



# WEB APPLICATION DEVELOPMENT

CIS2169

## Contents

<b>Repository URL:</b> .....	2
<b>Netlify URL:</b> .....	2
1.0 Introduction .....	3
2.0 Diagrams .....	3
3.0 Sketches .....	5
5.0 Code Implementation .....	7
6.0 Deployment planning and practice. ....	16
7.0 Final Application (final output when opening the application).....	21
8.0 Analysis Post-mortem.....	22
9.0 Conclusion .....	23
10.0 References.....	23

## Table of Figures

Figure 1: Activity Diagram.....	3
Figure 2: Class Diagram.....	4
Figure 3: Use case diagram .....	5
Figure 4: Sketch number 1 .....	6
Figure 5: Sketch number 2 .....	6
Figure 6: Sketch number 3 .....	7
Figure 7: Pushing in command line console.....	8
Figure 8: GitHub commits.....	9
Figure 9:GitHub commits.....	10
Figure 10: GitHub commits.....	11
Figure 11: Developers folder in Personal Computer.....	12
Figure 12: Create Module coding.....	13
Figure 13: Create degree coding.....	13
Figure 14: Create Assessment coding.....	14
Figure 15: Module Slots coding.....	14
Figure 16: Main JavaScript file coding .....	15
Figure 17: Netlify URL.....	17
Figure 18: Deployed Application.....	17
Figure 19: Test 1.....	18
Figure 20: Test 2.....	19
Figure 21: Test 3.....	19
Figure 22: Test 4.....	20
Figure 23: Test 5.....	20
Figure 24: Final Create Module Page.....	21
Figure 25: Final Create Degree Page.....	21
Figure 26: Final Create Assessment Page.....	22
Figure 27: Final Module Slots Page.....	22

Repository URL:

<https://github.com/IacovosIacovides/CW2-1.git>

Netlify URL:

<https://iacovoscw2.netlify.app/>

## 1.0 Introduction

An application developer has been asked to further develop and extend an already existing web-based academic management and tracking application, which at this point it tracks all academics research activities. The developer must enhance this application by improving several things that the application is lacking. For any type of work, easy and accurate planning should be followed to ensure that everything will be accurate and correct at the end. (Contributor, 2020) In this case, a correct plan includes, designing the system using appropriate UML diagrams and ensuring that the links between the builds are made possible. Understanding the way that something links to another are one of the most important aspects when it comes to developing such an application. When the diagrams are made, the implementation face will come, where the developer will use the diagram to build an application with the same links that he has designed (Bluescape, 2019). When the code is finished, various tests should be made, which will eliminate the chances of errors when the application is published. All of the steps made will be explained in this report.

## 2.0 Diagrams

In this section, the UML diagrams that have been designed will be explained.

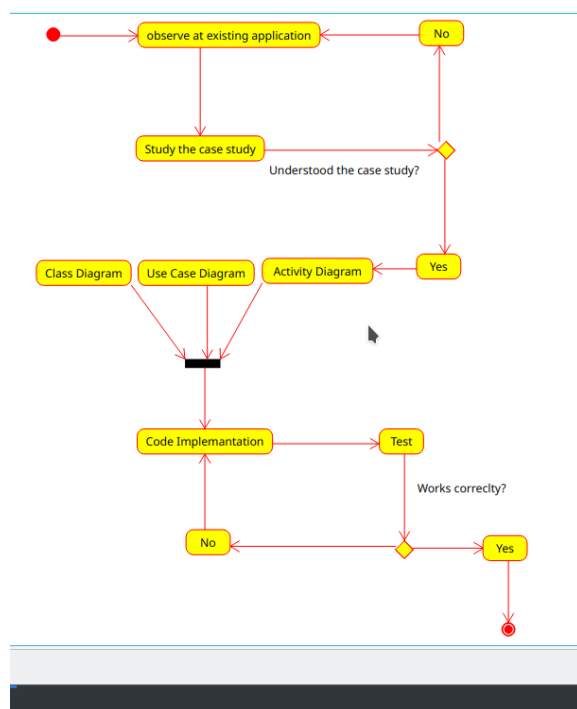


Figure 1: Activity Diagram

The diagram above is an Activity Diagram. It shows the way with which the developer will begin the process of understanding the case study, all the way to the very end which is deploying the application. The very first activity that the developer had to do is to observe the existing application, see what it is lacking, and then read and try to understand what the case study is asking, along with the aims and the objectives. If the developer understood what needs to be done, he would then create the diagrams and then continue to implement the code. After the appropriate tests have been done, he would deploy the application.

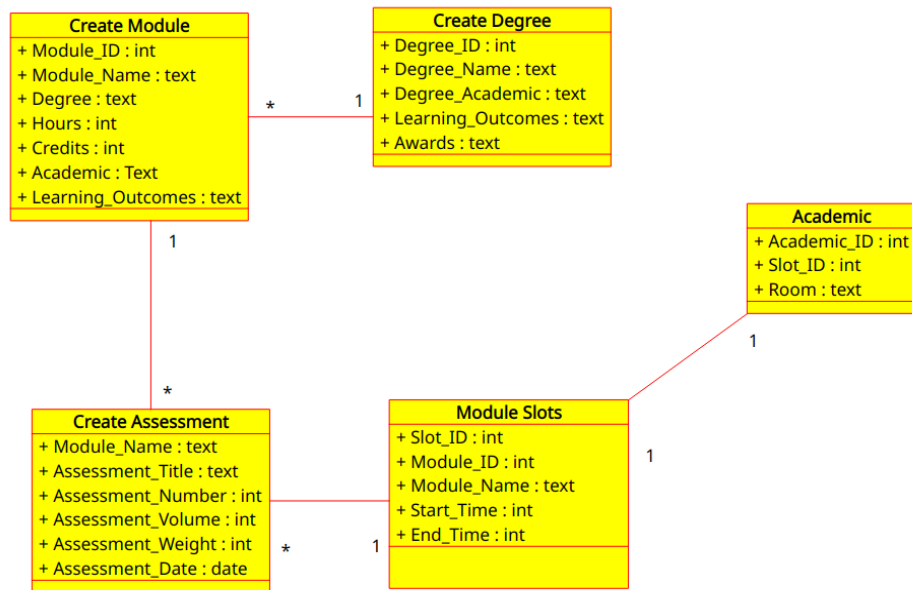


Figure 2: Class Diagram

The diagram above is a Class diagram. In this diagram, some classes have been developed with the aid of the given case study. The developer had read the case study over and over again until he managed to come up with these classes. The code will be built around these classes trying to connect every bit of code. Six classes have been made and each one has a primary key and a foreign key assigned to make the relationships between the classes possible and accurate.

