QUEEN'S UNIVERSITY BELFAST

'Queen ProCode'

A dissertation submitted in partial fulfilment of

The requirement for the degree of

MASTER OF SCIENCE in Software Development

in

The Queen's University of Belfast

By

'Ryan Kieran Smyth'

'23/05/2021'

Content Page

Chapter 1: Problem Specification

1.1 Original Problem Specification - November 2020 ………………………………………………………………… 6

1.2 Three Levels of Functionality ……………………………………………………………………………………6

1.3 Additional Functions…………………………………………………………………………………………7

1.4 New Project Specification – February 2021…………………………………………………………………7

1.5 Three Levels of Functionality ……………………………………………………………………………………8

1.6 Additional Functions…………………………………………………………………………………………8

1.7 Justification of Changes…………………………………………………………………………………………9

1.8 A further note on the target audience……………………………………………………………………..9

Chapter 2: Proposed solution and justification of the development model

2.1 Proposed Solution ……………………………………………………………………………………………11

2.2 Database ……………………………………………………………………………………………………11

2.3 Procedural Programming ……………………………………………………………………………………11

2.4 Justification of the Development Model ………………………………………………………………………11

2.5 The Development Model Description ……………………………………………………………………………12

2.5.1 The Incremental Approach………………………………………………………………………………12

2.6 A note on the implementation stage of the incremental approach ………………………………………………..13

2.7 The iterative approach ………………………………………..…………………………………………..13

2.8 Justification of the development model ……………………………………………………………………14

2.8,1 Advantages of this Approach ………………………………………………………………………………14

2.8.2 Disadvantages of this Approach……………………………………………………………………………15

Chapter 3: Requirement's analysis and specification

3.1 Overview ……………………………………………………………………………16

3.2.1 Use Case Diagram ……………………………………………………………16

3.2 Functional Requirements……………………………………………………………18

3.2.1 Tutor Functions ……………………………………………………………18

3.2.2 Student Functions……………………………………………………………21

3.3 Non-Functional Requirements ………………………………………………………22

3.3.1 Overview of Non-Functional Requirements…………………………………22

3.3.2 Non-Functional Requirements ………………………………………………22

3.4 The Data Model ……………………………………………………………………24

3.4.1 Overview……………………………………………………………………24

3.4.2 MySQL ……………………………………………………………………24

3.4.3 Exploring Regional Databases ………………………………………………24

3.4.4 Normalisation ………………………………………………………………25

Chapter 4: Design

Part 1 – User Interface Design ………………………………………………………29

4.1 Introduction ………………………………………………………………………29

4.2 Learnability and Simplicity …………………………………………………………30

4.3 Robustness …………………………………………………………………………31

4.4 Observability ………………………………………………………………………32

4.5 Recoverability ………………………………………………………………………32

4.7 The Feedback System ………………………………………………………………32

Part 2 – Software System Design

4.8 Overview 34

4.9 Architectural Design 35

4.9.1 Description 35

4.9.2 Advantages..36

4.9.3 Disadvantages 36

4.10 Three Layer Architecture Implementation 36

4.11 MVC 36

4.12 MVC Implementation 38

4.13 Component Design 39

Chapter 5: Implementation

5.1 Overview 40

5.2 Implementation 40

5.2.1 Feedback Component 40

5.2.4 General Feedback Component..45

5.2.7 Student Registration 47

5.2.9 Content Completion..48

5.3 Testing Strategies 53

5.3.1 Justification of Testing Strategies 54

Chapter 6: Evaluation and Conclusion

6.1 Overview ………..53

6.2 Success of the Project……..53

6.3 Development Languages ……………………………………………………………………………………………………54

6.4 Shortcomings of this Project. ………………………………………………………………………………………………54

1. Security and SQL Injection ……………………………………………………………………………………………54
2. Chat Component ……55
3. Feedback Representation and the UI ……………………………………………………………………………55
4. The use of Restful APIS …………………………………………………………………………………………………55
5. Increased Useability ……………………………………………………………………………………………………55

Bibliography……….60

Appendix ………58

Tutor Components…….58

Student Components………102

Tutor Component Testing………………………………………………………………………………………………………131

Student Component Testing……………………………………………………………………………………………………141

Security Testing ……145

.

1.1 Original Problem Specification - November 2020

How will I solve the problem?

The student demonstrated that they understood the problem, described a sensible approach to solving it and specified the project's objectives.

I am attempting to create a mobile app that will allow students of the EEECS school to access their year one Programming Module. Incoming students will face many difficulties in their first year of university. This software will help them make sure that they pass one of their core modules.

The software's functionality will allow students to monitor their progress, track their attendance, and complete their modules. This application needs to be user-friendly with a modern front-end design.

There should be an analysis of the problem domain and an indication of the project's proposed interface design and content. You could, for example, invent a fictional user/s (the more realistic, the better) and use this to explain how the user would use the application and the features which they might need or want to use.

I know from this point that I will need to use the technology from my Web Development Module. These technologies will most certainly include HTML, CSS, JavaScript, PHP and MySQL as the core technologies. . I plan to create a regional database for this application through MySQL as I am confident in using this technology.

There will be three levels of functionality. This structure is necessary to make a secure and functional application; The levels of functionality include an admin user or users and tutors and students. I plan to create the application through Android Studios to expand my learning of relevant and valuable technologies. I am unsure, however, if I will be able to create a sophisticated web equivalent in the time allocated.

1.2 Three Levels of Functionality

Tutor Mobile Application

1. Tutors will add and set up content concerning the module.

2. Tutors will be able to create classes to register for.

3. Tutors will communicate with students publicly concerning the module content.

4. Tutors will be able to make announcements.

Student Mobile Application

1. Students will be able to view and complete their module content.

2. Students will record their attendance concerning lectures and other learning sessions.

3. Students will be able to give feedback on their complete content.

4. Students will communicate with each other concerning the module content.

1.3 Additional Functions

The concept of this app requires a heavy focus on security. Security must be of paramount importance and something that I am interested in and focused on developing. I want to create advanced login security through PHP. At this point, I am not fully aware of the different security levels are necessary for this type of application.

1.4 New Project Specification – February 2021

How will I solve the problem?

The student demonstrated that they understood the problem, described a sensible approach to solving it and specified the project's objectives.

I am trying to create a Web-based browser application that will allow the students of QUB's EEECS school to access their year one programming module. Incoming students will face many difficulties in their first year of university. This software will help them make sure that they pass one of their core modules.

The software's functionality will allow students to check their progress, track their attendance and request help from their tutors. This application needs to be a user-friendly, multi-platform application with a modern front-end design.

There should be an analysis of the problem domain and an indication of the project's proposed interface design and content. You could, for example, invent a fictional user/s (the more realistic, the better) and use this to explain how the users would use the application and the features which they might need or want to use.

The content of this Web Application will be the following technologies. HTML 5, CSS, JavaScript, PHP, MySQL, Ajax, JQuery. These will be used as a full-stack technology to create a functional Web Application with a modern and clean user interface.

There will exist two different types of accounts that will make up this application. There will be a tutor level which will provide much more control and over the content of the application. The other level will be a student-level with limited functionality compared to the tutor level of the application.

1.5 Three Levels of Functionality

Tutor Level

1. Tutors will add and set up content concerning the module.

2. Tutors will create classes for the students to attend.

3. Tutors will communicate with students publicly and privately concerning the module content.

4. Tutors will be able to make announcements.

5. Tutors will be checking the students' progress concerning class attendance and module feedback on the content.

Student Level

1. Students will be able to view and complete their module content.

2. Students will record their attendance concerning lectures and other learning sessions.

3. Students will be able to provide feedback on the content.

4. Students will communicate with each other concerning the module content.

5. Student will be able to register for their classes that the tutor has set.

1.6 Additional Functions

The concept of this app requires a heavy focus on security. Security must be of the paramount importance of the non-functional requirements. Security is something that I am focused on developing. I want to create advanced login security through PHP. At this point, I am not fully aware of the different security levels are necessary for this type of application.

Other added functions would be concerning the students' statistics. These functions will make the Web Application all that more valuable from a teaching standpoint. Queens Canvas has a similar function that can track which students are online during live teaching. My application won't be sophisticated, but the students will have to enter a randomly generated code linked to that class to confirm their registration.

1.7 Justification of Changes

Several significant changes need to be acknowledged and explained. In the original project specification, I have stated that I planned to develop this project using Android Studio Code so that the application would have been available through a smartphone. However, as I progressed through many exercises concerning Android Studio Code, I decided to change to a browser-based Web Application. Here are the primary reasons why I decided to change to a browser-based full-stack approach.

1. The lack of resources during the Covid Pandemic.
2. The lack of free resources online.
3. The lack of knowledge and experience of the software that I was using.
4. Confidence in traditional full-stack web development.
5. A lack of peers to deliberate with to expand my understanding of the software.
6. The architectural design process was easier to implement using a browser-based web application.
7. I have used Sources code in the earlier development of a project of a similar nature and functionality.

1.8 A Further Note on the Target Audience

The Target Audience for this application will be the EEECS staff and students. Both staff and students should track the students' progress using this application; Three main components will exist to track this progress. These components will be in the form of attendance, performance and completion. Therefore, both the student and the tutor should have a good experience using this technology.

Chapter 2 Proposed Solution and the justification of the development model.

2. 1 Proposed Solution

My proposed solution is to develop a reliable web application that will capture student data. This data can then be easily manipulated to feedback a clear message to how the student performs in the labs set for them. This application is implemented using the traditional web stack technologies and some other minor technologies such as JQuery and Ajax. The

2.2 Database

The application will be linked to a database that will collect all the relevant data for performance, completion and attendance, as well as other data for other secondary functions. This database will be a regional database that will be developed through phpMyAdmin. The most appropriate choice for the database will work well with the PHP scripting language. See more about the data model in chapter 3.

2.3 Procedural Programming

Procedural programming is a programming paradigm that uses a linear or top-down approach. It relies on procedures or subroutines to perform computations.

Procedural Programming has many advantages. The main advantage of this programming approach is that each activity's function will stand alone. Therefore, any sudden changes to the software will be easier to implement and change. The activities functionality is written within the activity. Therefore if the customer required another function, this would be much easier to implement as no other activities would need to be changed. The program flow is easily tracked with the correct annotation of the code. Furthermore, Procedural Programming was how we were taught during the Web Development module. This style is what I am most confident in implementing.

2.4 The development model

This section will outline the development model I used to develop this web application. I initially believed that I would be using an Agile Model. But to be more specific, I am using the Incremental Model.

[1] The definition of Incremental Development is as follows "An Approach to software development where the software is delivered and deployed in increments."

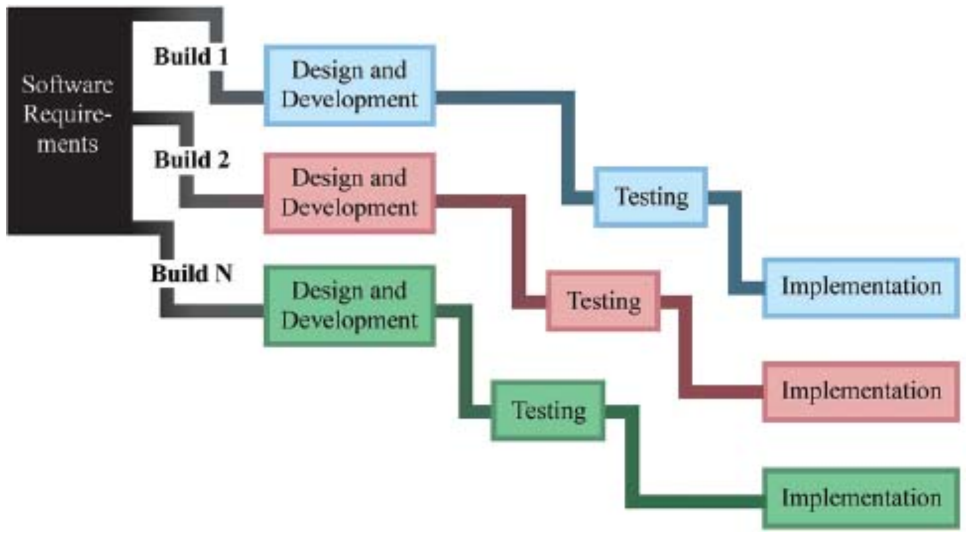
However, upon further research of iterative models, I discovered that my development model was more of a hybrid or the "*Iterative Model*" and an "*Incremental Model*". [2]Rod Stephens describes and Iterative Model as the following "Initially provides all three features at a low (but unstable) fidelity. Later iterations provide higher fidelity until all the features are provided with complete fidelity.

On its face, that does not make a lot of sense. In the next section, I will explain what I mean; further, that will hopefully make this statement clear.

2.5 The Development Model Description

2.5.1 The Incremental Approach

In Figure 2.5.2 you will see the type of development model that I wanted to use to get this web application successfully developed.



[1] Sommerville Ian, 2016, "Software Engineering", 2nd Edition, Pearson Education Limited, 1-292-09613-6
[2] Stephens Rod, 2015, "Beginning Software Engineering", 1st Edition, John Wiley & Sons, Inc

Build 1 – The Database using MySQL.

After completing the Software Requirements stage, I started on the Design and Development of a regional database that could easily interact with PHP to pass the necessary data needed for the application's functionality. This increment was undoubtedly the shortest increment necessary for the development of this application.

The most extended phase of the first build was the Design and the Development, with the Testing phase being short and of ease. However, the Implementation stage of Build 1 was not completed until the Design and Development of Build Two was completed. This is since the core software used in Build 2 was PHP which is inextricably linked with MySQL.

I could go far as to say that the Build 1 implementation phase and the Build 2 Design and Development stage are the same stages.

Build 2 – Functionality and Security using PHP, MySQL and JavaScript

Build 2 is the phase that will focus on developing the website's functionality. Build 2 will be the longest of the three increments. The primary focus of this functionality is to capture and manipulate the data for the user. The Design and Development section of Build 2 is the longest and the most significant phase of the entire development process.

PHP will supply the core software of the functionality for this web application.

Build 3 – User Interface using CSS and JavaScript

Build 3 is the final build of the web application. This build will focus on designing the User Interface and displaying the data in a specific format that will make the web application look modern and professional. The Design and Development phase of Build 3, like the last Builds, will be the longest, with the Testing phase and implementation phase being of similar length.

2.6 A Note on the Implementation Stage of the Incremental Approach

This project is a university project that means that the only feedback that I will receive is from my supervisor, given to them on a two-week basis. At the end of an Increment, I will display the entire level of the system to my supervisor to receive feedback to seek clarity before I formally end the increment.

2.7 The Iterative approach

I use the Iterative approach concerning many Uses cases inside an Increment. When I develop an individual Use Case, I will start with prototyping, testing, analysing, and refining the Use Case.

There is no guarantee that the formula set out in the Software Requirements phase will supply the correct set of objects to solve the problem at hand. Therefore, the Iterative approach supplies a flexible, less stressful approach in any immediate changes. Since this is an informal process, the Use Case will evolve depending on when I can get feedback or support.

[3]Furthermore, the Iterative approach suits a web-based browser application as each Use Case is acting independently of any other Use Cases that exist within the same increment. An immediate change concerning the functionality of the specific Use Case will not be felt widely throughout the application. An iterative approach may lead to the discovery and need for a Use Case that was not considered in the Software Requirement phase; this means that the Use Case can be easily implemented.

This Iterative approach, however, was not applied during Build 1. MySQL as software has a helpful user interface that allows me to quickly develop the database tables of each Use Case without much time spent on Testing or Implementation. MySQL is reliable.

2. 8 Justification

In the introduction to this section, I outlined that the Development Model that I had adopted was a hybrid between an Incremental and Iterative Model. In this section, I will justify this claim. The Incremental approach is a more general approach adopted during each level needed to build a web application.

Within each increment, there will be many Use Cases that have to be complete for the level functionality to be successful. Each Use Case will adopt an iterative approach whereby the functionality. An Iterative is only adopted for an independent Use Case within an Increment.

There will be two levels of the break down that

2.8.1 Advantages of this Approach

1. The time cost of implementing requirement changes is reduced. The amount of analysis and change in the documentation is minimised compared to a Waterfall approach.
2.
3. It is easier to get feedback from my supervisor on the development work that has been completed. My supervisor can comment on the software demonstrations and observe what work is completed.
4. Earlier delivery and deployment of helpful software to my supervisor is possible even if all the functionality has not been included or developed yet. Much more value is gained from this than adopting the Waterfall Model for development.
5. This model supplies flexibility. Since this is a Web Application whereby each Use Case acts independently, there is room for developing other functionality sections if I need a break from developing one Use Case.

[3] Douglas Bell, 2005, 'Software Engineering for Students', 4th Edition, Edition, Pearson Education Limited, 0-321-26127-5

2.8.2 Disadvantages of this Approach

1. The system structure tends to degrade as increments are applied. The regular and unexpected change will lead to messy code as new functionality is added. It becomes increasingly time-consuming to add new features to the system that has not been outlined in the software requirements.
2. Each phase of an iteration is rigid and do not overlap.
3. Problems may arise about system architecture because not all requirements are gathered upfront for the entire software life cycle. For example, I may have to add a new table into the database that will need to be used appropriately within the code.

<u>Requirements Analysis and Specification</u>

<u>3.1 Overview</u>

[4]Before developing software, it is essential to understand and document the customer's exact requirements. Throughout the MSC Software Development teaching side, our tutors tried to stress that you should only develop what the customer wants. Developers often fall into the trap of implementing something that the customer has not asked for and therefore does not meet the requirements. When software is developed with a contract involved, a dissatisfied client could lead to disputes and even legal battles.

Therefore, a crucial role is played by documentation of precise requirements, something that cannot be overstated. There is rarely a situation whereby the requirements from a customer are provided in the form of a single document. Often, experienced team members gather the requirements analysis and specification by visiting the customer. For this dissertation, there is no customer. My supervisor will act as the client and scrutinise my work.

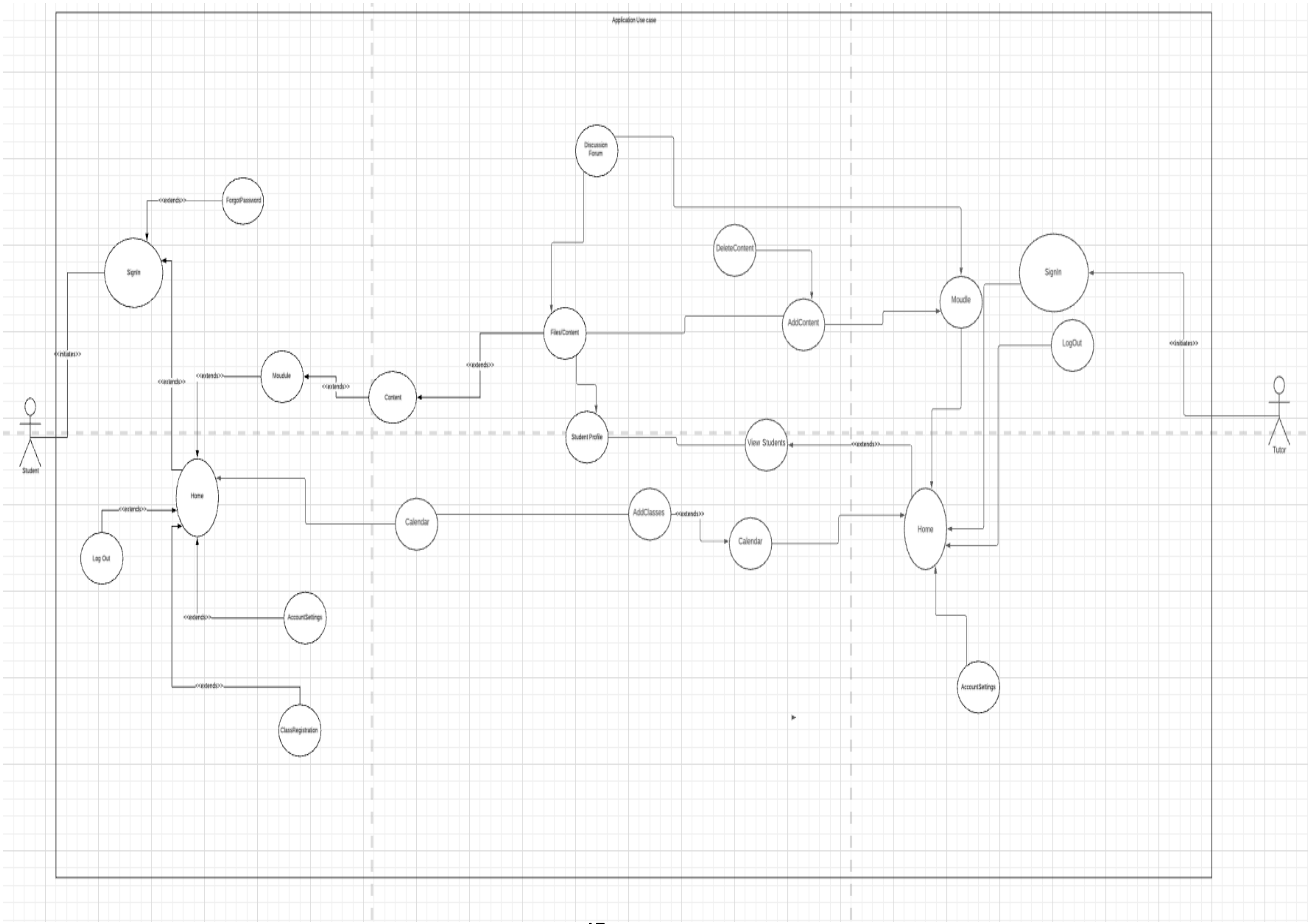
<u>The goal of the requirements analysis and specification phase is to clearly understand the customer requirements and to systematically organise the requirements into a specification document.</u>

This type of documentation is popularly known as requirement elicitation. An analyst starts requirement gathers information that could be useful to the development of the system.

<u>3.2.1 Use Case Diagram</u>

I designed this Use Case Diagram to illustrate the system that I wanted to create. Each Use Case would consist of a modular set of components, each with multiple functions. These Use Cases would form the name of the files that contains the code of the application. For example, the Use Case that would shows the module content would be named tutor_module.php if it was being accessed by the Tutor. I found this diagram useful as it acted as a check list of the Use Cases that I needed to implement.

[4]Mall Rajib, 2013, 'Fundamentals of Software Engineering, 3rd Edition, Asoke K. Ghosh, PHI Learning, 978-81-203-3819-7


Application Use case
Student
Tutor
SignIn
ForgotPassword
<<extends>>
<<initiates>>
Home
Moudule
Content
Files/Content
Discussion Forum
DeleteContent
AddContent
Moudle
SignIn
LogOut
Student Profile
View Students
Calendar
AddClasses
Calendar
Home
Log Out
AccountSettings
ClassRegistration
AccountSettings

Functional Requirements 3.2

Tutor Functions List 3.2.1

This list will outline the list of the Tutor level functionalities that will be implemented to meet the requirements specification.

Name	Login Page 1.1
Description	Authorisation of the user that will navigate them to their home page.
Pre-condition	The user is on the login page, and the authorisation is successful.
Post-condition	The user is on their home page and can access all the functionality.
Inputs	User inputs legitimate details
Outputs	The user is navigated to the home page.

Name	Create Announcement 1.2
Description	The tutor will create an announcement for all students to view.
Pre-condition	The tutor selects the added announcement on the home page.
Post-condition	The tutor provides a title and a description for the announcement.
Inputs	The tutor adds the relevant details and clicks send.
Outputs	All users, including the tutor, can see the announcement.

Name	Create Week 1.3
Description	The tutor will create a week for content to be added.
Pre-condition	The tutor selects the add week function on the module page.
Post-condition	The tutor creates a week that has not already been created.
Inputs	The tutor adds a week and a title for that week.
Outputs	All users, including the tutor, can see the week.

Name	Create Module Content 1.4
Description	The tutor will create module content that will belong to the week.
Pre-condition	The tutor selects a week to add the module content to it.
Post-condition	The tutor creates content that belongs to the week.
Inputs	The tutor adds module content to the week.
Outputs	All users, including the tutor can see the module content within a week.

Name	Create Content 1.5
Description	The tutor will create content that will belong to the module content.
Pre-condition	The tutor selects add content within the specific module content page.
Post-condition	The tutor adds the content to the module content.
Inputs	The tutor adds the relevant information to the module content.

Outputs	All users, including the tutor, can see the content's content.

Name	Create Student 1.6
Description	The tutor will be able to create a student.
Pre-condition	The tutor selects add student on the view student page.
Post-condition	The tutor adds the student to the system.
Inputs	The tutor adds the relevant information such as the names and email of the student to the form.
Outputs	The account is now an active account that can use all the system's functionality.

Name	Log Out 1.7
Description	The tutor will be able to log out of the system.
Pre-condition	The tutor selects the logout button that is available on every page.
Post-condition	The tutor will be returned to the authorisation page.
Inputs	The tutor clicks on the logout function.
Outputs	The tutor is then returned to the login page.

Name	Create Class 1.8
Description	The tutor will be able to create a class for students to attend.
Pre-condition	The tutor must create a class on the calendar page.
Post-condition	The tutor creates a class for students to attend.
Inputs	The tutor creates a class with a time, date and place.
Outputs	The tutor and all the students can then see the classes that have been set.

Name	Student data 1.9
Description	The tutor will be able to view the data of each student that has been enrolled in the system. The data should relate to attendance feedback and completion.
Pre-condition	The tutor must select a student to view through the view student page.
Post-condition	The tutor views the data relating to the student.
Inputs	The tutor selects the student that they want to view.
Outputs	The system will return all the data relating to the students' performance.

Name	General Attendance 1.10
Description	The tutor will be able to view the data relating to the general attendance of all the classes that exist with all the students that exist.
Pre-condition	The tutor must be able to access the home page of their account.
Post-condition	The tutor views the data relating to general attendance.
Inputs	N/A

Outputs	N/A

Name	Content Completion 1.11
Description	The tutor will be able to view the data relating to the module content completion rate of the content that has been created.
Pre-condition	The tutor must be able to access the module page in their account.
Post-condition	The tutor views the data relating to content completion.
Inputs	The tutor selects the module page of their account.
Outputs	The system will output the content completion rate of the module content.

Name	General Content Completion 1.12
Description	The tutor will be able to view the data relating to the module content completion rate of the content that has been created.
Pre-condition	The tutor must be able to access the module page in their account.
Post-condition	The tutor views the data relating to content completion.
Inputs	The tutor selects the module page of their account.
Outputs	The system will output the content completion rate of the module content.

Name	Class Attendance 1.13
Description	The tutor will view the data relating to the individual class attendance.
Pre-condition	The tutor must be able to access the home page in their account.
Post-condition	The tutor views the data relating to individual class attendance.
Inputs	The tutor selects the home page of their account.
Outputs	The system will output the individual attendance of the selected class.

Name	General Feedback 1.14
Description	The tutor will be able to view the data relating to the general feedback of the student in question.
Pre-condition	The tutor will select the student they want to view through the view student page.
Post-condition	The tutor views the data relating to the general feedback for the student.
Inputs	The tutor selects the student through the view student's page.
Outputs	The system will provide the tutor with general feedback about the students' performance.

Name	Individual Feedback 1.15
Description	The tutor will view the data relating to the individual feedback about specific module content.
Pre-condition	The tutor will select the student they want to view through the view student page.
Post-condition	The tutor views the data relating to the individual feedback for the student.
Inputs	The tutor selects the student through the view student's page.
Outputs	The system will provide the tutor with individual feedback about the students' performance.

Name	Change Profile Picture 1.16
Description	The tutor will be able to change their profile picture to personalise their account.
Pre-condition	The tutor will be able to change their profile picture through the account settings page.
Post-condition	The Tutors profile picture will be changed so that all users can see this.
Inputs	The tutor accesses the account settings page.
Outputs	The Tutors profile will be changed for all users to see.

Student Level Functions 3.2.2

This list will outline the list of the student level functionalities that will be implemented to meet the requirements specification.

Name	Login Page 2.1
Description	Authorisation of the user that will navigate them to their home page.
Pre-condition	The user is on the login page, and the authorisation is successful.
Post-condition	The user is on their home page and can access all the functionality.
Inputs	User inputs legitimate details
Outputs	The user is navigated to their personalised home page.

Name	Give Feedback 2.2
Description	The student will give feedback about the module content.
Pre-condition	The student is on the specific module content through the module page.
Post-condition	The student will have provided feedback concerning the module content.
Inputs	The student rates the module content based on how they feel.
Outputs	The student provides feedback that can be displayed to the tutor.

Name	Class Registration 2.3
Description	The student will be able to register once for their classes.
Pre-condition	The student is on the registration page and selects the relevant class.
Post-condition	The student will be registered for the class.
Inputs	The student must pass data to the system that is unique to the class to register for it.
Outputs	The student confirms their registration which the tutor can see.

Name	View Content 2.4
Description	The student will be able to view the content that has been provided.
Pre-condition	The student is on the view module page, which will show them the content set out for them.
Post-condition	The student will be able to follow the content that they need for their learning.
Inputs	The student must select the content that they want to view.
Outputs	The system will display the content to the module.

Name	View Announcements 2.5
Description	The student will be able to view the announcements.
Pre-condition	The student can view the announcements set by the tutor through their home page.

Post-condition	The student will be able to follow the direction of the announcement.
Inputs	The student must be on their home page to view announcements.
Outputs	The system will display the announcements set by the tutor.

Name	Discussion Forum
Description	The student will be able to view and contribute to the discussion forum.
Pre-condition	The student can access the discussion forum through the individual module page.
Post-condition	The student will be able to view and contribute to the discussion forum.
Inputs	The student must be on the relevant content page to view the discussion forum.
Outputs	The system will display what all users have contributed to the discussion forum.

3.3 Non-Functional Requirements

Overview of Non-Functional Requirements 3.3.1

Non-Functional Requirements relate to the characteristics of a system that cannot be expressed as functions.

Due to the time constraints of this project and lack of resources, only the Non-Functional Requirements that are vital to the functionality of the Web Application were thoughtfully considered and implemented.

These requirements address aspects concerning:

1. Maintainability – How easy it is to identify and fix defects in the system.
2. Portability – How easy is it to port the system to another platform.
3. Data Storage - How often is data stored, what data is stored and how should data be stored.
4. Security – How secure is the website, does the system hold sensitive data, and how is that data stored.
5. Reusability – How well designed is the system so that components can be extracted and reused.
6. Testability – How testable is the system?
7. Usability – How useable is the system? Does the system require training?
8. Efficiency – How efficient is the system performing the functions it is supposed to?

These requirements must be documented and implemented for performance and customer satisfaction.

Non-Functional Requirements 3.3.2

These are the Non-Functional Requirements that I decided that needed to be implemented.

Title	Browser Security
Description	The secure browser against unauthorised users
Pass Condition	The user can only access the web page when a session has been created.
Fail Condition	The user can access the webpage when they have not been authorised.

Title	Data Security
Description	Secure, personalised Data
Pass Condition	All the users' data is safely stored and is only accessible to the student and the tutor.
Fail Condition	Personal Data can be accessed by users who are not authorised.

Title	Data Storage
Description	Data is stored in the correct location to be easily retrieved when required.
Pass Condition	The data is stored in the correct location.
Fail Condition	The data is not stored in the correct location.

Title	Application Useability
Description	The application should be easy to and should not require training.
Pass Condition	The application is easy to use for the target audience.
Fail Condition	The application is not easy to use for the target audience.

Title	Application Useability
Description	The application should be easy to and should not require training.
Pass Condition	The application is easy to use for the target audience.
Fail Condition	The application is not easy to use for the target audience.

Title	Loading Time
Description	The application should take no longer than 5 seconds to load in.
Pass Condition	The application takes no longer than 5 seconds to load in with a perfect internet connection.
Fail Condition	The application takes longer than 5 seconds to load in with a perfect internet connection.

Title	Application Portability
Description	The application should be easy to move to a different server.
Pass Condition	The application should be easy to move to a different server.
Fail Condition	The application is not easy to move to a different server.

Title	Application Maintainability
Description	The application should communicate where the defects in the system are.
Pass Condition	The application should be easy to maintain and develop.
Fail Condition	The application isn't easy to maintain or develop.

The Data Model 3.4

Overview 3.4.1

This chapter will outline the techniques used to model the data associated with this application.
Data modelling is the process used to structure how data is stored and modelling relationships. The goal is to create a visual data map that accurately describes the data structure, how the data will be passed through the system whilst highlighting the vital data relationship. In this application, I have implemented a relational database system using MySQL.

A database is any collection of information that has been systematised for searching and retrieval via the use of a computer. Databases are intended to provide the means for modifying, deleting, storing, and retrieving data connected with different operations.
A Relational Database (RDBMS) allows us to identify and access data concerning another piece of data in the database.

MySQL 3.4.2

SQL is a programming language used to communicate with the data stored in the relational database system.
MySQL syntax is similar to the English Language, making it easy to write, read and interpret.

Exploring Relational Database 3.4.3

In my application, the data is sorted into tables that are named after the data they are storing. E.g. the student's personalised data in the student details table.
Each row of data in each table has a primary key in the form of an integer known as the unique identifier.
When this key is called, the database will return the row of data associated with it. In other words, all the non-attributed data must have a non-transitive functional dependency.
Relational Databases also have the advantage of creating foreign keys (usually the primary key), which can be stored in a separate table to claim ownership of other data that may need to be stored elsewhere.

Normalisation 3.4.4

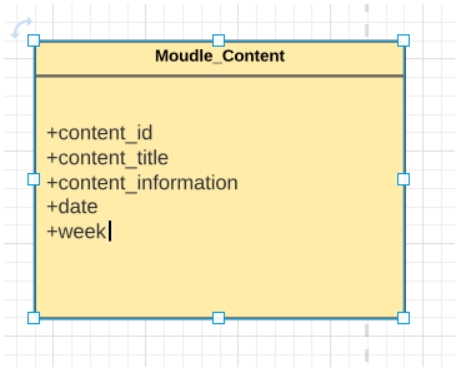
Normalisation is a technique that is used for a relational database. It means that we condense the number of tables included in our database while maintaining data integrity. We eliminate repeating data and weak entities and minimise redundancies by implementing this. This is especially important when adding, deleting or updating our database. Failing to normalise our database could result in a violation of our data.

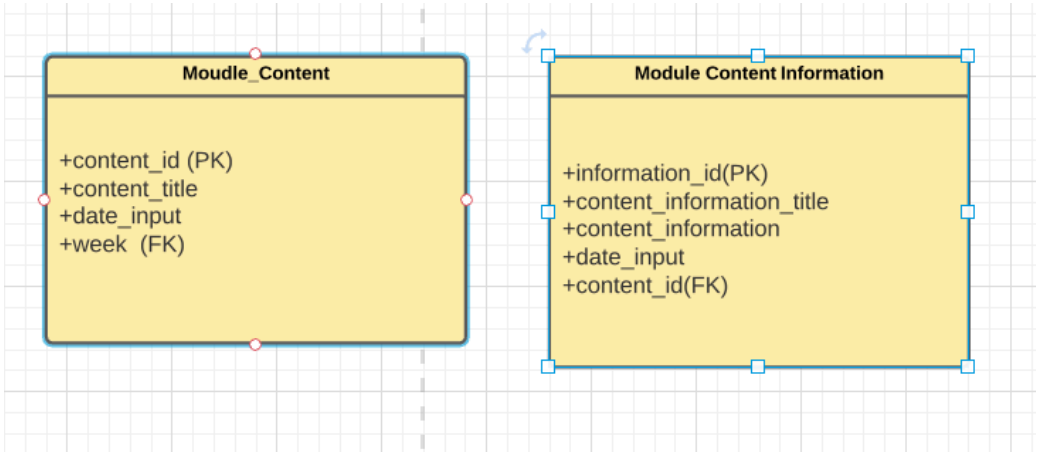
It was important that I normalized the database tables from the outset. This meant that I had to create separate tables for each set of related data. E.g The likes belonging to the comments in the discussion forum are separated to their own table so that each like would be recorded along with ownership to the user and the comment. One like will have one comment but one comment can have many likes therefore this is logical. Without normalization this would result in duplicated data and other anomalies.

Another example would be information that is found in the individual lab content that is set by the tutor. Module Content and Module Content Information cannot be listed in the same table as that would mean that there is reputative data. Once again, one set of module content information data belongs to one module content, but module content will have more than one set of module content information data. This would lead to duplicated data so it was required that I created another table called module content information and then link the two table together.

In Figure 3.4.4.1 you can see a representation of the raw data with no normalisation and then how the raw data was split into two tables using the correct procedure in the two tables below.

