->cookie_remember));
            $data['expiretime'] = strtotime($data['expire']);
        } else {
            $data['expire'] = date("Y-m-d H:i:s", strtotime($this->config->cookie_forget));
            $data['expiretime'] = 0;
        }
        $data['cookie_crc'] = sha1($data['hash'] . $this->config->site_key);
        $query = $this->dbh->prepare("INSERT INTO {$this->config->table_sessions} (uid, hash,
         expiredate, ip, agent, cookie_crc) VALUES (?, ?, ?, ?, ?, ?)");
        if (!$query->execute(array($uid, $data['hash'], $data['expire'], $ip, $agent,
        $data['cookie_crc']))) {
            return false;
        }
        $data['expire'] = strtotime($data['expire']);
        return $data;
    }
    /**
    * Removes all existing sessions for a given UID
    * @param int $uid
    * @return boolean
    */
    protected function deleteExistingSessions($uid)
    {
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_sessions} WHERE uid =
         ?");
        $query->execute(array($uid));
        return $query->rowCount() == 1;
    }
    /**
    * Removes a session based on hash
    * @param string $hash
    * @return boolean
    */
    protected function deleteSession($hash)
    {
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_sessions} WHERE hash
         = ?");
        $query->execute(array($hash));
        return $query->rowCount() == 1;
    }
    /**
    * Function to check if a session is valid
    * @param string $hash
    * @return boolean
    */
    public function checkSession($hash)
    {
        $ip = $this->getIp();
        $block_status = $this->isBlocked();
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return false;
        }
        if (strlen($hash) != 40) {
            return false;
        }
        $query = $this->dbh->prepare("SELECT id, uid, expiredate, ip, agent, cookie_crc FROM
         {$this->config->table_sessions} WHERE hash = ?");
        $query->execute(array($hash));
        if ($query->rowCount() == 0) {
            return false;
        }
        $row = $query->fetch(\PDO::FETCH_ASSOC);
        $sid = $row['id'];
        $uid = $row['uid'];
        $expiredate = strtotime($row['expiredate']);
        $currentdate = strtotime(date("Y-m-d H:i:s"));
        $db_ip = $row['ip'];
        $db_agent = $row['agent'];
        $db_cookie = $row['cookie_crc'];
        if ($currentdate > $expiredate) {
            $this->deleteExistingSessions($uid);
            return false;
        }
        if ($ip != $db_ip) {
            return false;
        }
        if ($db_cookie == sha1($hash . $this->config->site_key)) {
            return true;
        }
        return false;
    }
    /**
    * Retrieves the UID associated with a given session hash
    * @param string $hash
    * @return int $uid
    */
    public function getSessionUID($hash)
    {
        $query = $this->dbh->prepare("SELECT uid FROM {$this->config->table_sessions} WHERE
         hash = ?");
        $query->execute(array($hash));
        if ($query->rowCount() == 0) {
            return false;
        }
        return $query->fetch(\PDO::FETCH_ASSOC)['uid'];
    }
    /**
    * Checks if an email is already in use
    * @param string $email
    * @return boolean
    */
    public function isEmailTaken($email)
    {
        $query = $this->dbh->prepare("SELECT count(*) FROM {$this->config->table_users} WHERE
         email = ?");
        $query->execute(array($email));
        if ($query->fetchColumn() == 0) {
            return false;
        }
        return true;
    }
    /**
    * Adds a new user to database
    * @param string $email      -- email
    * @param string $password   -- password
    * @param array $params      -- additional params
    * @return int $uid
    */
    protected function addUser($name, $email, $password, $params = array(), &$sendmail)
    {
        $return['error'] = true;
        $query = $this->dbh->prepare("INSERT INTO {$this->config->table_users} VALUES ()");
        if (!$query->execute()) {
            $return['message'] = $this->lang["system_error"] . " #03";
            return $return;
        }
        $uid = $this->dbh->lastInsertId();
        $email = htmlentities(strtolower($email));
        if ($sendmail) {
            $addRequest = $this->addRequest($uid, $email, "activation", $sendmail);
            if ($addRequest['error'] == 1) {
                $query = $this->dbh->prepare("DELETE FROM {$this->config->table_users} WHERE
                 id = ?");
                $query->execute(array($uid));
                $return['message'] = $addRequest['message'];
                return $return;
            }
            $isactive = 0;
        } else {
            $isactive = 1;
        }
        $password = $this->getHash($password);
        if (is_array($params)&& count($params) > 0) {
            $customParamsQueryArray = array();
            foreach ($params as $paramKey => $paramValue) {
                $customParamsQueryArray[] = array('value' => $paramKey . ' = ?');
            }
            $setParams = ', ' . implode(', ', array_map(function ($entry) {
                return $entry['value'];
            }, $customParamsQueryArray));
        } else {
            $setParams = '';
        }
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET name = ?, email
         = ?, password = ?, isactive = ?, permission = ? WHERE id = ?");
        $bindParams = array_values(array_merge(array($name, $email, $password, $isactive, 1),
         array($uid)));
        if (!$query->execute($bindParams)) {
            $query = $this->dbh->prepare("DELETE FROM {$this->config->table_users} WHERE id =
             ?");
            $query->execute(array($uid));
            $return['message'] = $this->lang["system_error"] . " #04";
            return $return;
        }
        $return['error'] = false;
        return $return;
    }
    /**
    * Gets basic user data for a given UID and returns an array
    * @param int $uid
    * @return array $data
    */
    protected function getBaseUser($uid)
    {
        $query = $this->dbh->prepare("SELECT email, password, isactive FROM {$this->config-
         >table_users} WHERE id = ?");
        $query->execute(array($uid));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        $data['uid'] = $uid;
        return $data;
    }
    /**
    * Gets public user data for a given UID and returns an array, password is not returned
    * @param int $uid
    * @return array $data
    */
    public function getUser($uid)
    {
        $query = $this->dbh->prepare("SELECT * FROM {$this->config->table_users} WHERE id
         = ?");
        $query->execute(array($uid));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        $data['uid'] = $uid;
        // unset($data['password']);
        return $data;
    }
    /**
    * Allows a user to delete their account
    * @param int $uid
    * @param string $password
    * @param string $captcha = NULL
    * @return array $return
    */
    public function deleteUser($uid, $captcha = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }

        $user = $this->getBaseUser($uid);