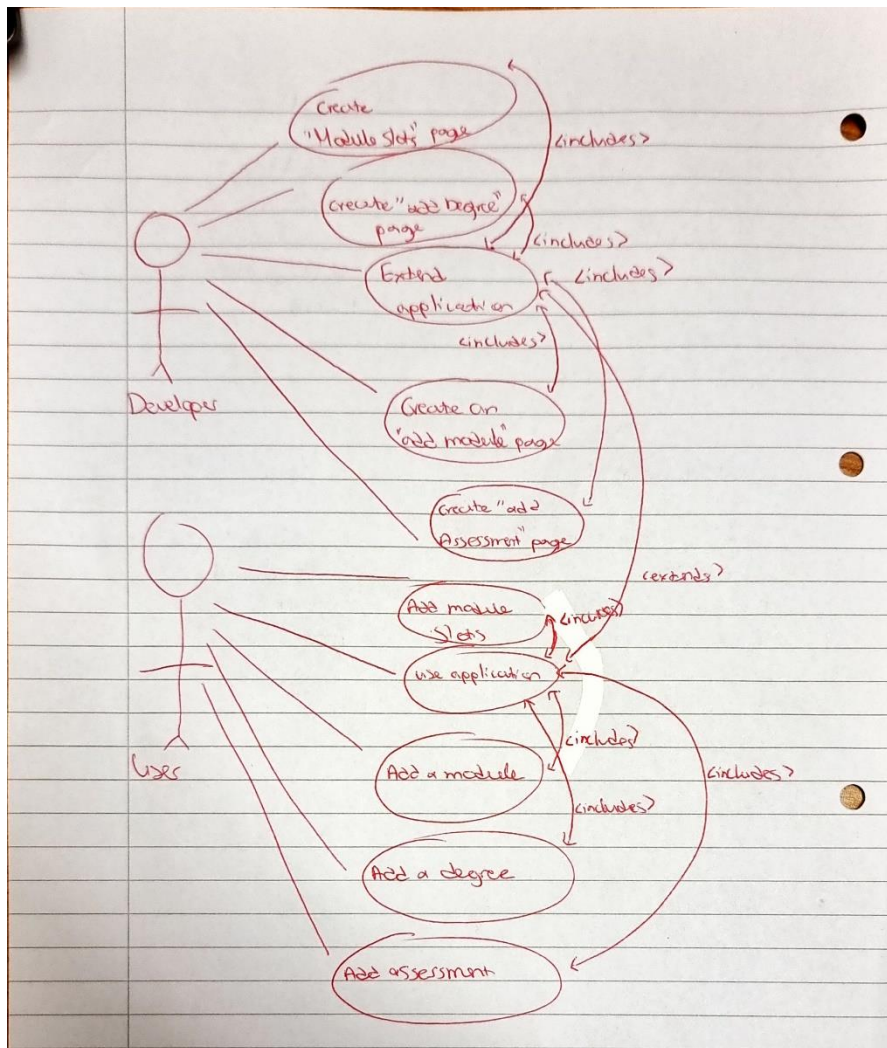


Figure 3: Use case diagram

The diagram above is the Use Case diagram that the developer has designed. The diagram shows a possible way with which the user will interact with the application's user interface.

### 3.0 Sketches

When a developer is asked to develop an application, one important thing to do before starting to code is to sketch some basic ways the page might look like. Thus, the developer has created some sketches about what the website's home page will look like along with the other pages. In this section, the sketches will be provided along with an explanation about each one.

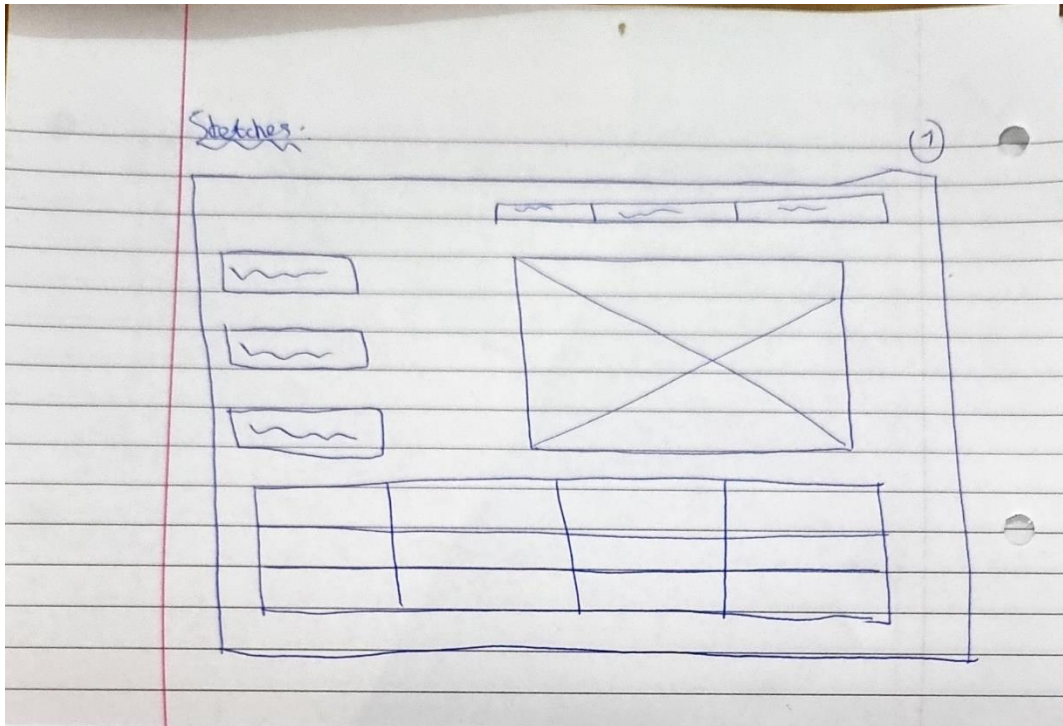


Figure 4: Sketch number 1

In the sketch above the home page is presented. There will be a bar on the top with which the user will be able to do various things, like adding a new module, creating a degree, or an assessment. There will be a quick way to add a module on the left-hand side of the page where the user can enter the name of the new module, the hours that it takes, the academic, etc. At the bottom of the page, there will be a table where all the modules will be displayed.