Figure 3.4.4.1 Depicting Raw Data and how it was broken into separate tables to stop redundancies



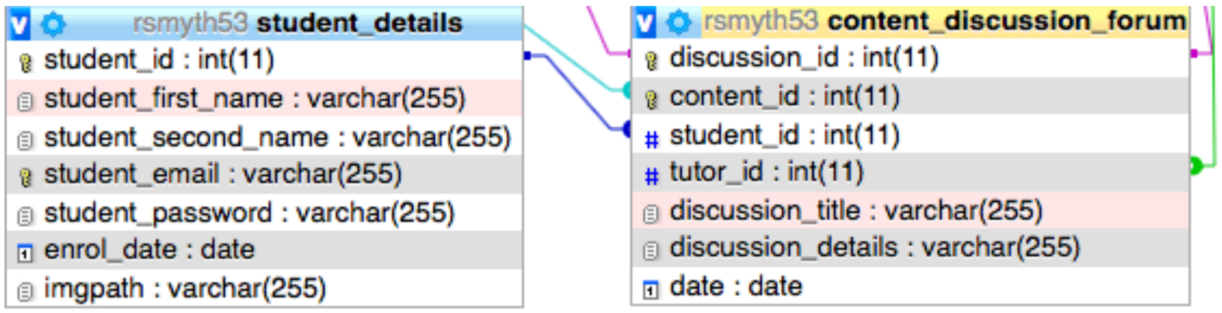


My database has been normalised out to the 3rd standard form. A relational database to qualify for the 3rd standard form must first be the 2nd standard form and have no transitive dependencies. This means that no other non-key attribute needs to be changed if you change a non-key attribute. For example, if the student's name needs to be changed, only the column in the student details table needs to be changed and nothing else.

There is a high emphasis on foreign Keys in this relational database. A Foreign Key is a column or columns in a relational database that links data in two tables. Foreign Keys are especially usefully for applications like mine, which have a lot of data that needs to be normalised but can be easily retained using join statements within the code.

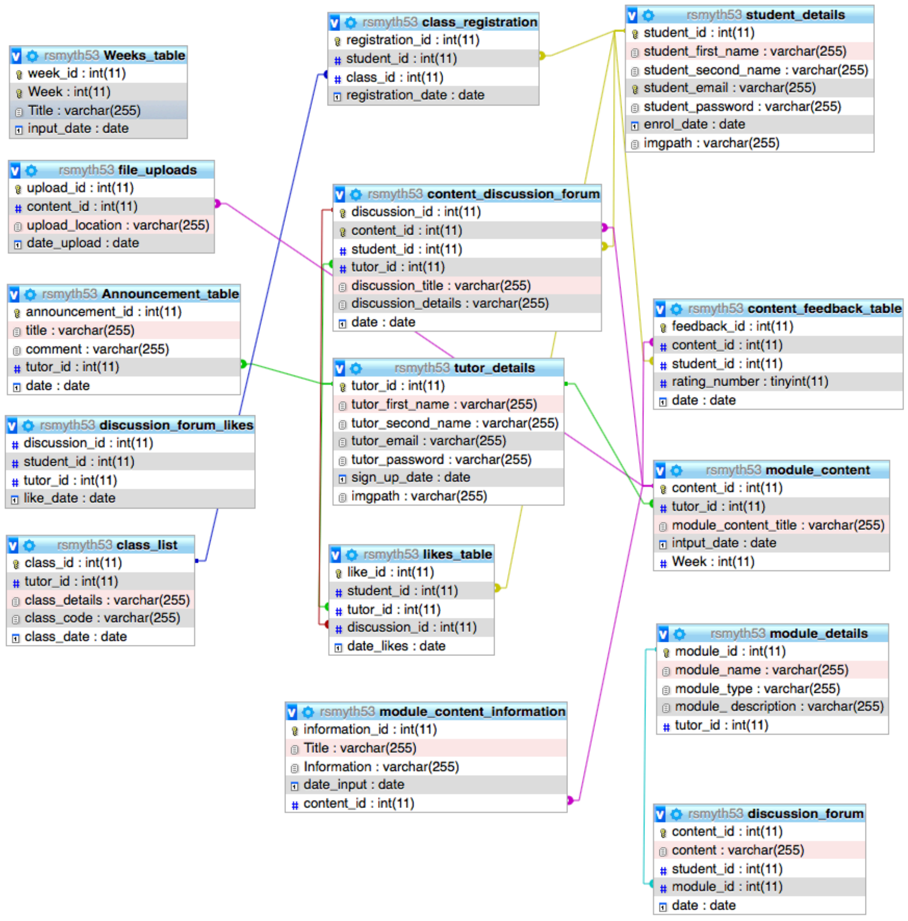
I have a Discussion Forum for students and tutors to comment under concerning content associated with the module in my application. A student or tutor can have many comments, but comment can only have one student or tutor. Figure 3.4.3.1, you can see the student details table and the content discussion forum table. In between the tables, you can see the blue link that extends from the student id in the student details table to the student id in the discussion forum table to represent the Foreign Key.

Figure 3.4.4.2 Depicting a Regional Database Table and how the Student ID is represented as a Foreign Key



rsmyth53 student_details
student_id : int(11)
student_first_name : varchar(255)
student_second_name : varchar(255)
student_email : varchar(255)
student_password : varchar(255)
enrol_date : date
imgpath : varchar(255)
rsmyth53 content_discussion_forum
discussion_id : int(11)
content_id : int(11)
student_id : int(11)
tutor_id : int(11)
discussion_title : varchar(255)
discussion_details : varchar(255)
date : date

Figure 3.4.4.3 Depicting the Normalised Tables with Entity Relationships



Chapter 4

4.1 Introduction

The interface that the user will observe when they use the software is the single, paramount aspect of the system. The user interface is the packaging for software. The user does not care how the system works (providing that it is reliable and fast) but rather how it looks and how simple it is to use. Unfortunately, user interface design is often the yardstick by which the system is judged proper. An interface that is difficult to navigate will, at best, result in a high amount of user errors. At worst will result in the software being discarded.

User interface design is an inter-disciplinary subject with contributions from computer science, sociology, cognitive psychology and ergonomics. The user interface design has as much to do with the study of people as with the technology. For example, Facebook developed their notorious Like button in 2009. This completely changed the way that people used Facebook. Initially, the notifications were blue. But when the notification button was changed to red, the level of interaction with each users' notifications increased.

User Interface design cannot be overstated. It will either make or break a professional Web Application. For that reason, I have decided to follow these principles to develop my application.

- Learnability – How easily can inexperienced users learn to use the system.
- Flexibility – does the interface support a variety of interaction styles.
- Robustness – how much feedback the system gives the user to confirm what is going on.
- Recoverability – does the system allow the user to quickly recover from an unintended situation.
- Observability – does the system display information that will help prompt the user to navigate the application.
- Simplicity - the application is simple and takes little skill to use.

Please note that many of the images in this report were taken before I had time to style the application correctly.

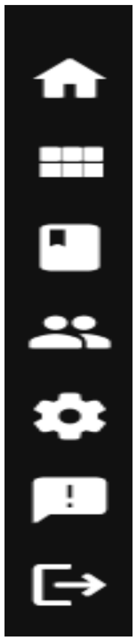
In the following sections I will outline how I adopted these principles and how I applied them to the individual components that make up application.

4.2 Learnability and Simplicity

The user interface will be simple, user-friendly, lightweight with a colourful design that will provide a comfortable and easy user experience. The interface will be designed so that the user can easily navigate the application without any difficulty. On the left-hand side of the application will be a navigation bar to access all the application's main functionality.

The home icon will always direct the user back to their original landing page. The module icon will direct the user back to the module landing page, displaying all the content in the Programming Module.
At the bottom of the navigation bar is will be the icon that gives the user the ability to log out, which will return them to the login page. See Figure 4.2.1

Figure 4.2.1 Depicting the Sidebar of the Application



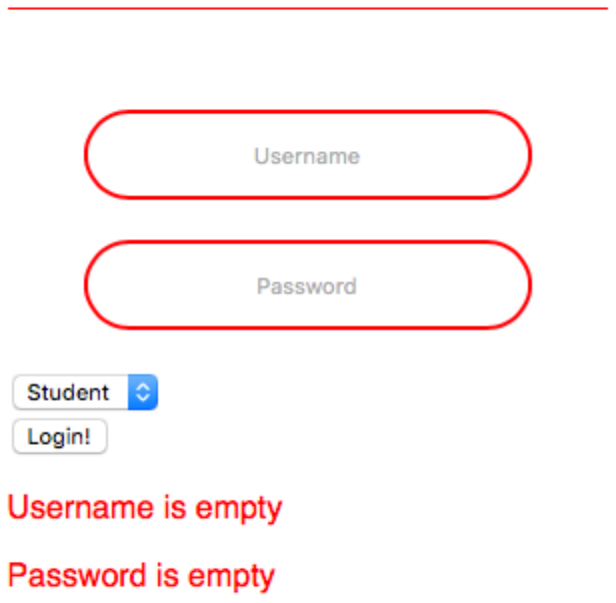
Much of the functionality will be accessible through each activity. But in certain circumstances, you can only access some functionality through the activity that it is logically related. For example, in the view module content activity, you can view the module content that you have set for the students. However, you can only access the discussion forum through this activity rather than from anywhere on the application. This helps create Learnability and Simplicity because the majority of the components are accessible through less than 3 clicks once the user has logged in.

4.3 Robustness – how much feedback the system gives the user to confirm what is going on.

A vital component of any GUI is how the application handles errors and communicates to the user. These error messages need to be accurate, practical and prompt the user to make the correct action. Failure to do this could result in your application being discarded. There is nothing that frustrates a user more than the inability to carry out a task by the system refusing to do it without them being able to figure out why it is being rejected.

It was important that error handling was carefully considered and done during the implementation stage. For example, on the login page, when the user enters the incorrect details, it is vital that the user was greeted with a "Username or Password invalid", the message should not specify what the user is missing but that they are refused access to the website on this basis. The actually message also carries importance to help the user solve the problem.. Another important example was returning the message that there is no content to show within a particular activity. When there is no content active within a database, the GUI will return a message outlining that ”There is no content to show”.

Figure 4.3.1 Depicting the Login Page handling an error.



Username is empty

Password is empty

4.4 Observability – does the system display information that will help prompt the user to navigate the system.

To achieve observability the application does display the right amount of information necessary so that the user can successfully navigate the system. As stated above the links to each of the pages can be found in the side bar of the application. These links are represented in the form of icons so that the user can navigate to the correct page. However, there could be more messaging to increase useability such as names beside the icons on the navigation bar.

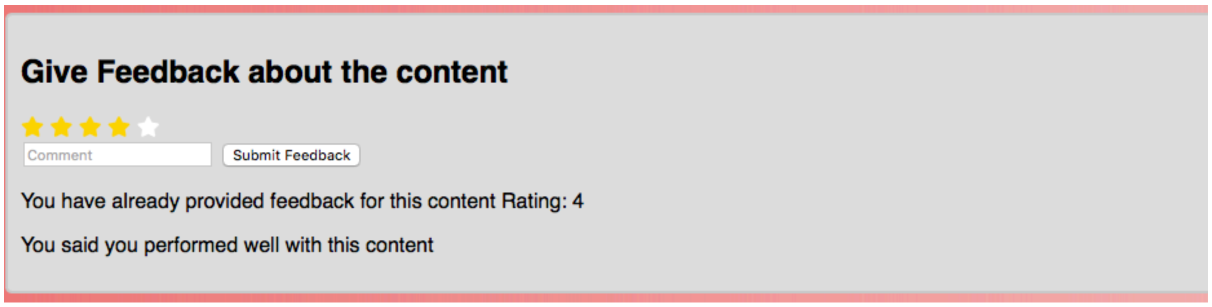
4.5 Recoverability – does the application allow for the user to recover from an unintended situation.

Recoverability relates to the ability of the system to recover from an unintended situation. From testing the system, it is fair to say that the system is designed so that the user can recover form an unintended situation easily. As I stated above the user can navigate back to the home page through the use of the home icon on the side bar or to any other page.

A significant unintended situation may be that the tutor has created or added something by accident that they wish to retract. For example, the tutor may create content with the incorrect title. This can be easily deleted through the activity which it exists. The same can be applied to incorrect classes or student accounts with incorrect details. There is a delete function that can be easily obtained.

4.7 Feedback Rating System

Figure 4.7.1 Depicting the Rating System that the student will use to give feedback about the system.


Give Feedback about the content

★★★★☆

You have already provided feedback for this content Rating: 4

You said you performed well with this content

The purpose of the Feedback Rating system is for the student to provide feedback to their tutor about how they performed with the lab that was set for them. The requirements do not state how the feedback is recorded, so I decided to use my imagination concerning what I, as a student, would like to see implemented. When the student uses their mouse and hovers over a star, they will turn gold to represent a potential rating being returned. Only when the user physically clicks on the rating will the value be returned to the database. This is a quick and uncomplicated way that the students can give feedback about the topic.

The feedback container is placed at the top of the view module content activity on a grey background to be identified and recorded. Throughout the application, the key components used to collect, and display data are at the top of the browser to be easily identified.

This is my most prominent example of a component that encompasses all of the principles that I outlined above. The component is Learnable as all the user had to do is understand a basic concept to provide feedback about the content that they have completed. The higher that they want to rate the topic the more stars that they will provide. The system is also robust. If the user has provided feedback, then the system will return the feedback that was provided, therefore confirming to the user what is going on. I should note that the GUI will have a more mature design than the depiction of Figure 4.8.1.

The component was implemented with a focus on recoverability. This means that if the user makes a mistake then the user can easily recover and provide the correct feedback in relation to the content. The back end of this component is implemented so that the rating will update every time that the use provided feedback. This means that the user can easily change their mind with further revision of the topic or if that have provided the incorrect input.

Part 2 – Software System Design

4.8 Overview

System Design is a crucial element to the development of software. A solid design must be developed. The criteria of judging a good design can vary widely across a diverse range of applications. There is not a broad consensus across the community about a good design for a specific application.

There were numerous sources that I studied which led me to implement this style of design for my application. But these this source stood out to me. This source was called "*The Fundamentals of Software Engineering"* by Rajib Mall. This source outlines that generally a good design will prevent many problems, and that finding a good design is not an easy task. However, the principles that I have outlined below are crucial to any design according to professional designers and engineers.

1. Correctness – A good design should be correct. It should implement all the functionalities of the system.
2. Understandability – The design should be easy to understand. Without this, it would be challenging to implement and maintain.
3. Maintainability – A good design should be easy to change. Maintainability is important as requests will usually keep coming for the customer long after the application has been released.

Another motivation forms this source is in relation to making the design modular. A module in a software engineering context is an encapsulation of code to implement a certain functionality. This means that each PHP file of my application is a module. Unfortunately, there is no metric by which an application can be measured in its modularity. A modular designed combined with a Three Layered Architecture Design has may advantages.

4.9 Architectural design - Three Layer Architecture

4.9.1 Description:

In software engineering, tier architecture or multilayered architecture is a client–server architecture in which presentation, application processing and data management functions are physically separated. The most widespread use of multitier architecture is the three-tier architecture.

The system is organised into layers with related functionality associated with each layer—a layer supplies services to the layer above it. The lowest layer, which is always the database, is the core service necessary to the functionality of the Web Application.

At the top of the system is the user interface, followed by the functionality layer in the middle. This layer will then interact with the System Support, the database directly to manipulate data.

For different components, there are distinctive designs that are implemented. For example, there is functionality provided to the user without them requesting this specifically. For other components, the MVC is adopted so that they request the information and that it is returned. General information such as the general attendance component will supply the information without request, and this will automatically update when the data is updated. This architectural style allows for a comfortable incremental design.

4.9.2 Advantages:

1. Redundant facilities can be provided in each layer to increase the system's dependability and system.
2. The architecture is changeable and portable. E.g. If the interface is unchanged, a new layer of improved/ extended functionality can be implemented without breaking other Web Application functionality.
3. Many development teams have separate developers who specialise in front-end, server backend, and data backend development. Modularising these parts of an application, you no longer have to rely on full-stack developers and better utilise each team's specialities.

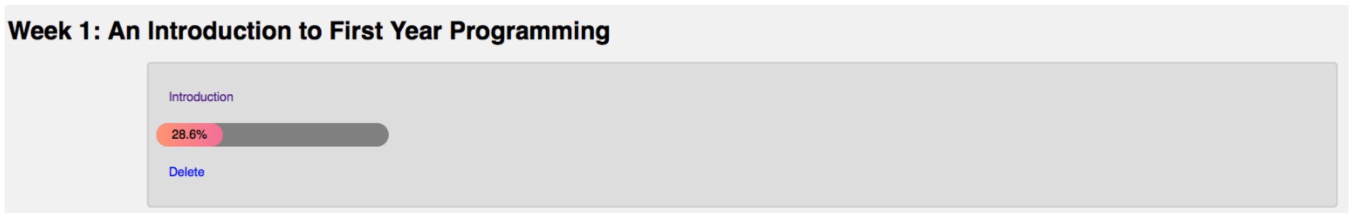
4.9.3 Disadvantages:

1. 1. Supplying a clear separation between the layers is often complicated and not desirable due to the time constraints of this project. E.g., the application's opportunity to be designed with each layer separately existed. But with the constraint of time and resources and keeping the complexity of the system as low as possible, it was better than some overlap in between the layer of the functionality and the user interface of the application.
2. 2. In some cases, higher-level systems may have to interact with lower-level systems rather than the level directly below them.

4.10 Three Layer Architecture Implementation

This section will describe how the Three Layer Architecture is implemented. The Content Completion component provides data about the completion rate of the content set by the tutor. The user does not request this information. Design using this pattern means that the user will be provided information without requesting it. This is the norm for most of the application, apart from some components that require user requests. It is important to note that these separate components would not be able to function without using the Three Layer Architecture. They need the information provided by this design to make the correct decision when asking for specific data.

Figure 4.10.1 Depicting the Content Completion component.


Week 1: An Introduction to First Year Programming

Introduction

28.6%

Delete

4.11 MVC

The MVC is a software design pattern commonly used for User Interface Design. This pattern divides the related programming logic into three interconnected elements. The MVC helps the user alert the system to change and display the information they request.

Model –The model is the domain layer that holds all the data, e.g., sensitive data concerning the users and their feedback.

View - The user interface layer is holding all the code relating to windows and buttons.

Controller –The controller handles requests to get information from the model or request the model to act.

Figure 4.11.1 Depicting the visual representation of the MVC pattern.

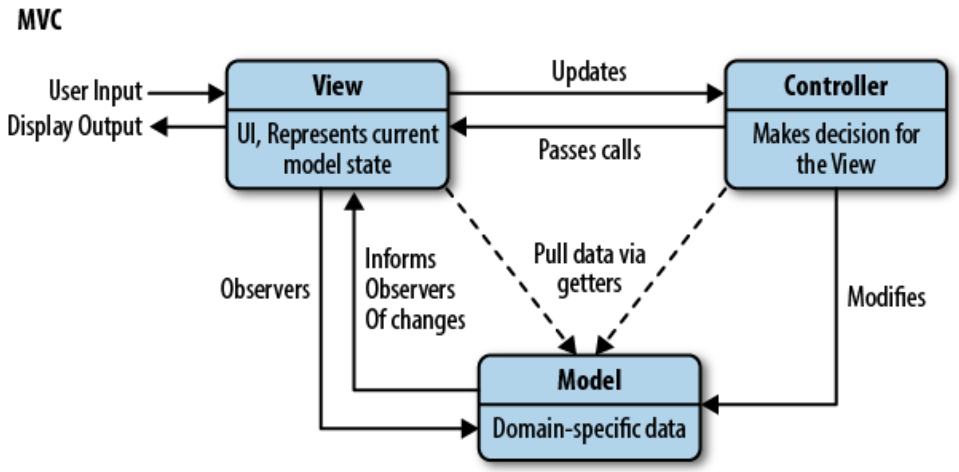
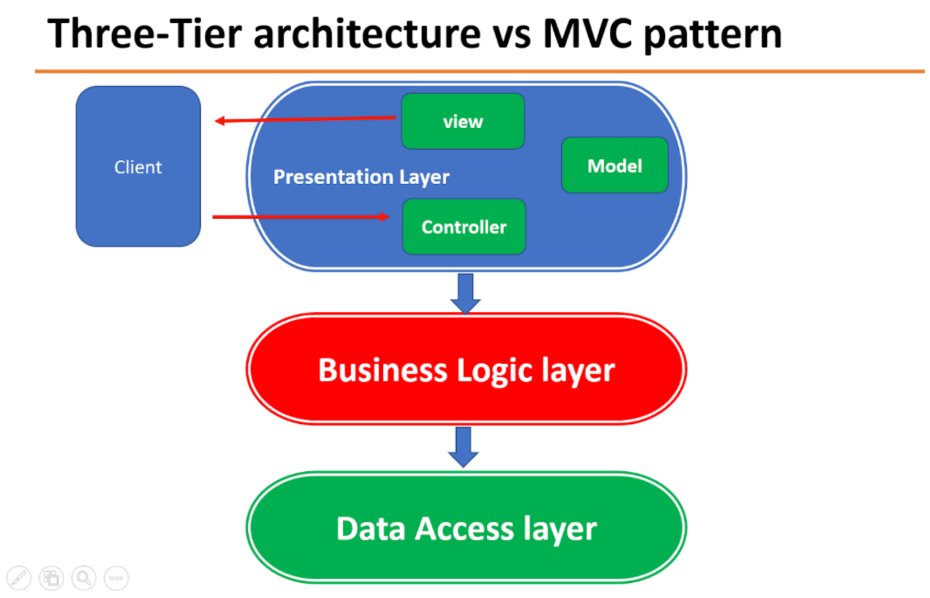


Figure 4.11.2 Depicting a visual representation of the Three Layer Architectural Design with the MVC pattern.

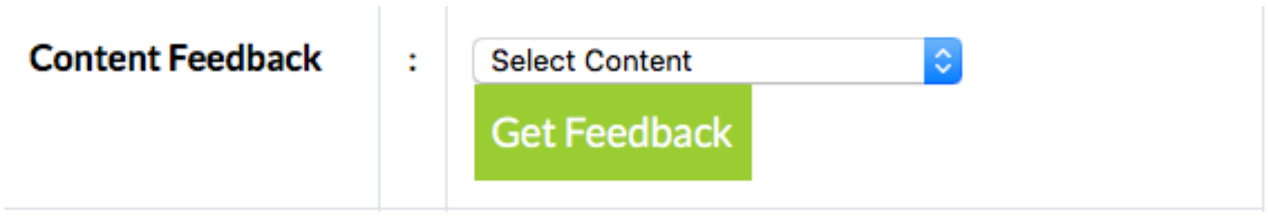


4.12 MVC Implementation

This section will show the MVC pattern implemented in the Web Application. To provide the reader with an example of how this MVC pattern is implemented, I will describe how the Individual Content Feedback is passed to the user. I have adopted a procedural programming style, and therefore the three elements of the MVC pattern all exist within the same activity.

To access the individual content feedback of a particular student. The tutor must access the students account through the Student Profile activity. The tutor uses the view to access the controller; the tutor can then use the view to update the controller about what content they want to view. This request is then modified and passed to the model, which will notify the view and the controller of the sudden change.

Figure 4.12.1 Depicting the view of the Individual Content Feedback Component.



4.13 Component Design

Chapter 3 outlined the non-functional requirements that are needed so the application can perform correctly. One of the principles that I outlined was reusability. This means that components can be taken from the system and used elsewhere to help productivity. For the most part, all of the main components that work to provide the main functionality can be reusable if the customer demanded further functionality.

The reusability of components is due to the database and how the components are written in the code. The fact that the database is modeled as a relational database means that the components were easier to write and, therefore, easier to implement in the context of extending that functionality to a new component.

Chapter 5 – Implementation

5.1 Overview

During the software construction in this application, it was essential to understand that the code that I am writing will be read more than it is written. Therefore, I implement good coding practices such as correct naming conventions, commenting, and making sure that the code is in line with the intended design and architecture. I used the Code Editor Visual Studio Code as it helped help me navigate activity to activity without any help.

I adopted Procedural Programming to develop this software which I have already outlined in Chapter 2. This meant that much of the code written within the file would only be relevant to that file. There is the odd occasion whereby there is a separate that is used to Delete the data from the database and then return to the page same page without the user even knowing they went there. When it came to the GUI design, there are external CSS and JavaScript files for the sake of practicality.

Implementation 5.2

5.2.1 Feedback System

The Feedback system was implemented to leave feedback on the labs that they are obliged to carry out as part of the module's module. Once the student was finished with the lab, they would give the work feedback through the star rating system automatically created once the tutor has created the content. There is five stars rating from poor to excellent.

Implementing the rating system was a demanding task that took a few weeks to implement. The aim was to make the system functional, but a critical emphasis was on making the system dynamic and user friendly. The main hurdle to creating a dynamic system was that I needed an intermediate knowledge of JavaScript. Furthermore, the difficulty lay in the ability to pass the data from JavaScript to PHP, which would then save the data to the database. Due to the functionality of this system, all the necessary code had to be written in the same activity so that it was easier for me to implement it successfully. It is good practice for JavaScript to be written in a JavaScript file rather than in a pair of tags in the activity.

In Figure 5.2.1, you can see the JavaScript relevant to making this system work dynamically. The first function in the code is called the $(document).ready function, which will be the primary function used to capture the student feedback. In the ready function, the resetStarColours(); is called so that all the stars are empty whenever the student clicks onto the individual content page.

Name	**Description**
$('.fastar').mouseleave(function(){	When the mouse is not hovering over the star element the stars will be returned to their default value white.
$('.fa-star').mouseover(function(){	When the mouse is hovering over the star function then the stars will change to gold.

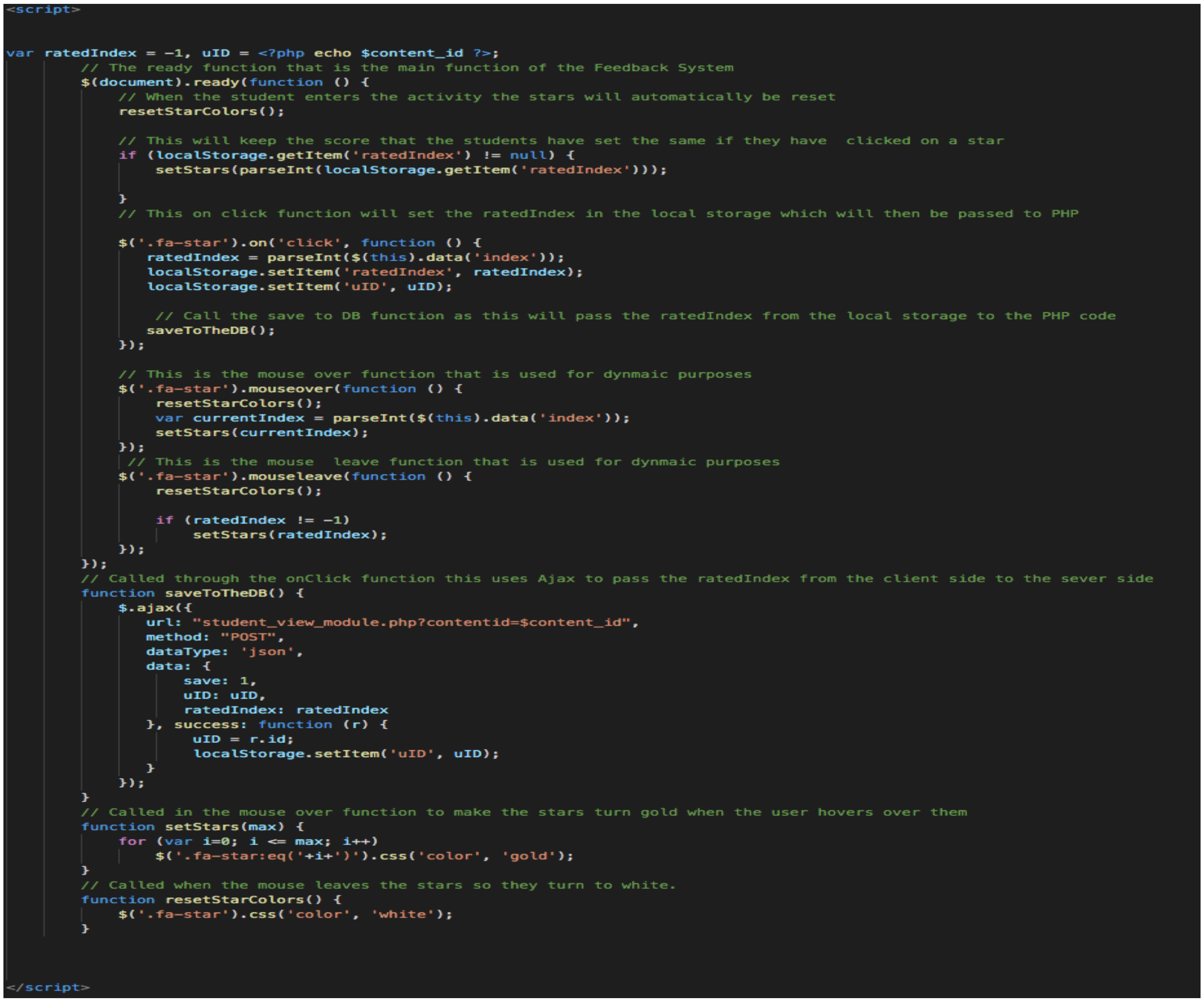
To make the feedback system dynamic, I added a $('.fa-star').mouseover(function () { to make the stars turn yellow when the user would hover over them with their mouse. There is also a $('.fa-star').mouseleave(function () { which was used so the stars would turn white if the user does not choose what they want to rate the content. These functions are the functions implemented to make the page dynamic.

I used both JavaScript and PHP to achieve this in terms of functionality and saving the data in the database. The variable that I used to pass the data was called the ratedIndex, which would be saved in the local storage of the browser through the $('.fa-star').on('click', function () {.

Inside the on click function is this saveToTheDB() { function which would then get the ratedIndex variable from the local storage and pass it to PHP through the Ajax function in the saveToDB();. This made it possible for the data to be stored into a local variable in PHP created using the super global variable in PHP called $POST['ratedIndex'];.

The data that was passed to PHP would then be incremented because the ratedIndex found in the local storage would be the incorrect value. After all, computers count from 0. E.g. If the user had clicked on the 5th star, the value passed through the local storage would then be 4. Therefore, the local variable needs to be incremented by adding the correct value to the database. To see the PHP code used to pass the data to the MySQL database, see figure 5.2.

Figure 5.2.1 JavaScript used for the Feedback System.

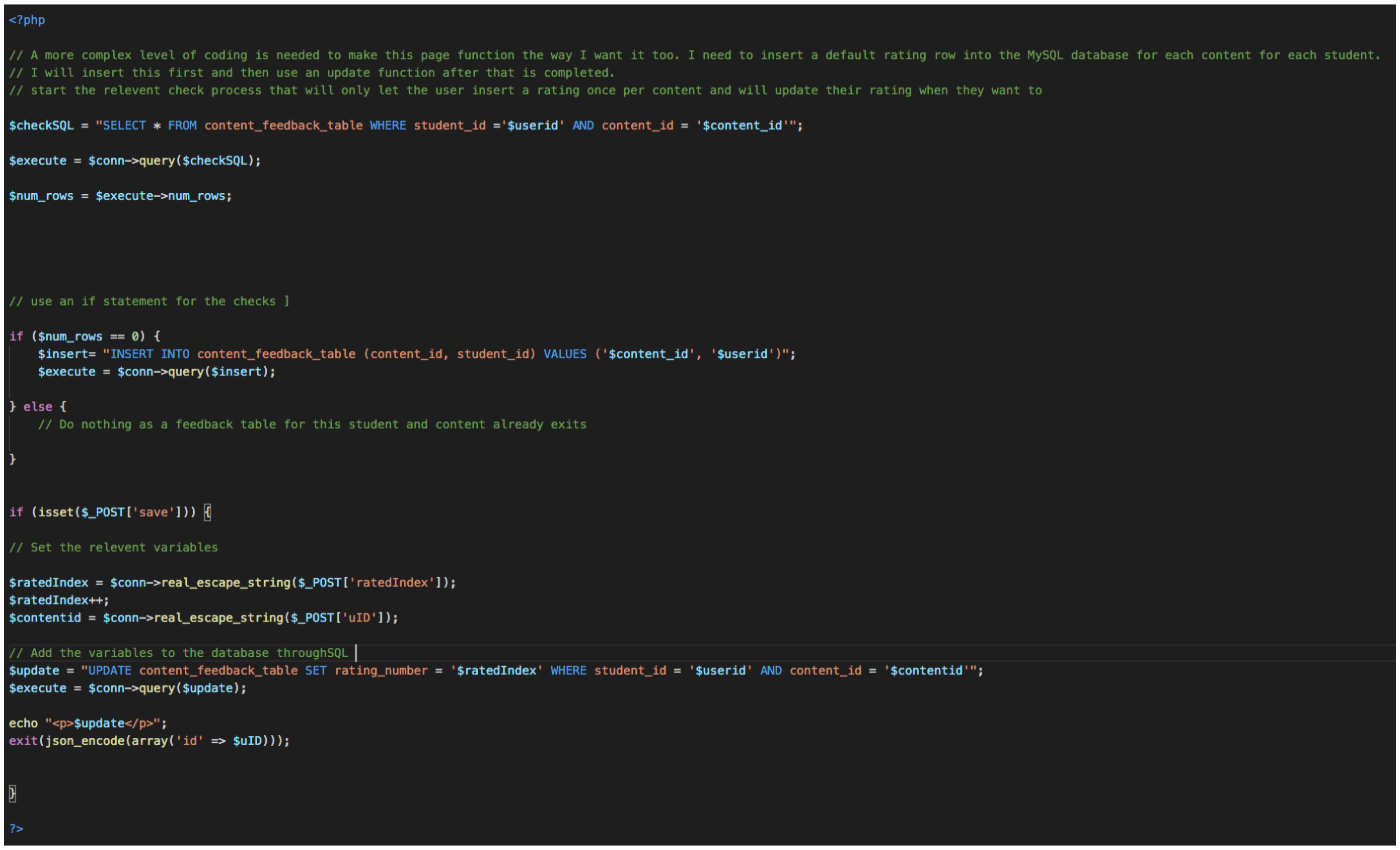


```
<script>


var ratedIndex = -1, uID = <?php echo $content_id ?>;
        // The ready function that is the main function of the Feedback System
        $(document).ready(function () {
            // When the student enters the activity the stars will automatically be reset
            resetStarColors();

            // This will keep the score that the students have set the same if they have  clicked on a star
            if (localStorage.getItem('ratedIndex') != null) {
                setStars(parseInt(localStorage.getItem('ratedIndex')));

            }
            // This on click function will set the ratedIndex in the local storage which will then be passed to PHP

            $('.fa-star').on('click', function () {
               ratedIndex = parseInt($(this).data('index'));
               localStorage.setItem('ratedIndex', ratedIndex);
               localStorage.setItem('uID', uID);

                // Call the save to DB function as this will pass the ratedIndex from the local storage to the PHP code
               saveToTheDB();
            });

            // This is the mouse over function that is used for dynmaic purposes
            $('.fa-star').mouseover(function () {
                resetStarColors();
                var currentIndex = parseInt($(this).data('index'));
                setStars(currentIndex);
            });
             // This is the mouse  leave function that is used for dynmaic purposes
            $('.fa-star').mouseleave(function () {
                resetStarColors();

                if (ratedIndex != -1)
                    setStars(ratedIndex);
            });
        });
        // Called through the onClick function this uses Ajax to pass the ratedIndex from the client side to the sever side
        function saveToTheDB() {
            $.ajax({
               url: "student_view_module.php?contentid=$content_id",
               method: "POST",
               dataType: 'json',
               data: {
                   save: 1,
                   uID: uID,
                   ratedIndex: ratedIndex
               }, success: function (r) {
                    uID = r.id;
                    localStorage.setItem('uID', uID);
               }
            });
        }
        // Called in the mouse over function to make the stars turn gold when the user hovers over them
        function setStars(max) {
            for (var i=0; i <= max; i++)
                $('.fa-star:eq('+i+')').css('color', 'gold');
        }
        // Called when the mouse leaves the stars so they turn to white.
        function resetStarColors() {
            $('.fa-star').css('color', 'white');
        }


</script>
```

Furthermore, whenever the student clicks onto the content page from the module page, I have implemented a set of checks to check if the user has already clicked on that page before. If the user has not clicked on that page before, the code is instructed to create a row in the database table called content_feedback_table belonging to the student and linked to the content but with a NULL value in the rating column.

Only when the user would click on the stars would the MySQL statement be executed due to the ‘save variable’ passed through the Ajax function called in the saveToTheDB(); function, which can only be called through the on click function. When the data is passed to the variable in the PHP code, there is an UPDATE statement to update the already existing column. Every time the user clicks the stars, the user can update the feedback of the content. This gives the user the option to change their minds or correct a mistake from a mis click.

Figure 5.2.3 PHP code used for recording feedback



```php
<?php

// A more complex level of coding is needed to make this page function the way I want it too. I need to insert a default rating row into the MySQL database for each content for each student.
// I will insert this first and then use an update function after that is completed.
// start the relevent check process that will only let the user insert a rating once per content and will update their rating when they want to

$checkSQL = "SELECT * FROM content_feedback_table WHERE student_id ='$userid' AND content_id = '$content_id'";

$execute = $conn->query($checkSQL);

$num_rows = $execute->num_rows;




// use an if statement for the checks ]

if ($num_rows == 0) {
    $insert= "INSERT INTO content_feedback_table (content_id, student_id) VALUES ('$content_id', '$userid')";
    $execute = $conn->query($insert);

} else {
    // Do nothing as a feedback table for this student and content already exits

}


if (isset($_POST['save'])) {

// Set the relevent variables

$ratedIndex = $conn->real_escape_string($_POST['ratedIndex']);
$ratedIndex++;
$contentid = $conn->real_escape_string($_POST['uID']);

// Add the variables to the database throughSQL
$update = "UPDATE content_feedback_table SET rating_number = '$ratedIndex' WHERE student_id = '$userid' AND content_id = '$contentid'";
$execute = $conn->query($update);

echo "<p>$update</p>";
exit(json_encode(array('id' => $uID)));


}

?>
```

This is the most vital component of the entire system, as the data collected from that page is the foundation that makes this entire system of any use. The data collection allows me to expand the system for other things such as completion and general feeling about the module.   If the feedback is completed, this will allow us to check the users progress with the work that has been set. If the feedback exists in the relevant table, the work has been completed in theory. After all, it is in the student's interest that they complete their University work.