        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_users} WHERE id
         = ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #05";
            return $return;
        }
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_sessions} WHERE uid =
         ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #06";
            return $return;
        }
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_requests} WHERE uid =
         ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #07";
            return $return;
        }
        if (!$this->deleteDir("https://codeset.co.uk/private_server/guard/CELL-$uid/")) {
            $return['message'] = $this->lang["system_error"];
            return $return;
        }
        $return['error'] = false;
        $return['message'] = $this->lang["account_deleted"];
        return $return;
    }
    /**
    * Delete a student's account
    * @param int $uid
    * @return array $return
    */
    public function deleteStudent($uid)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        $user = $this->getBaseUser($uid);
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_users} WHERE id
         = ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #05";
            return $return;
        }
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_sessions} WHERE uid =
         ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #06";
            return $return;
        }
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_requests} WHERE uid =
         ?");
        if (!$query->execute(array($uid))) {
            $return['message'] = $this->lang["system_error"] . " #07";
            return $return;
        }
        if (!$this->deleteDir("https://codeset.co.uk/private_server/guard/CELL-$uid/")) {
            $return['message'] = $this->lang["system_error"];
            return $return;
        }
        $return['error'] = false;
        $return['message'] = $this->lang["account_deleted"];
        return $return;
    }
    /**
    * Creates an activation entry and sends email to user
    * @param int $uid
    * @param string $email
    * @param string $type
    * @param boolean $sendmail = NULL
    * @return boolean
    */
    protected function addRequest($uid, $email, $type, &$sendmail)
    {
        $return['error'] = true;
        if ($type != "activation" && $type != "reset") {
            $return['message'] = $this->lang["system_error"] . " #08";
            return $return;
        }
        // if not set manually, check config data
        if ($sendmail === null) {
            $sendmail = true;
            if ($type == "reset" && $this->config->emailmessage_suppress_reset === true) {
                $sendmail = false;
                $return['error'] = false;
                return $return;
            }
            if ($type == "activation" && $this->config->emailmessage_suppress_activation ===
             true) {
                $sendmail = false;
                $return['error'] = false;
                return $return;
            }
        }
        $query = $this->dbh->prepare("SELECT id, expire FROM {$this->config->table_requests}
         WHERE uid = ? AND type = ?");
        $query->execute(array($uid, $type));
        if ($query->rowCount() > 0) {
            $row = $query->fetch(\PDO::FETCH_ASSOC);
            $expiredate = strtotime($row['expire']);
            $currentdate = strtotime(date("Y-m-d H:i:s"));
            if ($currentdate < $expiredate) {
                $return['message'] = $this->lang["reset_exists"];
                return $return;
            }
            $this->deleteRequest($row['id']);
        }
        if ($type == "activation" && $this->getBaseUser($uid)['isactive'] == 1) {
            $return['message'] = $this->lang["already_activated"];
            return $return;
        }
        $key = $this->getRandomKey(20);
        $expire = date("Y-m-d H:i:s", strtotime($this->config->request_key_expiration));
        $query = $this->dbh->prepare("INSERT INTO {$this->config->table_requests} (uid, rkey,
         expire, type) VALUES (?, ?, ?, ?)");
        if (!$query->execute(array($uid, $key, $expire, $type))) {
            $return['message'] = $this->lang["system_error"] . " #09";
            return $return;
        }
        $request_id = $this->dbh->lastInsertId();
        if ($sendmail === true) {
            $from = $this->config->site_name . "<" . $this->config->site_email . ">";
            $to = "<" . $email . ">";
            if ($type == "activation") {
                $this->codesetMail->Send($from, $to, sprintf($this-
                >lang['email_activation_subject'], $this->config->site_name), sprintf($this-
                >lang['email_activation_body'], $this->config->site_url, $this->config-
                >site_activation_page, $key));
            } else {
                $this->codesetMail->Send($from, $to, sprintf($this-
                >lang['email_reset_subject'], $this->config->site_name), sprintf($this-
                >lang['email_reset_body'], $this->config->site_url, $this->config-
                >site_password_reset_page, $key));
        }
        $return['error'] = false;
        return $return;
    }
    /**
    * Returns request data if key is valid
    * @param string $key
    * @param string $type
    * @return array $return
    */
    public function getRequest($key, $type)
    {
        $return['error'] = true;
        $query = $this->dbh->prepare("SELECT id, uid, expire FROM {$this->config-
         >table_requests} WHERE rkey = ? AND type = ?");
        $query->execute(array($key, $type));
        if ($query->rowCount() === 0) {
            $this->addAttempt();
            $return['message'] = $this->lang[$type."key_incorrect"];
            return $return;
        }
        $row = $query->fetch();
        $expiredate = strtotime($row['expire']);
        $currentdate = strtotime(date("Y-m-d H:i:s"));
        if ($currentdate > $expiredate) {
            $this->addAttempt();
            $this->deleteRequest($row['id']);
            $return['message'] = $this->lang[$type."key_expired"];
            return $return;
        }
        $return['error'] = false;
        $return['id'] = $row['id'];
        $return['uid'] = $row['uid'];
        return $return;
    }
    /**
    * Deletes request from database
    * @param int $id
    * @return boolean
    */
    protected function deleteRequest($id)
    {
        $query = $this->dbh->prepare("DELETE FROM {$this->config->table_requests} WHERE id
         = ?");
        return $query->execute(array($id));
    }
    /**
    * Verifies that a password is valid and respects security requirements
    * @param string $password
    * @return array $return
    */
    protected function validatePassword($password)
    {
        $return['error'] = true;
        if (strlen($password) < (int)$this->config->verify_password_min_length) {
            $return['message'] = $this->lang["password_short"];
            return $return;
        }
        $return['error'] = false;
        return $return;
    }
    /**
    * Verifies that an email is valid
    * @param string $email
    * @return array $return
    */
    protected function validateEmail($email)
    {
        $return['error'] = true;
        if (strlen($email) < (int)$this->config->verify_email_min_length) {
            $return['message'] = $this->lang["email_short"];
            return $return;
        } elseif (strlen($email) > (int)$this->config->verify_email_max_length) {
            $return['message'] = $this->lang["email_long"];
            return $return;
        } elseif (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $return['message'] = $this->lang["email_invalid"];
            return $return;
        }
        if ((int)$this->config->verify_email_use_banlist) {
            $bannedEmails = json_decode(file_get_contents(__DIR__ . "/files/domains.json"));
            if (in_array(strtolower(explode('@', $email)[1]), $bannedEmails)) {
                $return['message'] = $this->lang["email_banned"];
                return $return;
            }
        }
        $return['error'] = false;
        return $return;
    }
    /**
    * Allows a user to reset their password after requesting a reset key.
    * @param string $key
    * @param string $password
    * @param string $repeatpassword
    * @param string $captcha = NULL
    * @return array $return
    */
    public function resetPass($key, $password, $repeatpassword, $captcha = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        if (strlen($key) != 20) {
            $return['message'] = $this->lang["resetkey_invalid"];
            return $return;
        }
        $validatePassword = $this->validatePassword($password);
        if ($validatePassword['error'] == 1) {
            $return['message'] = $validatePassword['message'];
            return $return;
        }
        if ($password !== $repeatpassword) {
            // Passwords don't match
            $return['message'] = $this->lang["newpassword_nomatch"];
            return $return;
        }
        $data = $this->getRequest($key, "reset");
        if ($data['error'] == 1) {
            $return['message'] = $data['message'];
            return $return;
        }
        $user = $this->getBaseUser($data['uid']);
        if (!$user) {
            $this->addAttempt();
            $this->deleteRequest($data['id']);
            $return['message'] = $this->lang["system_error"] . " #11";
            return $return;
        }
        if (password_verify($password, $user['password'])) {
            $this->addAttempt();
            $return['message'] = $this->lang["newpassword_match"];
            return $return;
        }
        $password = $this->getHash($password);
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET password = ?
         WHERE id = ?");
        $query->execute(array($password, $data['uid']));
        if ($query->rowCount() == 0) {
            $return['message'] = $this->lang["system_error"] . " #12";
            return $return;
        }
        $this->deleteRequest($data['id']);
        $return['error'] = false;
        $return['message'] = $this->lang["password_reset"];
        $return['user'] = $user;
        return $return;
    }
    /**
    * Recreates activation email for a given email and sends
    * @param string $email
    * @return array $return
    */
    public function resendActivation($email, $sendmail = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        if ($sendmail == null) {
            $return['message'] = $this->lang['function_disabled'];
            return $return;
        }
        $validateEmail = $this->validateEmail($email);
        if ($validateEmail['error'] == 1) {
            $return['message'] = $validateEmail['message'];
            return $return;
        }
        $query = $this->dbh->prepare("SELECT id FROM {$this->config->table_users} WHERE email
         = ?");
        $query->execute(array($email));
        if ($query->rowCount() == 0) {
            $this->addAttempt();
            $return['message'] = $this->lang["email_incorrect"];
            return $return;
        }
        $row = $query->fetch(\PDO::FETCH_ASSOC);
        if ($this->getBaseUser($row['id'])['isactive'] == 1) {
            $this->addAttempt();
            $return['message'] = $this->lang["already_activated"];
            return $return;
        }
        $addRequest = $this->addRequest($row['id'], $email, "activation", $sendmail);
        if ($addRequest['error'] == 1) {
            $this->addAttempt();
            $return['message'] = $addRequest['message'];
            return $return;
        }
        $return['error'] = false;
        $return['message'] = $this->lang["activation_sent"];
        return $return;
    }
    /**
    * Changes a user's name
    * @param int $uid
    * @param string $newname
    * @return array $return
    */
    public function changeName($uid, $newname)
    {
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET name = ? WHERE
         id = ?");
        if (!$query->execute(array(htmlspecialchars($newname), $uid))) {
            return false;
        }
        return true;
    }
    /**
    * Changes a user's password
    * @param int $uid
    * @param string $currpass
    * @param string $newpass
    * @param string $repeatnewpass
    * @param string $captcha = NULL
    * @return array $return
    */
    public function changePassword($uid, $currpass, $newpass, $repeatnewpass, $captcha =
       null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        $validatePassword = $this->validatePassword($currpass);
        if ($validatePassword['error'] == 1) {
            $this->addAttempt();
            $return['message'] = $validatePassword['message'];
            return $return;
        }
        $validatePassword = $this->validatePassword($newpass);
        if ($validatePassword['error'] == 1) {
            $return['message'] = $validatePassword['message'];
            return $return;
        } elseif ($newpass !== $repeatnewpass) {
            $return['message'] = $this->lang["newpassword_nomatch"];
            return $return;
        }
        $user = $this->getBaseUser($uid);
        if (!$user) {
            $this->addAttempt();
            $return['message'] = $this->lang["system_error"] . " #13";
            return $return;
        }
        if (!password_verify($currpass, $user['password'])) {
            $this->addAttempt();
            $return['message'] = $this->lang["password_incorrect"];
            return $return;
        }
        $newpass = $this->getHash($newpass);
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET password = ?
         WHERE id = ?");
        $query->execute(array($newpass, $uid));
        $return['error'] = false;
        $return['message'] = $this->lang["password_changed"];
        return $return;
    }
    /**
    * Changes a user's email
    * @param int $uid
    * @param string $email
    * @param string $password
    * @param string $captcha = NULL
    * @return array $return
    */
    public function changeEmail($uid, $email, $password, $captcha = null)
    {
        $return['error'] = true;
        $block_status = $this->isBlocked();
        if ($block_status == "verify") {
            if ($this->checkCaptcha($captcha) == false) {
                $return['message'] = $this->lang["user_verify_failed"];
                return $return;
            }
        }
        if ($block_status == "block") {
            $return['message'] = $this->lang["user_blocked"];
            return $return;
        }
        $validateEmail = $this->validateEmail($email);
        if ($validateEmail['error'] == 1) {
            $return['message'] = $validateEmail['message'];
            return $return;
        }
        $validatePassword = $this->validatePassword($password);
        if ($validatePassword['error'] == 1) {
            $return['message'] = $this->lang["password_notvalid"];
            return $return;
        }
        $user = $this->getBaseUser($uid);
        if (!$user) {
            $this->addAttempt();
            $return['message'] = $this->lang["system_error"] . " #14";
            return $return;
        }
        if (!password_verify($password, $user['password'])) {
            $this->addAttempt();
            $return['message'] = $this->lang["password_incorrect"];
            return $return;
        }
        if ($email == $user['email']) {
            $this->addAttempt();
            $return['message'] = $this->lang["newemail_match"];
            return $return;
        }
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET email = ? WHERE
        id = ?");
        $query->execute(array($email, $uid));
        if ($query->rowCount() == 0) {
            $return['message'] = $this->lang["system_error"] . " #15";
            return $return;
        }
        $return['error'] = false;
        $return['message'] = $this->lang["email_changed"];
        return $return;
    }
    public function changeProfile($uid, $path)
    {
        $query = $this->dbh->prepare("UPDATE {$this->config->table_users} SET profile = ?
         WHERE id = ?");
        if (!$query->execute(array($path, $uid))) {
            return false;
        }
        return true;
    }
    /**
    * Informs if a user is locked out
    * @return string
    */
    public function isBlocked()
    {
        $ip = $this->getIp();
        $this->deleteAttempts($ip, false);
        $query = $this->dbh->prepare("SELECT count(*) FROM {$this->config->table_attempts}
         WHERE ip = ?");
        $query->execute(array($ip));
        $attempts = $query->fetchColumn();
        if ($attempts < intval($this->config->attempts_before_verify)) {
            return "allow";
        }
        if ($attempts < intval($this->config->attempts_before_ban)) {
            return "verify";
        }
        return "block";
    }
    /**
     * Verifies a captcha code
     * @param string $captcha
     * @return boolean
     */
    protected function checkCaptcha($captcha)
    {
        return true;
    }
    /**
    * Adds an attempt to database
    * @return boolean
    */
    protected function addAttempt()
    {
        $ip = $this->getIp();
        $attempt_expiredate = date("Y-m-d H:i:s", strtotime($this->config-
         >attack_mitigation_time));
        $query = $this->dbh->prepare("INSERT INTO {$this->config->table_attempts} (ip,
         expiredate) VALUES (?, ?)");
        return $query->execute(array($ip, $attempt_expiredate));
    }
    /**
    * Deletes all attempts for a given IP from database
    * @param string $ip
```