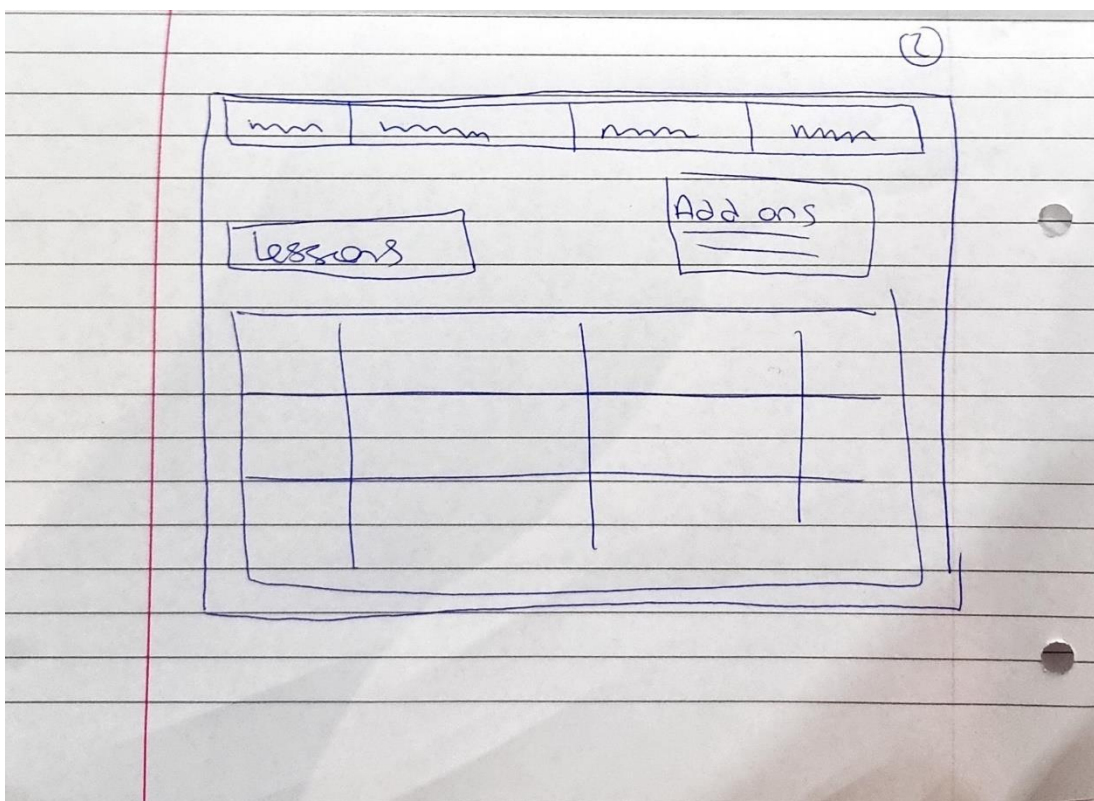


Figure 5: Sketch number 2



In the sketch above, another way in which the home page might look is presented. There will be a top navigation bar where the user can navigate to the rest adding pages, which include adding modules, creating degrees, and creating assessments. There will be a button called lessons which will be another gateway to the adding lessons page. At the bottom of the page, there will be a table showing the modules along with the rest of the important information about each module.

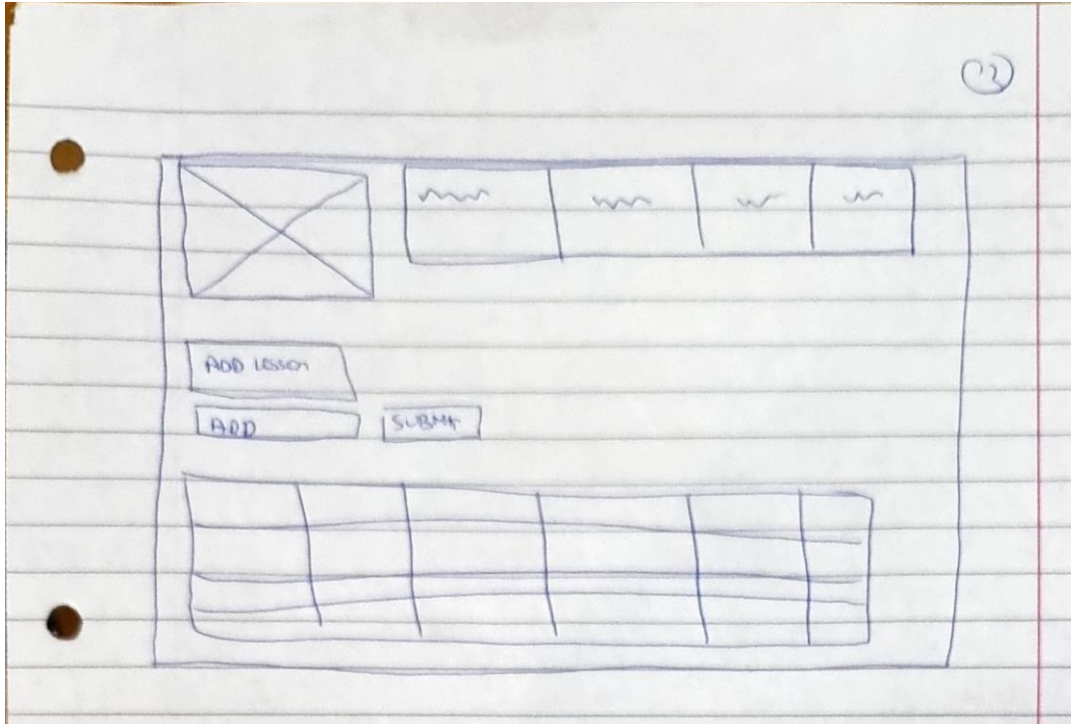


Figure 6: Sketch number 3

In the sketch above, the third way with which the home page might look like. As in the other sketches, there is a navigation bar on the top along with a logo. Below that there are some boxes with which the user can add a module. At the bottom of the page, there is a table where the modules will be presented along with important information about each module.

## 5.0 Code Implementation

When the developer has finished with all of the designs and the diagrams that will help him with the development, the code needs to be written. There were some basic files already developed for the application.

The developer has used a local repository through GitHub to achieve developing this application. Every time the developer had been changing anything in the code, he would “push” those changes in the repository, along with an explanation message telling what he changed.

The latter will start writing the code and when the code is done he will do some tests, to ensure that there are no errors and then deploy the page.