Name	**Description**
$('.fa-star').on('click', function () {.	When the user clicks on the star, this rating will be stored in the local storage.

saveToDB();.	This function is called through the onClick function and this will grab the data passed to the local storage so that it can be passed to PHP through Ajax and stored in our relational database.

5.2.4 Student Profile – General Feedback

The student profile component is a significant part of the system. It is used for the tutor to follow the progress of the student about attendance, feedback and completion rate of the content that the tutor has added. The student profile system relies heavily on the student feedback system. This data needs to be captured so that the tutor can see a correct representation of the student's progress within the module.

A significant section of code within the student profile page is the getAverageFeedback(), which will show the tutor how the student is generally feeling about the content they have completed so far. As I stated in the section above, there are five stars from feeling poor to confident. To get the average feedback, I need to divide the highest possible rating that the student could give on all the content they have completed by the actual rating provided in the student feedback system on all the content. I used many SQL SELECT statements that would return the number of rows that belonged to the student within the feedback table to get this data.

Figure 5.2.5 depicting the formula used to calculate the mean average.

Formula

$$\text{Average} = \frac{\sum_{i=1}^{n} x_i}{n}$$

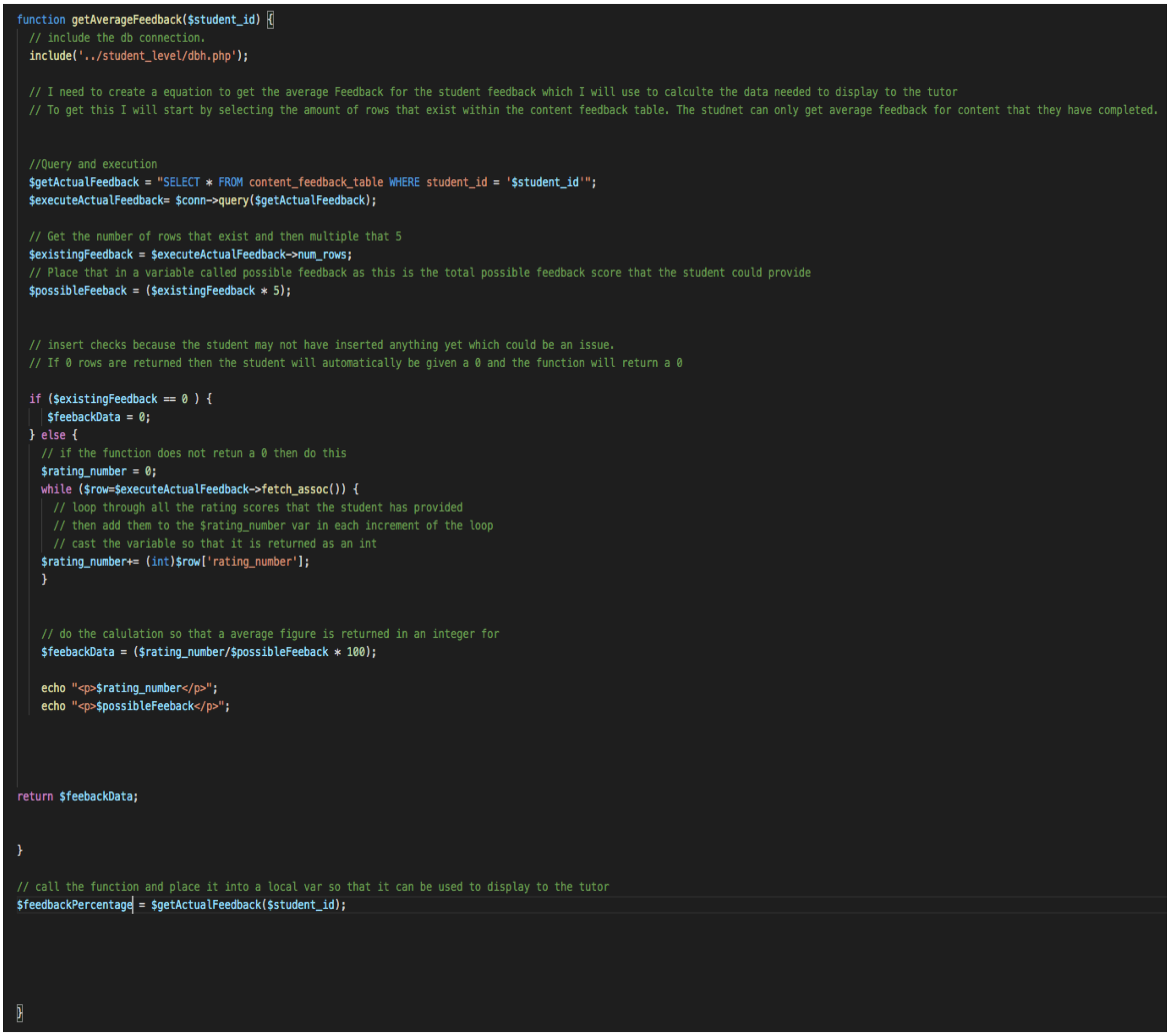
$$\sum_{i=1}^{n} x_i \longrightarrow x_1 + x_2 + x_3 + x_4 + \dots\dots + x_n$$

$$n \longrightarrow \text{total number of terms}$$

getcalc.com

The sum of the two variables was then be multiplied by 100 so that it put the average into a percentage figure which could be accurately manipulated in the GUI so that the tutor could see a visual representation of the progress belonging to the student. See figure 5.3 for the getAverageFeedback(); function.

Figure 5.2.6 depicting the getAverageFeedback(); function.



```php
function getAverageFeedback($student_id) {
  // include the db connection.
  include('../student_level/dbh.php');

  // I need to create a equation to get the average Feedback for the student feedback which I will use to calculte the data needed to display to the tutor
  // To get this I will start by selecting the amount of rows that exist within the content feedback table. The studnet can only get average feedback for content that they have completed.


  //Query and execution
  $getActualFeedback = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id'";
  $executeActualFeedback= $conn->query($getActualFeedback);

  // Get the number of rows that exist and then multiple that 5
  $existingFeedback = $executeActualFeedback->num_rows;
  // Place that in a variable called possible feedback as this is the total possible feedback score that the student could provide
  $possibleFeeback = ($existingFeedback * 5);


  // insert checks because the student may not have inserted anything yet which could be an issue.
  // If 0 rows are returned then the student will automatically be given a 0 and the function will return a 0

  if ($existingFeedback == 0 ) {
     $feebackData = 0;
  } else {
    // if the function does not retun a 0 then do this
    $rating_number = 0;
    while ($row=$executeActualFeedback->fetch_assoc()) {
      // loop through all the rating scores that the student has provided
      // then add them to the $rating_number var in each increment of the loop
      // cast the variable so that it is returned as an int
    $rating_number+= (int)$row['rating_number'];
    }


    // do the calulation so that a average figure is returned in an integer for
    $feebackData = ($rating_number/$possibleFeeback * 100);

    echo "<p>$rating_number</p>";
    echo "<p>$possibleFeeback</p>";



return $feebackData;


}

// call the function and place it into a local var so that it can be used to display to the tutor
$feedbackPercentage = $getActualFeedback($student_id);





}
```

Concerning displaying this data to the tutor, this was achieved by returning a word to the GUI screen. For example, if the getAverageFeedback(); function had returned an average rating of 60%, then the system will return the word 'Good'. See figure 5.4 to see the collection of if statements used so that the system returned the correct word that reflected the students' progress.

The General Feedback component of the system is a crucial attribute used to deliver feedback to the tutor, who can then make contact with the students. This was the first function that I developed to collect and display data. Many of the other functions that I used to display student data to the tutor are implemented in a similar nature but often displayed differently (More on this in the following component).

5.2.7 Student Registration

The student registration component of the system is used to track the student's attendance for the classes associated with this module. This is done in a similar nature to the Rating System component. This requires a form of input from the student. In theory, the component should work like this. The tutor will create a class that will call a function called getRandomKey(); function. This will then insert a random code into the database and the other data belonging to the class. The tutor has no option to not generate a code for the class. See Figure 5.5.

*Figure 5.2.8 depicting the getRandomKey(); function and the code for inserting it to MySQL*To register for the class, the student will go to the registration activity. They will then select the class they need to register for and enter the code given to them by the tutor. The tutor should hand out the code to the physically in the class (The tutor puts the code up at the end of the class on the screen). This has some potential flaws making it possible for students who didn't attend the class to register. That is not the problem that I am trying to solve.

Further development would have me place a time limit on how long the window of opportunity to register is to only register during the class. It would be possible to complete the lab whenever suited them, and in some cases, the system would be helpful to do that, especially in times such as the COVID-19. It should be noted that the content that is associated with that class will not be restricted if the student fails to register for the relevant class.

Name	**Description**
getRandomKey();	This function will generate a random key and then assign this to the class that the tutor just created. The key that was generate will be used to allow students to register for their classes.

5.2.9 Content Completion

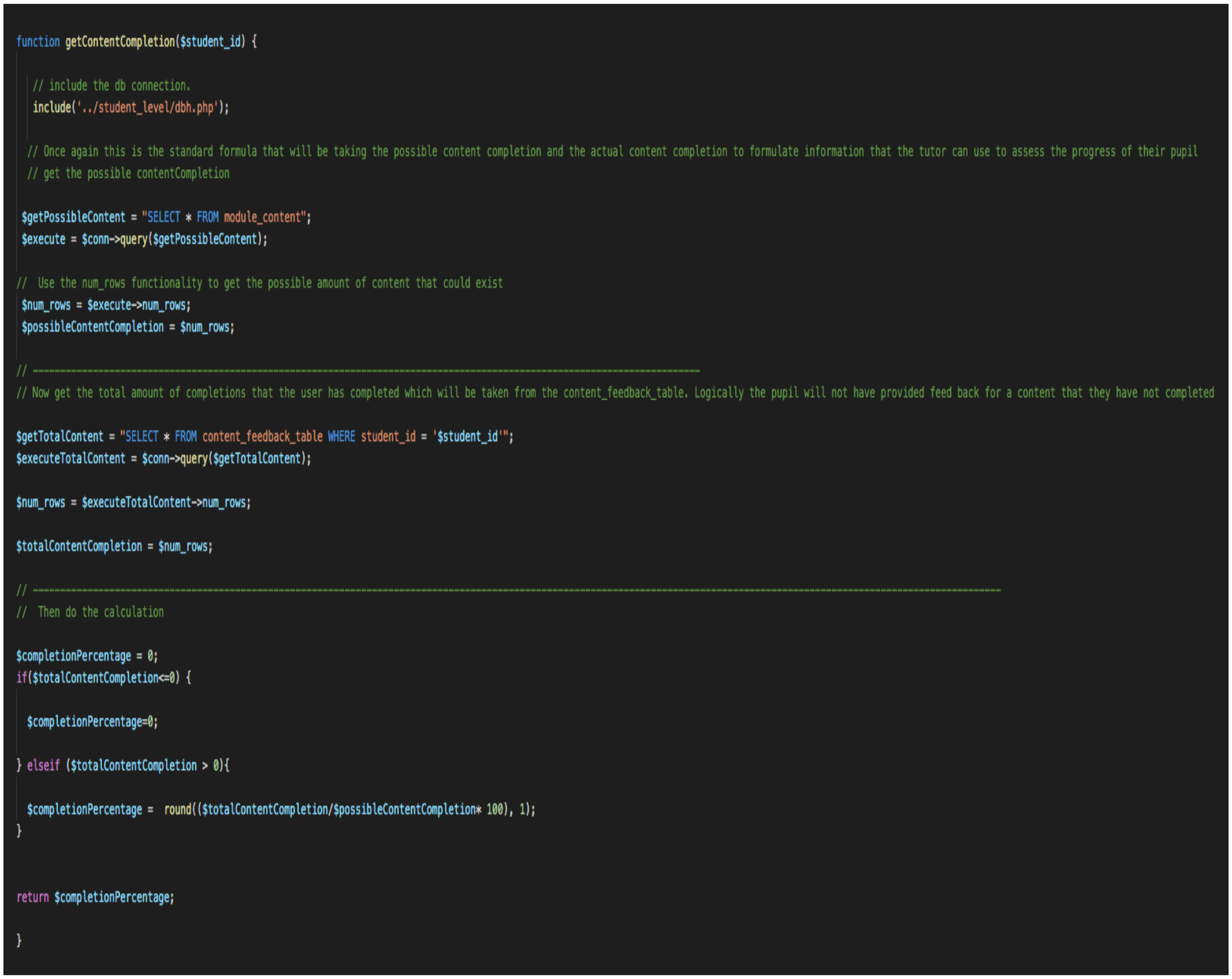
The Content Completion component of my system is to present data to the tutor concerning the student's completion rate of the content that has been set. Using the function getContentCompletion(); I was able to return a percentage and then present that in the HTML and CSS in a percentage bar. The function follows the same general idea of the previous getAverageFeedback();.

Name	**Description**
getContentCompletion();	This function will get the content completion rate of the student that is being viewed by the tutor. The function will compare the amount of rows feedback table and the amount of content that exists. I can they grab a percentage of the content that has been completed

To get the average completion rate, I needed to get the total amount of content in the system and then compare that to the amount of content the student has completed. I can determine if the student has completed the task if they have provided feedback through the feedback system. I needed to get the student data through a SELECT statement, pass the local variable stored into a num_rows function and then execute the equation to get the average. Once the average was completed and returned in a local variable, it was displayed to the user.

The returned value was displayed in a percentage bar to make this dynamic and easier to read for the tutor. This was easy to do as PHP can be written withing CSS. I simply passed the local variable that I used to call the getContentCompletion(); into the CSS in the width attribute of the progress bar.

Figure 5.2.10 depicting the getContentCompletion(); function.



```
function getContentCompletion($student_id) {

   // include the db connection.
   include('../student_level/dbh.php');

  // Once again this is the standard formula that will be taking the possible content completion and the actual content completion to formulate information that the tutor can use to assess the progress of their pupil
  // get the possible contentCompletion

 $getPossibleContent = "SELECT * FROM module_content";
 $execute = $conn->query($getPossibleContent);

//  Use the num_rows functionality to get the possible amount of content that could exist
 $num_rows = $execute->num_rows;
 $possibleContentCompletion = $num_rows;

// -----------------------------------------------------------------------------------------------------------------------
// Now get the total amount of completions that the user has completed which will be taken from the content_feedback_table. Logically the pupil will not have provided feed back for a content that they have not completed

$getTotalContent = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id'";
$executeTotalContent = $conn->query($getTotalContent);

$num_rows = $executeTotalContent->num_rows;

$totalContentCompletion = $num_rows;

// ------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
//  Then do the calculation

$completionPercentage = 0;
if($totalContentCompletion<=0) {

  $completionPercentage=0;

} elseif ($totalContentCompletion > 0){

  $completionPercentage =  round(($totalContentCompletion/$possibleContentCompletion* 100), 1);
}


return $completionPercentage;

}
```

A majority of the functionality in this Web Application centres around student feedback. Once they complete this, it is easy to display valuable data to the tutor to help them make important decisions to support their students. Many functions within this Web Application will show valuable data to the tutor. Other notable mentions found in the appendix are the student attendance(); function, found in the student profile activity. The generalAttendence(); that can be found in the tutor home page activity and the getModuleCompletionRate(); that can be found in the tutor module activity.

5.3 Testing Strategies

A crucial part of the Software Development lifecycle is the testing strategies employed to make sure the system works as required. Testing Strategies depend on the type of system that is being developed.

For a Web Application, the testing strategies that I am going to implement are as follows:

1. Functionality Testing: This tests several parameters such as the user interface, security testing, database testing and other basic website functionality.
2. Usability Testing: This would ensure that the website's navigation works with no spelling or grammatical errors.
3. Interface Testing: Test requests are sent to the correct location in the database. Errors must be caught by the system, and an appropriate message given to the user about the result. No server-side errors should be displayed to the user but only to the administrator. Ensure that the queries being executed to the database return the expected results by testing them in the database interface.
4. Database Testing: Testing to see if any errors appear while executing queries. Data integrity is maintained while creating, updating or deleting the data in the database. Test data retrieved from the database is relevant and display accurately to the user.
5. Security Testing: Testing that unauthorised access to secure pages should not be permitted. Restricted Files should not be downloaded without the appropriate access. Checking that the sessions are automatically killed if there is a lack of user activity.
6. Component Testing: This is the longest stage of testing. This confirms that each individual component of which makes up the system will do what it is implemented to do. All the components that are listed in the appendix will be tested so that they meet all their requirements.

By implementing individual component testing we can ensure that Interface Testing and Database testing are completed. Logically if the component does what it is required to do then the database queries are sound along with the Interface.

Many other testing strategies could be used, but many require other resources that I simply do not have at my disposal. I will be focusing on testing that I can carry out by myself, which will help improve the quality of my Web Application.

5.3.1 Justification of Testing Strategies

These testing strategies can be justified as they provide an essential checklist to develop the overall useability of my website. Due to the lack of resources that I have at my disposal, general functionality testing is exhausted. Due to the pandemic, there was very few people that I could use to acceptance test my software. All of the strategies listed above could be carried out by the developer.

6.1 Overview

This chapter will be the conclusion for the entire project. I will drawback to the specification requirements in the first chapter as a measurement of success. This chapter will outline the success and shortcomings of this project and the suitability of the technologies that have been used to implement the functions. The chosen development model discussed in chapter 2 will also be evaluated. Finally, we will outline other functions and further developments that would be crucial for this application to be considered successful.

6.2 Success of the Project

I outline the functional requirements and the non-functional requirements in the original specification. For the most part, the non-functional requirements stayed the same when outlined in Chapter 3. The core functionality and the problem that needed to be solved has undoubtedly be solved. That is the projects most tremendous success.

However, some changed that had to be executed when it comes to some functional requirements and other functions were discarded to successfully implement the most critical components of the project. The discarded requirements didn't directly solve the problem but would have made the application more useful.

The requirements of the student lab progress monitor stated that the application should:

1. Record the Attendance of the Students
2. Record data in Completion of Lab Exercises
3. Progress with the Current Topic
4. The need for additional support
5. The application should provide tutors with relevant and timely information to facilitate support when needed.

I believe that the application does meet most of the requirements outlined in the proposal. The application prompts the user to record their data in the form of a star function that will return that data to the tutor. This data is manipulated to best express how the student is performing at their lab exercises. This is the core function of the entire application, and without this, the application is obsolete. All the aspects of the core functionality have been implemented to solve the problem. This is the main reason why this project is mainly successful.

This project took a lot of independent work and research of concepts of Software Development that had not been covered in the modules during the previous academic year. My understanding of JavaScript, CSS and HTML has improved, along with my problem-solving skills, creativity and resilience when a setback is suffered.

6.3 Development Languages

I have used traditional Full-Stack Development Technologies to carry out the development of this application. These were the best-suited combinations of technologies used to develop a web application. These technologies are as follows, HTML, CSS, JavaScript, JQuery, Ajax, PHP and MYSQL. I have thoroughly enjoyed using these technologies, and I hope to expand my knowledge in these technologies as soon as this project is completed. I have been forced to step outside of my comfort zone with these languages. I have often been forced to think for days about how I can make the components work to solve the individual problem.

6.4 Shortcoming of this Project.

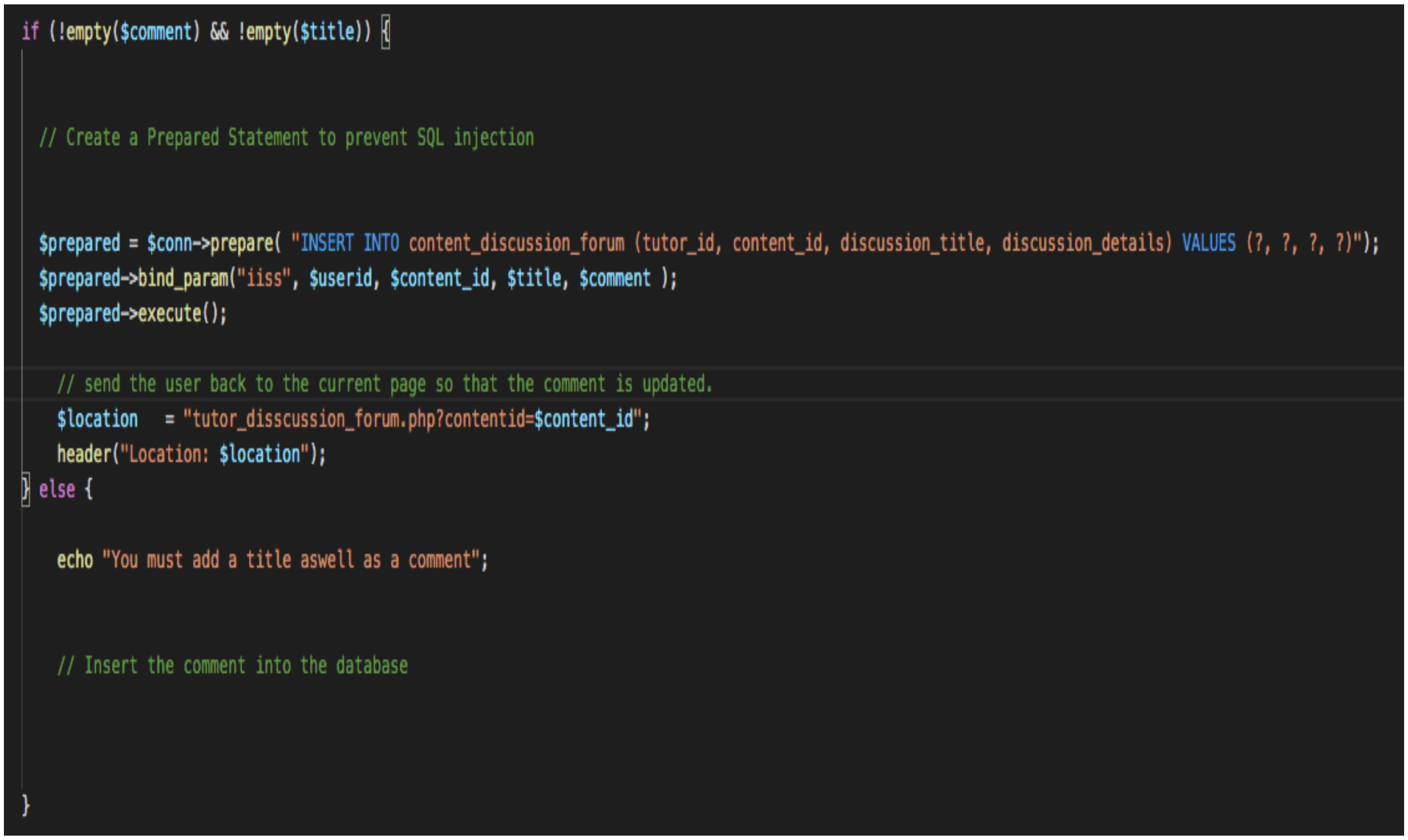
1. Security and SQL Injection

Many of the shortcomings of this project relate to the non-functional requirements needed for a live and professional web application. The biggest issue with this project is the security of the application. Due to time constraints, resources and the priority to implement the core functionality, only the most basic security aspects were implemented. I made certain web pages were locked, and that access to the website was only for authorised users. For the most part, the application is wide open to MySQL injection.

I have attempted to implement components that would protect against protentional harmful injections in some cases. MySQL injections have long been a criticism of PHP. I have learned that the best way to combat MySQL injection is using prepared statements in the PHP code.

Using a prepared statement allows for protecting personalised data and malicious users from altering your database. I have learned how to use prepared statements and that they are essential for gathering input from a user. Figure 6.4.1 is a prepared statement written so that the application can capture the student's comments or tutor for the discussion forum.

Figure 6.4.1 Depicting a Prepared Statement and avoid MySQL injection.



```
if (!empty($comment) && !empty($title)) {


  // Create a Prepared Statement to prevent SQL injection


  $prepared = $conn->prepare( "INSERT INTO content_discussion_forum (tutor_id, content_id, discussion_title, discussion_details) VALUES (?, ?, ?, ?)");
  $prepared->bind_param("iiss", $userid, $content_id, $title, $comment );
  $prepared->execute();

    // send the user back to the current page so that the comment is updated.
    $location   = "tutor_disscussion_forum.php?contentid=$content_id";
    header("Location: $location");
} else {

    echo "You must add a title aswell as a comment";


    // Insert the comment into the database



}
```

2. **Chat Component**

Another shortcoming of this application is the lack of private communication that a tutor and student can have concerning the tutor sets' labs. There is no private chat that exists. I believe that to improve this application's experience and productivity, this would need to be the next implemented.

3. **Feedback Representation and the UI**

The way the feedback is communicated to the tutor should be improved by developing a higher standard UI. Unfortunately, I have not thought of how this should be communicated through the UI. However, I understand that this requires looking at similar applications. The use of progression bars and unique words will suffice for this prototype, but further development would be mandatory.

<u>This use of Restful APIs</u>

A commonly new way that applications like the one I have developed are being implemented is in by using Restful APIs. This was not something that was covered in my original Web Development module and once I had learnt what this technology was I had already drafted the Software Requirements and begun implementing the code.

According to the PHP website this is a description of what an API is…

An Application Programming Interface, or API, defines the classes, methods, functions and variables that your application will need to call in order to carry out its desired task. In the case of PHP applications that need to communicate with databases the necessary APIs are usually exposed via PHP extensions.

APIs can be procedural or object-oriented. With a procedural API, you call functions to carry out tasks, with the object-oriented API you instantiate classes and then call methods on the resulting objects. Of the two, the latter is usually the preferred interface, as it is more modern and leads to better-organized code.

This is a useful technology that is best suited to organize and easily call code that needs to be reused. When this Project is complete, I will be hoping to gain experience using this technology

4. **<u>Increased Useability</u>**
   Another shortcoming of the project is lack of information in relation to the user navigation bar. I was hoping to use a bootstrap framework to solve the problem. Unfortunately, the framework was starting to override my own CSS so it had to be removed. There was not enough time for me to create hover functionality using JavaScript to make the application more user friendly by allowing the names of pages to pop up when hovered over. Apart from this minor shortcoming I think that the useability of the application is very solid. This could easily have been implemented with more time

Bibliography

Bell, Douglas. *Software Engineering for Students*. 4th ed., Addison-Wesley.

Mall, Rajib. *Fundamentals of Software Engineering*. 3rd ed., PHI Learning, 2005.

Sommerville, Ian. *Software Engineering*. Tenth, Pearson.

Appendix

This Appendix is used to document testing and the individual components that make this application work. Due to time constraints, I have only included primary functions. The remaining code can be found in the source files for further reference.

Tutor Level – Code Appendix

General Attendance Component

This component will show the tutor the amount general attendance of the students in the classes that have been created for the semester so far. To make the code clearer I split some of the key code into functions so that they can be called multiple times if necessary. This is the full attendance function.

```
//  Create the functions that will be needed to display the statistics for the tutor to view and make decision on

//  Lets start with class attendance


function fullAttendance() {

    include('../student_level/dbh.php');

    //  This will display to the user what the attendance for the entire semester


    // Get the number of classes from MySQL Database

    $getClasses = "SELECT * FROM class_list";

    $execute = $conn->query($getClasses);
```

```
$numClasses = $execute->num_rows;


// Get the number of students
$getStudents = "SELECT * FROM student_details";
$execute2 = $conn->query($getStudents);


$numStudents = $execute2->num_rows;


// The number of attend is $numStudents multiplied by $numClasses
$possibleAttends = $numStudents * $numClasses;


// get the total attends so far


$totalAttends = "SELECT * FROM class_registration";


$getAttends = $conn->query($totalAttends);


$allAttends = $getAttends->num_rows;


// calulate the attendance
```

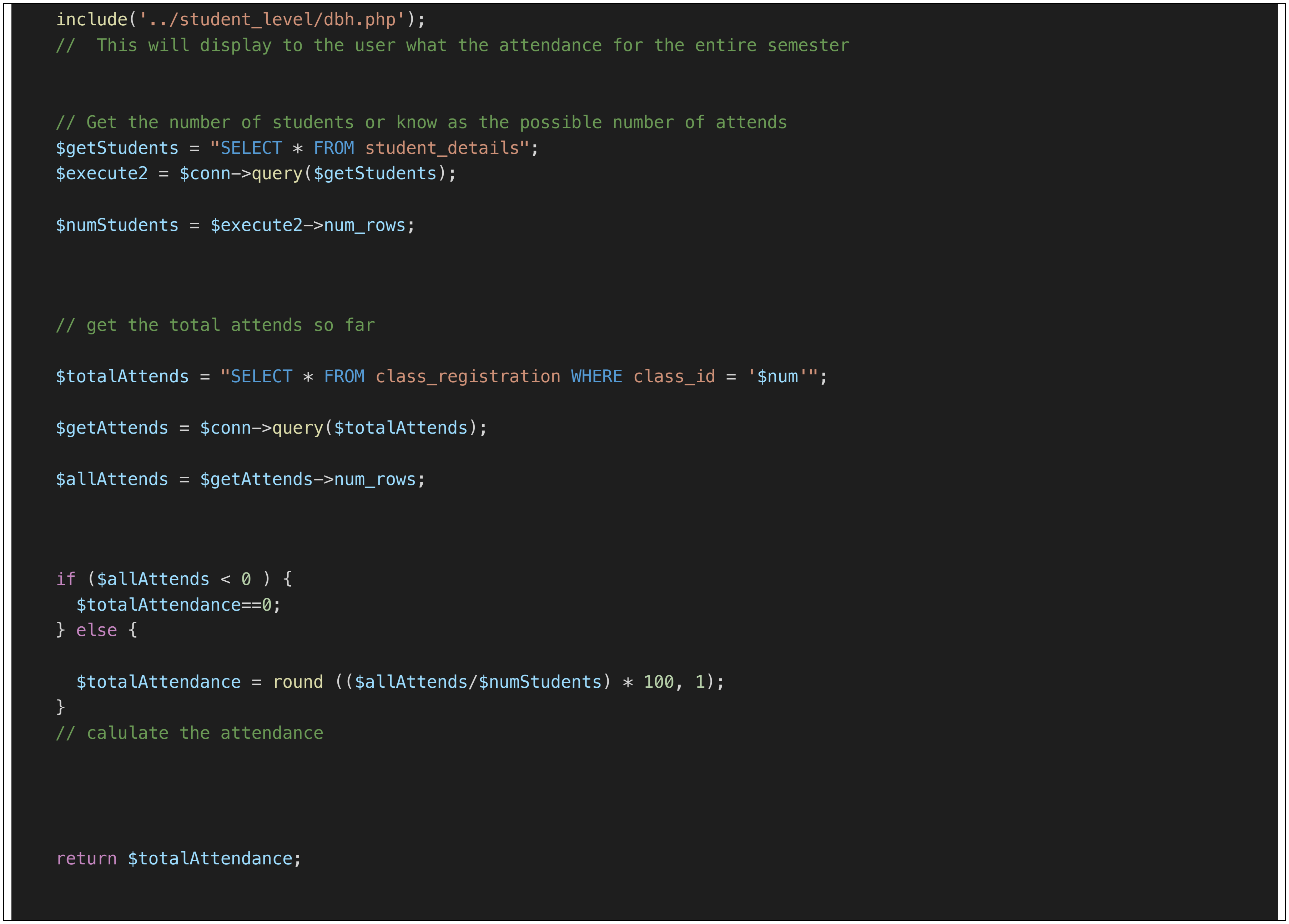
```
    if ($allAttends < 0 ) {
      $generalAttendance==0;
    } else {

      $generalAttendance =  round(($allAttends/$possibleAttends) * 100, 1);
    }


    return $generalAttendance;


}





<div class='container darker'>



        <h2> General Attendence</h2>
        <?php


         $showAttendance = fullAttendance();
         echo "<div class='progress-g'>";

echo "<div class='progress-done-g' data-done='$showAttendance'> $showAttendance%";
         echo '</div>';

         echo "</div>";

        ?>
```

<u>Class Attendance Component</u>

This is the individual class attendance component that is used so that the tutor can determine what the individual class attendance stands at.

To make the code clearer, I split some of the key code into functions so that they can be called multiple times if necessary. This is the class attendance function.