```
        * @param boolean $all = false
    * @return boolean
    */
    protected function deleteAttempts($ip, $all = false)
    {
        if ($all==true) {
            $query = $this->dbh->prepare("DELETE FROM {$this->config->table_attempts} WHERE
             ip = ?");
            return $query->execute(array($ip));
        }
        $query = $this->dbh->prepare("SELECT id, expiredate FROM {$this->config-
         >table_attempts} WHERE ip = ?");
        $query->execute(array($ip));
        while ($row = $query->fetch(\PDO::FETCH_ASSOC)) {
            $expiredate = strtotime($row['expiredate']);
            $currentdate = strtotime(date("Y-m-d H:i:s"));
            if ($currentdate > $expiredate) {
                $queryDel = $this->dbh->prepare("DELETE FROM {$this->config->table_attempts}
                WHERE id = ?");
                $queryDel->execute(array($row['id']));
            }
        }
    }
    /**
    * Returns IP address
    * @return string $ip
    */
    protected function getIp()
    {
        if (isset($_SERVER['HTTP_X_FORWARDED_FOR']) && $_SERVER['HTTP_X_FORWARDED_FOR'] !=
         '') {
            return $_SERVER['HTTP_X_FORWARDED_FOR'];
        } else {
            return $_SERVER['REMOTE_ADDR'];
        }
    }
    /**
    * Returns is user logged in
    * @return boolean
    */
    public function isLogged()
    {
        return (isset($_COOKIE[$this->config->cookie_name]) && $this-
         >checkSession($_COOKIE[$this->config->cookie_name]));
    }
    /**
     * Returns current session hash
     * @return string
     */
    public function getSessionHash()
    {
        return $_COOKIE[$this->config->cookie_name];
    }
    /**
     * Compare user's password with given password
     * @param int $userid
     * @param string $password_for_check
     * @return bool
     */
    public function comparePasswords($userid, $password_for_check)
    {
        $query = $this->dbh->prepare("SELECT password FROM {$this->config->table_users} WHERE
         id = ?");
        $query->execute(array($userid));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        return password_verify($password_for_check, $data['password']);
    }
    /**
     * Return's user ID from database using session hash
     * @param int $userid
     * @param string $password_for_check
     * @return bool
     */
    public function getUIDFromHash($hash)
    {
        $query = $this->dbh->prepare("SELECT uid FROM {$this->config->table_sessions} WHERE
         hash = ?");
        $query->execute(array($hash));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        return $data;
    }
    /**
     * Generates a random string
     * @return string
     */
    public function getRandomKey($length = 20)
    {
        $alphabet = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890';
        $pass = array();
        $alphaLength = strlen($alphabet) - 1;
        for ($i = 0; $i < $length; $i++) {
            $n = rand(0, $alphaLength);
            $pass[] = $alphabet[$n];
        }
        return implode($pass);
    }
    /**
     * Generates a random password
     * @return string
     */
    public function getRandomPassword($length = 2)
    {
        $text = file('https://codeset.co.uk/vendor/phpauth/phpauth/words.txt',
         FILE_IGNORE_NEW_LINES);
        $password = "";
        for ($i=0; $i < $length; $i++) {
            if (strlen($password) > 0) {
                $word = $text[array_rand($text)];
                while(strlen($word) < 3) {
                    $word = $text[array_rand($text)];
                }
                $password = $password."-".$word;
            } else {
                $word = $text[array_rand($text)];
                while(strlen($word) < 3) {
                    $word = $text[array_rand($text)];
                }
                $password = $word;
            }
        }
        return $password;
    }
    private function deleteDir($dirPath) {
        if (! is_dir($dirPath)) {
            return False;
        }
        if (substr($dirPath, strlen($dirPath) - 1, 1) != '/') {
            $dirPath .= '/';
        }
        $files = glob($dirPath . '*', GLOB_MARK);
        foreach ($files as $file) {
            if (is_dir($file)) {
                if (!self::deleteDir($file)) {
                    return False;
                }
            } else {
                if (!unlink($file)) {
                    return False;
                }
            }
        }
        if (!rmdir($dirPath)) {
            return False;
        }
        return True;
    }
}
```

CODESET.CO.UK/PRIVATE_SERVER/INCLUDES/CODESETCOURSE/CLASS.COURSE.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-12-23 19:17:17
 * @Project: codeset.co.uk
 * @File Name: class.classes.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-25 20:26:31
**/
namespace CodeSet;
/**
 * Classes class
 * Manage CodeSet classes
 */
class Classes
{
    private $dbh;
    public function __construct(\PDO $dbh)
    {
        $this->dbh = $dbh;
    }
    /**
     * Retrieves all user's classes
     * @param string $userID
     * @return array $data
     */
    public function getClasses($userID)
    {
        $query = $this->dbh->prepare("SELECT * FROM classes WHERE author_id = ?");
        $query->execute(array($userID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        return $data;
    }
    /**
     * Gets class user is in
     * @param string $userID
     * @param bool $teacher
     * @return int $id
     */
    public function getUsersClass($userID, $teacher = false)
    {
        $query = $this->dbh->prepare("SELECT `class_id` FROM `membership` WHERE `user_id`
         = ?");
        $query->execute(array($userID));
        if ($query->rowCount() == 0) {
            return false;
        }
        if ($teacher) {
            $id = $query->fetchAll(\PDO::FETCH_ASSOC);
        } else {
            $id = $query->fetch(\PDO::FETCH_ASSOC);
        }
        if (!$id) {
            return false;
        }
        if ($teacher) {
            $ids_final = array();
            foreach ($id as $_id) {
                array_push($ids_final, $_id['class_id']);
            }
            return $ids_final;
        }
        return $id['class_id'];
    }
    /**
     * Retrieves class details
     * @param string $classID
     * @return array $data
     */
    public function getClass($classID)
    {
        $query = $this->dbh->prepare("SELECT * FROM classes WHERE id = ?");
        $query->execute(array($classID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        return $data;
    }
    /**
     * Gets a class leaderboard
     * @param string $classID
     * @param object $auth
     * @param int $questions
     * @return array $leaderboard
     */
    public function getClassLeaderboard($classID, $auth, $questions)
    {
        $students = $this->getStudents($classID);
        $ids = array();
        foreach ($students as $student) {
            if ($student['type'] != 1) {
                array_push($ids, intval($student['user_id']));
            }
        }
        $ids = implode(',', $ids);
        $query = $auth->prepare("SELECT * FROM users WHERE id IN ($ids) ORDER BY progress
         DESC");
        $query->execute();
        if ($query->rowCount() == 0) {
            return false;
        }
        $students = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$students) {
            return false;
        }
        $leaderboard = array();
        foreach ($students as $student) {
            $progress = $this->percentScore(array($student['progress']), $questions);
            array_push($leaderboard, array($progress, $student['name'], $student['profile'],
             $student['progress']));
        }
        return $leaderboard;
    }
     /**
     * Returns leaderboard for user's classes based on total score
     * @param string $userID
     * @param object $auth
     * @return array $data
     */
    public function getClassesLeaderboard($userID, $auth, $questions)
    {
        $query = $this->dbh->prepare("SELECT * FROM classes WHERE author_id = ?");
        $query->execute(array($userID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $classes = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$classes) {
            return false;
        }
        $averages = array();
        foreach ($classes as $class) {
            $query = $this->dbh->prepare("SELECT user_id FROM membership WHERE class_id = ?
             AND type != 1");
            $query->execute(array($class['id']));
            if ($query->rowCount() != 0) {
                $students = $query->fetchAll(\PDO::FETCH_ASSOC);
                if (!$students) {
                    return false;
                }
                $scores = array();
                $total = 0;
                foreach ($students as $student) {
                    $query = $auth->prepare("SELECT progress FROM users WHERE id = ?");
                    $query->execute(array($student['user_id']));
                    if ($query->rowCount() == 0) {
                        return false;
                    }
                    $progress = $query->fetch(\PDO::FETCH_ASSOC);
                    if (!$progress) {
                        return false;
                    }
                    array_push($scores, $progress['progress']);
                    $total += $questions;
                }
                array_push($averages, array($this->percentScore($scores, $total),
                $class['name']));
            } else {
                array_push($averages, array(0, $class['name']));
            }
        }
        usort($averages, function($b, $a) {
            return $a[0] - $b[0];
        });
        return $averages;
    }
    /**
     * Retrieves class's students
     * @param string $classID
     * @return array $data
     */
    public function getStudents($classID)
    {
        $query = $this->dbh->prepare("SELECT * FROM membership WHERE class_id = ? AND type =
         0");
        $query->execute(array($classID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        return $data;
    }
    /**
     * Adds a user to a class
     * @param string $userID
     * @param string $classID
     * @return bool
     */
    public function addUser($classID, $userID, $type)
    {
        $query = $this->dbh->prepare("INSERT INTO membership (class_id, user_id, type) VALUES
         (?,?,?)");
        if (!$query->execute(array($classID, $userID, $type))) {
            return false;
        }
        return true;
    }
    /**
     * Deletes a user from a class
     * @param string $userID
     * @param string $classID
     * @return bool
     */
    public function deleteUser($classID, $userID)
    {
        $query = $this->dbh->prepare("DELETE FROM membership WHERE class_id = ? AND user_id =
         ?");
        if (!$query->execute(array($classID, $userID))) {
            return false;
        }
        return true;
    }
    /**
     * Creates a CodeSet class
     * @param string $name
     * @param string $user
     * @param int $remember
     * @param string $captcha = NULL
     * @return bool
     */
    public function create($name, $userID)
    {
        $query = $this->dbh->prepare("SELECT id FROM classes");
        $query->execute();
        $occuredIDs = $query->fetchAll(\PDO::FETCH_ASSOC);
        $query = $this->dbh->prepare("INSERT INTO classes (id, name, author_id) VALUES
         (?,?,?)");
        if (!$query->execute(array($this->generateID(20, $occuredIDs), $name, $userID))) {
            return false;
        }
        $query = $this->dbh->prepare("SELECT id FROM classes WHERE name = ? AND author_id
         = ?");
        $query->execute(array($name, $userID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $classID = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$classID) {
            return false;
        }
        if (!$this->addUser($classID['id'], $userID, 1)) {
            return false;
        }
        return true;
    }
    /**
     * Deletes a CodeSet class
     * @param string $classID
     * @param string $userID
     * @return bool
     */
    public function delete($classID, $userID)
    {
        if ($this->checkUserClassPermission($classID, $userID)) {
            $query = $this->dbh->prepare("DELETE FROM classes WHERE id = ?");
            if (!$query->execute(array($classID))) {
                return "Could not delete class. Please try again";
            } else {
                $query = $this->dbh->prepare("DELETE FROM membership WHERE class_id = ?");
                if (!$query->execute(array($classID))) {
                    return "Could not remove users from class";
                }
                return true;
            }
        } else {
            return "You do not have permission to delete that class";
        }
    }
    /**
     * Renames a CodeSet class
     * @param string $newName
     * @param string $classID
     * @param string $userID
     * @return bool
     */
    public function rename($newName, $classID, $userID)
    {
        if ($this->checkUserClassPermission($classID, $userID)) {
            $query = $this->dbh->prepare("UPDATE classes SET name = ? WHERE id = ?");
            if (!$query->execute(array(htmlspecialchars($newName), $classID))) {
                return "Could not rename class. Please try again";
            }
            return true;
        } else {
            return "You do not have permission to rename that class";
        }
    }
    /**
     * Composes subject and body for student help
     * @param string $teacherEmail
     * @param string $studentName
     * @param string $className
     * @param int $questionID
     * @param string $code
     * @param string $instructions
     * @param string $comments
     * @return bool
     */
    public function composeStudentToTeacherEmail($teacherEmail, $studentName, $studentEmail,
      $className, $questionID, $code, $instructions, $comments)
    {
        $subject = "$studentName is stuck on question $questionID";
        $body = "
            <h2>$studentName, from the class $className, is stuck on question $questionID</
             h2>
            <br>
            <p><b>Question:</b> ".nl2br($instructions)."</p>
            <p><b>Student's Comments:</b> ".nl2br($comments)."</p>
            <p><b>$studentName's code:</b> <br><code>".nl2br($code)."</code></p>
            <br>
            <p>You can reply to $studentName here: <a href='mailto:$studentEmail?
             Subject=Codeset - Question $questionID Help'>$studentEmail</a></p>
             ";
        return ['subject' => $subject, 'body' => $body, 'email' => $teacherEmail];
    }
    /**
     * Composes subject and body for student help
     * @param string $teacherEmail
     * @param string $studentName
     * @param string $className
     * @param int $questionID
     * @param string $code
     * @param string $instructions
     * @param string $comments
     * @return bool
     */
    public function composeStudentLoginEmail($studentName, $studentEmail, $username,
       $password)
    {
        $subject = "Your CodeSet Account";
        $body = "
            <h2>Hello $studentName</h2>
            <br>
            <p>Your teacher has created your CodeSet account. You can log in with the details
             below:</p>
            <p><b>Username:</b> <code>$username</code></p>
            <p><b>Password:</b> <code>$password</code></p>
            <br>
            <p>You can log into your account <a href='https://codeset.co.uk/public_html/
             pages/public/signin.php'>here</a></p>
        ";
        return ['subject' => $subject, 'body' => $body, 'email' => $studentEmail];
    }
    /**
     * Checks if a user has permission for a class
     * @param string $classID
     * @param string $userID
     * @return bool
     */
    private function checkUserClassPermission($classID, $userID)
    {
        $query = $this->dbh->prepare("SELECT author_id FROM classes WHERE id = ?");
        $query->execute(array($classID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $data = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$data) {
            return false;
        }
        if ($data['author_id'] == $userID) {
            return true;
        }
    }
    /**
     * Calculates percentage complete for all students in a class
     * @param array $scores
     * @param int $max
     * @return int
     */
    private function percentScore($scores, $max)
    {
        $scoreTotal = 0;
        foreach ($scores as $score) {
            $scoreTotal += $score;
        }
        return round(($scoreTotal/$max)*100);
    }
    /**
     * Generates random ID for class
     * @param int $length
     * @param $array $occured
     * @return int
     */
    private function generateID($length, $occured)
    {
        if (function_exists("random_bytes")) {
            $bytes = random_bytes(ceil($length / 2));
        } elseif (function_exists("openssl_random_pseudo_bytes")) {
            $bytes = openssl_random_pseudo_bytes(ceil($length / 2));
        } else {
            return false;
        }
        $id = substr(bin2hex($bytes), 0, $length);
        if (in_array($id, $occured)) {
            return $this->generateID($length, $occured);
        } else {
            return $id;
        }
    }
}
```