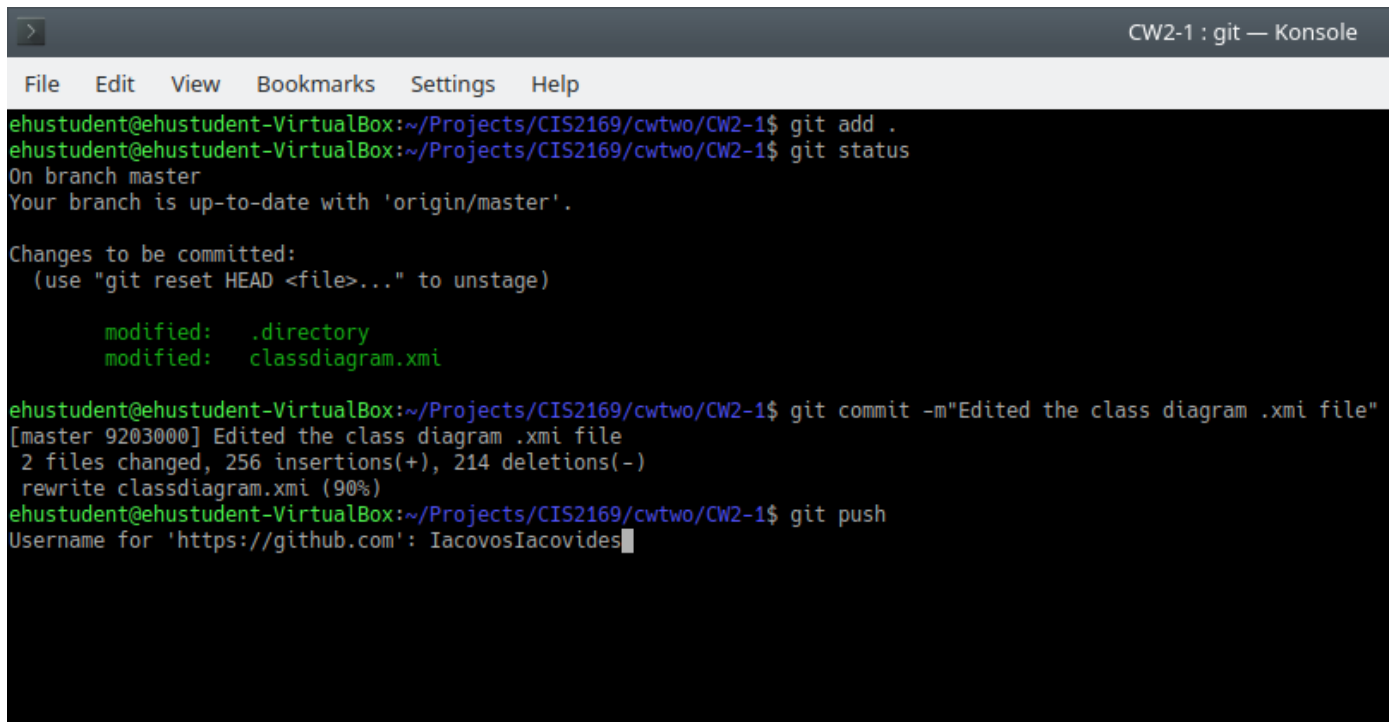
A screenshot of a terminal window titled "CW2-1 : git — Konsole". The terminal shows a series of Git commands and their outputs. The user is in a directory ~/Projects/CIS2169/cwtwo/CW2-1. The commands executed are: 'git add .' (staging files), 'git status' (showing staged files: .directory and classdiagram.xmi), 'git commit -m"Edited the class diagram .xmi file"' (creating a commit), and 'git push' (pushing to GitHub). The output shows the commit hash [master 9203000] and the commit message. The terminal also shows the progress of pushing the commit to GitHub, with a progress bar for 'rewrite classdiagram.xmi (90%)'. The terminal prompt is 'ehustudent@ehustudent-VirtualBox:~/Projects/CIS2169/cwtwo/CW2-1\$'.

Figure 7: Pushing in command line console.

- ➔ The first line of code is ***"git add ."***. This code **adds new or changed files in your working directory to the Git staging area**.
- ➔ The second line of code is ***"git status"***. With this code, the current state of the working directory and staging area is shown. It shows you which changes have been staged, which haven't, and which files Git isn't tracking.
- ➔ The third line of code is ***"git commit -m"Edited the class diagram .xmi file"***. With this code, the developer is adding an explanatory message next to the commit which will show on GitHub.

➔ The final line of code is **“git push”** which pushes the commitments to the repository in GitHub.

The screenshot shows a GitHub repository interface for a forked repository from 'profharmohanpandey/CW2'. The repository is named 'IacovosIacovides' and is on the 'master' branch. It is 18 commits ahead of the upstream repository. The commit history shows a recent commit by 'IacovosIacovides' with the message 'Added another two rows with data in the table in degree page' 10 minutes ago. Below the commit history is a list of files in the repository, including .directory, iacovides\_I\_24128341\_CW2 (1).docx, README.md, activitydiagram.xmi, assessment.html, assessment.jpg, books.jpg, classdiagram.xmi, degree.html, degree.jpg, general.jpg, ggg.xmi, and home.html. The right sidebar shows repository statistics: 0 stars, 0 watching, 13 forks, and a language distribution chart for JavaScript (48.1%), HTML (47.1%), and CSS (4.8%).

Figure 8: GitHub commits.

As you can see there have been 51 commits to this application which means that the developers who worked for this application have committed their changes 51 times.