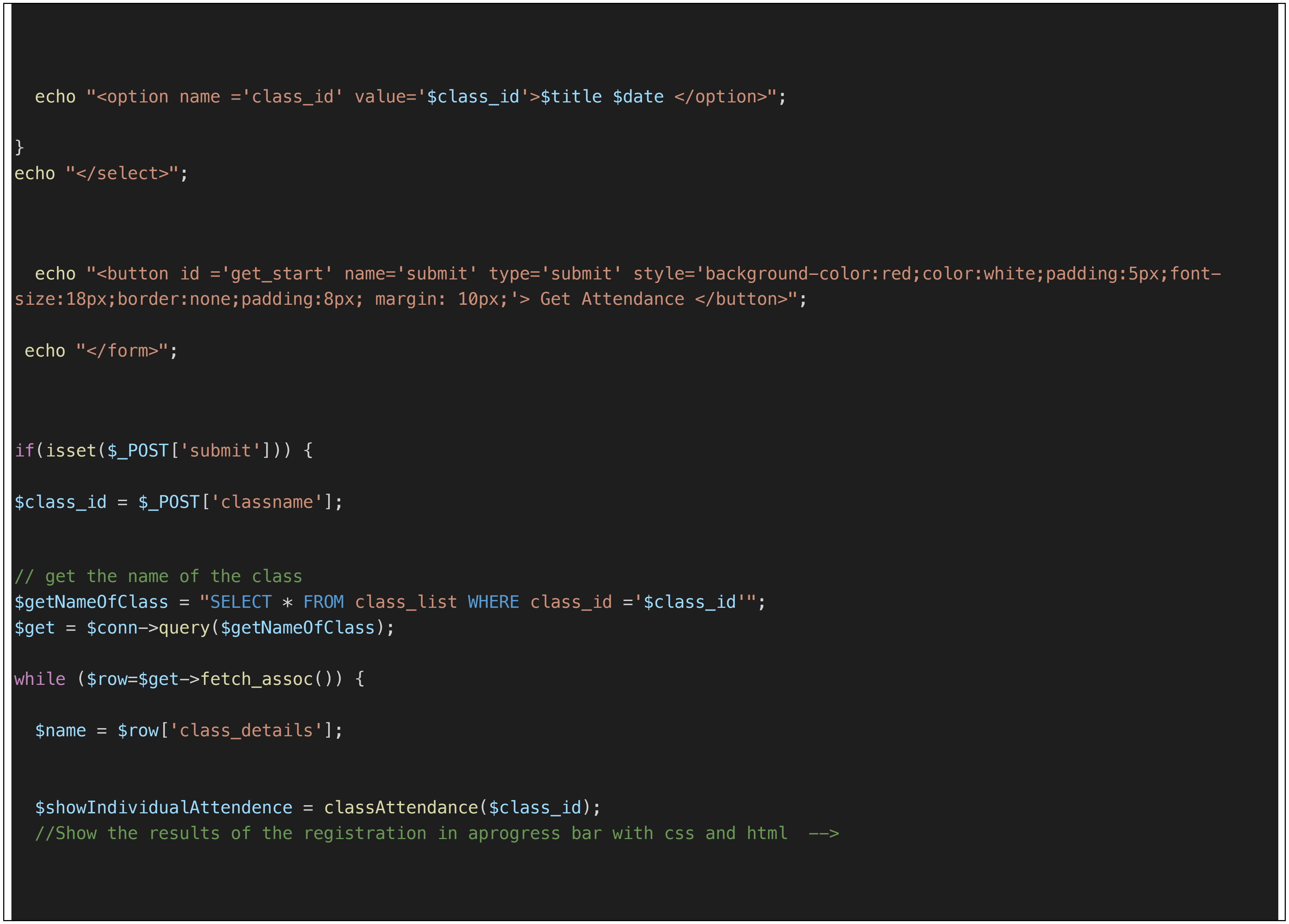
```
function classAttendance($num) {
```



```
include('../student_level/dbh.php');
//  This will display to the user what the attendance for the entire semester


// Get the number of students or know as the possible number of attends
$getStudents = "SELECT * FROM student_details";
$execute2 = $conn->query($getStudents);

$numStudents = $execute2->num_rows;



// get the total attends so far

$totalAttends = "SELECT * FROM class_registration WHERE class_id = '$num'";

$getAttends = $conn->query($totalAttends);

$allAttends = $getAttends->num_rows;



if ($allAttends < 0 ) {
  $totalAttendance==0;
} else {

  $totalAttendance = round (($allAttends/$numStudents) * 100, 1);
}
// calulate the attendance




return $totalAttendance;
```

```
}
?>
```

```
<div class='container darker'>

<h2> Check individual class Attendence</h2>



<form action="tutor_home.php" class="update-resource" method ="POST">


<select  placeholder= "Class Name" name="classname" >

<?php


// put this code into a for loop to display all the conetent of the  classes on the screen




$classList = "SELECT * FROM class_list";
 $result = $conn->query($classList);
 echo "<option> Select a Class </option>";
 while($row=$result->fetch_assoc()) {
   $class_id = $row['class_id'];
   $title = $row['class_details'];
   $date = $row['class_date'];
```

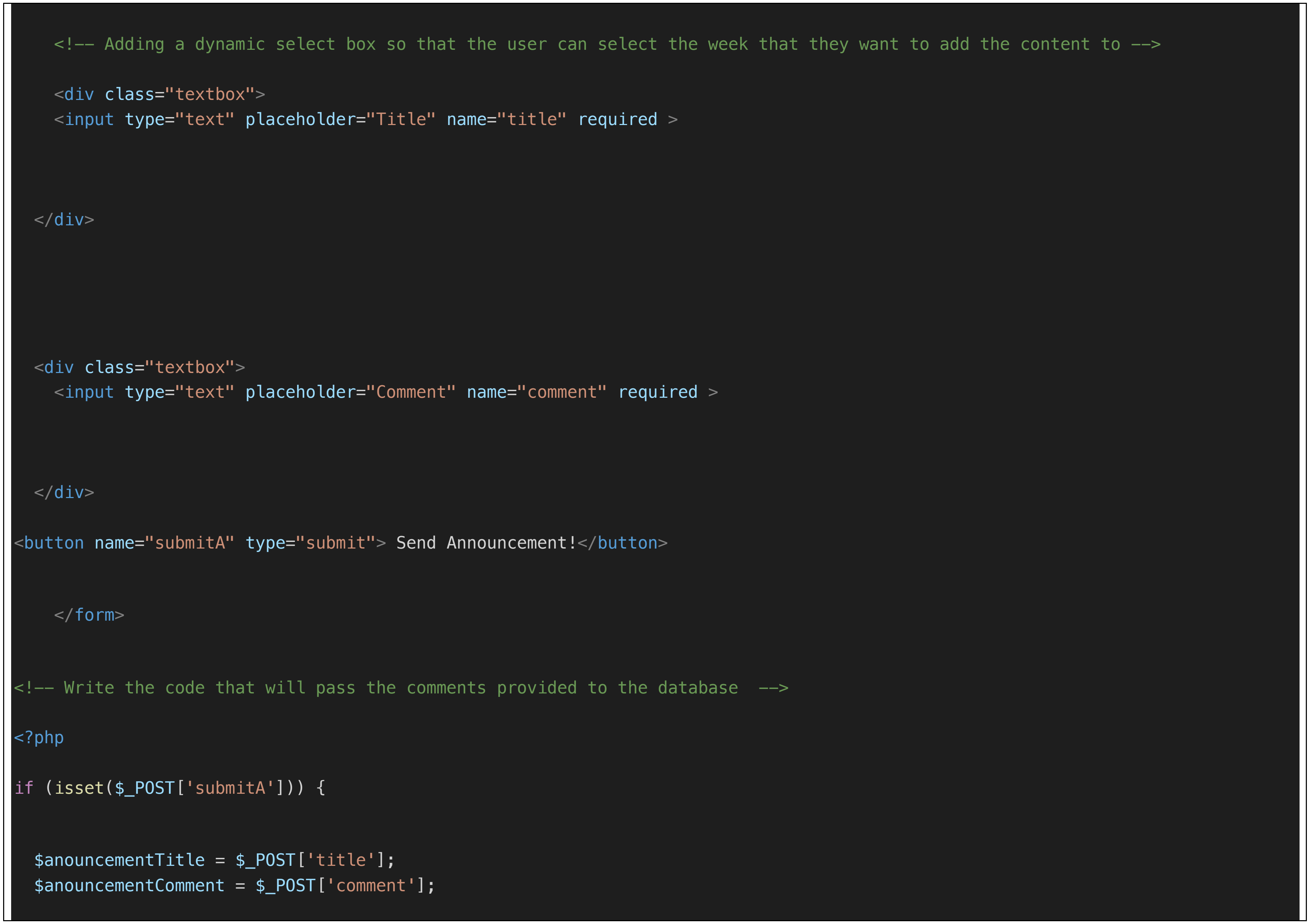


```
  echo "<option name ='class_id' value='$class_id'>$title $date </option>";

}
echo "</select>";



  echo "<button id ='get_start' name='submit' type='submit' style='background-color:red;color:white;padding:5px;font-
size:18px;border:none;padding:8px; margin: 10px;'> Get Attendance </button>";

 echo "</form>";



if(isset($_POST['submit'])) {

$class_id = $_POST['classname'];


// get the name of the class
$getNameOfClass = "SELECT * FROM class_list WHERE class_id ='$class_id'";
$get = $conn->query($getNameOfClass);

while ($row=$get->fetch_assoc()) {

  $name = $row['class_details'];


  $showIndividualAttendence = classAttendance($class_id);
  //Show the results of the registration in aprogress bar with css and html  -->
  echo "<p>  $name </p>";
  echo "<div class='progress'>";
  echo "<div class='progress-done' data-done='$showIndividualAttendence'  style = 'background: linear-gradient(to left, #F2709C, #FF9472) ;height:100%;width:<?php echo $showIndividualAttendence ?>%  ;border-radius: 20px;height: 100%; display: flex; align-items:center; justify-content: center; animation: progress-done 2s; '> $showIndividualAttendence%";

  echo '</div>';

  echo "</div>";

}






}
```

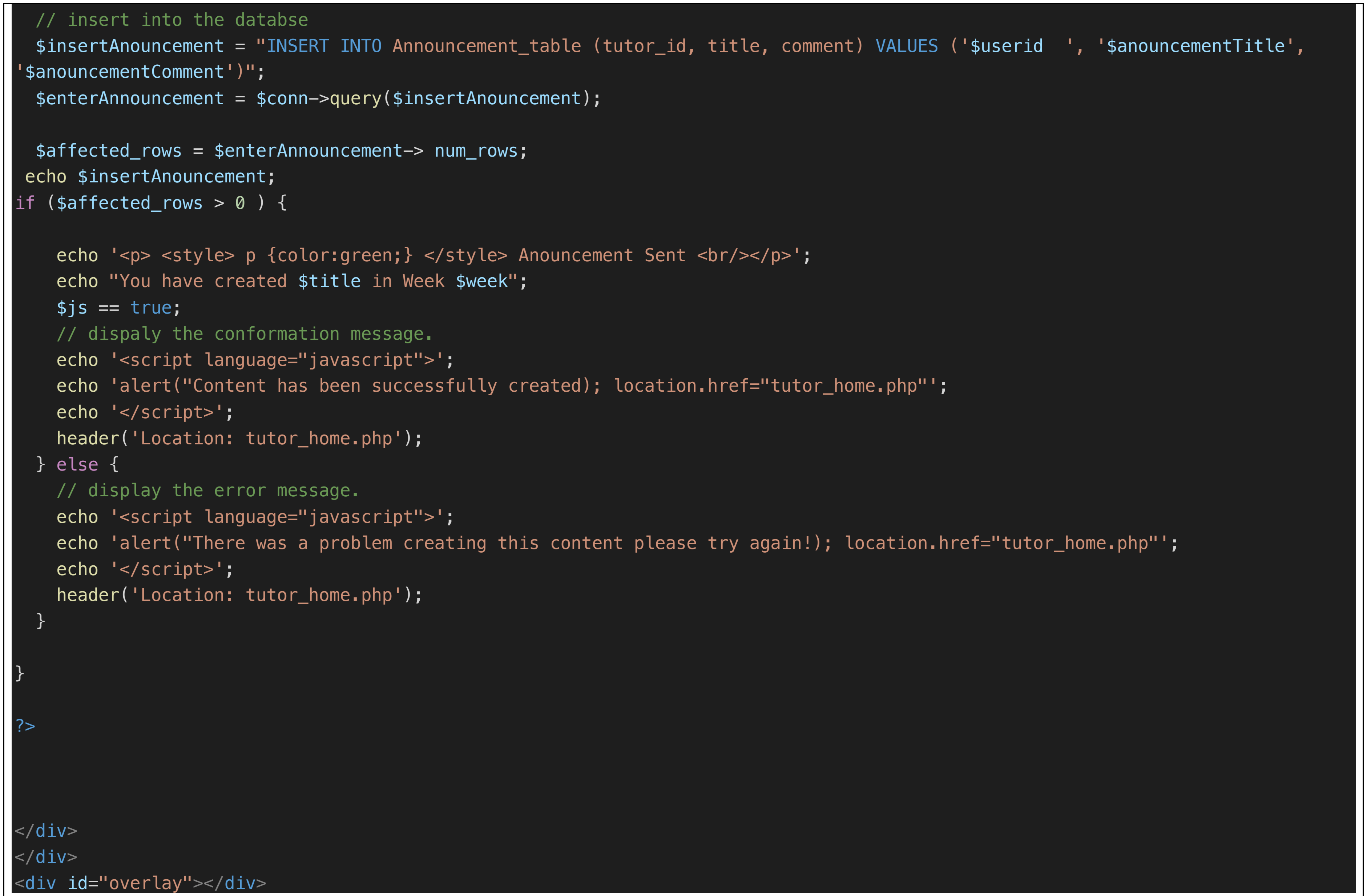
Add Announcement Component.

This Component allows for the user to be able to add announcement to system which can be viewed but other users. The component creates a pop up window on the page for the tutor to be able to add a new announcement.

```
<div class="modal" id="modal">
    <div class="modal-header">
      <div class="title"> Make Anouncement Here! </div>
      <button data-close-button class="close-button">&times;</button>
    </div>
    <div class="modal-body">
    <form action="tutor_home.php" class="form" method = "POST">
```



```
    <!-- Adding a dynamic select box so that the user can select the week that they want to add the content to -->

    <div class="textbox">
    <input type="text" placeholder="Title" name="title" required >



  </div>






  <div class="textbox">
    <input type="text" placeholder="Comment" name="comment" required >



  </div>

<button name="submitA" type="submit"> Send Announcement!</button>


    </form>


<!-- Write the code that will pass the comments provided to the database  -->

<?php

if (isset($_POST['submitA'])) {


  $anouncementTitle = $_POST['title'];
  $anouncementComment = $_POST['comment'];
```



```
  // insert into the databse
  $insertAnnouncement = "INSERT INTO Announcement_table (tutor_id, title, comment) VALUES ('$userid  ', '$anouncementTitle',
'$anouncementComment')";
  $enterAnnouncement = $conn->query($insertAnouncement);

  $affected_rows = $enterAnnouncement-> num_rows;
 echo $insertAnouncement;
if ($affected_rows > 0 ) {

    echo '<p> <style> p {color:green;} </style> Anouncement Sent <br/></p>';
    echo "You have created $title in Week $week";
    $js == true;
    // dispaly the conformation message.
    echo '<script language="javascript">';
    echo 'alert("Content has been successfully created); location.href="tutor_home.php"';
    echo '</script>';
    header('Location: tutor_home.php');
  } else {
    // display the error message.
    echo '<script language="javascript">';
    echo 'alert("There was a problem creating this content please try again!); location.href="tutor_home.php"';
    echo '</script>';
    header('Location: tutor_home.php');
  }

}

?>




</div>
</div>
<div id="overlay"></div>
```

JavaScript

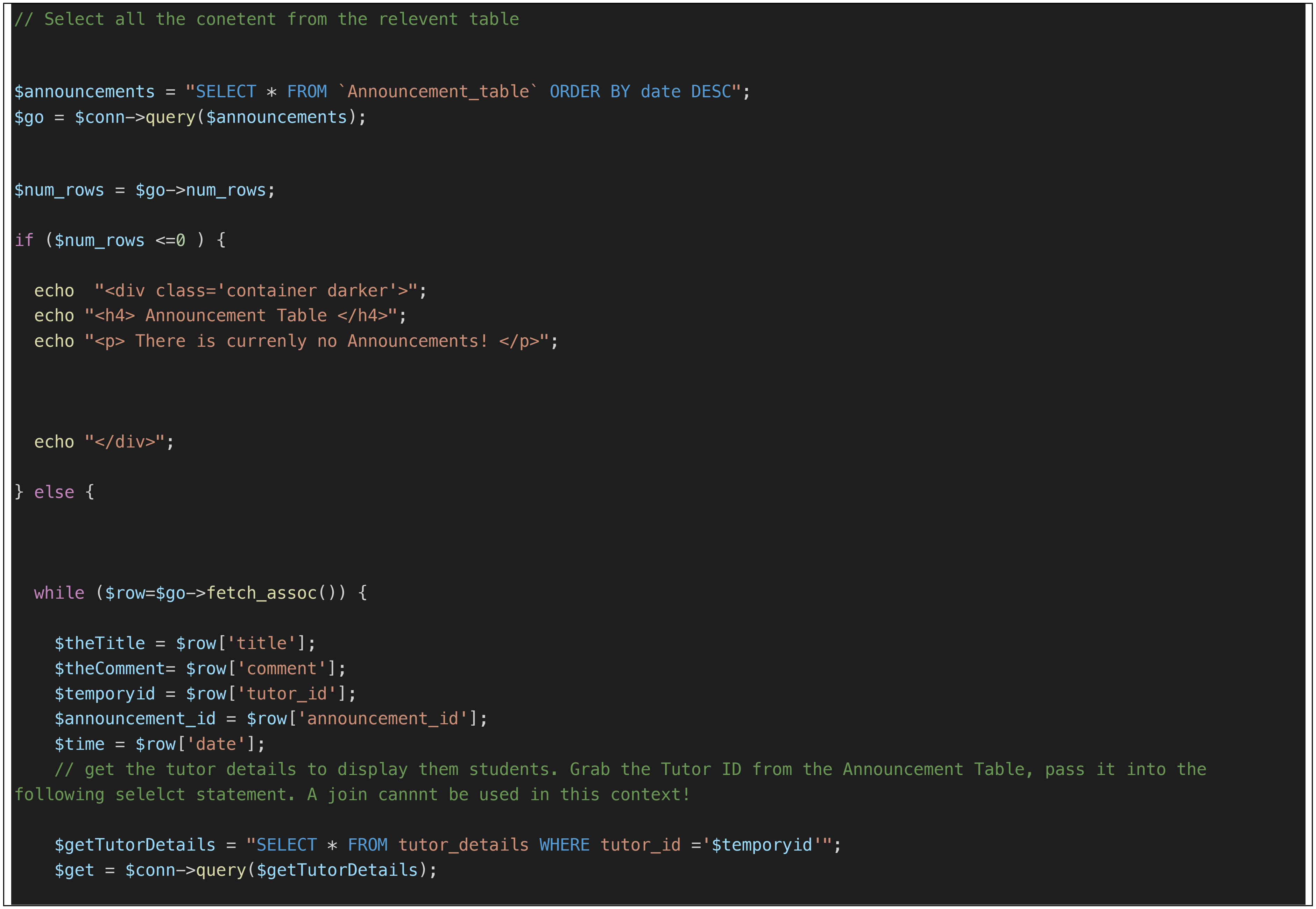
```javascript
const openModalButtons = document.querySelectorAll('[data-modal-target]')
const closeModalButtons = document.querySelectorAll('[data-close-button]')
const overlay = document.getElementById('overlay')
const submitData = document.querySelectorAll('[data-submit]')

openModalButtons.forEach(button => {
  button.addEventListener('click', () => {
    const modal = document.querySelector(button.dataset.modalTarget)
    openModal(modal)
  })
})

overlay.addEventListener('click', () => {
  const modals = document.querySelectorAll('.modal.active')
  modals.forEach(modal => {
    closeModal(modal)
  })
})

submitData.forEach(button => {
  button.addEventListener('click', () => {
    const modal = document.querySelector(button.dataset.modalTarget)
    openModal(modal)
  })
})

closeModalButtons.forEach(button => {
  button.addEventListener('click', () => {
    const modal = button.closest('.modal')
    closeModal(modal)
  })
})
```

```
function openModal(modal) {
  if (modal == null) return
  modal.classList.add('active')
  overlay.classList.add('active')
}

function closeModal(modal) {
  if (modal == null) return
  modal.classList.remove('active')
  overlay.classList.remove('active')
}
```

View Announcement Function

This component will allow for tutors to view the announcements that they have made to their students. This will be returned in order of how recent the tutor created the announcement. This can be found in tutor_level/tutor_homepage.php

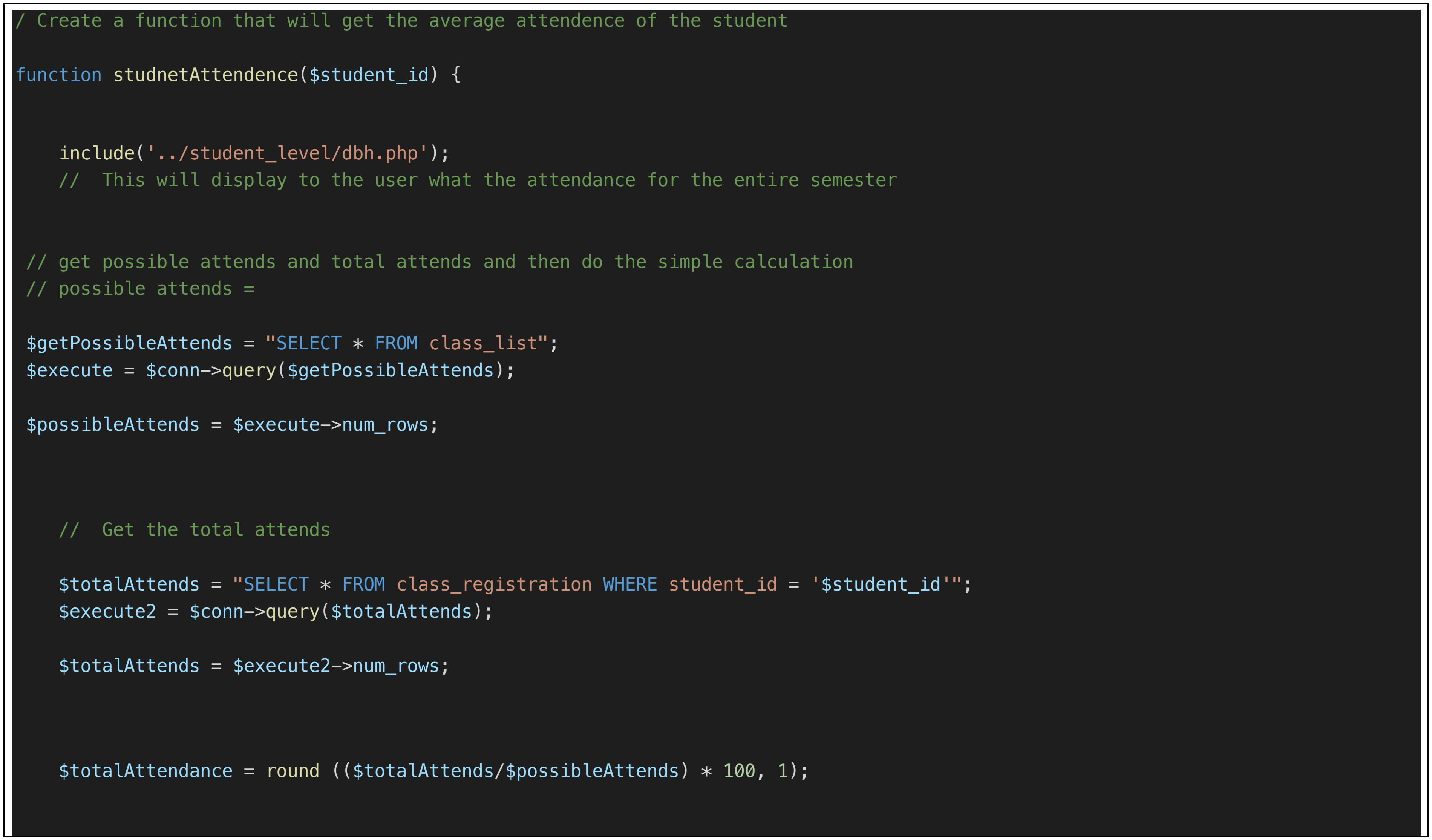
```
<!-- This is the code that will show the tutor the amount announcements that they have made recently to their students -->
<!-- This is the container that will hold all the announcements for the platform -->
<div class="container darker">

<div class="page-title">
<h4>Announcements </h4>

</div>




<?php
```



```
// Select all the conetent from the relevent table


$announcements = "SELECT * FROM `Announcement_table` ORDER BY date DESC";
$go = $conn->query($announcements);


$num_rows = $go->num_rows;

if ($num_rows <=0 ) {

  echo  "<div class='container darker'>";
  echo "<h4> Announcement Table </h4>";
  echo "<p> There is currenly no Announcements! </p>";



  echo "</div>";

} else {



  while ($row=$go->fetch_assoc()) {

    $theTitle = $row['title'];
    $theComment= $row['comment'];
    $temporyid = $row['tutor_id'];
    $announcement_id = $row['announcement_id'];
    $time = $row['date'];
    // get the tutor details to display them students. Grab the Tutor ID from the Announcement Table, pass it into the
following selelct statement. A join cannnt be used in this context!

    $getTutorDetails = "SELECT * FROM tutor_details WHERE tutor_id ='$temporyid'";
    $get = $conn->query($getTutorDetails);
    echo "<div class='container darker'>";


    while ($row2=$get->fetch_assoc()) {
      $fname = $row2['tutor_first_name'];
      $sname = $row2['tutor_second_name'];
      $file = $row2['imgpath'];

      echo "<img src='$file' alt='Avatar'>";
      echo "<p> Tutor : $fname $sname</p>";
    }
    echo "<p>$theTitle</p>";

    echo "<p>$theComment</p>";
    echo "<span class='time-right'>$time</span>";
    echo "<button id = 'module-content-delete' style=''> <a
href='php_functionality/delete_announcement.php?contentid=$announcement_id'> Delete </a>   </button>";





    echo "</div>";

  }
}
?>
```

Individual Student Attendance Component.

This component will return the individual attendance of the student on the student profile page.

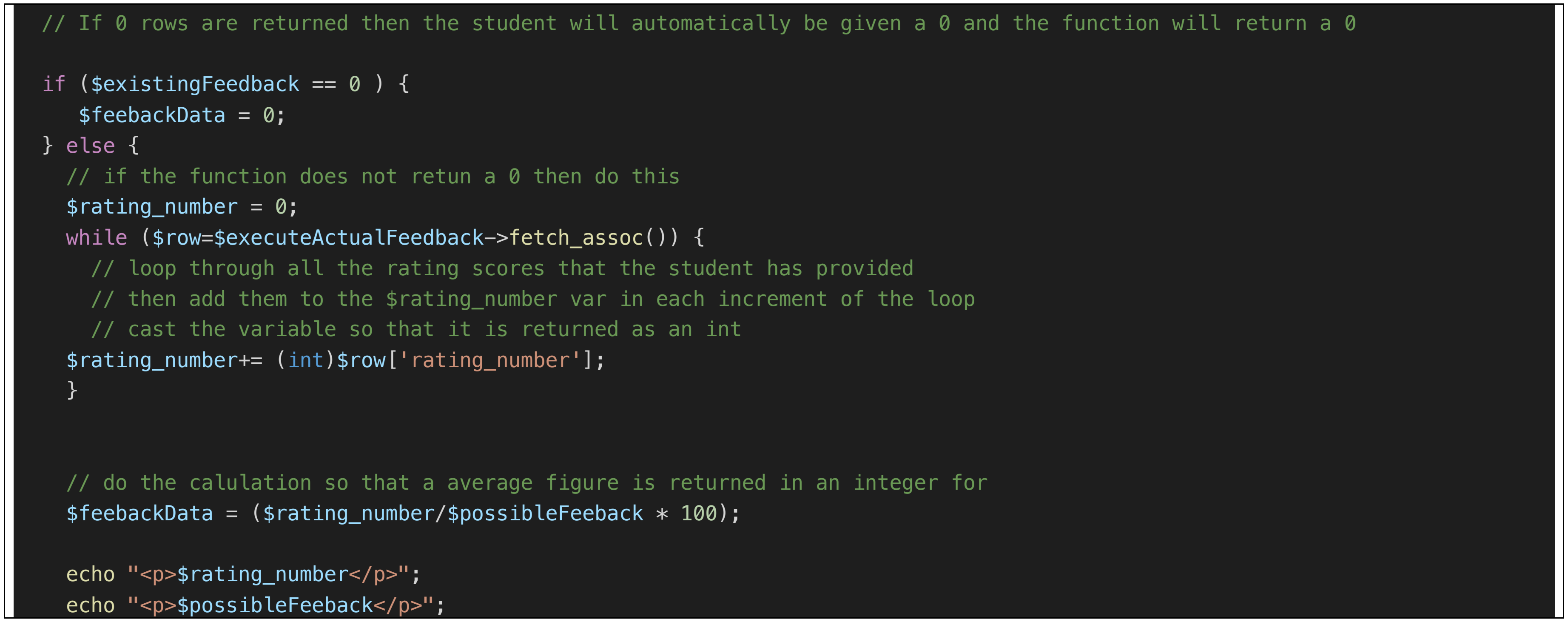


```
/ Create a function that will get the average attendence of the student

function studnetAttendence($student_id) {


    include('../student_level/dbh.php');
    //  This will display to the user what the attendance for the entire semester


 // get possible attends and total attends and then do the simple calculation
 // possible attends =

 $getPossibleAttends = "SELECT * FROM class_list";
 $execute = $conn->query($getPossibleAttends);

 $possibleAttends = $execute->num_rows;



    //  Get the total attends

    $totalAttends = "SELECT * FROM class_registration WHERE student_id = '$student_id'";
    $execute2 = $conn->query($totalAttends);

    $totalAttends = $execute2->num_rows;



    $totalAttendance = round (($totalAttends/$possibleAttends) * 100, 1);
    return $totalAttendance
```

Average Content Feedback

This is the Average Content Feedback Component which is used to determine the average feedback of the student in relation to all the feedback that they have given on the content that they have completed.

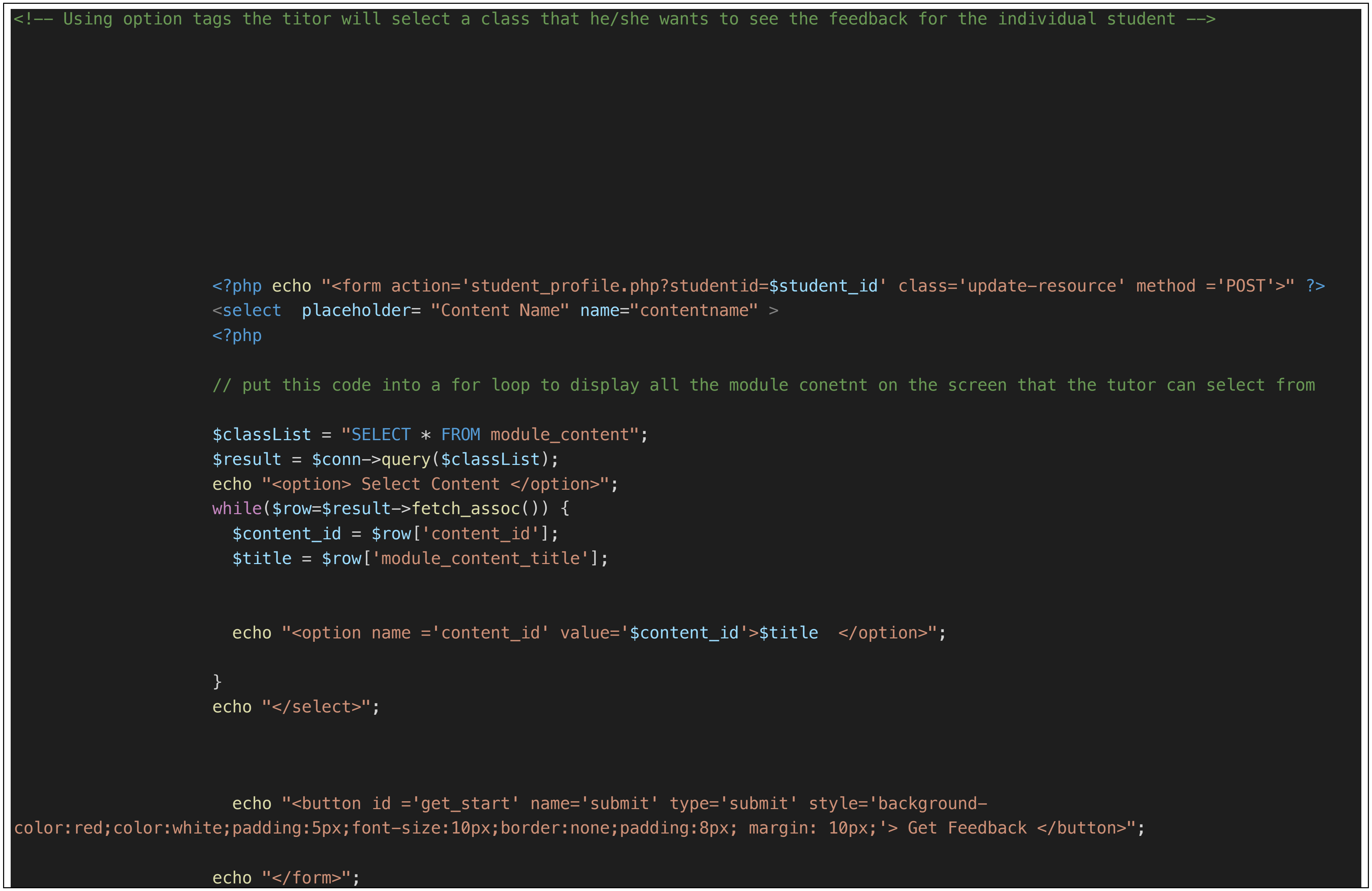
```
unction getAverageFeedback($student_id) {
  // include the db connection.
  include('../student_level/dbh.php');

  // I need to create a equation to get the average Feedback for the student feedback which I will use to calculte the data needed to display to the tutor
  // To get this I will start by selecting the amount of rows that exist within the content feedback table. The studnet can only get average feedback for content that they have completed.


  //Query and execution
  $getActualFeedback = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id'";
  $executeActualFeedback= $conn->query($getActualFeedback);

  // Get the number of rows that exist and then multiple that 5
  $existingFeedback = $executeActualFeedback->num_rows;
  // Place that in a variable called possible feedback as this is the total possible feedback score that the student could provide
  $possibleFeeback = ($existingFeedback * 5);


  // insert checks because the student may not have inserted anything yet which could be an issue.
```



```
// If 0 rows are returned then the student will automatically be given a 0 and the function will return a 0

if ($existingFeedback == 0 ) {
   $feebackData = 0;
} else {
  // if the function does not retun a 0 then do this
  $rating_number = 0;
  while ($row=$executeActualFeedback->fetch_assoc()) {
    // loop through all the rating scores that the student has provided
    // then add them to the $rating_number var in each increment of the loop
    // cast the variable so that it is returned as an int
  $rating_number+= (int)$row['rating_number'];
  }


  // do the calulation so that a average figure is returned in an integer for
  $feebackData = ($rating_number/$possibleFeeback * 100);

  echo "<p>$rating_number</p>";
  echo "<p>$possibleFeeback</p>";
```

<u>Individual Content Feedback</u>

This is the component that will be used so that the tutor can view the individual content feedback in relation to the content that has been provided. This will return a word as a representation as an indicator of how the student is performing.



```
<!-- Using option tags the titor will select a class that he/she wants to see the feedback for the individual student -->










                    <?php echo "<form action='student_profile.php?studentid=$student_id' class='update-resource' method ='POST'>" ?>
                    <select  placeholder= "Content Name" name="contentname" >
                    <?php

                    // put this code into a for loop to display all the module conetnt on the screen that the tutor can select from

                    $classList = "SELECT * FROM module_content";
                    $result = $conn->query($classList);
                    echo "<option> Select Content </option>";
                    while($row=$result->fetch_assoc()) {
                      $content_id = $row['content_id'];
                      $title = $row['module_content_title'];


                      echo "<option name ='content_id' value='$content_id'>$title  </option>";

                    }
                    echo "</select>";



                      echo "<button id ='get_start' name='submit' type='submit' style='background-
color:red;color:white;padding:5px;font-size:10px;border:none;padding:8px; margin: 10px;'> Get Feedback </button>";

                    echo "</form>";
                    if (isset($_POST['submit'])) {
                      // Select the student feedback concern this content in the module
                        $content_id = $_POST['contentname'];

                      if ($content_id == 0) {
                        echo '<p> <style> color:black; </style> Nothing Selected </p>';

                      } else {


                      $getContentFeedBack = "SELECT * FROM content_feedback_table INNER JOIN module_content ON
content_feedback_table.content_id = module_content.content_id WHERE student_id = '$student_id' AND content_feedback_table.content_id =
'$content_id' AND content_feedback_table.rating_number IS NOT NULL";
                      $execute = $conn->query($getContentFeedBack);
                      $rating = null;

                      while ($row=$execute->fetch_assoc()) {
                        $rating = (int) $row['rating_number'];
                        $name = $row['module_content_title'];



                        // Construct the If else statement that will display to the tutor what the stading is for the student in
relation to this module
                        echo "<p> $name :  </p>";
                        if ($rating==0 || $rating == null) {
                          echo '<p> <style> color:black;</style> No Feedback Given </p>';
                        } elseif ($rating == 1 ) {
                          echo '<p> <style> color:red; </style> Poor  </p>';
                        } elseif ($rating ==  2 ) {
                          echo '<p> <style> color:red; </style> Unsatisfactory  </p>';
                        } elseif ($rating  == 3) {
                          echo '<p> <style> color:gold; </style> Satisfactory </p>';
                        } elseif ($rating == 4) {
```

```
            echo '<p> <style> color:green; </style> Good </p>';
          } elseif ($rating == 5) {
            echo '<p> <style>  color: green; </style> Excellent </p>';
          }  elseif ($feedbackPercentage > 5 ) {
            echo '<p> <style> color: green; </style> Excellent </p>';
          }

        }

      }
    }

      ?>

</tr>
```

<u>Content Completion</u>

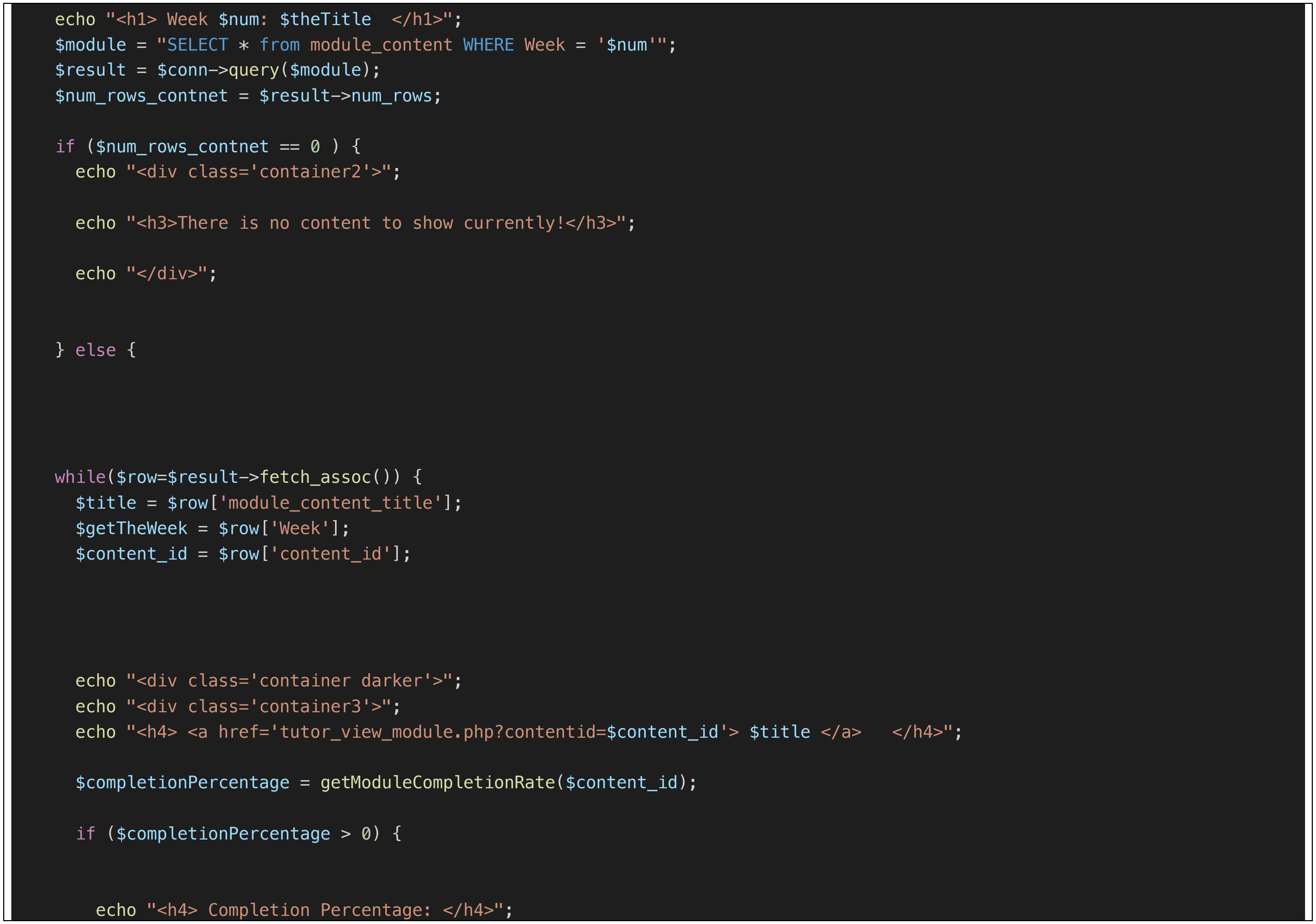
 This is the content completion which get the individual content completion percentage of each student. Found in tutor_level/ student_profile.php.

```
function getContentCompletion($student_id) {

   // include the db connection.
   include('../student_level/dbh.php');

  // Once again this is the standard formula that will be taking the possible content completion and the actual content
completion to formulate information that the tutor can use to assess the progress of their pupil
  // get the possible contentCompletion

 $getPossibleContent = "SELECT * FROM module_content";
 $execute = $conn->query($getPossibleContent);

//  Use the num_rows functionality to get the possible amount of content that could exist
 $num_rows = $execute->num_rows;
 $possibleContentCompletion = $num_rows;

// -----------------------------------------------------------------------------------------------------------------------
// Now get the total amount of completions that the user has completed which will be taken from the content_feedback_table.
Logically the pupil will not have provided feed back for a content that they have not completed

$getTotalContent = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id' AND
content_feedback_table.rating_number IS NOT NULL";
$executeTotalContent = $conn->query($getTotalContent);

$num_rows = $executeTotalContent->num_rows;

$totalContentCompletion = $num_rows;

// -------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------
//  Then do the calculation

$completionPercentage = 0;
if($totalContentCompletion<=0) {
  $completionPercentage=0;

} elseif ($totalContentCompletion > 0){

  $completionPercentage =  round(($totalContentCompletion/$possibleContentCompletion* 100), 1);
}


return $completionPercentage;

}
```