CODESET.CO.UK/PRIVATE_SERVER/INCLUDES/CODESETNEWS/CLASS.NEWS.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2016-11-26 12:45:30
 * @Project: codeset.co.uk
 * @File Name: class.news.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-29 19:13:58
**/
namespace CodeSet;
/**
 * News class
 * Class news system
 */
class News{

    private $dbh;
    public function __construct(\PDO $dbh) {
        $this->dbh = $dbh;
    }
    public function setClassDeadline($classID, $point, $expiration, $maxQuestions) {
        $return = ['error' => True, 'message' => ''];
        if ($point > $maxQuestions) {
            $return['message'] = "Please enter a valid question number";
            return $return;
        }
        $now = new \DateTime("now");
        $date = explode('-', $expiration);
        $expiration = new \DateTime();
        $expiration->setDate($date[0], $date[1], $date[2]);
        $expiration->setTime(23, 59, 59);
        if ($now > $expiration) {
            $return['message'] = "Please enter a valid date";
            return $return;
        }
        $query = $this->dbh->prepare("INSERT INTO deadlines (`class_id`, `point`,
        `expiration`, `date`) VALUES (?,?,?,?)");
        if (!$query->execute(array($classID, $point, $expiration->format('Y-m-d H:i:s'),
         $now->format('Y-m-d')))) {
            $return['message'] = "Could not add deadline. Please try again.";
        }
        $return['error'] = False;
        return $return;
    }
    public function getClassDeadlines($classID) {
        $query = $this->dbh->prepare("SELECT * FROM deadlines WHERE class_id = ? ORDER BY
         expiration DESC");
        $query->execute(array(strval($classID)));
        if ($query->rowCount() == 0) {
            return false;
        }
        $deadlines = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$deadlines) {
            return false;
        }
        return $deadlines;
    }
    public function getDeadlineInfo($deadlineID) {
        $query = $this->dbh->prepare("SELECT * FROM deadlines WHERE id = ?");
        $query->execute(array($deadlineID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $deadline = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$deadline) {
            return false;
        }
        return $deadline;
    }
    public function getActiveDeadlines($classID) {
        $deadlines = $this->getClassDeadlines($classID);

        if ($deadlines == False) {
            return False;
        }
        $activeDeadlines = array();
        foreach ($deadlines as $deadline) {
            $date = new \DateTime($deadline['expiration']);

            if ($date >= new \DateTime()) {
                array_push($activeDeadlines, $deadline);
            }
        }
        return array_reverse($activeDeadlines);
    }
    public function deleteClassDeadline($deadlineID) {
        $query = $this->dbh->prepare("DELETE FROM deadlines WHERE id = ?");
        if (!$query->execute(array($deadlineID))) {
            return False;
        }
        return True;
    }
    public function getClassIdFromDeadlineId($deadlineID) {
        $query = $this->dbh->prepare("SELECT class_id FROM deadlines WHERE id = ?");
        $query->execute(array($deadlineID));
        if ($query->rowCount() == 0) {
            return false;
        }
        $classID = $query->fetch(\PDO::FETCH_ASSOC);
        if (!$classID) {
            return false;
        }
        return $classID['class_id'];
    }
    public function addClassNews($classID, $title, $body) {
        $previousNews = $this->getClassNews($classID);
        foreach ($previousNews as $previousNew) {
            if (in_array($body, $previousNew)) {
                return False;
            }
        }
        $now = new \DateTime();
        $query = $this->dbh->prepare("INSERT INTO news (`class_id`, `title`, `body`, `date`)
         VALUES (?,?,?,?)");
        if (!$query->execute(array($classID, htmlspecialchars($title),
         htmlspecialchars($body), $now->format('Y-m-d')))) {
            return False;
        }
        return True;
    }
    public function getClassNews($classID) {
        $query = $this->dbh->prepare("SELECT * FROM news WHERE class_id = ? ORDER BY id DESC
         LIMIT 10");
        $query->execute(array(strval($classID)));
        if ($query->rowCount() == 0) {
            return false;
        }
        $news = $query->fetchAll(\PDO::FETCH_ASSOC);
        if (!$news) {
            return false;
        }
        return $news;
    }
    public function deleteClassNews($newsID) {
        $query = $this->dbh->prepare("DELETE FROM news WHERE id = ?");
        if (!$query->execute(array($newsID))) {
            return False;
        }
        return True;
    }
}
```

CODESET.CO.UK/PRIVATE_SERVER/INCLUDES/CODESETPROFILEUPLOAD/HANDLER.PHP

```
<?php
/**
 * @Author: Ben
 * @Date: 2017-04-12 23:55:59
 * @Project: codeset.co.uk
 * @File Name: handler.php
 * @Last Modified by:   Ben
 * @Last Modified time: 2017-08-29 22:52:04
**/
include '../../../config.php';

$uid = $auth->getUIDFromHash($auth->getSessionHash())['uid'];

$user = $auth->getUser($uid);

$target_dir = "../../../public_html/assets/images/profiles/";
$extension = end(explode(".", basename($_FILES["account-profile-file"]["name"])));
$target_file = $target_dir . "user-" . $uid . "." . strtolower($extension);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Check if image file is an image
if(isset($_POST["account-profile-change-submit"])) {
    $check = getimagesize($_FILES["account-profile-file"]["tmp_name"]);
    if($check !== false) {
        $uploadOk = 1;
    } else {
        $uploadOk = 0;
    }
}

if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    $uploadOk = 0;
}

// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    header('Location: /public_html/pages/private/account.php');
// Try to upload file
} else {
    if ($user['profile'] != "user-default.png" && file_exists($target_dir .
$user['profile'])) {
        unlink($target_dir . $user['profile']);
    }

    if (move_uploaded_file($_FILES["account-profile-file"]["tmp_name"], $target_file)) {

        // Update path in database
        $auth->changeProfile($uid, "user-" . $uid . "." . strtolower($extension));

        // Rotate image if needed
        $image = imagecreatefromstring(file_get_contents($target_file));
        $exif = exif_read_data($target_file);

        if(isset($exif['Orientation'])) {
            switch($exif['Orientation']) {
                case 8:
                    $image = imagerotate($image,90,0);
                    break;
                case 3:
                    $image = imagerotate($image,180,0);
                    break;
                case 6:
                    $image = imagerotate($image,-90,0);
                    break;
                default:
                    break;
            }

            switch(strtolower($extension)) {
                    case 'jpg':
                        imagejpeg($image, $target_file, 90);
                        break;
                    case 'jpeg':
                        imagejpeg($image, $target_file, 90);
                        break;
                    case 'png':
                        imagepng($image, $target_file);
                        break;
                    case 'gif':
                        imagegif($image, $target_file);
                        break;
                    default:
                        // Unsupported format
                        break;
                }
        }
        imagedestroy($image);

        // Crop image to square
        function square_crop($src_image, $dest_image, $thumb_size = 200, $jpg_quality = 90) {

            // Get dimensions of existing image
```

```
$image = getimagesize($src_image);

// Check for valid dimensions
if( $image[0] <= 0 || $image[1] <= 0 ) return false;

// Determine format from MIME-Type
$image['format'] = strtolower(preg_replace('/^.*?\//', '', $image['mime']));

// Import image
switch( $image['format'] ) {
    case 'jpg':
    case 'jpeg':
        $image_data = imagecreatefromjpeg($src_image);
        break;
    case 'png':
        $image_data = imagecreatefrompng($src_image);
        break;
    case 'gif':
        $image_data = imagecreatefromgif($src_image);
        break;
    default:
        // Unsupported format
        return false;
        break;
}

// Verify import
if( $image_data == false ) return false;

// Calculate measurements
if( $image[0] > $image[1] ) {
    // For landscape images
    $x_offset = ($image[0] - $image[1]) / 2;
    $y_offset = 0;
    $square_size = $image[0] - ($x_offset * 2);
} else {
    // For portrait and square images
    $x_offset = 0;
    $y_offset = ($image[1] - $image[0]) / 2;
    $square_size = $image[1] - ($y_offset * 2);
}