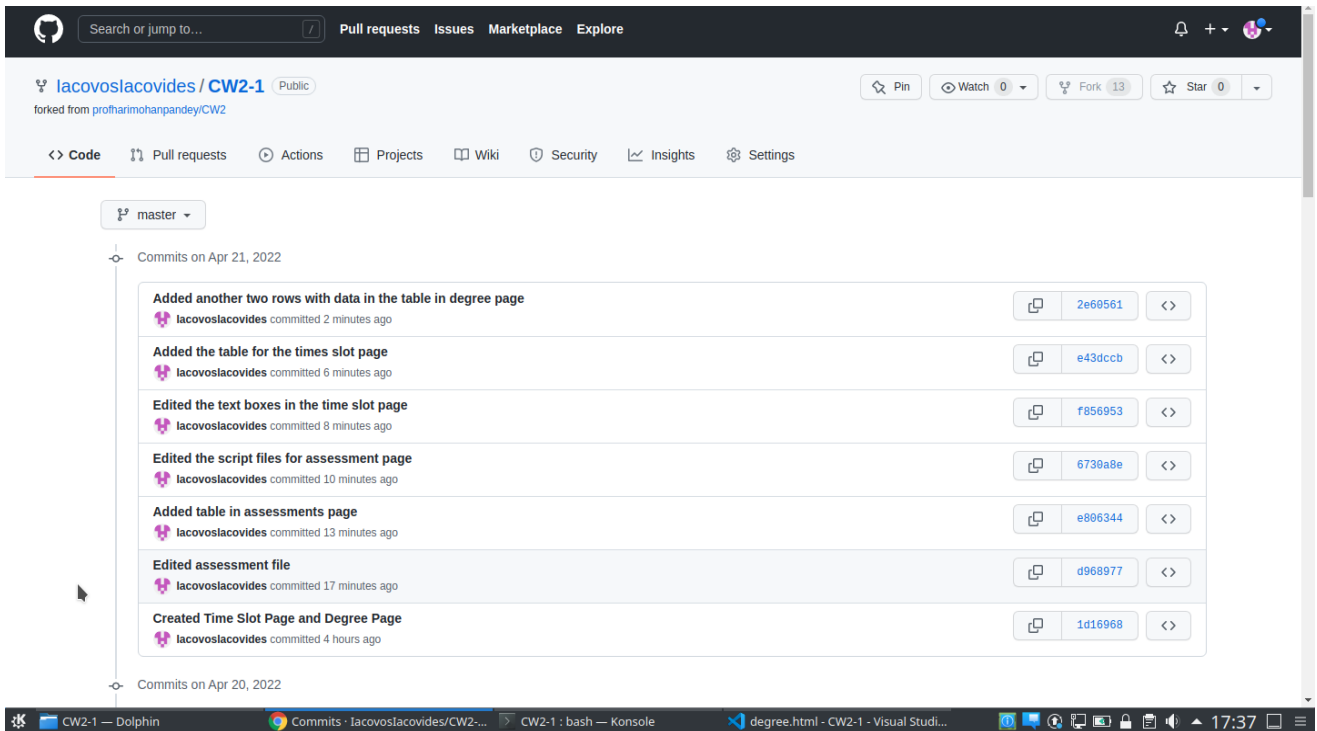


Figure 9:GitHub commits.

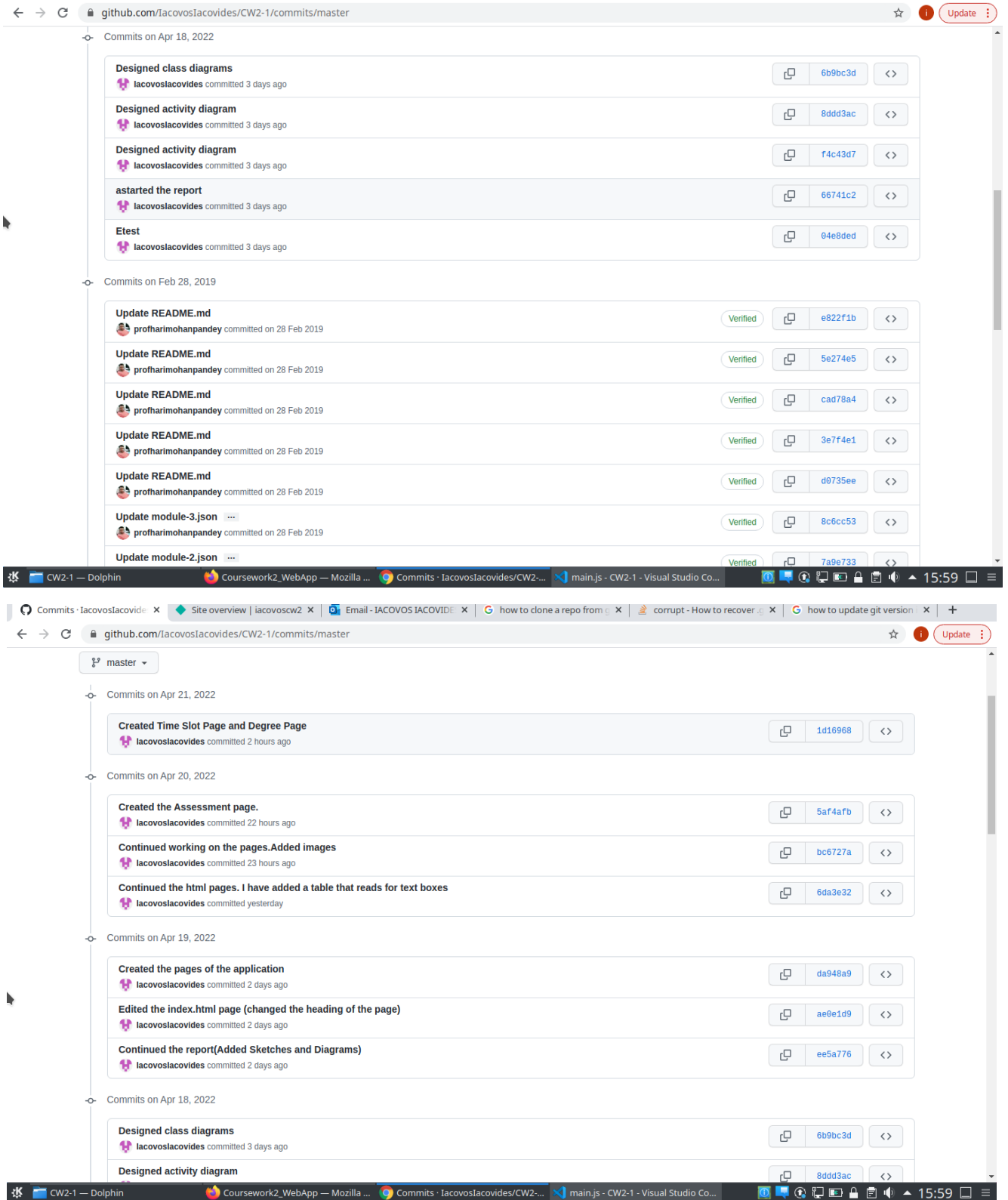


Figure 10: GitHub commits.

On the annotations above, the commits that have been pushed to the Git are provided. The oldest commits have been made by another developer three years ago until a new developer has been asked to enhance the application this year. Each commit has its identification code where someone can see the details of the commit. The details include the code that has been changed, the documents that have been changed, and many more. The repository's URL that the developer has been using is <https://github.com/IacovosIacovides/CW2-1.git>.