View Module

This is the component that will be used so that the tutor can see the content that they have originally created for the students benefit. This can be found in the tutor_level/tutor_module.php.

```
<?php

  // put this code into a for loop to display all the conetent of the module on the screen
  //get all the data associated with the week
$getWeek = "SELECT * from Weeks_table ORDER BY Week ASC";
  $query = $conn->query($getWeek);
  while($row=$query->fetch_assoc()) {

    $theTitle = $row['Title'];
    $num = $row['Week'];

    echo  "<div class='container'>";
    echo  "<div class='container2'>";
```



```
echo "<h1> Week $num: $theTitle  </h1>";
$module = "SELECT * from module_content WHERE Week = '$num'";
$result = $conn->query($module);
$num_rows_contnet = $result->num_rows;

if ($num_rows_contnet == 0 ) {
  echo "<div class='container2'>";

  echo "<h3>There is no content to show currently!</h3>";

  echo "</div>";


} else {




while($row=$result->fetch_assoc()) {
  $title = $row['module_content_title'];
  $getTheWeek = $row['Week'];
  $content_id = $row['content_id'];




  echo "<div class='container darker'>";
  echo "<div class='container3'>";
  echo "<h4> <a href='tutor_view_module.php?contentid=$content_id'> $title </a>   </h4>";

  $completionPercentage = getModuleCompletionRate($content_id);

  if ($completionPercentage > 0) {


    echo "<h4> Completion Percentage: </h4>";
```

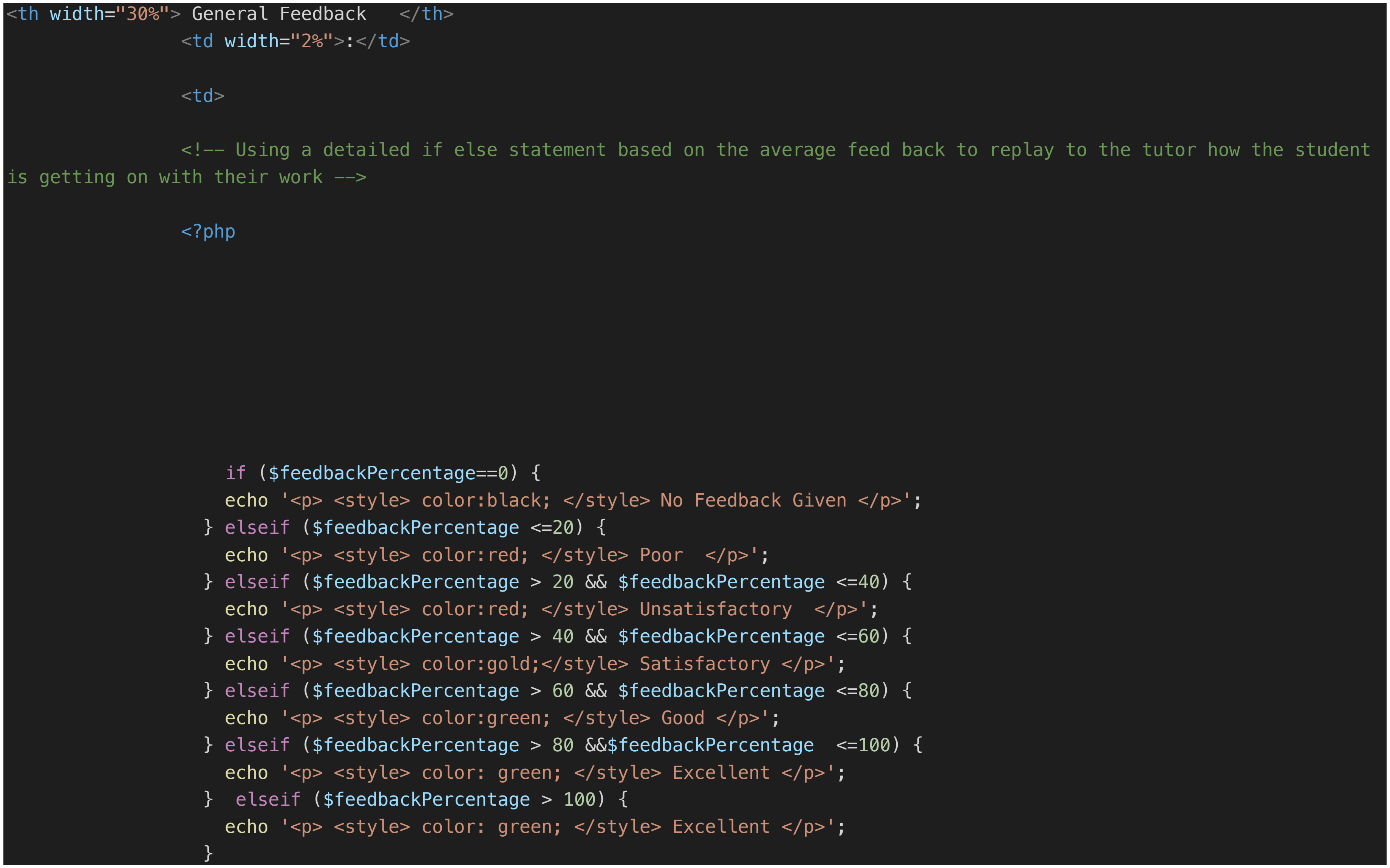
```
        echo "<div class='progress-g'>";
        echo "<div class='progress-done' data-done='$completionPercentage'  style = 'background: linear-gradient(to left, 
#F2709C, #FF9472) ;height:100%;width:$completionPercentage%  ;border-radius: 20px;height: 100%; display: flex; align-
items:center; justify-content: center; animation: progress-done 2s; '>";
        echo  "$completionPercentage%";
        echo "</div>";
        echo "</div>";


        echo "<button id = 'module-content-delete' style=''> <a 
href='php_functionality/delete_content.php?contentid=$content_id'> Delete </a>   </button>";

        echo "</div>";

      } else {

        echo "<h4> Completion Percentage: </h4>";
        echo "<div class='progress-g'>";

        echo "<div class='progress-done' data-done='$completionPercentage'  style = 'background: linear-gradient(to left, 
#F2709C, #FF9472) ;height:100%;width:$completionPercentage%  ;border-radius: 20px;height: 100%; display: flex; align-
items:center; justify-content: center; animation: progress-done 2s; text-align: center; '>";

        echo  "<p style = 'text-align:center;'>$completionPercentage%</p>";
        echo "</div>";
        echo "</div>";


        echo "<button id = 'module-content-delete' style=''> <a 
href='php_functionality/delete_content.php?contentid=$content_id'> Delete </a>   </button>";

        echo "</div>";
      }
      echo "</div>";
```

```
    }
    echo "</div>";
    echo "</div>";
  }
  }

  ?>
</body>
</html>
```

General Content Feedback

This is the General Content Feedback that is used to determine the general feedback of the student in relation to all the content that they have completed. This is done by returning a statement that represents how the student is doing.

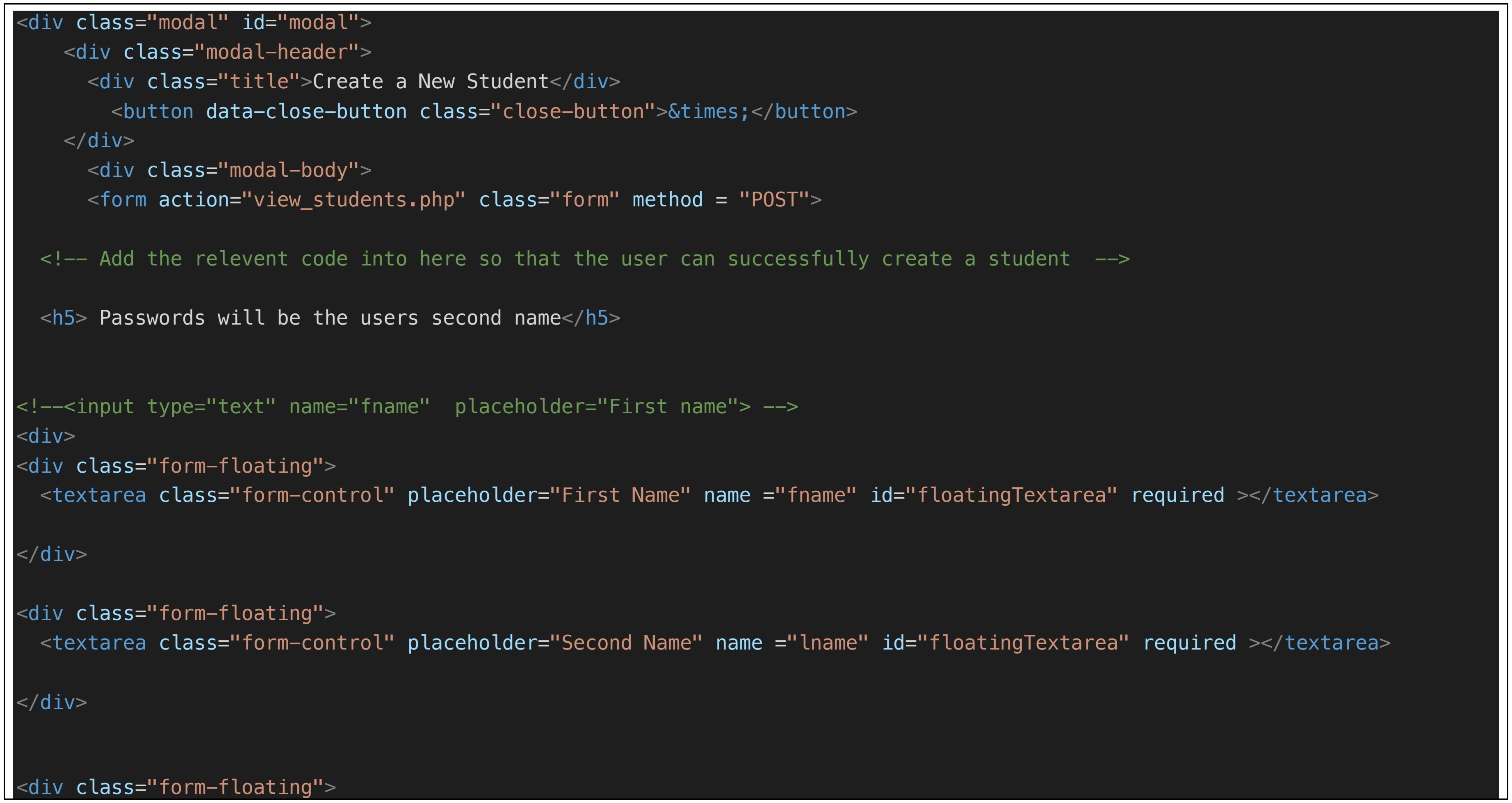
```
//////////////////////////////////////////////////////// GET AVERAGE FEEDBACK
////////////////////////////////////////////////////////

function getAverageFeedback($student_id) {
  // include the db connection.
  include('../student_level/dbh.php');

  // I need to create a equation to get the average Feedback for the student feedback which I will use to calculte the data needed to display to the tutor
  // To get this I will start by selecting the amount of rows that exist within the content feedback table. The studnet can only get average feedback for content that they have completed.


  //Query and execution
  $getActualFeedback = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id'";
  $executeActualFeedback= $conn->query($getActualFeedback);
  // Get the number of rows that exist and then multiple that 5
  $existingFeedback = $executeActualFeedback->num_rows;
  // Place that in a variable called possible feedback as this is the total possible feedback score that the student could
provide
  $possibleFeeback = ($existingFeedback * 5);


  // insert checks because the student may not have inserted anything yet which could be an issue.
  // If 0 rows are returned then the student will automatically be given a 0 and the function will return a 0

  if ($existingFeedback == 0 ) {
     $feebackData = 0;
  } else {
    // if the function does not retun a 0 then do this
    $rating_number = 0;
    while ($row=$executeActualFeedback->fetch_assoc()) {
      // loop through all the rating scores that the student has provided
      // then add them to the $rating_number var in each increment of the loop
      // cast the variable so that it is returned as an int
    $rating_number+= (int)$row['rating_number'];
    }


    // do the calulation so that a average figure is returned in an integer for
    $feebackData = ($rating_number/$possibleFeeback * 100);

    echo "<p>$rating_number</p>";
    echo "<p>$possibleFeeback</p>";




return $feebackData;


}
```



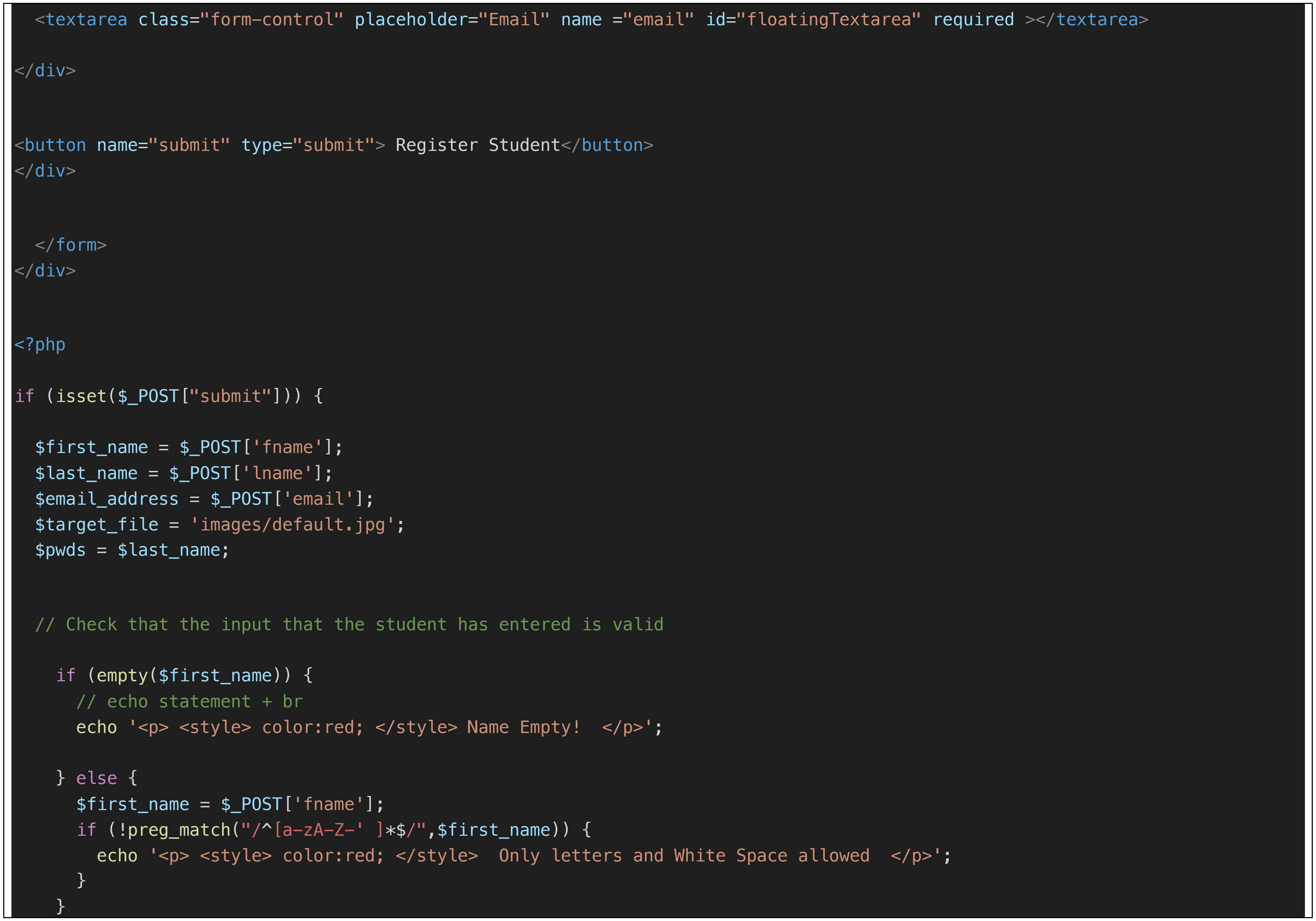
```
<th width="30%"> General Feedback   </th>
                <td width="2%">:</td>

                <td>

                <!-- Using a detailed if else statement based on the average feed back to replay to the tutor how the student
is getting on with their work -->

                <?php









                    if ($feedbackPercentage==0) {
                    echo '<p> <style> color:black; </style> No Feedback Given </p>';
                  } elseif ($feedbackPercentage <=20) {
                    echo '<p> <style> color:red; </style> Poor  </p>';
                  } elseif ($feedbackPercentage > 20 && $feedbackPercentage <=40) {
                    echo '<p> <style> color:red; </style> Unsatisfactory  </p>';
                  } elseif ($feedbackPercentage > 40 && $feedbackPercentage <=60) {
                    echo '<p> <style> color:gold;</style> Satisfactory </p>';
                  } elseif ($feedbackPercentage > 60 && $feedbackPercentage <=80) {
                    echo '<p> <style> color:green; </style> Good </p>';
                  } elseif ($feedbackPercentage > 80 &&$feedbackPercentage  <=100) {
                    echo '<p> <style> color: green; </style> Excellent </p>';
                  }  elseif ($feedbackPercentage > 100) {
                    echo '<p> <style> color: green; </style> Excellent </p>';
                  }
```

Add Student Function

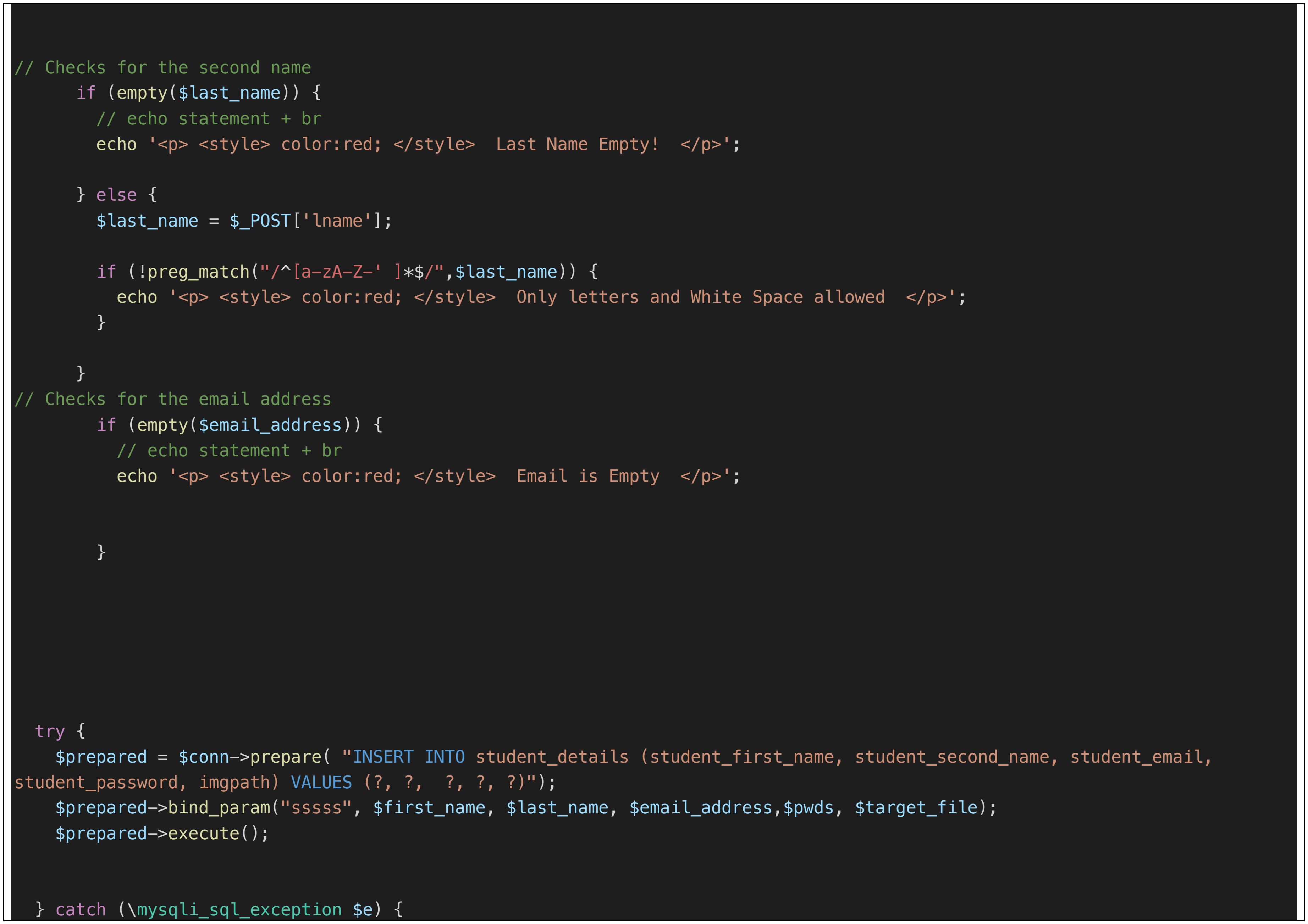
This is the add student function which will be used so that students can be added to the system successfully. This is done using a pop up modal. This can be found in the view_student.php



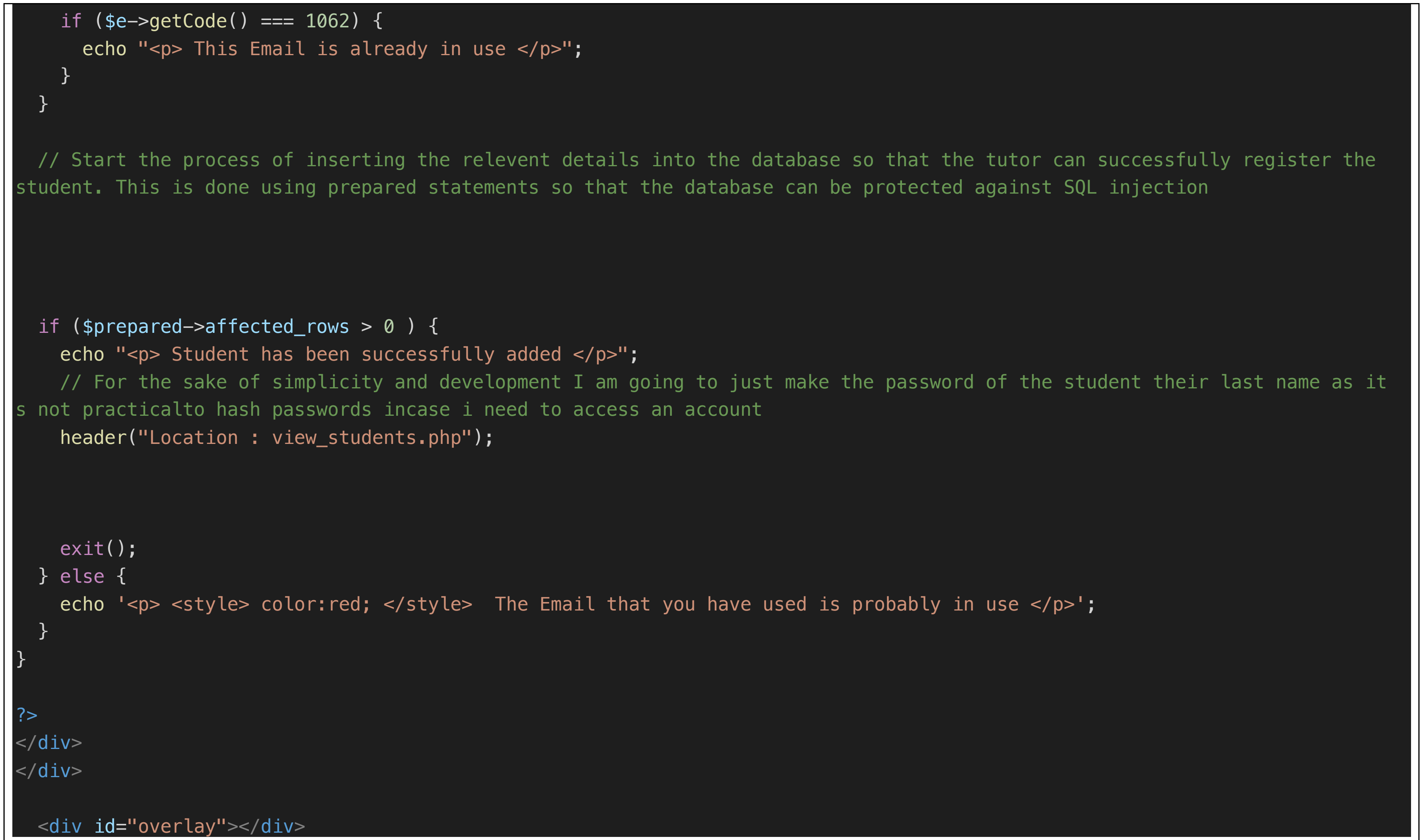
```
<div class="modal" id="modal">
    <div class="modal-header">
      <div class="title">Create a New Student</div>
        <button data-close-button class="close-button">&times;</button>
    </div>
      <div class="modal-body">
      <form action="view_students.php" class="form" method = "POST">

  <!-- Add the relevent code into here so that the user can successfully create a student  -->

  <h5> Passwords will be the users second name</h5>


<!--<input type="text" name="fname"  placeholder="First name"> -->
<div>
<div class="form-floating">
  <textarea class="form-control" placeholder="First Name" name ="fname" id="floatingTextarea" required ></textarea>

</div>

<div class="form-floating">
  <textarea class="form-control" placeholder="Second Name" name ="lname" id="floatingTextarea" required ></textarea>

</div>


<div class="form-floating">
```



```
  <textarea class="form-control" placeholder="Email" name ="email" id="floatingTextarea" required ></textarea>

</div>


<button name="submit" type="submit"> Register Student</button>
</div>


  </form>
</div>


<?php

if (isset($_POST["submit"])) {

  $first_name = $_POST['fname'];
  $last_name = $_POST['lname'];
  $email_address = $_POST['email'];
  $target_file = 'images/default.jpg';
  $pwds = $last_name;


  // Check that the input that the student has entered is valid

    if (empty($first_name)) {
      // echo statement + br
      echo '<p> <style> color:red; </style> Name Empty!  </p>';

    } else {
      $first_name = $_POST['fname'];
      if (!preg_match("/^[a-zA-Z-' ]*$/",$first_name)) {
        echo '<p> <style> color:red; </style>  Only letters and White Space allowed  </p>';
      }
    }
```



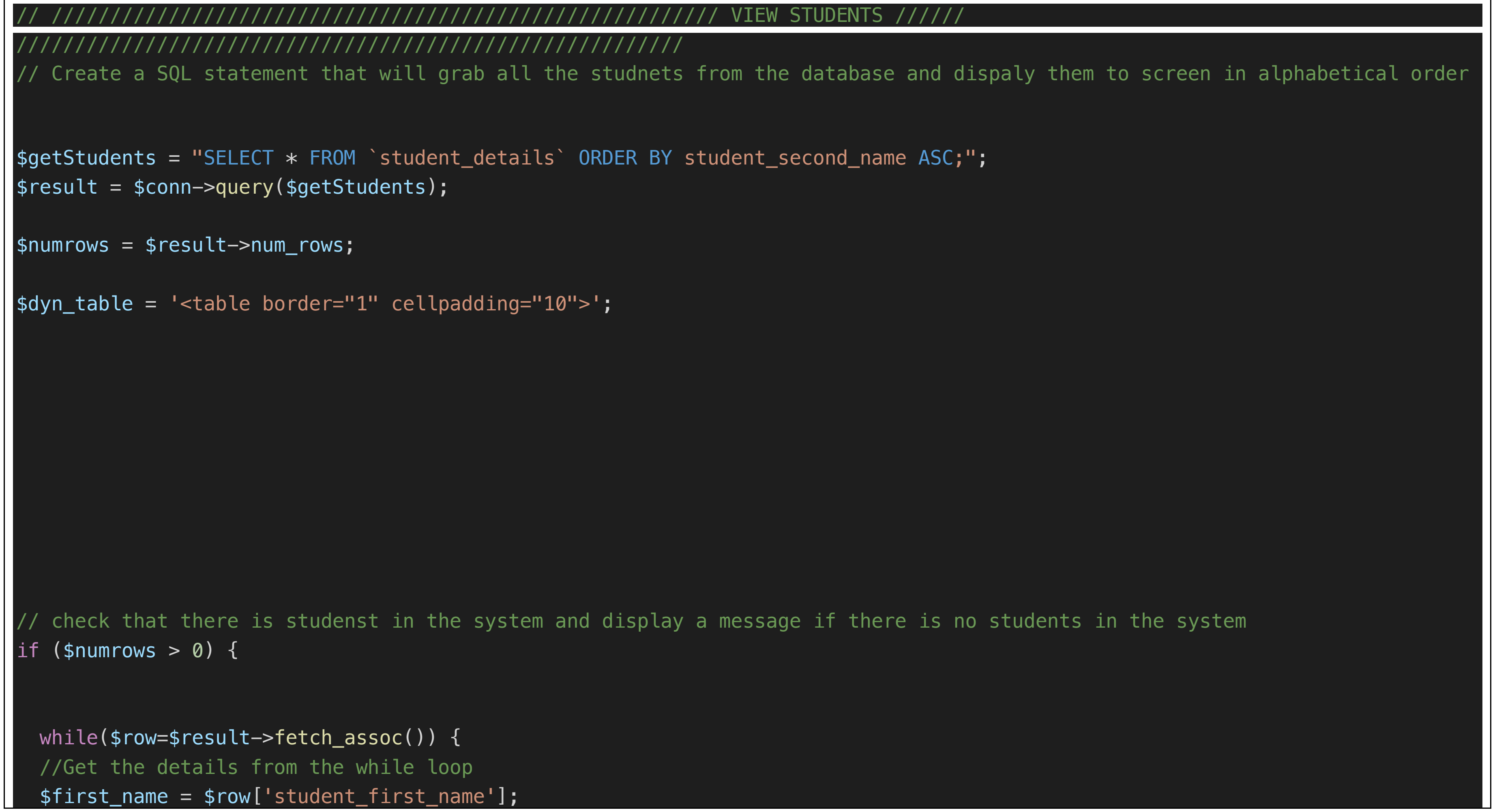
```
// Checks for the second name
   if (empty($last_name)) {
    // echo statement + br
    echo '<p> <style> color:red; </style>  Last Name Empty!  </p>';

   } else {
    $last_name = $_POST['lname'];

    if (!preg_match("/^[a-zA-Z-' ]*$/",$last_name)) {
     echo '<p> <style> color:red; </style>  Only letters and White Space allowed  </p>';
    }

   }
// Checks for the email address
    if (empty($email_address)) {
     // echo statement + br
     echo '<p> <style> color:red; </style>  Email is Empty  </p>';


    }






 try {
  $prepared = $conn->prepare( "INSERT INTO student_details (student_first_name, student_second_name, student_email,
student_password, imgpath) VALUES (?, ?,  ?, ?, ?)");
  $prepared->bind_param("sssss", $first_name, $last_name, $email_address,$pwds, $target_file);
  $prepared->execute();


 } catch (\mysqli_sql_exception $e) {
```



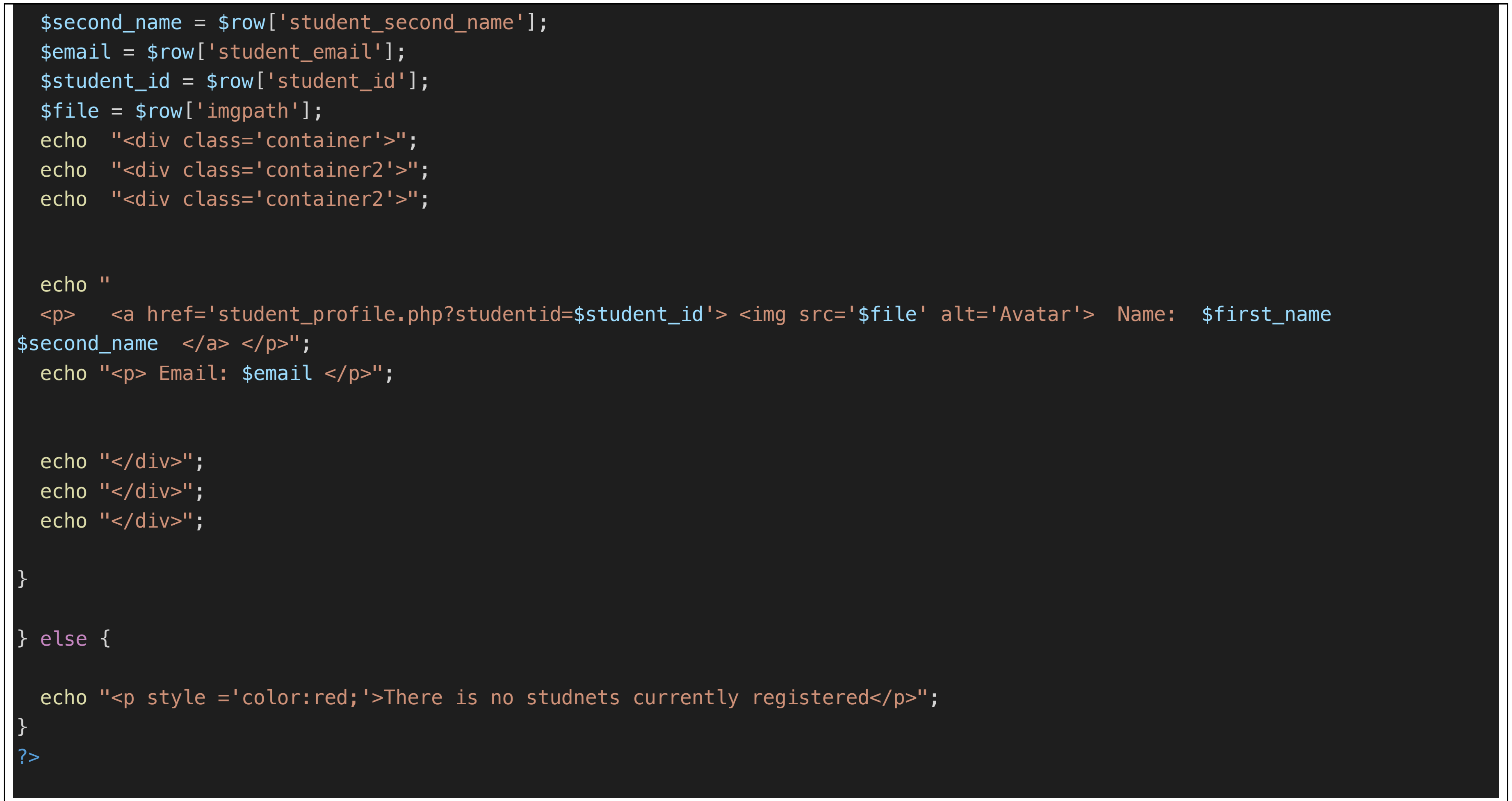
```
    if ($e->getCode() === 1062) {
      echo "<p> This Email is already in use </p>";
    }
  }

  // Start the process of inserting the relevent details into the database so that the tutor can successfully register the 
student. This is done using prepared statements so that the database can be protected against SQL injection





  if ($prepared->affected_rows > 0 ) {
    echo "<p> Student has been successfully added </p>";
    // For the sake of simplicity and development I am going to just make the password of the student their last name as it
s not practicalto hash passwords incase i need to access an account
    header("Location : view_students.php");




    exit();
  } else {
    echo '<p> <style> color:red; </style>  The Email that you have used is probably in use </p>';
  }
}

?>
</div>
</div>

  <div id="overlay"></div>
```

View Student Function

This is the view student function that will be used so that the tutor can view a list of the student that are registered in the system.