// Resize and crop
$canvas = imagecreatetruecolor($thumb_size, $thumb_size);
if( imagecopyresampled( $canvas, $image_data, 0, 0, $x_offset, $y_offset,
 $thumb_size, $thumb_size, $square_size, $square_size)) {
    // Create thumbnail
    switch( strtolower(preg_replace('/^.*\./', '', $dest_image)) ) {
        case 'jpg':
        case 'jpeg':
            return imagejpeg($canvas, $dest_image, $jpg_quality);
            break;
        case 'png':
            return imagepng($canvas, $dest_image);
            break;
        case 'gif':
            return imagegif($canvas, $dest_image);
            break;
        default:
            // Unsupported format
            return false;
            break;
    }

} else {
    imagedestroy($image);
    imagedestroy($canvas);
    return false;
```

```
            }
            imagedestroy($image);
            imagedestroy($canvas);
            return True;
        }

        square_crop($target_file, $target_file);

    }
}

header('Location: /public_html/pages/private/account?redirect=True.php');
```

TESTING

Before the site can be used by teachers and students in lessons, the system needs to undergo testing. This will be done by checking each feature of the site to make sure they work.

Testing Inputs

Test No.	Description	Input	Expected Outcome	Actual Outcome	Changes Needed
1	Entering data into name text box in create new user form	**Valid Data:** “Test name”	Account created	“Account created”	None
		Boundary Data: “%\&*£”	Account created	“Account created”	None
		Erroneous Data: “”	Asks user to input a name	“Please enter a name”	None
2	Entering data into email text box in create new user form	**Valid Data:** “test@test.com”	Account created	“Account created”	None
		Boundary Data: “%\&*£”	Tells user address is invalid	“Email address is invalid”	None
		Erroneous Data: “”	Tells user address is invalid	“Email address is too short”	None
3	Entering data into password text box in create new user form	**Valid Data:** “password”	Account created	“Account created”	None
		Boundary Data: “%\&*£”	Account created	“Account created”	None
		Erroneous Data: “”	Tells user password is invalid	“Password is too short”	None
4	Entering data into repeat password text box in create new user form	**Valid Data:** “password”	Account created	“Account created”	None
		Boundary Data: “%\&*£”	Account created	“Account created”	None
		Erroneous Data: “”	Tells user the passwords do not match or password is invalid	“Passwords do not match”	None
5	Entering data into email text box in sign in form	**Valid Data:** “test@test.com”	User taken to Dashboard	Redirected to dashboard	None
		Boundary Data: ‘ " or “”=" '	Tells user email is invalid or user does not exist	“Email address / password are invalid”	None
		Erroneous Data: “”	Tells user the email address is invalid	“Email address / password are invalid”	None
6	Entering data into password text box in sign in form	**Valid Data:** “password”	User taken to Dashboard	Redirected to dashboard	None
		Boundary Data: ‘ " or “”=" '	Tells user password is invalid	“Email address / password are invalid”	None
		Erroneous Data: “”	Tells user the password is invalid	“Email address / password are invalid”	None

Test No.	Description	Input	Expected Outcome	Actual Outcome	Changes Needed
7	Entering data into email text box in request password reset email form	**Valid Data:** “test@test.com”	Reset request sent to user’s email	“Password reset request sent to email address” And email in user’s inbox	None
		Boundary Data: “%\&*£”	Tells user email is invalid or user does not exist	“Email address is invalid"	None
		Erroneous Data: “”	Tells user the email address is invalid	“Email address is invalid"	None
8	Entering data into reset key text box in reset password form	**Valid Data:** “n6v0VbVfwNY2cEADIcMT”	Password reset and user taken to dashboard	Redirected to dashboard	None
		Boundary Data: “%\&*£” or old key	Tells user key is invalid or has expired	“Reset key is invalid” or “Reset key has expired”	None
		Erroneous Data: “”	Tells user the key is invalid	“Reset key is invalid”	None
9	Entering data into password text box in reset password form	**Valid Data:** “password”	Password reset and user taken to dashboard	Redirected to dashboard	None
		Boundary Data: “%\&*£” or old password	Tells user the password is invalid or is has been used before	“Password is invalid” or “New password is the same as the old password”	None
		Erroneous Data: “”	Tells user the password is invalid	“Password is invalid”	None
10	Entering data into repeat password key text box in reset password form	**Valid Data:** “password”	Password reset and user taken to dashboard	Redirected to dashboard	None
		Boundary Data: “%\&*£” or different password to one in text box above	Tells user password is invalid or does not match	“New passwords do not match”	None
		Erroneous Data: “”	Tells user the password is invalid	“New passwords do not match”	None
11	Entering data into class name text box in create new class form	**Valid Data:** “Class name”	Class created and page reloaded, showing new class tile	Page reloaded and new class tile displayed	None
		Boundary Data: “%\&*£”	Class created and page reloaded, showing new class tile	Page reloaded and new class tile displayed	None
		Erroneous Data: “”	Tells the user to enter a class name	"Please enter a name for the class”	None
12	Entering data into student name text box in create new	**Valid Data:** “Student name”	Account created and page reloaded showing new student in row	Page reloaded and student displayed	None
		Boundary Data: “%\&*£”	Account created and page reloaded showing new student in row	Page reloaded and student displayed	None

Test No.	Description	Input	Expected Outcome	Actual Outcome	Changes Needed
	student account form	**Erroneous Data:** “”	Tells the user to enter a name for the student	“Please enter a name and email for the class”	“class” needs to be changed to "student"
13	Entering data into student email text box in create new student account form	**Valid Data:** “test@test.com”	Account created and page reloaded showing new student in row	Page reloaded and student displayed	None
		Boundary Data: “%\&*£” or email of account already created	Tells user email address is invalid	“Email address is invalid” or “That email address is already in use”	None
		Erroneous Data: “”	Tells the user to enter an email for the student	“Please enter a name and email for the class”	“class” needs to be changed to "student"
14	Entering data into class name text box in change class name form	**Valid Data:** “New class name”	Class’s name changed and page reloaded	Redirected to dashboard	None
		Boundary Data: “%\&*£”	Class’s name changed and page reloaded	Redirected to dashboard	None
		Erroneous Data: “”	Tells user the to enter a name for the class	“Please enter a name for the class”	None
15	Entering data into question to reach text box in create deadline form	**Valid Data:** “25”	Deadline created, page reloaded and deadline displayed	Page reloaded and deadline shown	None
		Boundary Data: “-1”, “76”	Browser stops user from inputing any numbers that are not between 1 and 75, the last question	“Value must be greater than or equal to 1”, “Value must be less than or equal to 75”	None
		Erroneous Data: “”	Tells user they must enter a question to reach	“Question to reach and date cannot be empty”	None
16	Entering data into date text box in create deadline form	**Valid Data:** “2017-10-09”	Deadline created, page reloaded and deadline displayed	Page reloaded and deadline shown	None
		Boundary Data: “1917-10-09”, “2017-10-32”, “09/10/2017”, “%\&*£”	Stops any dates already gone from being inputted, and tells user if date is invalid	“Please enter a valid date”, if day is greater than last day in month it rolls into next month (e.g. 2017-10-32 becomes 2017-11-1), “Please enter a valid date”	None
		Erroneous Data: “”	Tells user they must enter a date	“Question to reach and date cannot be empty”	None
17	Change profile picture form button	**Valid Data:** JPG image	Profile picture uploaded, page reloaded and new profile picture displayed	Page reloaded and new picture shown	None