The application is stored in a folder on the developer's personal computer.

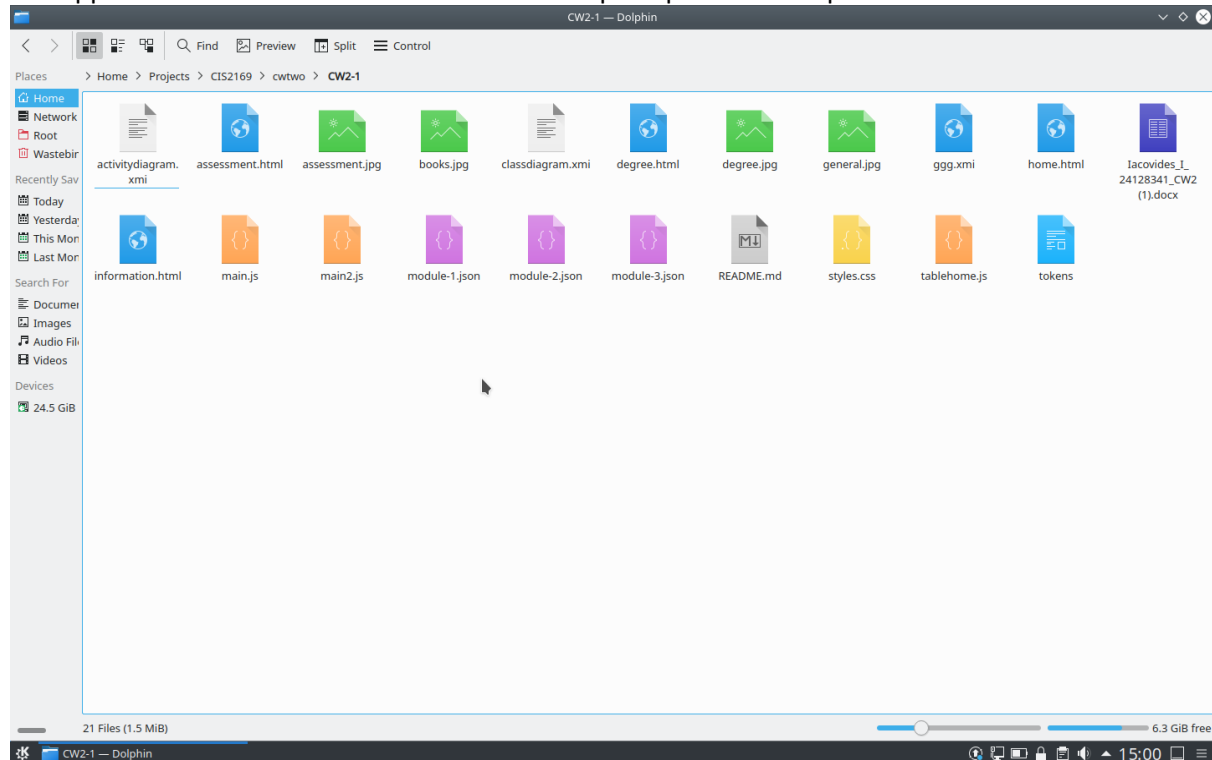


Figure 11: Developers folder in Personal Computer.

The code for the application is stored in the files you can see on the screen capture above. The files with the blue icon are the ones that store the coding which gives the website its looks.

The coding itself has comments above each different section of it. This makes it simpler and tidier for someone to edit if needed. When the coding is nested the correct way and there are useful comments, then the experience is much easier.

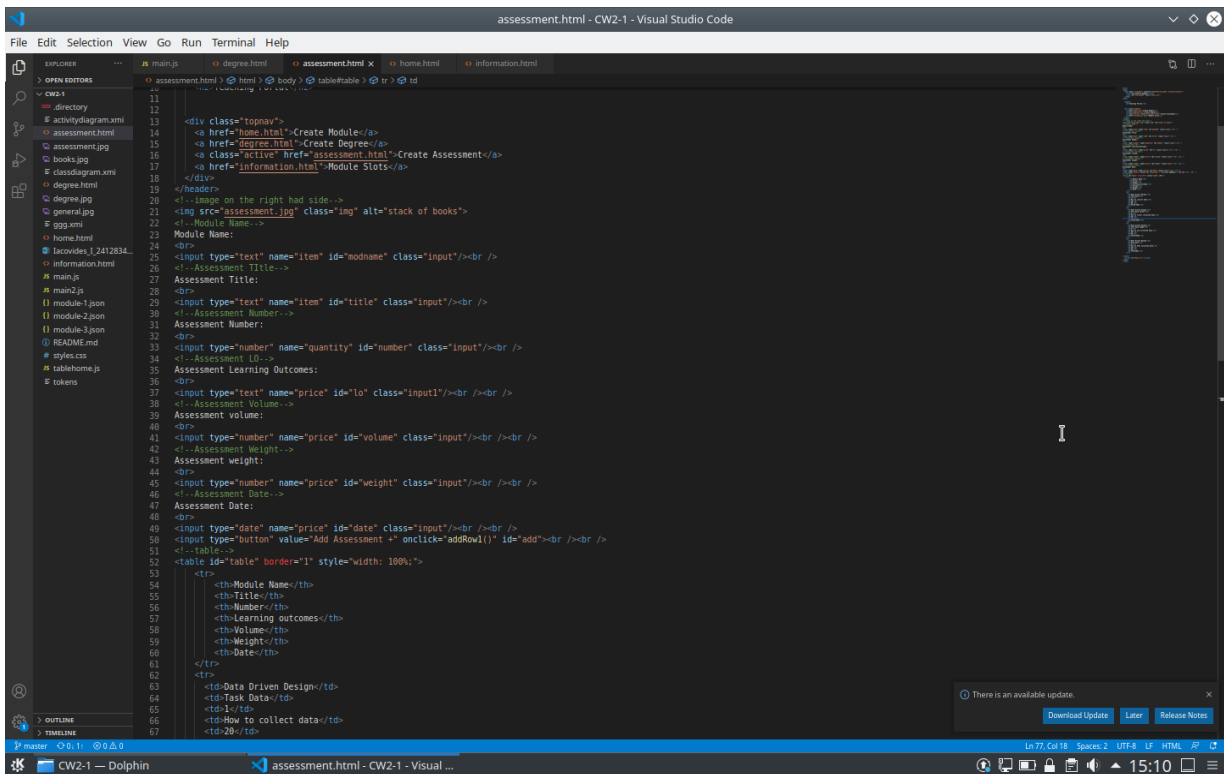


Figure 12: Create Module coding.