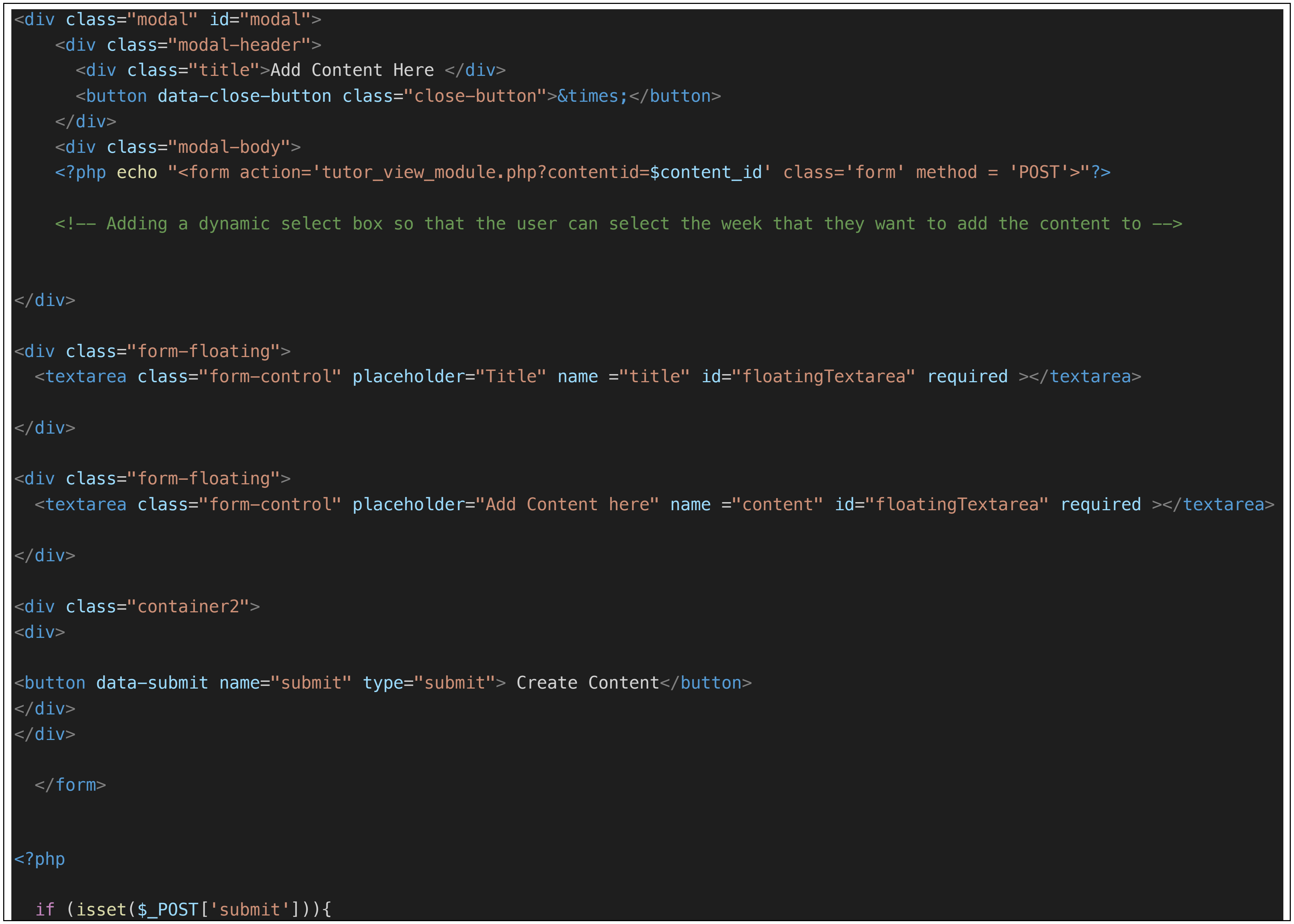
```
// ////////////////////////////////////////////////////////// VIEW STUDENTS //////
/////////////////////////////////////////////////////////
// Create a SQL statement that will grab all the studnets from the database and dispaly them to screen in alphabetical order


$getStudents = "SELECT * FROM `student_details` ORDER BY student_second_name ASC;";
$result = $conn->query($getStudents);

$numrows = $result->num_rows;

$dyn_table = '<table border="1" cellpadding="10">';










// check that there is studenst in the system and display a message if there is no students in the system
if ($numrows > 0) {


  while($row=$result->fetch_assoc()) {
  //Get the details from the while loop
  $first_name = $row['student_first_name'];
```

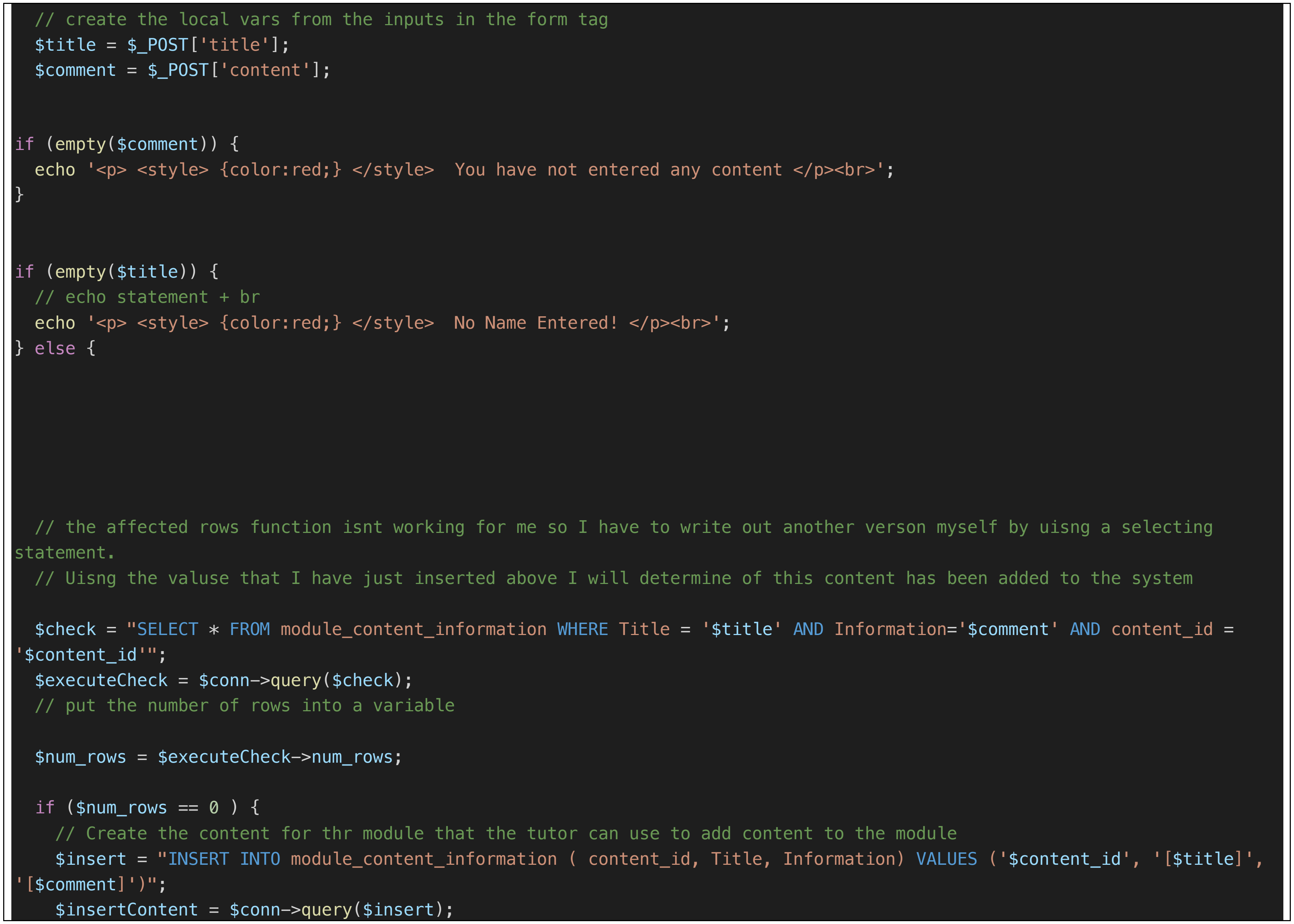


```
  $second_name = $row['student_second_name'];
  $email = $row['student_email'];
  $student_id = $row['student_id'];
  $file = $row['imgpath'];
  echo  "<div class='container'>";
  echo  "<div class='container2'>";
  echo  "<div class='container2'>";


  echo "
  <p>   <a href='student_profile.php?studentid=$student_id'> <img src='$file' alt='Avatar'>  Name:  $first_name
$second_name  </a> </p>";
  echo "<p> Email: $email </p>";


  echo "</div>";
  echo "</div>";
  echo "</div>";

}

} else {

  echo "<p style ='color:red;'>There is no studnets currently registered</p>";
}
?>
```

<u>Add Content Information</u>

This will allow us to add Content information to content that belongs to the module for the students learning


```
<div class="modal" id="modal">
    <div class="modal-header">
      <div class="title">Add Content Here </div>
      <button data-close-button class="close-button">&times;</button>
    </div>
    <div class="modal-body">
    <?php echo "<form action='tutor_view_module.php?contentid=$content_id' class='form' method = 'POST'>"?>

    <!-- Adding a dynamic select box so that the user can select the week that they want to add the content to -->


</div>

<div class="form-floating">
  <textarea class="form-control" placeholder="Title" name ="title" id="floatingTextarea" required ></textarea>

</div>

<div class="form-floating">
  <textarea class="form-control" placeholder="Add Content here" name ="content" id="floatingTextarea" required ></textarea>

</div>

<div class="container2">
<div>

<button data-submit name="submit" type="submit"> Create Content</button>
</div>
</div>

  </form>


<?php

  if (isset($_POST['submit'])){
```

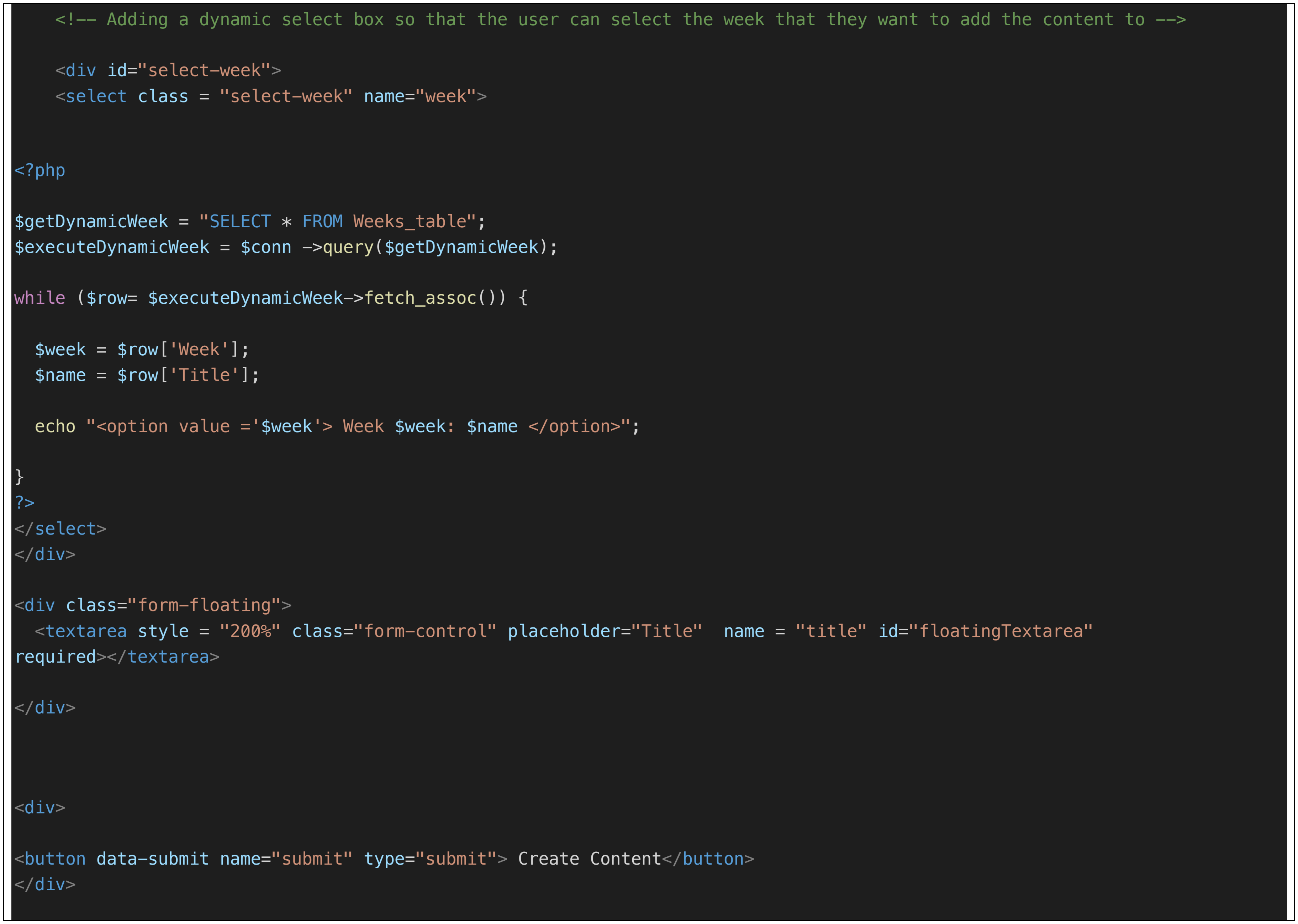


```
  // create the local vars from the inputs in the form tag
  $title = $_POST['title'];
  $comment = $_POST['content'];


if (empty($comment)) {
  echo '<p> <style> {color:red;} </style>  You have not entered any content </p><br>';
}


if (empty($title)) {
  // echo statement + br
  echo '<p> <style> {color:red;} </style>  No Name Entered! </p><br>';
} else {







  // the affected rows function isnt working for me so I have to write out another verson myself by uisng a selecting
statement.
  // Uisng the valuse that I have just inserted above I will determine of this content has been added to the system

  $check = "SELECT * FROM module_content_information WHERE Title = '$title' AND Information='$comment' AND content_id =
'$content_id'";
  $executeCheck = $conn->query($check);
  // put the number of rows into a variable

  $num_rows = $executeCheck->num_rows;

  if ($num_rows == 0 ) {
    // Create the content for thr module that the tutor can use to add content to the module
    $insert = "INSERT INTO module_content_information ( content_id, Title, Information) VALUES ('$content_id', '[$title]',
'[$comment]')";
    $insertContent = $conn->query($insert);
  } else {
    echo '<p> <style> {color:red;} </style>  Something went wrong please try again </p><br>';
  }







}
  }
?>




</div>
</div>
  <div id="overlay"></div>
```

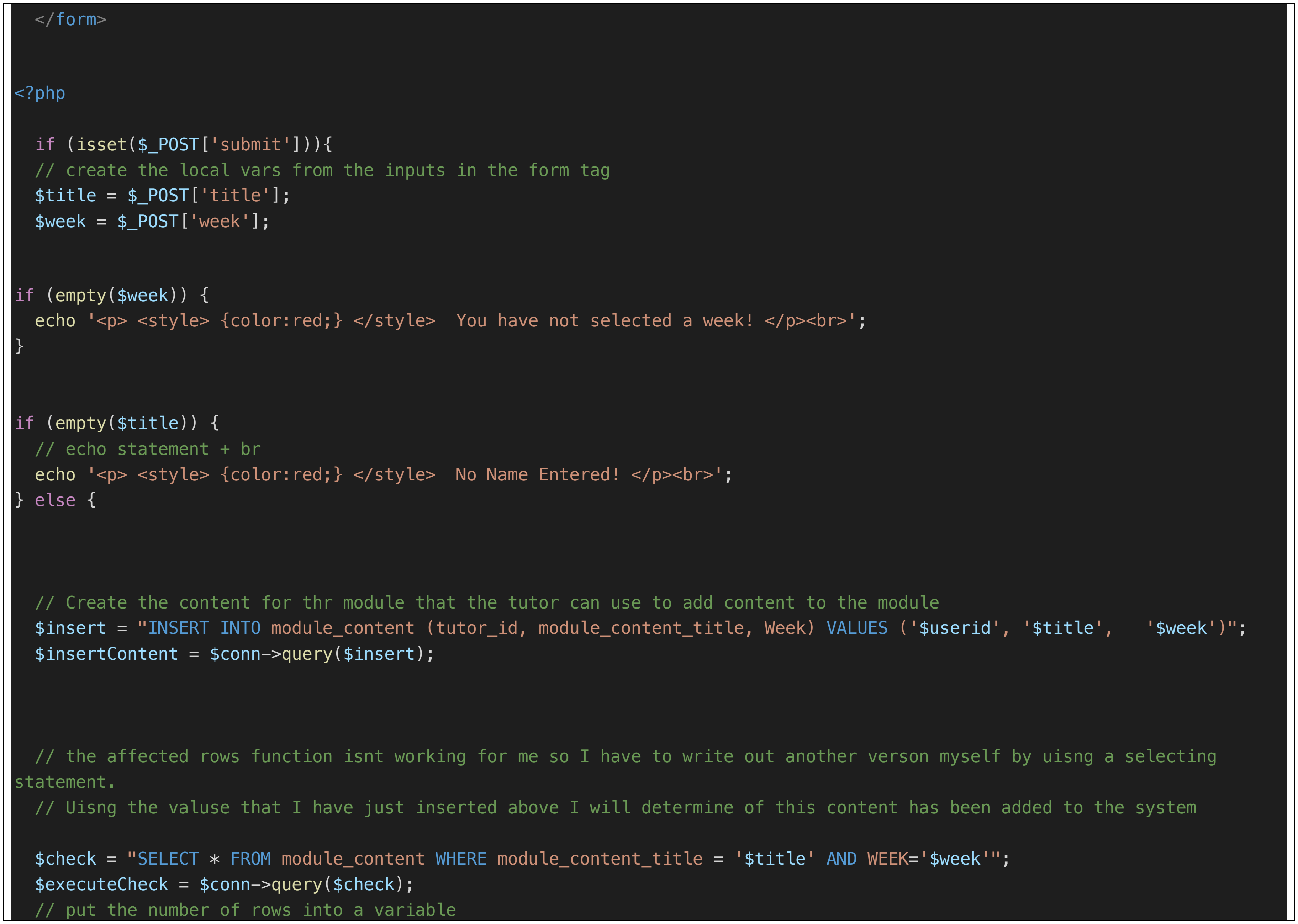
<u>Add Module Content</u>

This allows the tutor to add content to the module which the user can see. This will also create discussion forum that the user can use.

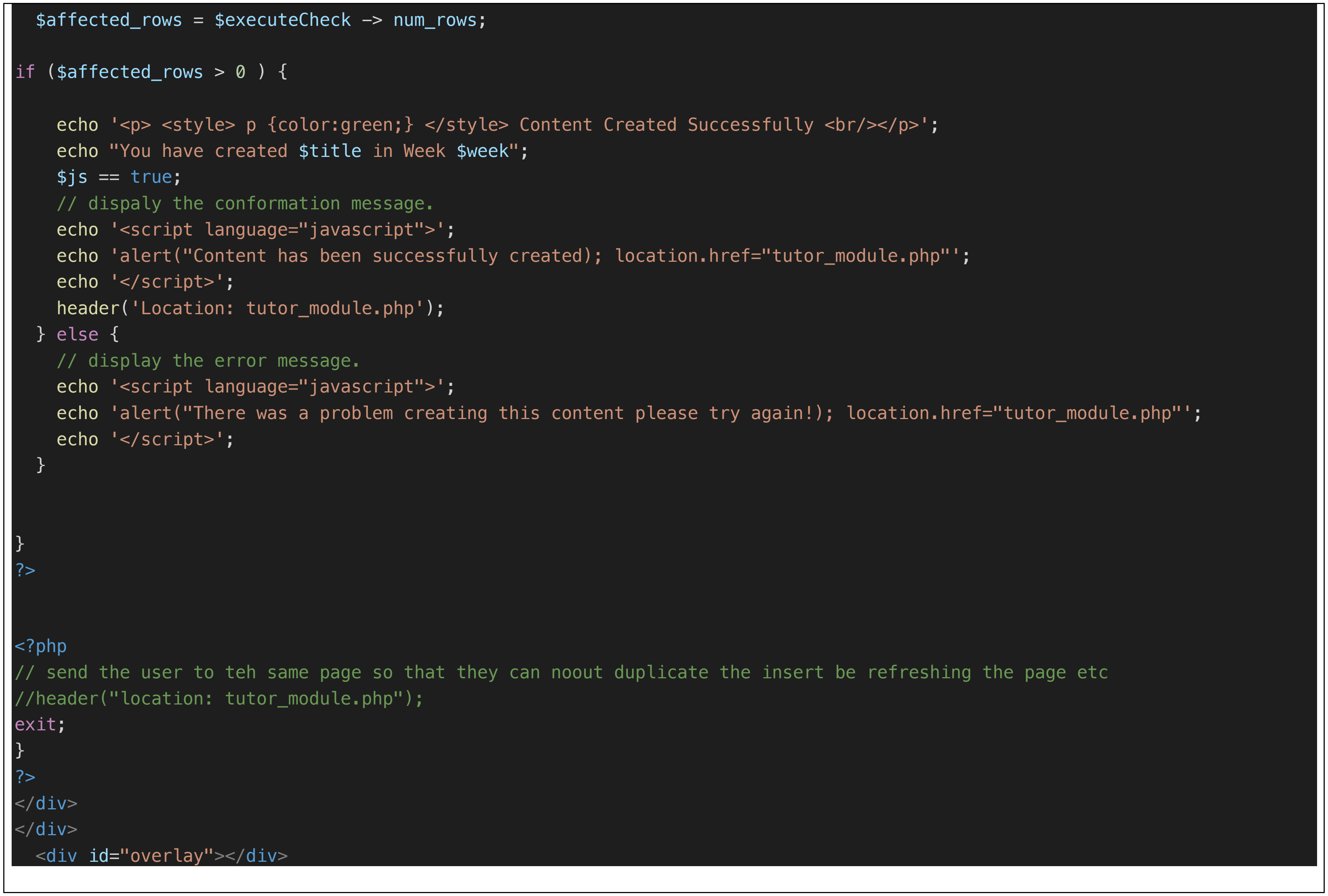
```
<div class="modal" id="modal">
    <div class="modal-header">
      <div class="title">Add Content Here </div>
      <button data-close-button class="close-button">&times;</button>
    </div>
    <div class="modal-body">
    <form action="tutor_module.php" class="form" method = "POST">
```



```
    <!-- Adding a dynamic select box so that the user can select the week that they want to add the content to -->

    <div id="select-week">
    <select class = "select-week" name="week">


<?php

$getDynamicWeek = "SELECT * FROM Weeks_table";
$executeDynamicWeek = $conn ->query($getDynamicWeek);

while ($row= $executeDynamicWeek->fetch_assoc()) {

  $week = $row['Week'];
  $name = $row['Title'];

  echo "<option value ='$week'> Week $week: $name </option>";

}
?>
</select>
</div>

<div class="form-floating">
  <textarea style = "200%" class="form-control" placeholder="Title"  name = "title" id="floatingTextarea"
required></textarea>

</div>



<div>

<button data-submit name="submit" type="submit"> Create Content</button>
</div>
```



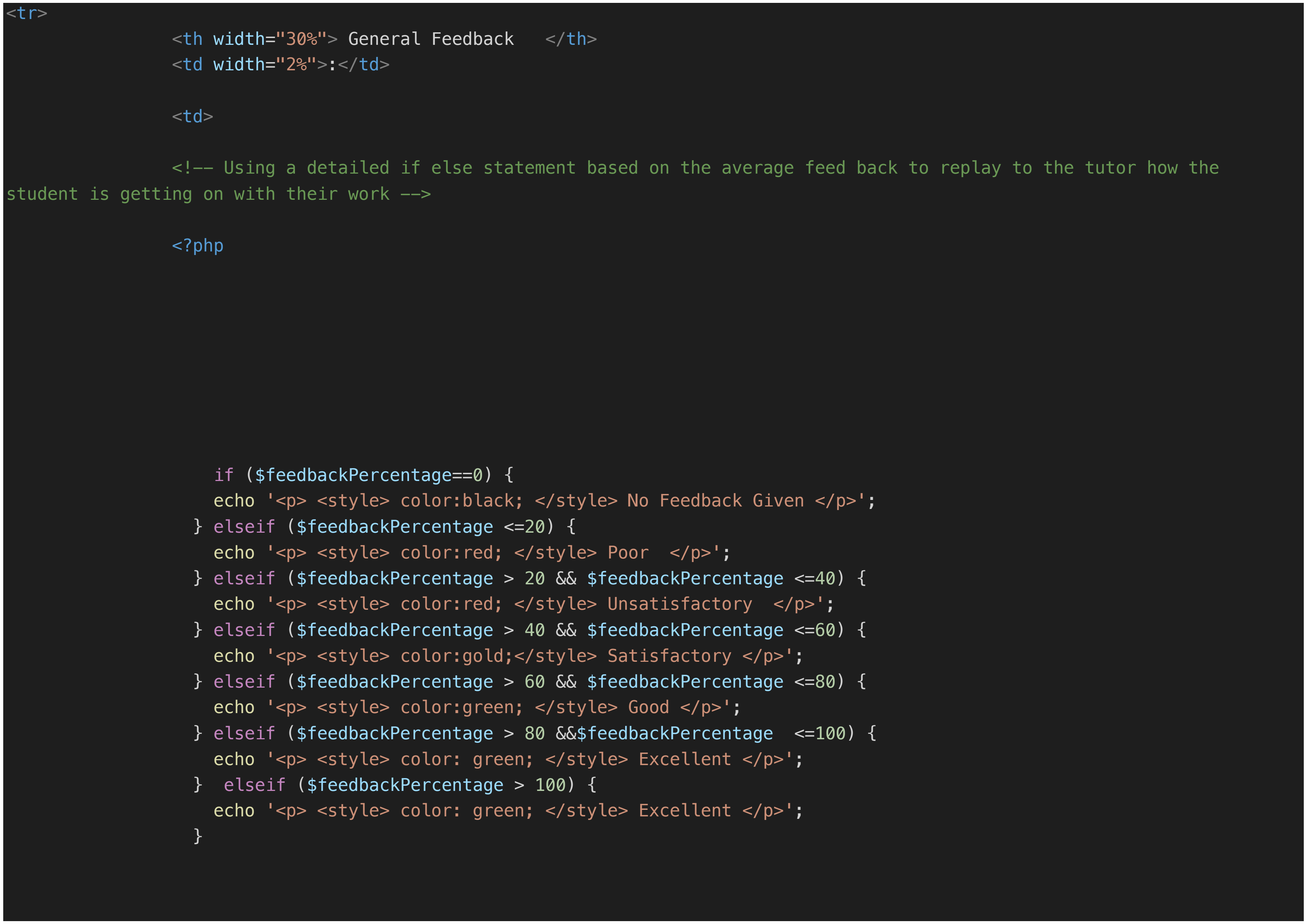
```
  </form>


<?php

  if (isset($_POST['submit'])){
  // create the local vars from the inputs in the form tag
  $title = $_POST['title'];
  $week = $_POST['week'];


if (empty($week)) {
  echo '<p> <style> {color:red;} </style>  You have not selected a week! </p><br>';
}


if (empty($title)) {
  // echo statement + br
  echo '<p> <style> {color:red;} </style>  No Name Entered! </p><br>';
} else {



  // Create the content for thr module that the tutor can use to add content to the module
  $insert = "INSERT INTO module_content (tutor_id, module_content_title, Week) VALUES ('$userid', '$title',   '$week')";
  $insertContent = $conn->query($insert);



  // the affected rows function isnt working for me so I have to write out another verson myself by uisng a selecting 
statement.
  // Uisng the valuse that I have just inserted above I will determine of this content has been added to the system

  $check = "SELECT * FROM module_content WHERE module_content_title = '$title' AND WEEK='$week'";
  $executeCheck = $conn->query($check);
  // put the number of rows into a variable
```



```
  $affected_rows = $executeCheck -> num_rows;

if ($affected_rows > 0 ) {

    echo '<p> <style> p {color:green;} </style> Content Created Successfully <br/></p>';
    echo "You have created $title in Week $week";
    $js == true;
    // dispaly the conformation message.
    echo '<script language="javascript">';
    echo 'alert("Content has been successfully created); location.href="tutor_module.php"';
    echo '</script>';
    header('Location: tutor_module.php');
  } else {
    // display the error message.
    echo '<script language="javascript">';
    echo 'alert("There was a problem creating this content please try again!); location.href="tutor_module.php"';
    echo '</script>';
  }


}
?>



<?php
// send the user to teh same page so that they can noout duplicate the insert be refreshing the page etc
//header("location: tutor_module.php");
exit;
}
?>
</div>
</div>
  <div id="overlay"></div>
```

Average Content Feedback

This is the Average Content Feedback which is used to determine the students full academic standing.



```
<tr>
                <th width="30%"> General Feedback   </th>
                <td width="2%">:</td>

                <td>

                <!-- Using a detailed if else statement based on the average feed back to replay to the tutor how the
student is getting on with their work -->

                <?php








                    if ($feedbackPercentage==0) {
                    echo '<p> <style> color:black; </style> No Feedback Given </p>';
                  } elseif ($feedbackPercentage <=20) {
                    echo '<p> <style> color:red; </style> Poor  </p>';
                  } elseif ($feedbackPercentage > 20 && $feedbackPercentage <=40) {
                    echo '<p> <style> color:red; </style> Unsatisfactory  </p>';
                  } elseif ($feedbackPercentage > 40 && $feedbackPercentage <=60) {
                    echo '<p> <style> color:gold;</style> Satisfactory </p>';
                  } elseif ($feedbackPercentage > 60 && $feedbackPercentage <=80) {
                    echo '<p> <style> color:green; </style> Good </p>';
                  } elseif ($feedbackPercentage > 80 &&$feedbackPercentage  <=100) {
                    echo '<p> <style> color: green; </style> Excellent </p>';
                  }  elseif ($feedbackPercentage > 100) {
                    echo '<p> <style> color: green; </style> Excellent </p>';
                  }
```

```
               ?>


                </td>

              </tr>
```

```
function getAverageFeedback($student_id) {
  // include the db connection.
  include('../student_level/dbh.php');

  // I need to create a equation to get the average Feedback for the student feedback which I will use to calculte the data
needed to display to the tutor
  // To get this I will start by selecting the amount of rows that exist within the content feedback table. The studnet can
only get average feedback for content that they have completed.


  //Query and execution
  $getActualFeedback = "SELECT * FROM content_feedback_table WHERE student_id = '$student_id'";
  $executeActualFeedback= $conn->query($getActualFeedback);

  // Get the number of rows that exist and then multiple that 5
  $existingFeedback = $executeActualFeedback->num_rows;
  // Place that in a variable called possible feedback as this is the total possible feedback score that the student could
provide
  $possibleFeeback = ($existingFeedback * 5);


  // insert checks because the student may not have inserted anything yet which could be an issue.
  // If 0 rows are returned then the student will automatically be given a 0 and the function will return a 0

  if ($existingFeedback == 0 ) {
     $feebackData = 0;
  } else {
```

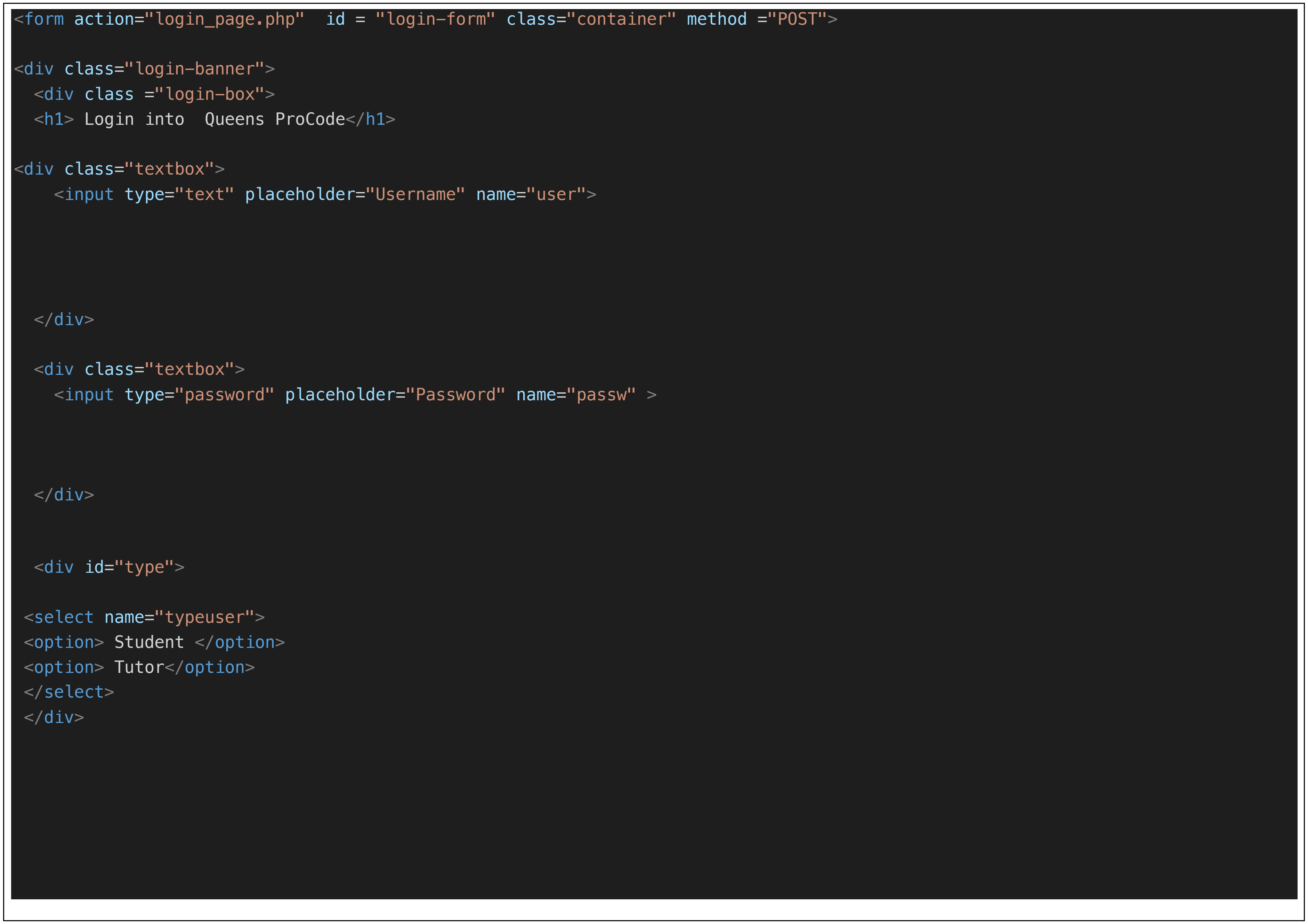
```php
    // if the function does not retun a 0 then do this
    $rating_number = 0;
    while ($row=$executeActualFeedback->fetch_assoc()) {
      // loop through all the rating scores that the student has provided
      // then add them to the $rating_number var in each increment of the loop
      // cast the variable so that it is returned as an int
    $rating_number+= (int)$row['rating_number'];
    }


    // do the calulation so that a average figure is returned in an integer for
    $feebackData = ($rating_number/$possibleFeeback * 100);

    echo "<p>$rating_number</p>";
    echo "<p>$possibleFeeback</p>";




return $feebackData;
```

Student Level

Login Component 1.1

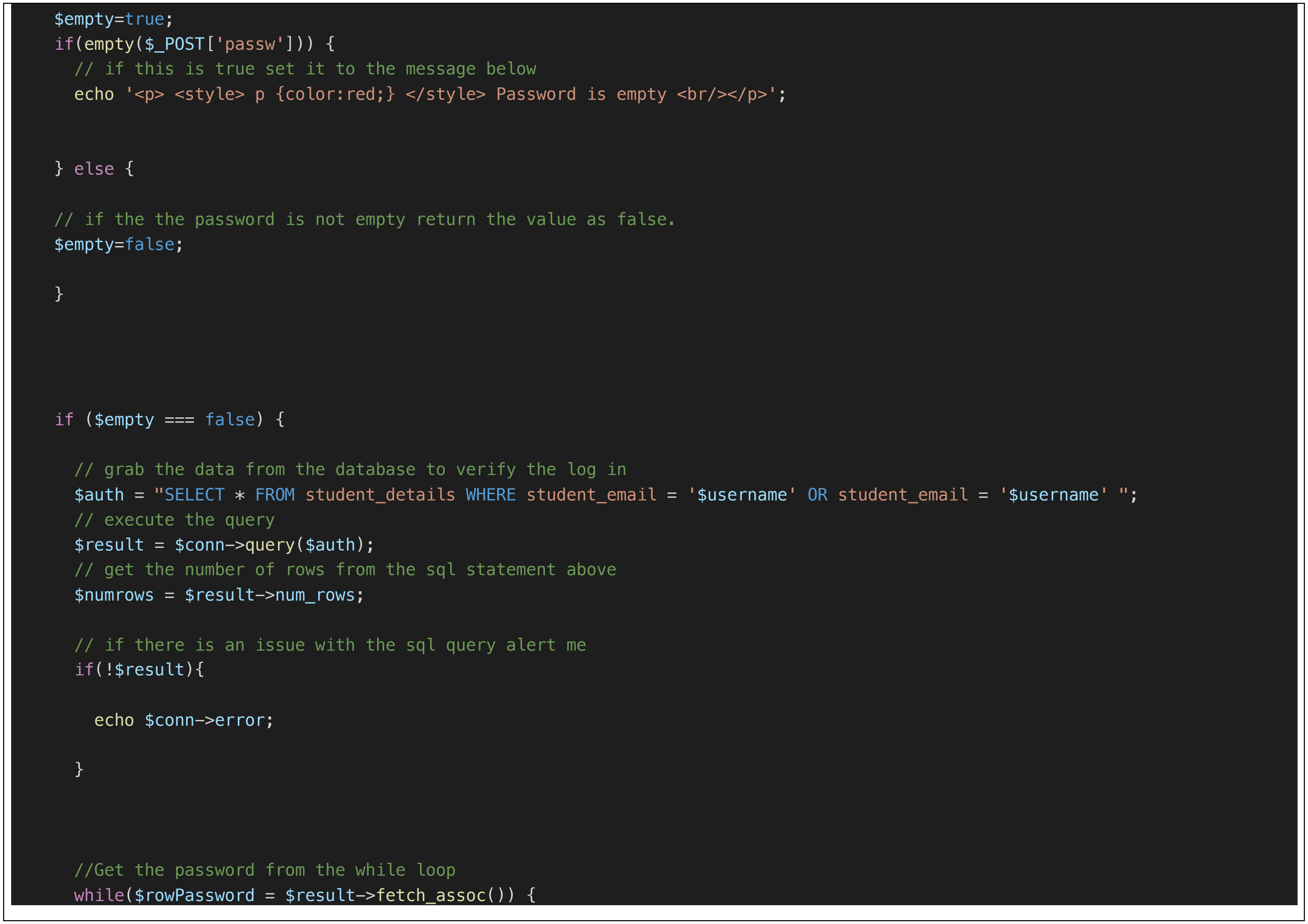
This is the login component that can be found in the file **student_level/login_page.php**.