		Boundary Data: DOC file	Profile picture does not change	Profile picture did not change	None

Test No.	Description	Input	Expected Outcome	Actual Outcome	Changes Needed
18	Entering data into name in change name form	**Valid Data:** “New user name”	User’s name changed and page reloaded displaying changes	Page reloaded and new name shown	None
		Boundary Data: “%\&*£”	User’s name changed and page reloaded displaying changes	Page reloaded and new name shown	None
		Erroneous Data: “”	Tells user they must enter a new name	Page reloaded, no error message shown but name not changed	May be more user friendly if and error message is displayed
19	Entering data into current password text box in change password form on account page	**Valid Data:** “password”	Password changed and page reloaded	Page reloaded	None
		Boundary Data: String that is not current password	Tells the user password is incorrect	Pop up closes and “Password changed” is displayed, although the password did not actually change	Should display error message telling the user password incorrect
		Erroneous Data: “”	Tells user they must enter a password	“Could not change password. Please try again”	May be more user friendly if user was told to enter a password
20	Entering data into new password text box in change password form on account page	**Valid Data:** “password”	Password changed and page reloaded	Page reloaded and password changed	None
		Boundary Data: “%\&*£”	Password changed and page reloaded	Page reloaded and password changed	None
		Erroneous Data: “”	Tells user to enter new password	“Could not change password. Please try again”	May be more user friendly if user was told to enter a new password
21	Entering data into repeat new password text box in change password form on account page	**Valid Data:** “password”	Password changed and page reloaded	Page reloaded and password changed	None
		Boundary Data: String that does not match	Tells user passwords do not match	Pop up closes and “Password changed” is displayed, although the password did not actually change	Should display error message telling the user passwords do not match
		Erroneous Data: “”	Tells user to re-enter password	“Could not change password. Please try again”	May be more user friendly if user was told to re-enter their new password

Testing Course System & Editor

Test No.	Description	Code / Method	Expected Outcome	Actual Outcome	Changes Needed
22	Import blocked libraries	“import os”	Block os module	“>> Module 'os' is blocked”	None
		“import sys”	Block sys module	“>> Module ‘sys' is blocked”	None
		“import shutil”	Block shutil module	“>> Module ‘shutil’ is blocked”	None
		“import subprocess”	Block subprocess module	“>> Module ‘subprocess’ is blocked”	None

23	Test memory limit	def recursion(i):
		return recursion(i*i)
		print(recursion(2))
			Process should stop as it exceeds the memory limit	line 2, in recursion
				return recursion(i*i)
				MemoryError
					None

24	Test all questions to make sure all variations of code solutions to questions are accepted	I tested each question with different variations, such as with no spaces between variable names and equal signs (e.g. number = 5 and number=5)	The different variations of the code should be marked as correct	Most accepted variations in the code, however a few marked some variations as just below the percentage it needed to be to be marked as correct	Correct some questions so that they accept variations of code
25	Changing question ID in address bar	id=0	Page handles the error as no question 0 exists	“Could not get question. Please try again” then displays blank question page -See screenshot-	Redirect user if question does not exits
		id=100 or any question number not yet reached	Page redirects user to first uncompleted question	Page loads question after the last one completed	None
		id=string	Page handles the error as no question ‘string’ exists	“Could not get question. Please try again” then displays blank question page -See screenshot-	Redirect user if question does not exits
26	Test request help from teacher feature	Normal question	Email is sent to teacher	Pop up box displayed, email sent successfully	None
		Set id=0 in address bar	Error displayed and email not sent	Pop up box displayed, email sent containing no question data	Redirect user if question does not exits
27	Test reset code feature	Normal question	Code is reset	Page reloaded, displaying original question code	None
		Set id=0 in address bar	Error handled with no exception displayed	Page reloaded, no error message	None
		Reach 25% complete	News item created and displayed on dashboard	News item created and shown to all users in class	None

Test No.	Description	Code / Method	Expected Outcome	Actual Outcome	Changes Needed
28	Test milestone news feature	Reach 50% complete	News item created and displayed on dashboard	News item created and shown to all users in class	None
		Reach 75% complete	News item created and displayed on dashboard	News item created and shown to all users in class	None
		Reach 100% complete	News item created and displayed on dashboard	News item created and shown to all users in class	None

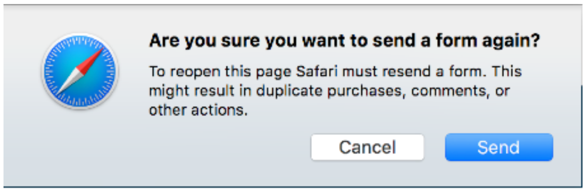
Testing Site In Different Browsers

For most of the development of my site I have been mainly using Safari on a Mac as my main browser, as well as occasionally Chrome on Windows. However there are many different browsers that the site may be used on, and each browser displays websites differently. To make sure the site displays and works properly on each browser I need to test them.

Test No.	OS	Browser	Conclusion	Changes Needed
29	Mac OS (*Version 10.13*)	Chrome	No issues	None
		Firefox	Name overlaps profile picture, course bar on dashboard is lower down section, but does not affect use, main blue colour is slightly lighter. “Submit” buttons say “Submit Query”, which may confuse the user and make the buttons larger, changing the layout	Move name to left, change Submit button text
		Opera	No issues	None
30	Windows 10	Chrome	“Show lessons” button overlapped “Hide lessons” button	Hide Show/Hide buttons when other is displayed
		Microsoft Edge	“Submit” button say “Submit Query”, which may confuse user and means the buttons are cut off because they are longer. Also, “Create Class” button icon is displayed at a lower resolution	Change Submit button text
		Internet Explorer	Login page does not display properly, “Create Class” button icon and site logo on the navigation bar are displayed at a lower resolution, Python editor did not work - the terminal did not display the output	Fix login page and Python editor

Testing Page Refreshes After Submitting Form

During development the pages should have been made so that when a form is submitted and the page reloads, if the user then refreshes the page the browser does not then ask them if they want to send the form again. This needs to be tested, as it means data could be duplicated or deleted from the databases.

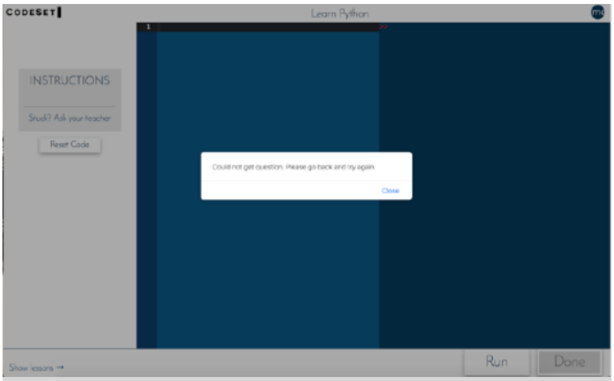
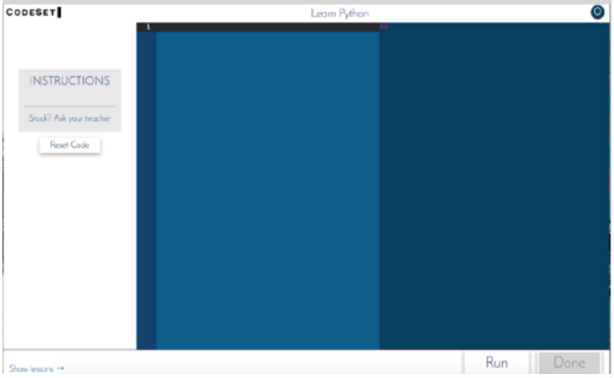
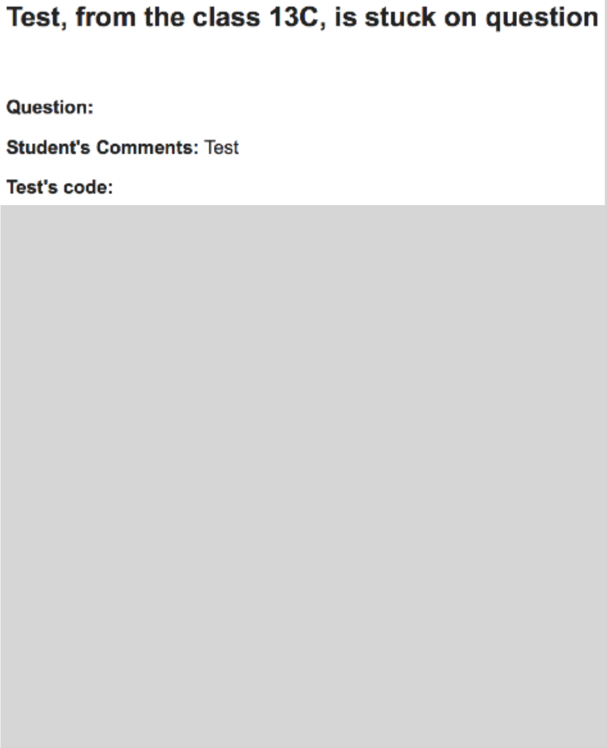
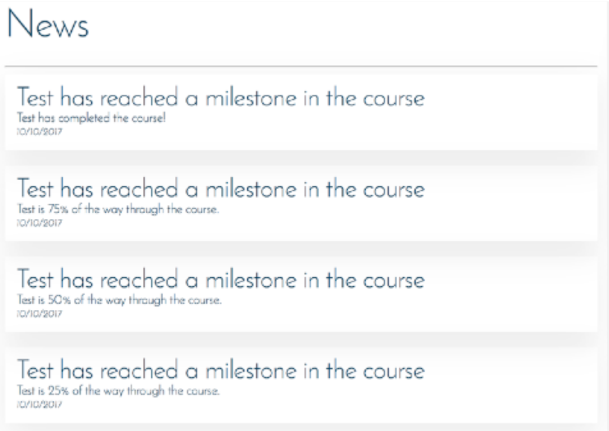
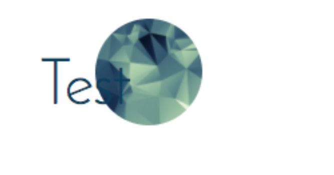
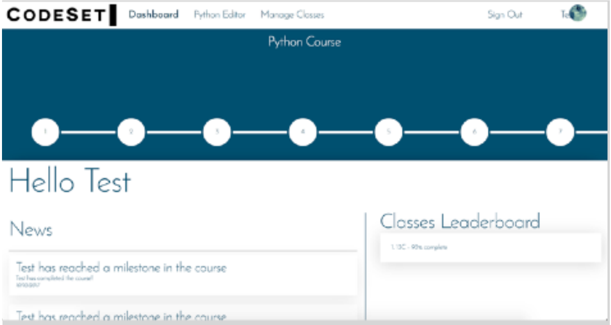
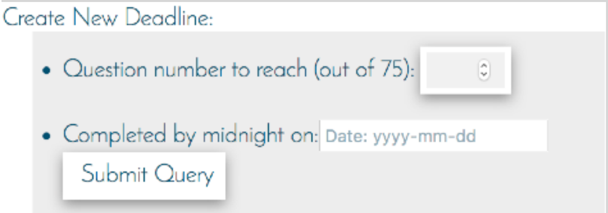


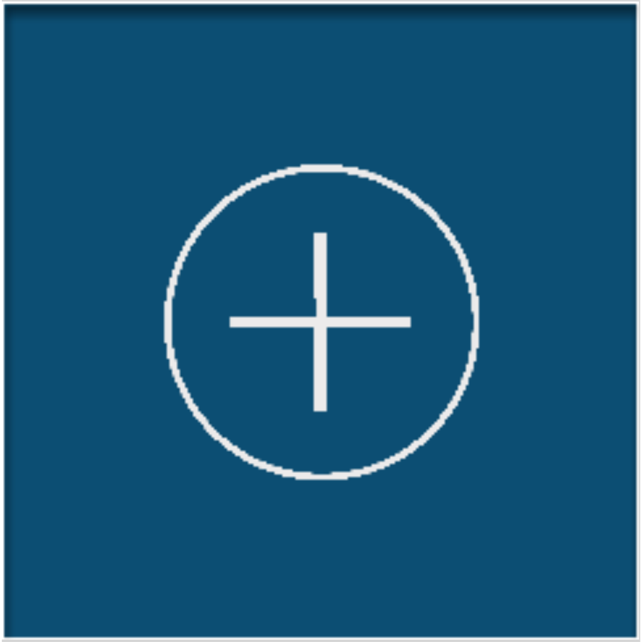
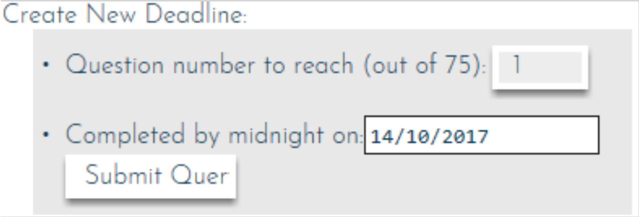
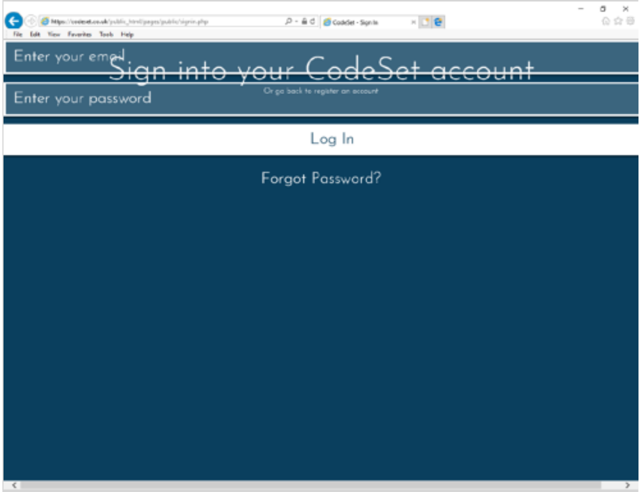
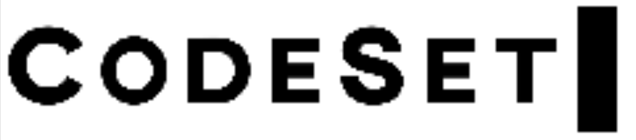
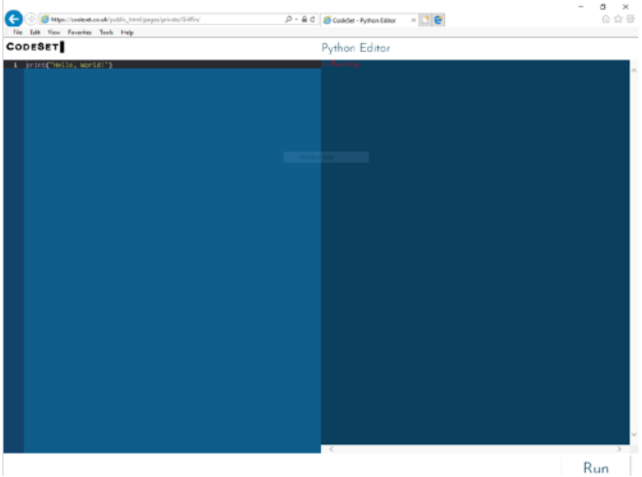
Are you sure you want to send a form again?

To reopen this page Safari must resend a form. This might result in duplicate purchases, comments, or other actions.

Test No.	Page	Form	Action	Outcome	Changes Needed
31	index.php	Create user	Submit create user form	Fail	Stop user from being able to resubmit form
32	public_html/pages/private/classes/	New class	Submit new class's name form	Pass	None
33	public_html/pages/private/classes/index.php	Delete class	Submit class id as a hidden input in delete form	Pass	None
34	public_html/pages/private/classes/class.php	Rename class	Submit new class name in rename class form	Pass	None
35	public_html/pages/private/classes/class.php	Delete class	Submit class id as a hidden input in delete form	Pass	None