The coding above is about the home page which is called “Create Module” when the application is being used. The help of comments makes the understanding of the coding much easier since it explains what is going on.

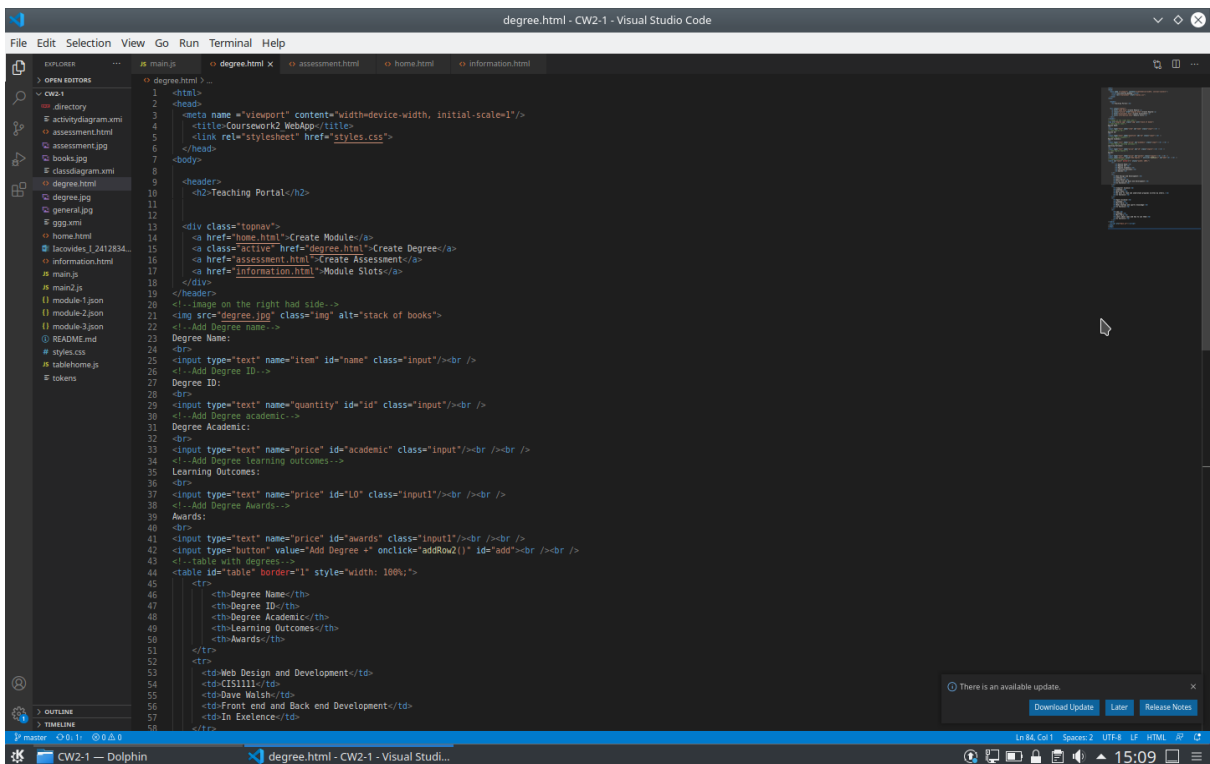


Figure 13: Create degree coding.

The coding above is about the degree addition page which is called “Create Degree” when the application is being used. The help of comments makes the understanding of the coding much easier since it explains what is going on.

```

11
12
13 <div class="topnav">
14 <a href="home.html">Create Module</a>
15 <a href="degree.html">Create Degree</a>
16 <a class="active" href="assessment.html">Create Assessment</a>
17 <a href="information.html">Module Slots</a>
18 </div>
19 </header>
20 <!-- Page on the right had side -->
21 
22 <!-- Module Name -->
23 Module Name:
24 <input type="text" name="item" id="modname" class="input"/><br />
25 <!-- Assessment Title -->
26 Assessment Title:
27 <input type="text" name="item" id="title" class="input"/><br />
28 <!-- Assessment Number -->
29 Assessment Number:
30 <input type="number" name="quantity" id="number" class="input"/><br />
31 <!-- Assessment LO -->
32 Assessment Learning Outcomes:
33 <input type="text" name="price" id="lo" class="input"/><br /><br />
34 <!-- Assessment Volume -->
35 Assessment volume:
36 <input type="number" name="price" id="volume" class="input"/><br /><br />
37 <!-- Assessment Weight -->
38 Assessment weight:
39 <input type="number" name="price" id="weight" class="input"/><br /><br />
40 <!-- Assessment Date -->
41 Assessment Date:
42 <input type="date" name="price" id="date" class="input"/><br /><br />
43 <input type="button" value="Add Assessment" onClick="addRow()" id="add"><br /><br />
44 <!-- table -->
45 <table id="table" border="1" style="width: 100%;>
46 <thead>
47 <tr>
48 <th>Module Name</th>
49 <th>Title</th>
50 <th>Number</th>
51 <th>Learning outcomes</th>
52 <th>Volume</th>
53 <th>Weight</th>
54 <th>Date</th>
55 </tr>
56 </thead>
57 <tbody>
58 <tr>
59 <td>Data Driven Design</td>
60 <td>Task Data</td>
61 <td>How to collect data</td>
62 <td>20</td>
63 </tr>
64 </tbody>
65 </table>
66
67

```

Figure 14: Create Assessment coding.

The coding above is about the assessment addition page which is called “Create Assessment” when the application is being used. The help of comments makes the understanding of the coding much easier since it explains what is going on.

```

1 <html>
2 <head>
3 <meta name="viewport" content="width=device-width, initial-scale=1"/>
4 <title>Coursework 2 Webpage</title>
5 <link rel="stylesheet" href="styles.css">
6 </head>
7 <body>
8 <header>
9 <h2>Teaching Portal</h2>
10
11 <div class="topnav">
12 <a href="home.html">Create Module</a>
13 <a href="degree.html">Create Degree</a>
14 <a href="assessment.html">Create Assessment</a>
15 <a class="active" href="information.html">Module Slots</a>
16 </div>
17 </header>
18 <!-- Page on the right had side -->
19 
20 <!-- Add Module Name -->
21 Module Name:
22 <input type="text" name="item" id="name" class="input"/><br />
23 <!-- Add Module ID -->
24 Module ID:
25 <input type="text" name="quantity" id="id" class="input"/><br />
26 <!-- Add Module start time -->
27 Start Time:
28 <input type="time" name="price" id="start" class="input"/><br />
29 <!-- Add Module end time -->
30 End Time:
31 <input type="time" name="price" id="finish" class="input"/><br />
32 <input type="button" value="Add Slots" onClick="addRow()" id="add"><br />
33 <!-- table with modules -->
34 <table id="table" border="1" style="width: 100%;>
35 <thead>
36 <tr>
37 <th>Module Name</th>
38 <th>Module ID</th>
39 <th>Time Starting</th>
40 <th>Time Finish</th>
41 </tr>
42 </thead>
43 <tbody>
44 <tr>
45 <td>Fundamentals of UX</td>
46 <td>CIS2168</td>
47 <td>12:08</td>
48 <td>14:00</td>
49 </tr>
50 <tr>
51 <td>Fundamentals of Web Coding</td>
52 <td>CIS2152</td>
53 <td>16:00</td>
54 <td></td>
55 </tr>
56 </tbody>
57 </table>
58
59