This component checks that the student or the tutor is authorized to be logging into the system. The type variable means that I can check if the user has selected to login as a student or a tutor and this will then direct them to the correct home page that is associated with their account.



```html
<form action="login_page.php"  id = "login-form" class="container" method ="POST">

<div class="login-banner">
  <div class ="login-box">
  <h1> Login into  Queens ProCode</h1>

<div class="textbox">
    <input type="text" placeholder="Username" name="user">





  </div>

  <div class="textbox">
    <input type="password" placeholder="Password" name="passw" >



  </div>


  <div id="type">

 <select name="typeuser">
 <option> Student </option>
 <option> Tutor</option>
 </select>
 </div>
```

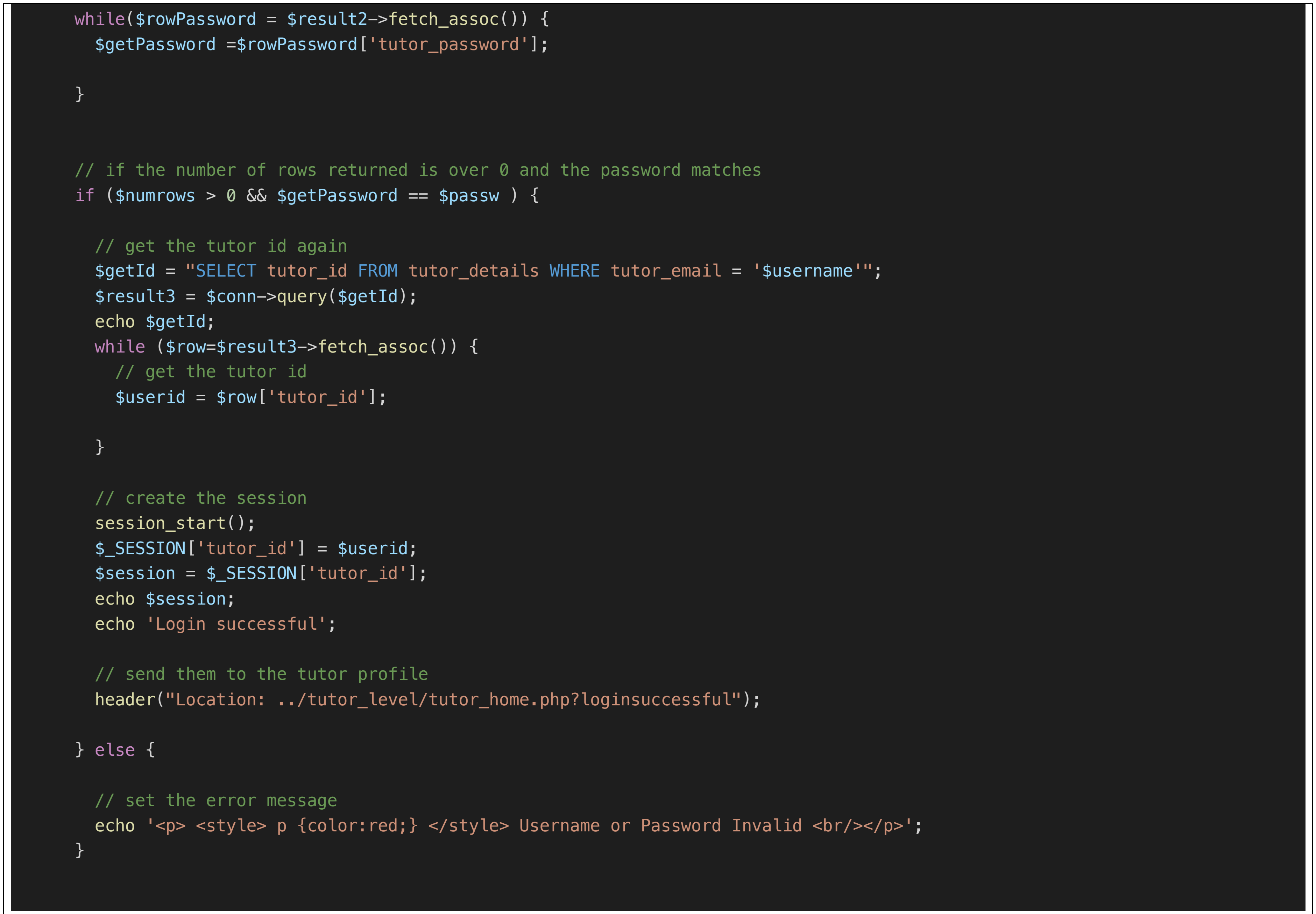
```
<button class ="btn" name="submit" type="submit"> Login!</button>



</form>



<?php
// This is the code that will check if the student or tutor is registered for the application
// If the user hits the login button. The file will be directed here


if (isset($_POST['submit'])){
  // There are the VARs captured by the form tag that exists above.
  $username = $_POST['user'];
  $passw = $_POST['passw'];
  $type = $_POST['typeuser'];
  $userid = null;

  // if the type is equal to the string "Student" then follow this logic.
  if ($type == 'Student') {

  // This will check if the $username variable is filled in.
    if(empty($_POST['user'])) {

      // if the VAR is empty then return this message to the user.
      echo '<p> <style> p {color:red;} </style> Username is empty <br/></p>';;

    } else{
      // else do nothing and continue with the logic


    }

    // check the password is empty
```



```
    $empty=true;
    if(empty($_POST['passw'])) {
      // if this is true set it to the message below
      echo '<p> <style> p {color:red;} </style> Password is empty <br/></p>';


    } else {

    // if the the password is not empty return the value as false.
    $empty=false;

    }





    if ($empty === false) {

      // grab the data from the database to verify the log in
      $auth = "SELECT * FROM student_details WHERE student_email = '$username' OR student_email = '$username' ";
      // execute the query
      $result = $conn->query($auth);
      // get the number of rows from the sql statement above
      $numrows = $result->num_rows;

      // if there is an issue with the sql query alert me
      if(!$result){

        echo $conn->error;

      }




      //Get the password from the while loop
      while($rowPassword = $result->fetch_assoc()) {
        $getPassword=$rowPassword['student_password'];

      }



      // check if the number of rows retured is more than 0 and that the password that has been returned is equal to the
password that has been passed
      // by the form tag.
      if ($numrows > 0 && $getPassword == $passw  ) {
        // get the tutor id again
        $getId = "SELECT student_id FROM student_details WHERE student_email = '$username'";
        $result2= $conn->query($getId);
        //echo $getId;
        while ($row=$result2->fetch_assoc()) {

          $userid = $row['student_id'];

        }

        // create the session
        session_start();
        $_SESSION['student_id'] = $userid;
        $session = $_SESSION['student_id'];
        //echo $session;
        //echo 'Login successful';
        header('Location: student_homepage.php?successful');

      } else {

        // if there is not data found set this statment
        echo '<p> <style> p {color:red;} </style> Username or Password invalid <br/></p>';
      }



    } else {
```

```php
  }




  // if the user hits tutor follow this route
} else if ($type == 'Tutor') {
  echo 'Check 1';



  //check username is empty

  if(empty($_POST['user'])) {
    // if this is true set it to the message below
    echo '<p> <style> p {color:red;} </style>  Username is empty <br/></p>';



  } else{


  }

  // check the password is empty
  $empty = true;
  if(empty($_POST['passw'])) {
    // if this is true set it to the message below
    echo '<p> <style> p {color:red;} </style>  Password is empty <br/></p>';
```

```
} else {
  // otherwise set this var used for verification
  $empty=false;

}





// If teh error checks or passed assert the sql statement
if ($empty == false) {

  $auth2 = "SELECT * FROM tutor_details WHERE tutor_email = '$username' ";

  // execute the sql statement
  $result2 = $conn->query($auth2);
  $numrows = $result2->num_rows;
  echo $numrows;
  //echo $auth;






  // if there is a error tell me
  if(!$result2){

    echo $conn->error;

  }


  // Get the password from a while loop
```



```php
    while($rowPassword = $result2->fetch_assoc()) {
      $getPassword =$rowPassword['tutor_password'];

    }


    // if the number of rows returned is over 0 and the password matches
    if ($numrows > 0 && $getPassword == $passw ) {

      // get the tutor id again
      $getId = "SELECT tutor_id FROM tutor_details WHERE tutor_email = '$username'";
      $result3 = $conn->query($getId);
      echo $getId;
      while ($row=$result3->fetch_assoc()) {
        // get the tutor id
        $userid = $row['tutor_id'];

      }

      // create the session
      session_start();
      $_SESSION['tutor_id'] = $userid;
      $session = $_SESSION['tutor_id'];
      echo $session;
      echo 'Login successful';

      // send them to the tutor profile
      header("Location: ../tutor_level/tutor_home.php?loginsuccessful");

    } else {

      // set the error message
      echo '<p> <style> p {color:red;} </style> Username or Password Invalid <br/></p>';
    }
    } else {


    }


  }
}




?>
</div>
</div>
```

Logout – 1.2

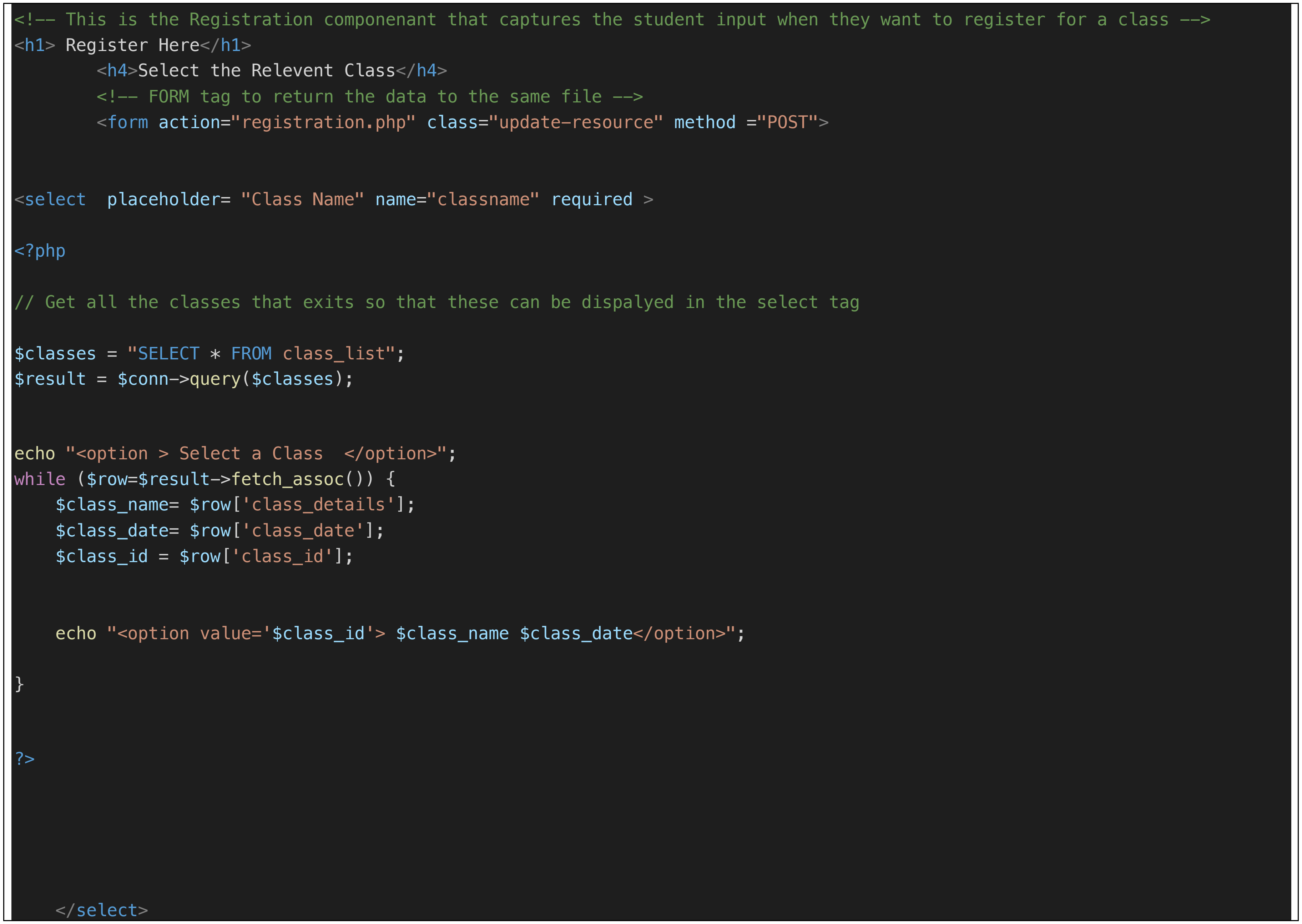
This is the log out function which is used to log students and tutors out of the account. This component will direct the user to the **student_level/login_page.php.** Visiting this file will destroy the current session that you have created when you log in.

```php
<?php

session_start();
session_destroy();
header('Location: ../student_level/login_page.php');
exit;

?>
```

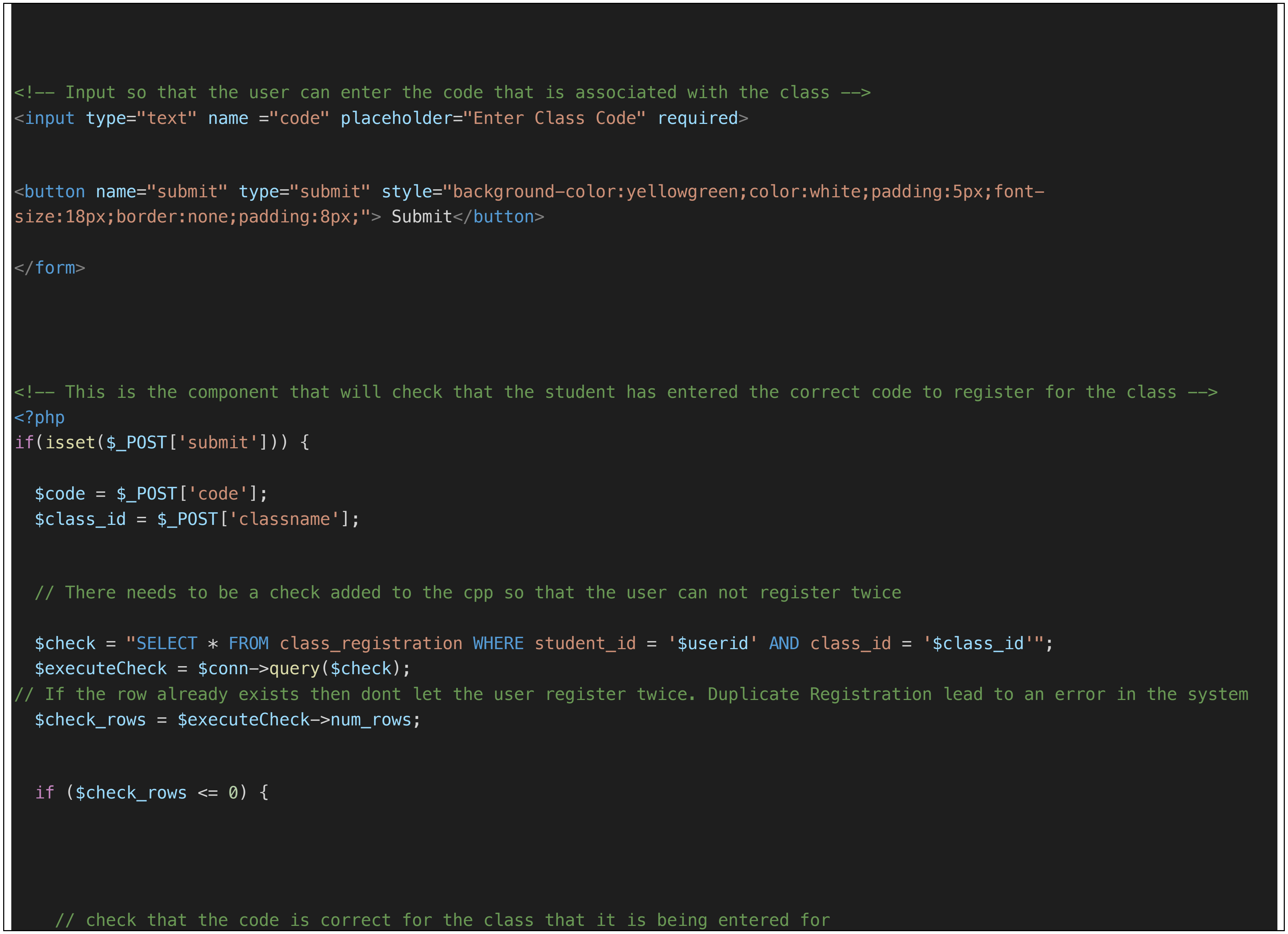
Class Registration

This is the Registration Component check that the student has registered for the class. This will register the student's successful registration and display below the Select component that I have outlined in the component above. This can be found in **student_level/registration.php**

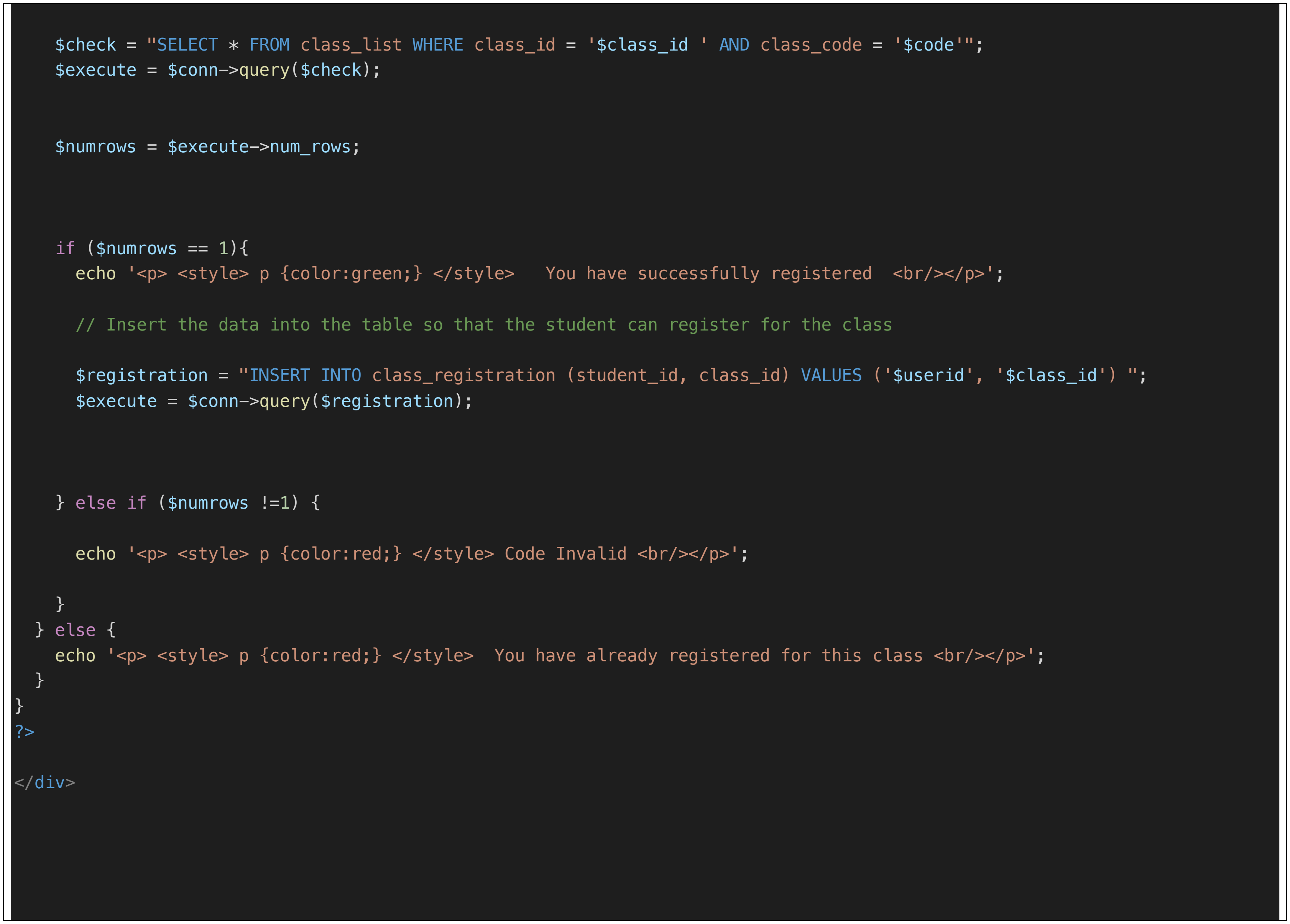
```html
<div class='container darker'>





<!-- Use  the JavaScript to send the rating to the database -->




```



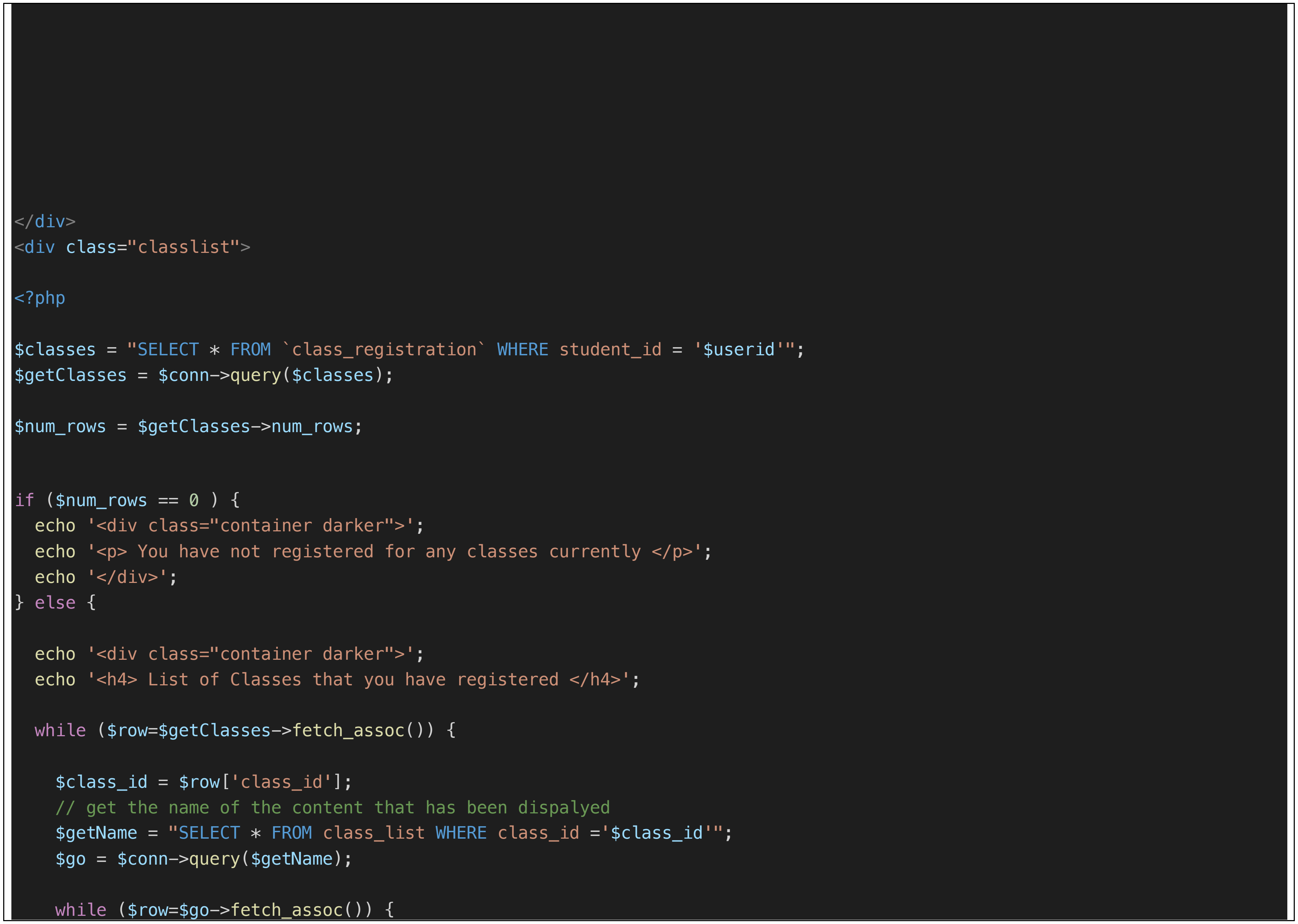
```
<!-- This is the Registration componenant that captures the student input when they want to register for a class -->
<h1> Register Here</h1>
        <h4>Select the Relevent Class</h4>
        <!-- FORM tag to return the data to the same file -->
        <form action="registration.php" class="update-resource" method ="POST">


<select  placeholder= "Class Name" name="classname" required >

<?php

// Get all the classes that exits so that these can be dispalyed in the select tag

$classes = "SELECT * FROM class_list";
$result = $conn->query($classes);


echo "<option > Select a Class  </option>";
while ($row=$result->fetch_assoc()) {
    $class_name= $row['class_details'];
    $class_date= $row['class_date'];
    $class_id = $row['class_id'];


    echo "<option value='$class_id'> $class_name $class_date</option>";

}


?>






    </select>
```



```
<!-- Input so that the user can enter the code that is associated with the class -->
<input type="text" name ="code" placeholder="Enter Class Code" required>


<button name="submit" type="submit" style="background-color:yellowgreen;color:white;padding:5px;font-
size:18px;border:none;padding:8px;"> Submit</button>

</form>




<!-- This is the component that will check that the student has entered the correct code to register for the class -->
<?php
if(isset($_POST['submit'])) {

  $code = $_POST['code'];
  $class_id = $_POST['classname'];


  // There needs to be a check added to the cpp so that the user can not register twice

  $check = "SELECT * FROM class_registration WHERE student_id = '$userid' AND class_id = '$class_id'";
  $executeCheck = $conn->query($check);
// If the row already exists then dont let the user register twice. Duplicate Registration lead to an error in the system
  $check_rows = $executeCheck->num_rows;


  if ($check_rows <= 0) {




    // check that the code is correct for the class that it is being entered for
```



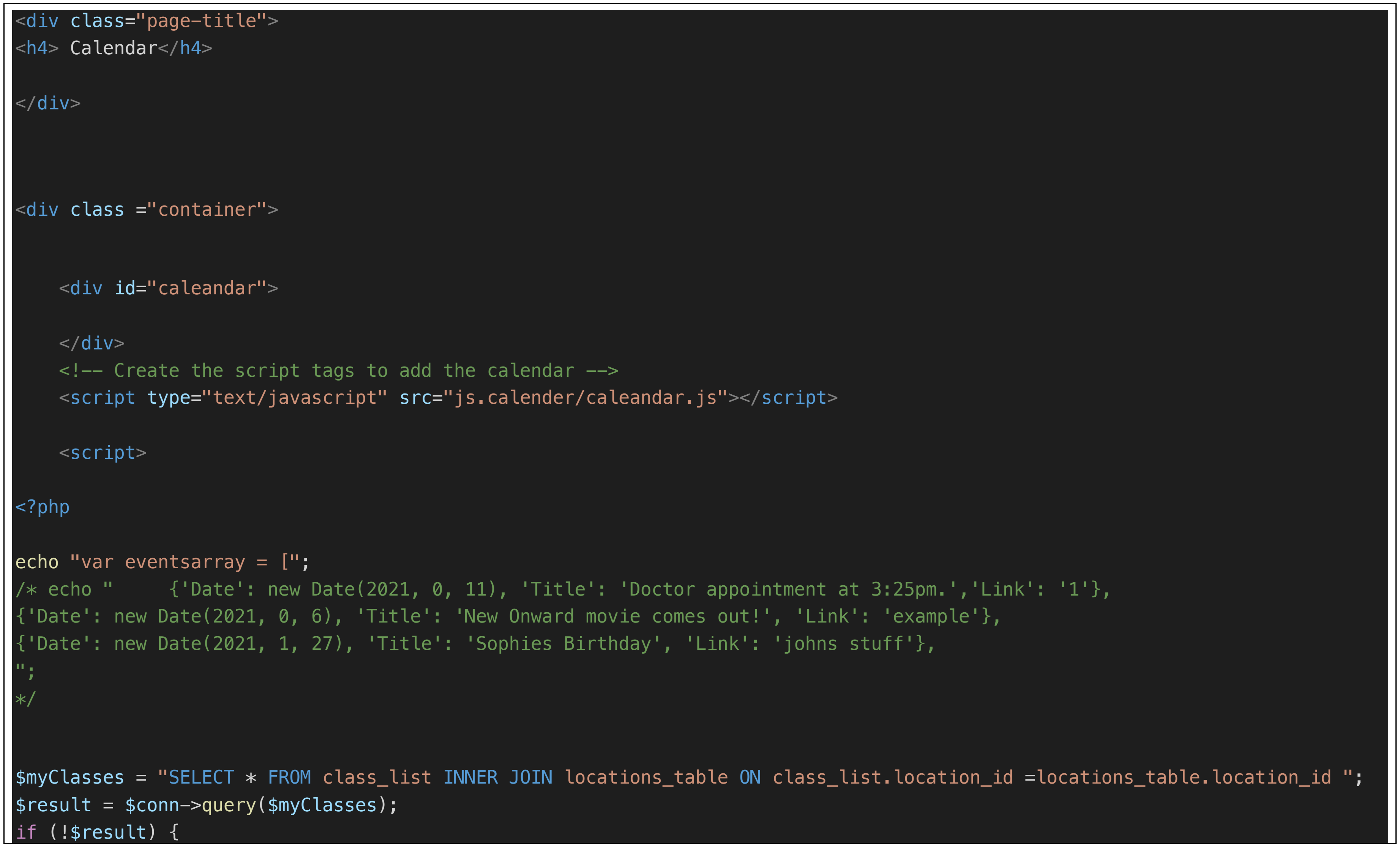
```
    $check = "SELECT * FROM class_list WHERE class_id = '$class_id ' AND class_code = '$code'";
    $execute = $conn->query($check);


    $numrows = $execute->num_rows;



    if ($numrows == 1){
      echo '<p> <style> p {color:green;} </style>   You have successfully registered  <br/></p>';

      // Insert the data into the table so that the student can register for the class

      $registration = "INSERT INTO class_registration (student_id, class_id) VALUES ('$userid', '$class_id') ";
      $execute = $conn->query($registration);



    } else if ($numrows !=1) {

      echo '<p> <style> p {color:red;} </style> Code Invalid <br/></p>';

    }
  } else {
    echo '<p> <style> p {color:red;} </style>  You have already registered for this class <br/></p>';
  }
}
?>

</div>
```



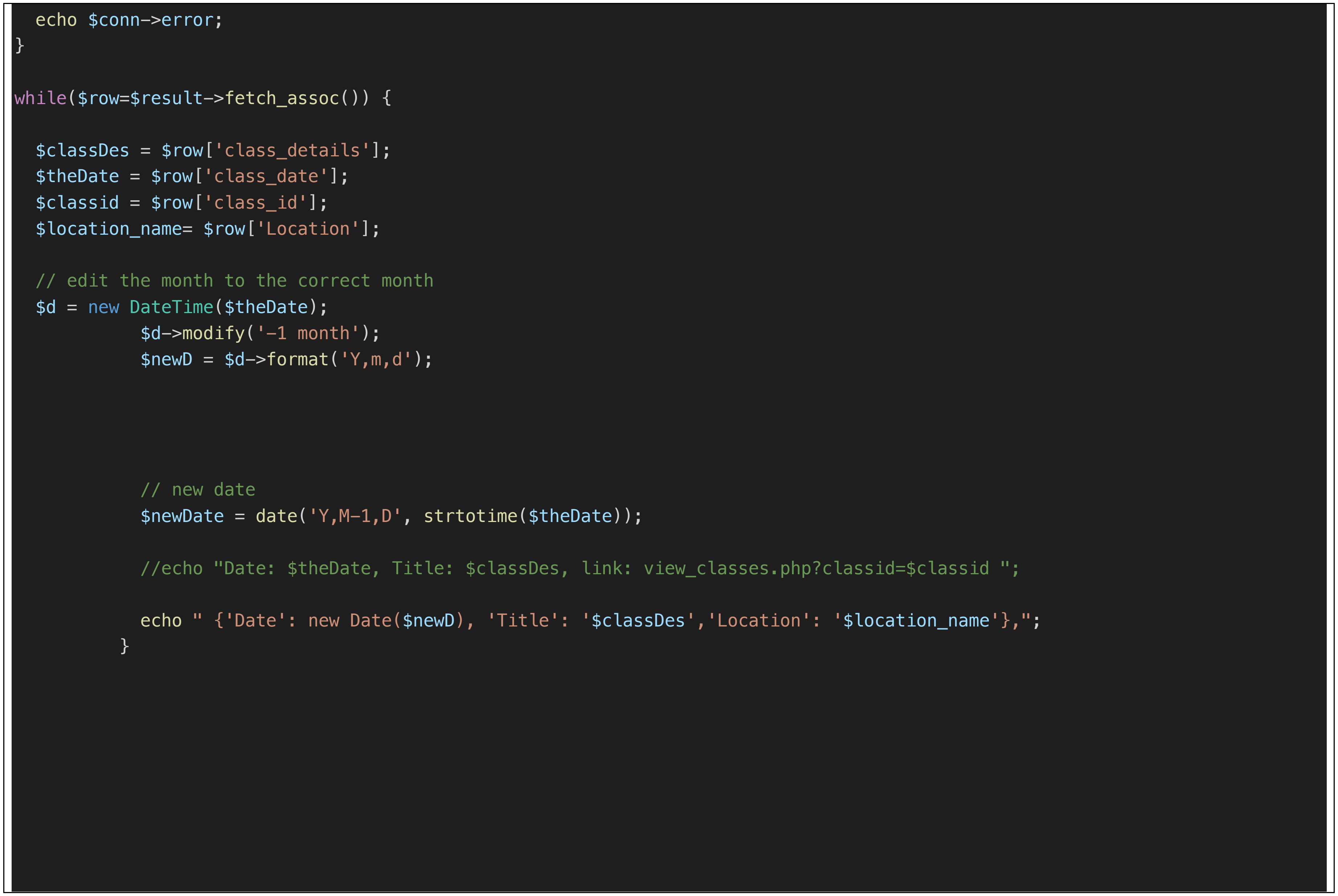
```
</div>
<div class="classlist">

<?php

$classes = "SELECT * FROM `class_registration` WHERE student_id = '$userid'";
$getClasses = $conn->query($classes);

$num_rows = $getClasses->num_rows;


if ($num_rows == 0 ) {
  echo '<div class="container darker">';
  echo '<p> You have not registered for any classes currently </p>';
  echo '</div>';
} else {

  echo '<div class="container darker">';
  echo '<h4> List of Classes that you have registered </h4>';

  while ($row=$getClasses->fetch_assoc()) {

    $class_id = $row['class_id'];
    // get the name of the content that has been dispalyed
    $getName = "SELECT * FROM class_list WHERE class_id ='$class_id'";
    $go = $conn->query($getName);

    while ($row=$go->fetch_assoc()) {
      echo '<div class="container darker">';
      $class = $row['class_details'];
      $class_date = $row['class_date'];

      echo " <img src='css.calender/tick.png' alt='Avatar'>";
      echo "<p>$class: $class_date</p>";
      echo '</div>';
    }







}
echo '</div>';


}
?>
```