36	public_html/pages/private/classes/class.php	Create student	Submit student name and email in new student form	Pass	None
37	public_html/pages/private/classes/class.php	Delete student	Submit student id as a hidden input in delete form	Fail	Stop user from being able to resubmit a form
38	public_html/pages/private/classes/class.php	New deadline	Submit deadline question and date in new deadline form	Fail	Stop user from being able to resubmit form
39	public_html/pages/private/classes/class.php	Delete deadline	Submit deadline id as hidden input in delete deadline form	Fail	Stop user from being able to resubmit form
40	public_html/pages/private/classes/deadline.php	Delete deadline	Submit deadline id as hidden input in delete deadline form	Pass	None
41	public_html/pages/private/Course/index.php	Request teacher help	Submit question information, student code and comment to email teacher form	Fail	Stop user from being able to resubmit form
42	public_html/pages/private/account.php	Upload profile picture	Upload user's file	Pass	None
43	public_html/pages/private/account.php	Change name	Submit user's new name to change name form	Pass	None
44	public_html/pages/private/account.php	Change password	Submit user's old password, new password and repeated password to change password form	Fail	Stop user from being able to resubmit form

Evidence

Test No.	Screenshot	Test No.	Screenshot
25	 Could not get question. Please go back and try again. 	26	 **Test, from the class 13C, is stuck on question** **Question:** **Student's Comments:** Test **Test's code:**
28	 News Test has reached a milestone in the course Test has completed the course! 10/10/2017 Test has reached a milestone in the course Test is 75% of the way through the course. 10/10/2017 Test has reached a milestone in the course Test is 50% of the way through the course. 10/10/2017 Test has reached a milestone in the course Test is 25% of the way through the course. 10/10/2017	29	Chrome:  Firefox:  Hello Test News Test has reached a milestone in the course Test has completed the course! Test has reached a milestone in the course Classes Leaderboard 1. 13C - 90% complete  Create New Deadline: - Question number to reach (out of 75): - Completed by midnight on: Date: yyyy-mm-dd Submit Query

Test No.	Screenshot
25	Edge plus icon:  Edge “Submit” button:  Create New Deadline: • Question number to reach (out of 75): 1 • Completed by midnight on: 14/10/2017 Submit Quer IE login page:  Enter your email Sign into your CodeSet account Or go back to register an account Enter your password Log In Forgot Password? IE CodeSet logo:  CODESET IE Python Editor:  CODESET Python Editor `print("Hello, world!")` Run

EVALUATION

While making my website, I had to overcome many hurdles. The most difficult part was sandboxing the Python environment, so that users could not run dangerous code or access other parts of the server. I went through many methods of doing this, each with their advantages and disadvantages, before settling on the current implementation. Many of the methods I tried only allowed one input to be passed to the program, which I used to send the program location. This would have meant the Python editor would not accept inputs, reducing the effectiveness of the Python course as inputs are a crucial part. While my current method does accept inputs, it is not perfect. As the inputs have to be submitted by the user before running the code, the user may be asked for more inputs than necessary and not always in the correct order, as inputs in functions, if statements and loops can be requested at any point in the program, as well as being requested multiple times. As my current method reads from the top to the bottom searching for inputs, it does not follow the order of the program. I looked at ways to make Python send a request back to page whenever it needs an input, but I could not find many documented approaches. One way that may have worked could have been to make a Python interpreter in PHP that could follow the program order, sending requests back whenever it comes across an input, however this would have been a lot of work and may not be possible. I could also have used a Node.js, which allows JavaScript to be run on the front end and back end. This would allow AJAX pipes to be kept open so that data could be transferred between the front and back ends multiple times. Also, to make the Python environment more secure, a better method if I had the resources would be to have a separate server for running the user's python code. This means the code could be run by a different user with less permissions and would keep user code separate from the site code.

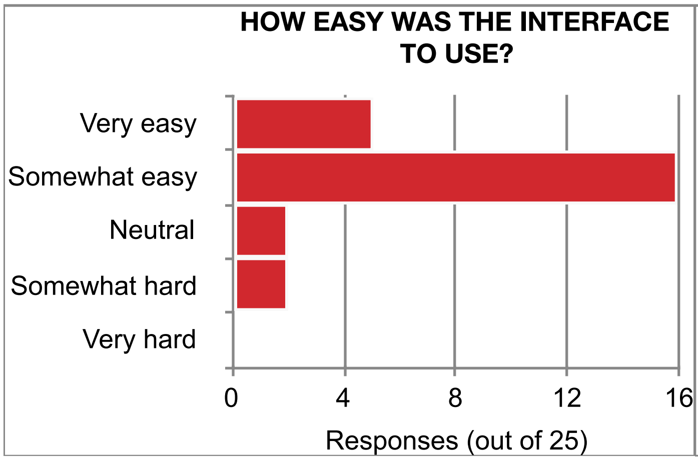
Another issue with the site is to do with the code checking algorithm. The idea of checking the output works well, although some questions do not produce an output so could be modified to print something to improve the verification that the code is correct. However, the other part to the checking algorithm - comparing the user's code to variations of the correct code that I have written - does not always work well. This is because, while Python is very strict when it comes to syntax and code layout compared to other languages, there are still many different ways to write a program, especially in the more complex questions towards the end of the course. This means that the current algorithm marks code that works and is correct as incorrect because I did not think of that variation when writing the question solutions. A possible way of improving this algorithm could be by writing individual Python checks for each question, which could include the user's code file then check that the variable and function names are assigned to the correct values, as well as make sure the other aspects of the code are correct. This would make the checking algorithm less finicky so the user does not get frustrated because their code is correct but is not being accepted, however this method would require a lot more work for each question and would make the system less modifiable as each question's individual Python checking code would have to be changed.

During the design and development process, I gave my client regular updates on the progress and to get their input. This meant that once the development was finished, he already knew how to use it and there were no unexpected surprises. He has been using the site for a few weeks, and has given me feedback on the site - he likes the modern and minimalistic design, and he said the system to manage the classes and students was useful. He also liked the Python editor without the questions, which he said allowed his students to experiment with Python on their own without having to install any software on their own computer. He said a possible improvement would be more detailed feedback on the students' code to help them correct any problems.

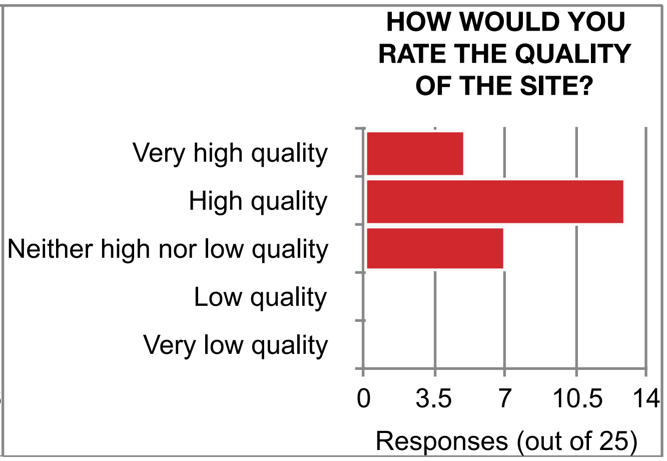
Finally, I think that while the site looks modern and minimalistic, which I like, in some places the design of the site has a negative impact on the user experience - the look is sometimes prioritised over how easy it is to use. This means that people who are new to the site may find it difficult to use and navigate, and users may get frustrated because a part of the site is difficult to use or find because making it obvious would affect the design.

Testing Against Objectives

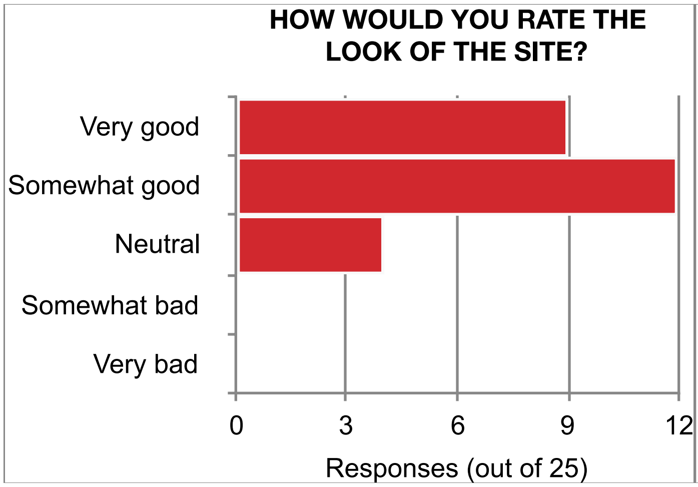
Objective	Met	Evaluation
1. The system needs to be accessible for students and teachers from school and at home, keeping their progress update across all devices the system is used on.	Yes	As the system is online and is always running, the system can be accessed from anywhere with an internet connection at any time. The user's information and progress is stored in databases, and updated as soon as it changes. The user's code is stored in files on the server instead of their computer, so as soon as the user clicks Run, the code is saved, making it available to the user from any computer.
2. The Python environment must be secure and isolated from the rest of the system, so that unsafe code cannot cause damage if it is run.	Yes	The user is not able to import libraries that give them the ability to cause damage to the server or other users. Each user has their own folder which acts as a cell - they do not have permission to access anything outside of their folder. This could be improved by running user code on a separate server.
3. The interactive course must cover the basics of Python so that the students can gain a solid understanding of the programming language, and learn at their own pace.	Yes	There are 75 questions, which means the course content is very spread out and new topics are introduced gradually, allowing the user to learn them. As the user can access the course at any time and can revisit completed lessons they are able to learn at their own pace. The lessons cover the basics of Python, giving students a foundation they can then build on.
4. The proposed interactive course must have clear explanations and questions, with sample code and hints to help the students.	Yes	All questions are made up of three parts: title, introduction and instructions. The title gives the user a short summary of what topic is being covered in the question. The introduction explains a topic and in most questions gives examples of code to help the user with the question. The instructions tell the user what to do, often giving names for any variables or functions they need to use, hints, and for more complicated questions the task is broken down into smaller parts so the user doesn't feel overwhelmed.
5. The system must allow students to get help from their teacher if they get stuck at home or in lesson.	Yes	On each question the user can request help from their teacher by clicking on a button and entering comments to explain to their teacher what they are stuck with. Teachers receive the question instructions, the student's code and their comments. The teacher can then email the student a response or if they are in the lesson they can help them in person.



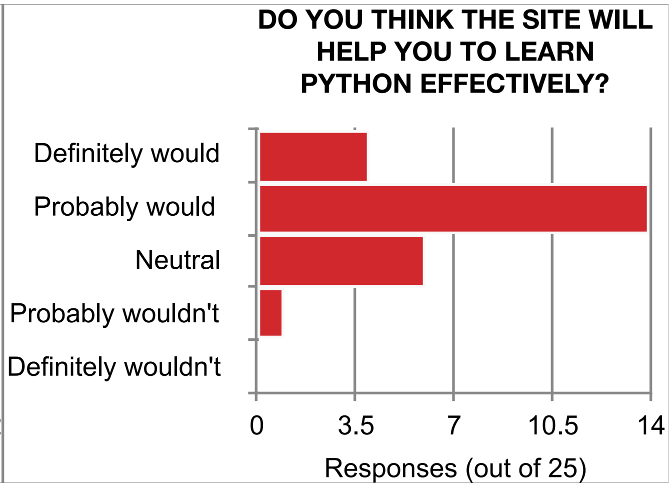
HOW EASY WAS THE INTERFACE TO USE?



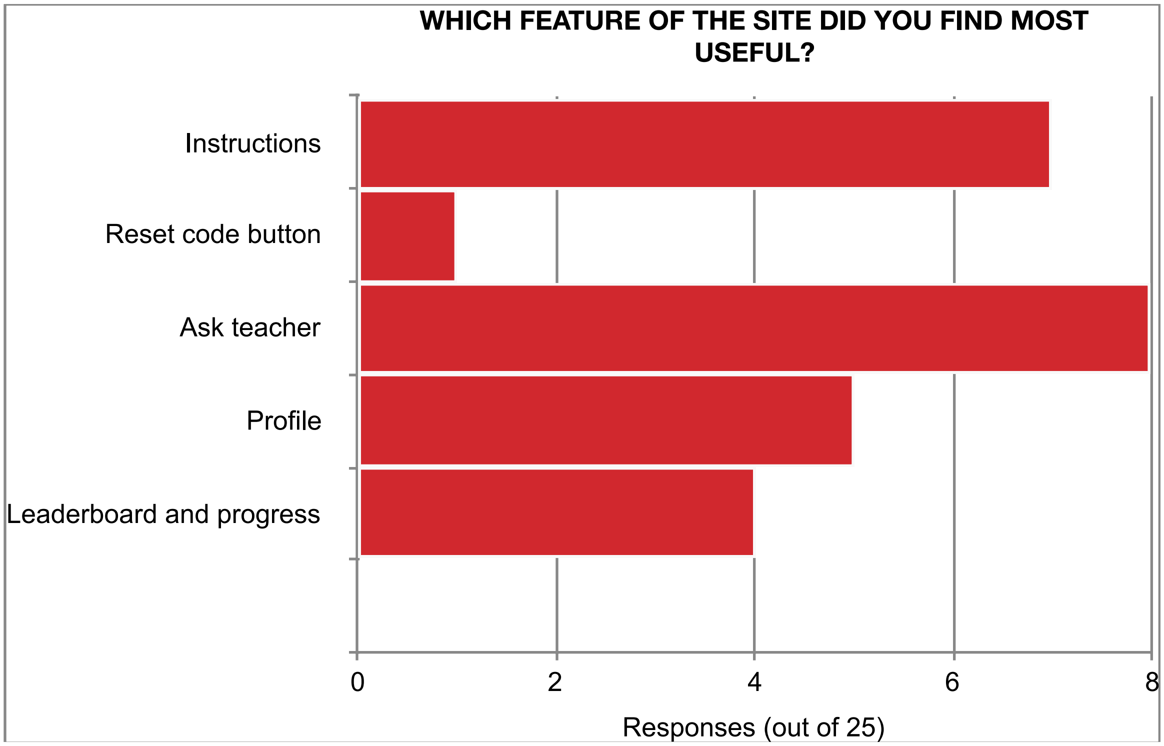
HOW WOULD YOU RATE THE QUALITY OF THE SITE?



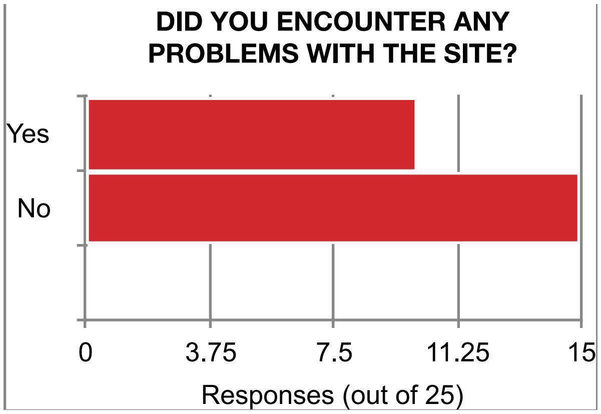
HOW WOULD YOU RATE THE LOOK OF THE SITE?



DO YOU THINK THE SITE WILL HELP YOU TO LEARN PYTHON EFFECTIVELY?



WHICH FEATURE OF THE SITE DID YOU FIND MOST USEFUL?



Objective	Met	Evaluation
6. The interactive system must allow students to be able to track their own progress, and compare their progress against other students in the class.	Yes	Progress is measured by the number of questions the user has completed. They can see how far they have reached on the Dashboard, with completed questions displayed as filled in circles. Also, on the right side of the Dashboard is a leaderboard, ranking users by percentage complete. Finally, in the News section of the Dashboard whenever a user reaches milestones in the course, the rest of the class can see.
7. Teachers must be able to track the progress of students against others in the class and set them deadlines, monitoring their progress in reaching these deadlines.	Yes	On the Dashboard, teachers can see their classes ranked by the percentage of the course the class's students have completed. This is also shown again in a more visual representation on the Classes page, showing bars that are filled up as the students progress. On each individual class page, the same bars are used, this time to show each student's progress. Teachers are able to set Deadlines on for each class on the class's page. When the teacher clicks on a deadline, they are taken to the Deadlines page, where they can see a list of students who have and haven't yet reached the Deadline, as well as bars at the top of the page showing the students' progress towards reaching the Deadline.
8. The users of the system must be able to easily update their details and preferences without access to the user database.	Yes	On the Account page, users can easily get an overview of the information stored about them. Users are able to change their name, password and profile picture using the interface on the Account page.
9. The interface of the system must be easy and intuitive to use so that users do not need to be trained on how to use it. The aesthetics of the system must also be taken into account during its design.	Partly	The site is very aesthetically pleasing, however in some places the aesthetics affect how easy and intuitive the site is to use. When designing the site, I should have created a prototype using placeholder data to test how user friendly the site is to use.

Objective	Met	Evaluation
10. Teachers must easily be able to create classes and add students to them.	Yes	On the Classes page, teachers can create new classes simply by clicking on the New Class tile, then inputting a name and pressing create. The class will then be displayed as a tile on the grid, where the user can then click on it to be taken to the individual class page. On the class page, students are displayed in a list. At the bottom of this list is a form, where teachers can enter the student's name and email, then press Submit to create the student's account. This could be improved as adding a large class of students can be very tedious - a better way may be to allow the user to upload a CSV file.

User Feedback

To get user feedback and to test my program to see if it worked as expected, a Year 10 GCSE Computing class used my site during a lesson and for homework. They found a number of bugs and suggested improvements, detailed below:

IF YOU ANSWERED YES, PLEASE DESCRIBE:

- *"It gave me the answers on each question"*
- *"Capital letters stopped me getting the correct result"*
- *"Sometimes the code has to be overly specific in order to be correct and also the text overlaps on the task bar at the top"*
- *"I couldn't pass a question but it was fixed"*
- *"Questions weren't working but they did get fixed fast"*
- *"Some of the instructions were a little bit confusing"*
- *"At home it cropped the thing to say what question you were on and the teacher help line"*
- *"Some times the words overlapped and I could not read it"*
- *"There was an error surpassing one question, however it was fixed"*

IN YOUR OWN WORDS, WHAT ARE THE THINGS YOU WOULD MOST LIKE TO IMPROVE?

- *"Make it a bit clearer on the questions"*
- *"Harder questions"*
- *"Add a hint button"*
- *"There were a few repeated sentences (typos)"*
- *"Tutorials so you know how to use the site"*
- *"I think the question number should be displayed more clearly while you're working, rather than having to go back to the home page"*
- *"Have a clear exit button from the actual programming tasks"*

- *“The login button could be made in a more clear way. The website could be easier to find*”
- No improvements needed (6 people)

This feedback has helped me to evaluate my project. They have suggested some useful improvements, and more importantly they have found a number of bugs that affected use of the site. The students gave me true criticism of my project, which helped me to identify where improvements were needed.

Improvements

Based on user feedback and testing, I identified the changes were needed (see Documented Design for more detail):

- Fix “Show Lessons” button in Google Chrome
- Allow teachers to choose lessons if they haven’t yet completed previous questions
- Add link to bottom of the email containing student information
- Add student’s email address at bottom of ask teacher for help email
- Change generated passwords to dictionary words
- Add a larger space between sign out and user’s name
- Show question number on Course page
- Fix some questions
- One user was able to get the answers to the questions
- Show numbers on progress bars
- Students found a way to get around method to stop users skipping questions
- Increase font size on Account page
- Stop user from being able to resubmit forms on some pages
- Redirect user to first uncompleted question if they change the question id in the address bar
- Show error messages on Account page
- Change “class” to “student” in error message on class page