Calendar Component

This Component is the Calendar Component that allows for students to see when their classes are taking place. This Component uses external Libraries so that it can function correctly. The student can only see the classes that exist for the current month. Once the month changes then the class cannot be viewed.



```
<div class="page-title">
<h4> Calendar</h4>

</div>



<div class ="container">


    <div id="caleandar">

    </div>
    <!-- Create the script tags to add the calendar -->
    <script type="text/javascript" src="js.calender/caleandar.js"></script>

    <script>

<?php

echo "var eventsarray = [";
/* echo "     {'Date': new Date(2021, 0, 11), 'Title': 'Doctor appointment at 3:25pm.','Link': '1'},
{'Date': new Date(2021, 0, 6), 'Title': 'New Onward movie comes out!', 'Link': 'example'},
{'Date': new Date(2021, 1, 27), 'Title': 'Sophies Birthday', 'Link': 'johns stuff'},
";
*/


$myClasses = "SELECT * FROM class_list INNER JOIN locations_table ON class_list.location_id =locations_table.location_id ";
$result = $conn->query($myClasses);
if (!$result) {
```



```
  echo $conn->error;
}

while($row=$result->fetch_assoc()) {

  $classDes = $row['class_details'];
  $theDate = $row['class_date'];
  $classid = $row['class_id'];
  $location_name= $row['Location'];

  // edit the month to the correct month
  $d = new DateTime($theDate);
            $d->modify('-1 month');
            $newD = $d->format('Y,m,d');




            // new date
            $newDate = date('Y,M-1,D', strtotime($theDate));

            //echo "Date: $theDate, Title: $classDes, link: view_classes.php?classid=$classid ";

            echo " {'Date': new Date($newD), 'Title': '$classDes','Location': '$location_name'},";
          }
```

```
   echo "  ];";

 ?>
     var settings = { NavShow: false,  EventTargetWholeDay: false, EventClick: function(Link) {console.log(Link);} };

     var calelement = document.getElementById('caleandar');

     caleandar(calelement, eventsarray, settings);

   </script>

</div>
```

Announcement Component

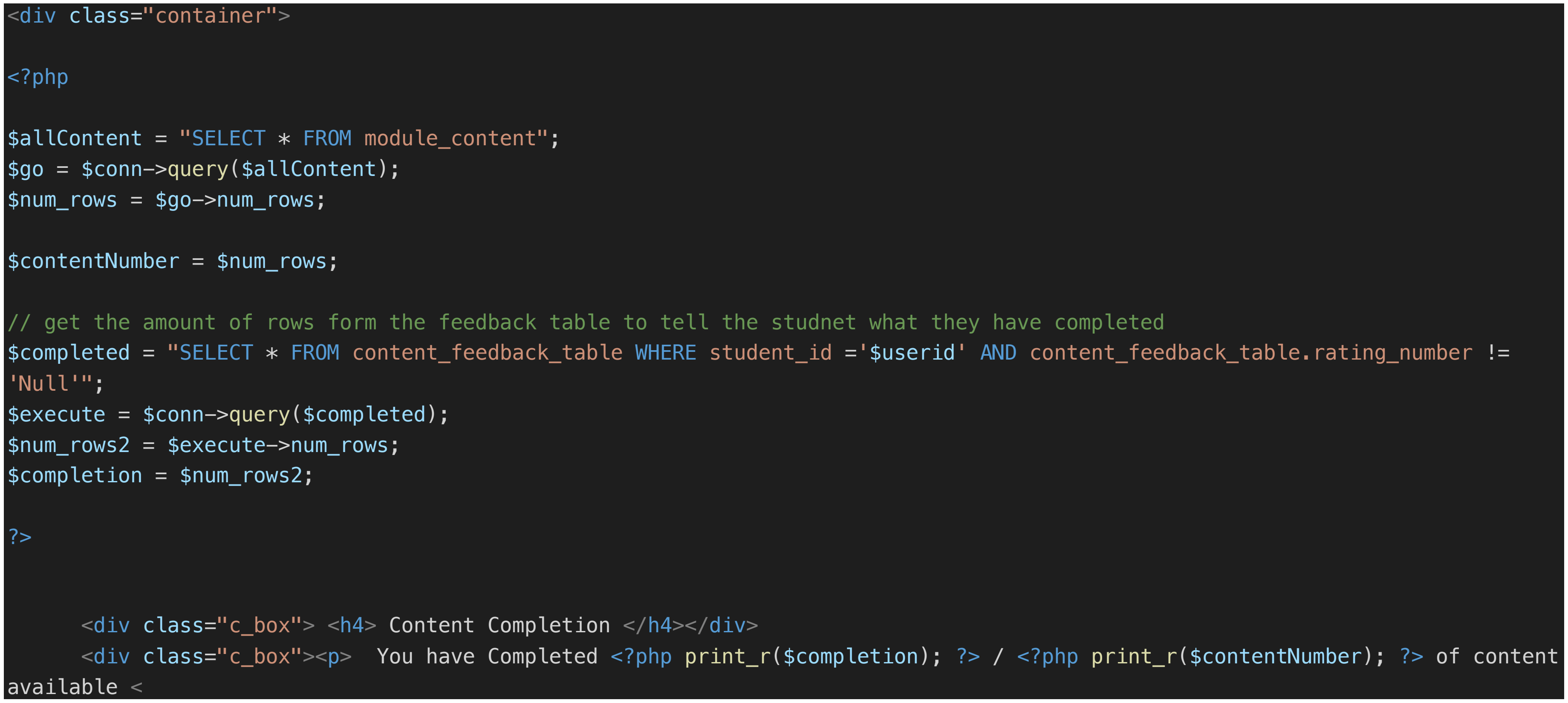
The Announcement Component will show the students the announcements that have been made by their tutor. This will be returned in order of how recent the tutor created the announcement. This can be found in student_level/student_homepage.php.

```
<?php

// Select all the conetent from the relevent table

// If there is no announcements then return a message saying that there is no announcements currently
$announcements = "SELECT * FROM `Announcement_table` ORDER BY date DESC";
$go = $conn->query($announcements);


$num_rows = $go->num_rows;
```

```
if ($num_rows <=0 ) {

  echo  "<div class='container darker'>";
  echo "<h4> Announcement Table </h4>";
  echo "<p> There is currenly no Announcements! </p>";



  echo "</div>";

} else {



while ($row=$go->fetch_assoc()) {

  $theTitle = $row['title'];
  $theComment= $row['comment'];
  $temporyid = $row['tutor_id'];
  $time = $row['date'];
  // get the tutor details to display them students. Grab the Tutor ID from the Announcement Table, pass it into the following
selelct statement. A join cannnt be used in this context!

  $getTutorDetails = "SELECT * FROM tutor_details WHERE tutor_id ='$temporyid'";
  $get = $conn->query($getTutorDetails);







  echo "<div class='container darker'>";


  while ($row2=$get->fetch_assoc()) {
    $fname = $row2['tutor_first_name'];
    $sname = $row2['tutor_second_name'];
    $file = $row2['imgpath'];

    // create a variable using the correct directory
    $studentFile = "../tutor_level/$file";



          echo "<img src='$studentFile' alt='Avatar'>";
          echo "<p> Tutor : $fname $sname</p>";
        }
        echo "<p>$theTitle</p>";

        echo "<p>$theComment</p>";
        echo "<span class='time-right'>$time</span>";







        echo "</div>";

}
}
?>





</body>
</html>
```

Content Completion

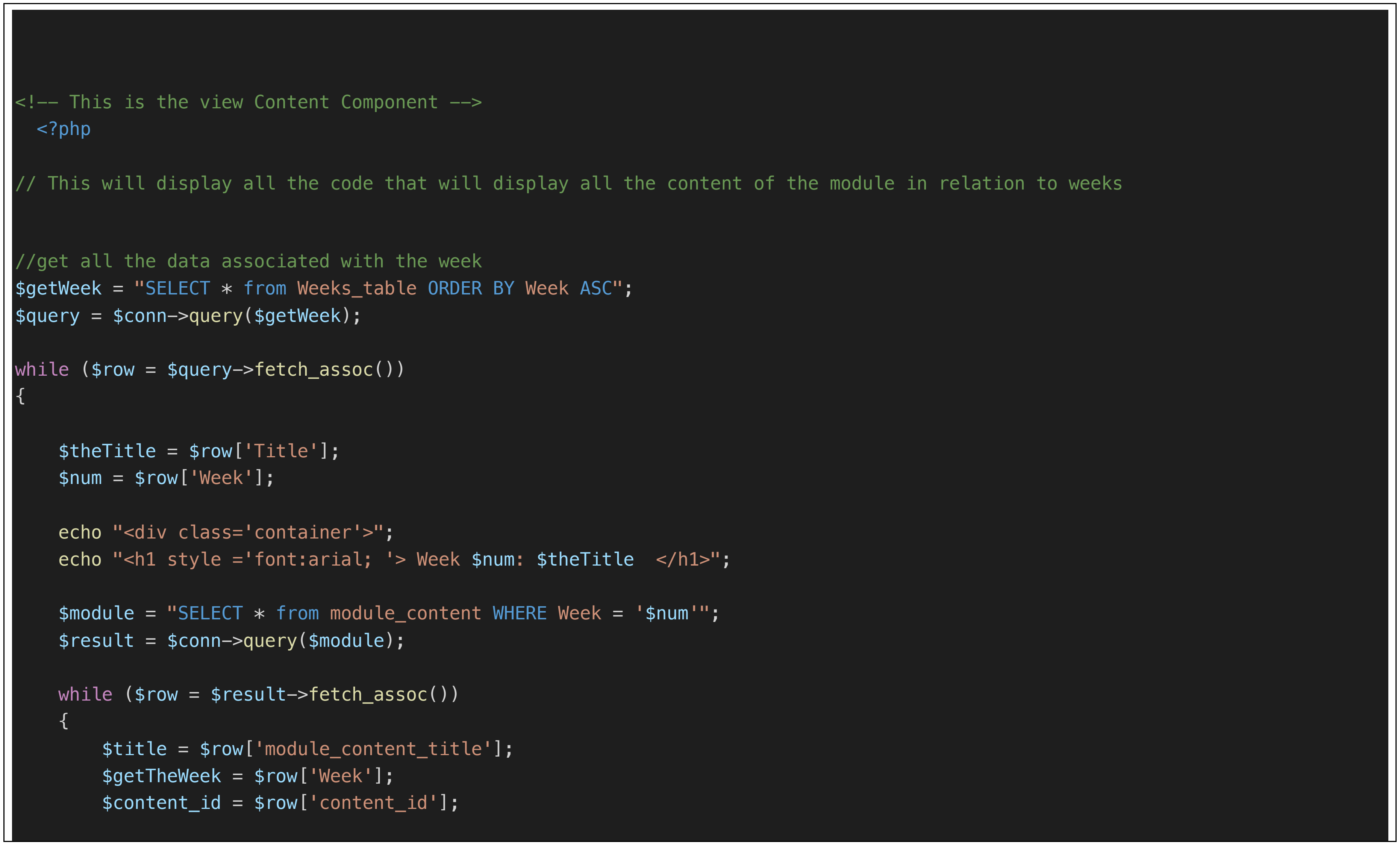
This is the component that will be used so that the student can track what content they have completed and how much content is left to complete. This component takes the total amount of content that exists within the module and compares this with the information in the content feedback table belonging to the student. This data is then passed in variables that are compared in the HTML and CSS to give the user a physical representation of the data. This component can be found in the student_level/ student_module.php.



```
<div class="container">

<?php

$allContent = "SELECT * FROM module_content";
$go = $conn->query($allContent);
$num_rows = $go->num_rows;

$contentNumber = $num_rows;

// get the amount of rows form the feedback table to tell the studnet what they have completed
$completed = "SELECT * FROM content_feedback_table WHERE student_id ='$userid' AND content_feedback_table.rating_number !=
'Null'";
$execute = $conn->query($completed);
$num_rows2 = $execute->num_rows;
$completion = $num_rows2;

?>


      <div class="c_box"> <h4> Content Completion </h4></div>
      <div class="c_box"><p>  You have Completed <?php print_r($completion); ?> / <?php print_r($contentNumber); ?> of content
available <
```

View Module Component

This is the component that will allow for students to view their module content. This component should display all the content that belongs to the weeks that the tutor has created. This should be displayed in a clear way with all the content directing the user to their own unique page within the system. This can be found in the student_level/student_module.php page



```
<!-- This is the view Content Component -->
  <?php

// This will display all the code that will display all the content of the module in relation to weeks


//get all the data associated with the week
$getWeek = "SELECT * from Weeks_table ORDER BY Week ASC";
$query = $conn->query($getWeek);

while ($row = $query->fetch_assoc())
{

    $theTitle = $row['Title'];
    $num = $row['Week'];

    echo "<div class='container'>";
    echo "<h1 style ='font:arial; '> Week $num: $theTitle  </h1>";

    $module = "SELECT * from module_content WHERE Week = '$num'";
    $result = $conn->query($module);

    while ($row = $result->fetch_assoc())
    {
        $title = $row['module_content_title'];
        $getTheWeek = $row['Week'];
        $content_id = $row['content_id'];
        // This will check if the student has completed the lab work or not yet
        $completed = "SELECT * FROM content_feedback_table WHERE student_id = '$userid' AND  content_id = '$content_id'";
        $execute = $conn->query($completed);

        $num_rows = $execute->num_rows;

        echo "<div class='container darker'>";
        echo "<p> <a href='student_view_module.php?contentid=$content_id' style ='color:black;  '> $title </a>   </p>";

        if ($num_rows > 0)
        {
            echo "<p style ='color:green;'> Completed </p>";
        }

        echo "</div>";

    }
```

__Student Feedback System Component.__

This component was referenced in chapter 5 and outlines how the. Application captures the data that it is suppose. The component captures data by prompting the user to give feedback on the content that they have just created. This component uses a significant amount of JavaScript and PHP to make it work functionally and dynamically. This can found in the student_level/ student_view_module.php.

This will prompt the user to input the feedback and will display the feedback once they have provided feedback.

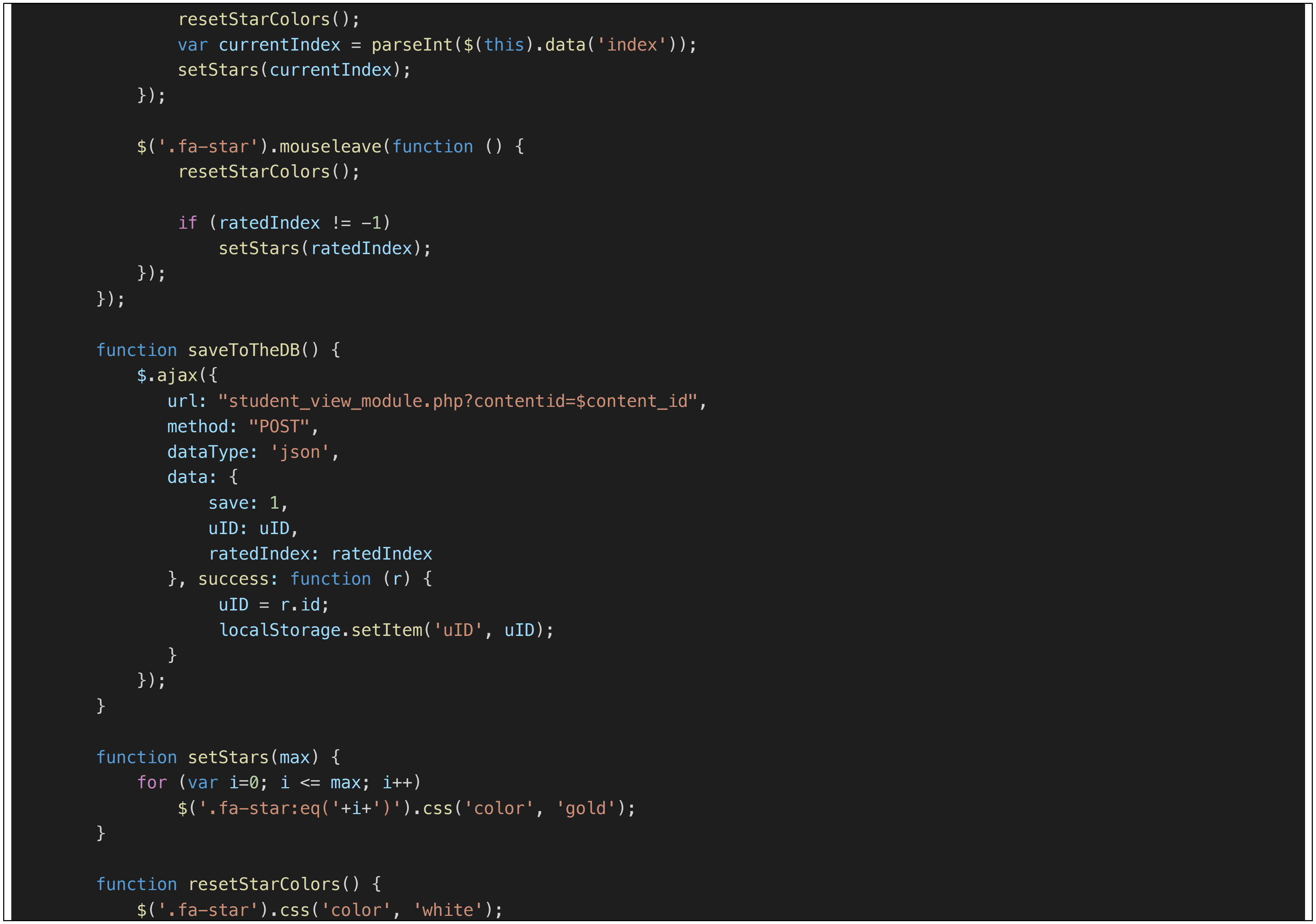
```
<div class='container darker'>
<div class="give-feedback-title">
<h2>Give Feedback about the content</h2>
```

```
</div>



<div  style = "background: #000 padding: 150px;">
  <i class = "fa fa-star" data-index = "0"></i>
  <i class = "fa fa-star" data-index = "1"></i>
  <i class = "fa fa-star" data-index = "2"></i>
  <i class = "fa fa-star" data-index = "3"></i>
  <i class = "fa fa-star" data-index = "4"></i>

</div>

<div>


<!-- Code that will create a feedback comment -->
<form action="student_view_module.php"  id = "feedback" class="feedback" method ="POST">

<div class="feedback">
    <input type="text" placeholder="Comment" name="comment" width ='100' height ='100' maxlength = '2000'>

    <button name="submit" type="submit"> Submit Feedback</button>




  </div>


</form>

<?php
// Check if the user has already provided feedback about this content and tell them if so
```

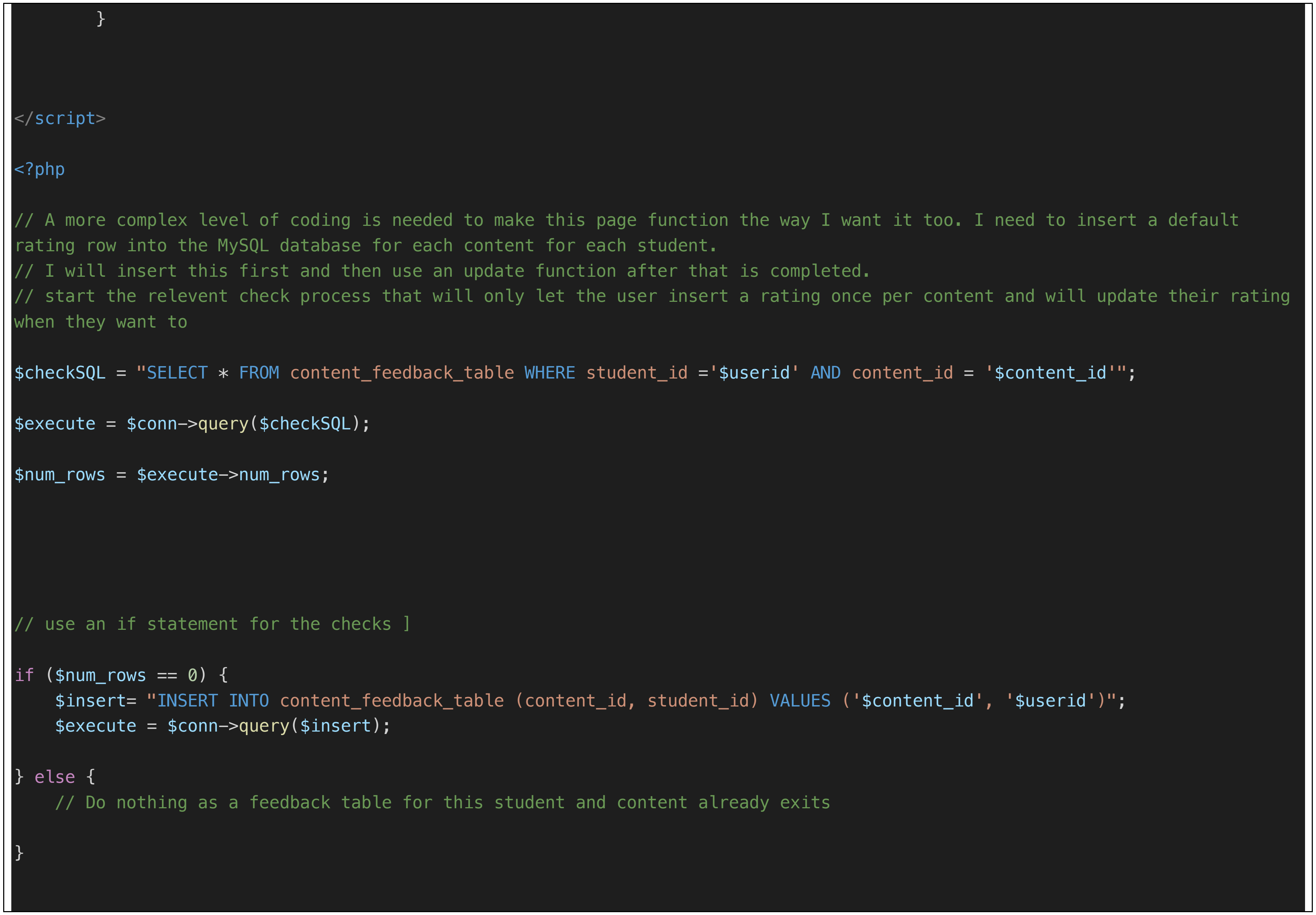
```php
$checkStars = "SELECT * FROM content_feedback_table WHERE student_id = '$userid' AND content_id ='$content_id'";
$executeStars = $conn->query($checkStars);

$exist = $executeStars->num_rows;

if ($exist > 0 ) {

    while ($row=$executeStars->fetch_assoc()) {
        $feedback = $row['rating_number'];

    }


  echo "<p> You have already provided feedback for this content Rating: $feedback</p>";

//   This will return a message based on the feedback that was given
  if ($feedback==1) {
    echo '<p> <style> color:black; </style>  You struggled with this content</p>';
  } elseif ($feedback==2) {
    echo '<p> <style> color:red; </style>  You said you felt unsatisfied with the content </p>';
  } elseif ($feedback==3) {
    echo '<p> <style> color:red; </style>  You said that you where satisfied with this content </p>';
  } elseif ($feedback==4) {
    echo '<p> <style> color:gold;</style>  You said you performed well with this content</p>';
  } elseif ($feedback==5) {
    echo '<p> <style> color:green; </style>  You said you preformed extremly well with this content </p>';
  }






}
```

This is the JavaScript that was used so that the student could interact with the feedback system in a dynamic way. This part is also responsible for passing the data to the database so that it can be successfully stored and then processed by other components.

```
<!--  This is the JavaScript that will be used for the functionality of the star system that is set up on the User Interface -
->
<script src="https://code.jquery.com/jquery-3.5.1.min.js" integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
crossorigin="anonymous"></script>
<script>


var ratedIndex = -1, uID = <?php echo $content_id ?>;

        $(document).ready(function () {
            resetStarColors();

            if (localStorage.getItem('ratedIndex') != null) {
                setStars(parseInt(localStorage.getItem('ratedIndex')));

            }

            $('.fa-star').on('click', function () {
               ratedIndex = parseInt($(this).data('index'));
               localStorage.setItem('ratedIndex', ratedIndex);
               localStorage.setItem('uID', uID);

               saveToTheDB();
            });

            $('.fa-star').mouseover(function () {
```



```
            resetStarColors();
            var currentIndex = parseInt($(this).data('index'));
            setStars(currentIndex);
        });

        $('.fa-star').mouseleave(function () {
            resetStarColors();

            if (ratedIndex != -1)
                setStars(ratedIndex);
        });
    });

    function saveToTheDB() {
        $.ajax({
           url: "student_view_module.php?contentid=$content_id",
           method: "POST",
           dataType: 'json',
           data: {
               save: 1,
               uID: uID,
               ratedIndex: ratedIndex
           }, success: function (r) {
                uID = r.id;
                localStorage.setItem('uID', uID);
           }
        });
    }

    function setStars(max) {
        for (var i=0; i <= max; i++)
            $('.fa-star:eq('+i+')').css('color', 'gold');
    }

    function resetStarColors() {
        $('.fa-star').css('color', 'white');
```



```
        }



</script>

<?php

// A more complex level of coding is needed to make this page function the way I want it too. I need to insert a default rating row into the MySQL database for each content for each student.
// I will insert this first and then use an update function after that is completed.
// start the relevent check process that will only let the user insert a rating once per content and will update their rating when they want to

$checkSQL = "SELECT * FROM content_feedback_table WHERE student_id ='$userid' AND content_id = '$content_id'";

$execute = $conn->query($checkSQL);

$num_rows = $execute->num_rows;





// use an if statement for the checks ]

if ($num_rows == 0) {
    $insert= "INSERT INTO content_feedback_table (content_id, student_id) VALUES ('$content_id', '$userid')";
    $execute = $conn->query($insert);

} else {
    // Do nothing as a feedback table for this student and content already exits

}
```

```
if (isset($_POST['save'])) {

// Set the relevent variables

$ratedIndex = $conn->real_escape_string($_POST['ratedIndex']);
$ratedIndex++;
$contentid = $conn->real_escape_string($_POST['uID']);


$update = "UPDATE content_feedback_table SET rating_number = '$ratedIndex' WHERE student_id = '$userid' AND content_id =
'$contentid'";
$execute = $conn->query($update);

echo "<p>$update</p>";
exit(json_encode(array('id' => $uID)));


}

?>
```

View Content Information

This component will display the information that belongs to the content page that we are currently viewing. This should display the information with a title making it easy to identify what content belongs to what title. This can be found on the student_level/ student_view_module.php.

```
!-- This will show the sub content that is associated with the content that student will user to learn  -->
<?php

$getContentInfo = "SELECT * FROM module_content_information WHERE content_id = '$content_id'";
$result = $conn->query($getContentInfo);

while($row=$result->fetch_assoc()) {
  $contentTitle = $row['Title'];
  $description = $row['Information'];


  echo  "<div class='container darker'>";
  echo "
  <p>    $contentTitle </p>";
  echo "<p>$description</p>";


  echo "</div>";




  //echo "<p> <a href='student_profile.php?studentid=$student_id'> Student: $first_name  $second_name </a>    </p>";




}

?>
</body>
```

Tutor Level Component Testing

I have included only components that are either Primary or Secondary Components due to time constraints.

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
General Attendance	Test that the General Attendance is returning the correct Attendance	View the tutor home page.	No input required.	Add all the values in the database to determine that the correct data is being returned.	The component should return the correct data.	The component is returning the correct data percentage.	Y
Class Attendance Component	Test that the individual class component is returning the correct data.	View the tutor home page.	Select a Class to return the attendance for.	Select a class to view if it returns the correct data.	The component should return the attendance data for the class selected.	If the component returns the correct data for the class selected.	Y
Add Announcement Function.	Test that the application can add Announcements to the system.	The user needs to click the add announcement button on the side bar.	Enter information to create an announcement.	Enter announcement details and click submit.	The system should pass the correct information, and this should be displayed.	If the information that was sent by the user has been sent.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
View Announcement function.	Test that the View Announcement is returning the correct Announcement.	View the tutor home page.	No input required.	Confirm that the data is being returned. And that the attendance is correct	The component should return the announcement to that was entered to the home page.	The announcement should be returned to the home page.	Y
Individual Student Attendance	Test that the attendance for the student's individual profile	View the view student page.	No input needed.	Confirm that the data is being returned. And that the attendance is correct	Confirm that the attendance of individual student is being returned/	The correct data should be returned with the correct value.	Y
Average Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Individual Content Feedback	Test that the individual content feedback component is working.	View the view student page.	Select the content that you want to view feedback.	Confirm that the data is being returned. And that the data is correct and relevant	The component should return the correct data in the correct format.	The correct data is being retuned in the correct format to the user.	Y
Content Completion	Test that the content completion component is working.	View the view student page.	No input needed.	Confirm that the data is being returned. And that the content completion is correct	Confirm that the content completion value is being returned.	The correct data should be returned with the correct value.	Y
Average Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
View Module	Test that the user can view the module content that exits	View the tutor home page.	No input needed.	Confirm that the data is being returned. And that the data is correct and relevant for the module content.	The component will return the correct data in the correct format.	The correct data is being retuned in the correct format to the user.	Y
View Module Content	Test that the user can view the module content that is needed.	View the view module content page	No input needed.	Confirm that the data is being returned. And that the content completion is correct.	Confirm that the content completion value is being returned.	The correct data should be returned with the correct value.	Y
Average Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Individual Content Feedback	Test that the individual content feedback component is working.	View the view student page.	Select the content that you want to view feedback.	Confirm that the data is being returned. And that the data is correct and relevant	The component should return the correct data in the correct format.	The correct data is being retuned in the correct format to the user.	Y
Content Completion	Test that the content completion component is working.	View the view student page.	No input needed.	Confirm that the data is being returned. And that the content completion is correct	Confirm that the content completion value is being returned.	The correct data should be returned with the correct value.	Y
General Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Add Content Information	Test that the user can add module content information	View the view module content page	Input relevant content information	Confirm that the data is being returned. And that the data is correct and relevant for the module content.	The component will return the correct data in the correct format.	The correct data is being retuned in the correct format to the user after being successfully added.	Y
Add module content	Test that the user can add module content to the application	View the tutor module page	Input relevant module content data.	Confirm that the data is being added to the application in the module content section	Confirm that the content completion value is being returned.	The correct data should be returned with the correct value.	Y
Average Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
View Discussion Forum	Test that the user can see the comments that have been added to the discussion forum	View the Discussion Forum page.	No Input needed	Confirm that the data is being returned. And that the data is correct and relevant.	The component will return the correct data in the correct format.	The correct data is being retuned in the correct format to the user when they view the page.	Y
Add Discussion Forum	Test that the user can add a comment to the discussion forum	View the Discussion Forum page.	Input relevant data.	Confirm that the data is being added to the application in the discussion forum.	Confirm that the component is returning the data to the discussion forum.	The correct data should be returned with the correct value to the discussion forum.	Y
Average Content Feedback	Test that the application can display the students content feedback rating.	The user needs to click on the view student page to view the content.	No input required.	View that the component is returning the correct data about the student.	The system should pass the correct data in relation to the student and their feedback.	The correct data it returned and is correct.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Add Student Function	Test that the application can create a new student profile	The user needs to view the view students page	Add the student's relevant details	View that the component is returning the correct data about the student.	The system should pass the students personal data to the system to be stored	The correct data is passed and retuned	Y
View Student Function	Test that the application will list all the students that exist in order.	The user needs to view the view students page	No input needed	View that the component is retuning the correct data in the correct format	The system should return the correct data in the correct format	The correct data is returned to the user.	Y

Student Level – Component Testing

I have included only components that are either Primary or Secondary Components due to time constraints.

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Login	Test that the user can sign in successfully with valid details.	Enter the details when prompted	A valid Username and Password.	Enter a valid Username and Password Combination	The system should direct the user to the home page.	If the student is returned to their personalised home page	Y
Log Out	Test that the user can log out of their account.	Click on the log out button	Click on the log out button.	Click on the log out button.	The system should destroy the session and return the user to the login page	If the session is destroyed and the user is returned to the login page.	Y
Class Registration Component	Test that the Class Registration component collects the correct data and registers the student.	The student needs to register for their classes.	Enter a valid class and class code for the component to pass to server side code.	Enter a valid code and class combination. View the SELECT statement that is being used.	The system should pass the correct details. The student should register for the class.	If the class and code combination is valid then the student should be registered for the class.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Calendar Component	The student can see the classes that have been created by the tutor.	The student opens the calendar pages.	No input required.	The user views if the calendar is returning the correct date	The calendar should return the correct class on the correct date.	If the calendar correctly returns the correct dates and name of the classes to the user.	Y
Announcement Component.	The student can see the announcement s that the tutor has made.	The user opens the student homepage.	No input required.	The user views that the system is returning tutor announcements	The announcement component should return announcement In the correct order.	The component will return all the announcements in the relevant order.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Content Completion	The system should return the correct amount of content that has been completed by the student.	The user will open the student Module page.	No Input required	The user will view the page that has the with the Content Completion Component.	The system will return the correct data.	If the system returns data that matches the how much data that the student has completed.	Y
View Module Component.	The student can see the module content and which week that the content belongs to.	The user opens the student view module page.	No input required.	The user views that the system is returning the module content in the correct format belonging to the correct week.	The view module component should return the module data in the correct format.	The component will return the module content in a structured way with the week that the module belongs to.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Content Completion	The system should return the correct amount of content that has been completed by the student.	The user will open the student Module page.	No Input required	The user will view the page that has the with the Content Completion Component.	The system will return the correct data.	If the system returns data that matches the how much data that the student has completed.	Y
View Module Component.	The student can see the module content and which week that the content belongs to.	The user opens the student view module page.	No input required.	The user views that the system is returning the module content in the correct format belonging to the correct week.	The view module component should return the module data in the correct format.	The component will return the module content in a structured way with the week that the module belongs to.	Y

Component	Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed	Result
Student Feedback Component	The component should be dynamic and capture the feedback that the student wants to give the content that has been set.	The user will open the view module content page.	The user will give the content a rating from 1-5.	The user will provide an input in relation to the content. All 5 different rating should be given to be sure that all are functional.	The system will capture all feedback and will display that to the user once the page is refreshed.	If the system returns the rating that the user has provided on all 5 different inputs.	Y
View Content Information Component	The student can see the module content information.	The user opens the student view module page.	No input required.	The user views that the system is returning the module content information in the correct format belonging to the correct week.	The component should return the information in the correct format so that the student can user the app correctly.	The component will return all the data that is associated with the content. Each part will be associated with a title so that it is easy to identify.	Y

Security Testing – Tutor Level and Student Level

Browser Security Testing – Tutor Level and Student Level

This test will involve the user trying to access the pages of the application without authorization. This is tested by accessing the URL through the browser and expecting the page to return you to the login page of the application.

Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed
Check that the user can't access the page without authorization	The user will try and access the page through the browser	Input the web address.	The user will attempt to access the website by typing the URL into the browser without creating a session first	The system will return the user to the login page	The system will return the user to the login page of the system after they have attempted to access the page without authorization.

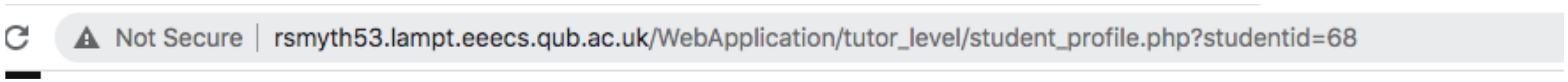
Results

Page Name	Result	Page Name	Result
Tutor Home	Passed	Student Home	Passed
Tutor Module	Passed	Student Module	Passed
Tutor View Module	Passed	Student View Module	Passed
Tutor Calendar	Passed	Student Calendar	Passed
Tutor Log out	Passed	Student Log out	Passed
View Students	Passed	Registration	Passed
Tutor Account Settings	Passed	Account Settings	Passed
Student Profile	Passed	Student Discussion Forum	Passed
Tutor Formulate Code	Passed		
Tutor Discussion Forum	Passed		

<u>Browser Security Testing – Tutor Level and Student Level</u>

This test will involve testing for pages that rely on the $GET Super variable. This means that some pages will have components that require access to the database to get data. Examples of this would be the individual student profile from the Tutor level or the discussion forum.

This is a depiction of a webpage with to access the unique profile with the "studentid" which is equal 68. If the user was to change the "studentid" in the URL to an invalid id then the user would be moved to the view students page. This will prevent the user from seeing any back-end errors



rsmyth53.lampt.eeecs.qub.ac.uk/WebApplication/tutor_level/student_profile.php?studentid=68

Description	Test Initialisation	Test Input	Test Procedure	Expected Results	Passed

Check that the user can’t access the page without entering data that exists	The user will try and access the page through the browser	Input the web address with an invalid ID	The user will attempt to access the website by typing the URL into the browser with data that doesn’t exist	The system will return the user to a page that does exist	The system will return the user to a page that does exist and only allow them to access pages that do exist.

Results

Page Name	Result	Page Name	Result
Student Profile	Passed	Student Module	Passed
Tutor View Module	Passed	Student View Module	Passed
Tutor Discussion Forum	Passed		Passed