```

Figure 15: Module Slots coding.



The coding above is about the slots page which is called “Module Slots” when the application is being used. The help of comments makes the understanding of the coding much easier since it explains what is going on.

On all of the screen captures above there is a line of code that says:

➔ `<input type="..." name="..." id="..." class="..." /><br></br>`

This line of code is found throughout the application multiple times. What this does, it adds an input text box on the page where the user can add their data and then return it to the table below the input boxes.

There is also another line of code that is found in every code file that says:

➔ `<input type="button" value="Add ... +" onclick="AddRowX()" id="add" /><br></br>`

This line of code is found in every coding file. This line is the one that produces a button below the input text boxes, where when the button is clicked, then a function in a JavaScript file adds that data in a new row in the table.

The main JavaScript file that I have used, holds all important functions that aid in adding the data from the input text boxes into the tables of each page. What this code does is that it takes the information from the text boxes, matches the data into the appropriate “id” and then returns the data to the table as a new row at the bottom of the table.

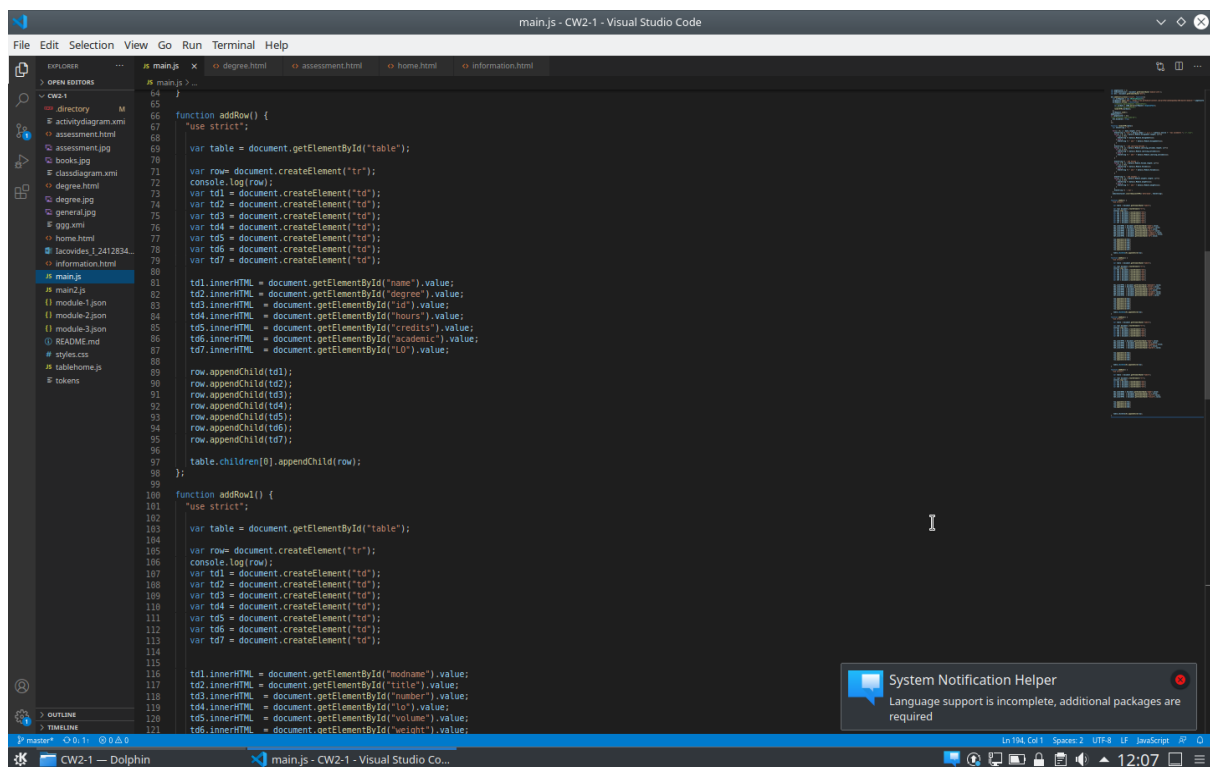


Figure 16: Main JavaScript file coding.

The coding that the developer has done for the tables of each page is a simple table code. The code that the developer uses is:

`<table id="....." border="1" style="width: 100%;">`

The “<table>” part in the code is the one that is responsible for creating the table. The way that the program knows how to end the table is by using the same piece of code but with a “/” before the word **table**. Thus, it is written like this: “</table>”. Inside these two pieces of code, all the columns and rows go. The developer assigned a “<tr>” bracket which is the new row and then for the row to end the developer used “</tr>”. Then with “<th>.....</th>” the developer assigned the table header element which are the cells with the bold text inside which are the table headers. For the rest of the rows the developer used “<td>.....</td>” for the ordinary text in the cells. Below is an example of table coding on the page “Create Degree”.

```
→ <tr>
    <th>Degree Name</th>
    <th>Degree ID</th>
    <th>Degree Academic</th>
    <th>Learning Outcomes</th>
    <th>Awards</th>
</tr>
<tr>
    <td>Web Design and Development</td>
    <td>CIS1111</td>
    <td>Dave Walsh</td>
    <td>Front end and Back end Development</td>
    <td>In Excellence</td>
</tr>
```

## 6.0 Deployment planning and practice.

When the coding has been finished, the developer will start testing whether everything is working the way it should, accurately and fast. The developer will connect his GitHub account to the system called “Netlify”, where he can deploy the website and test it. The link to deploy the application is <https://iacovoscw2.netlify.app/>.

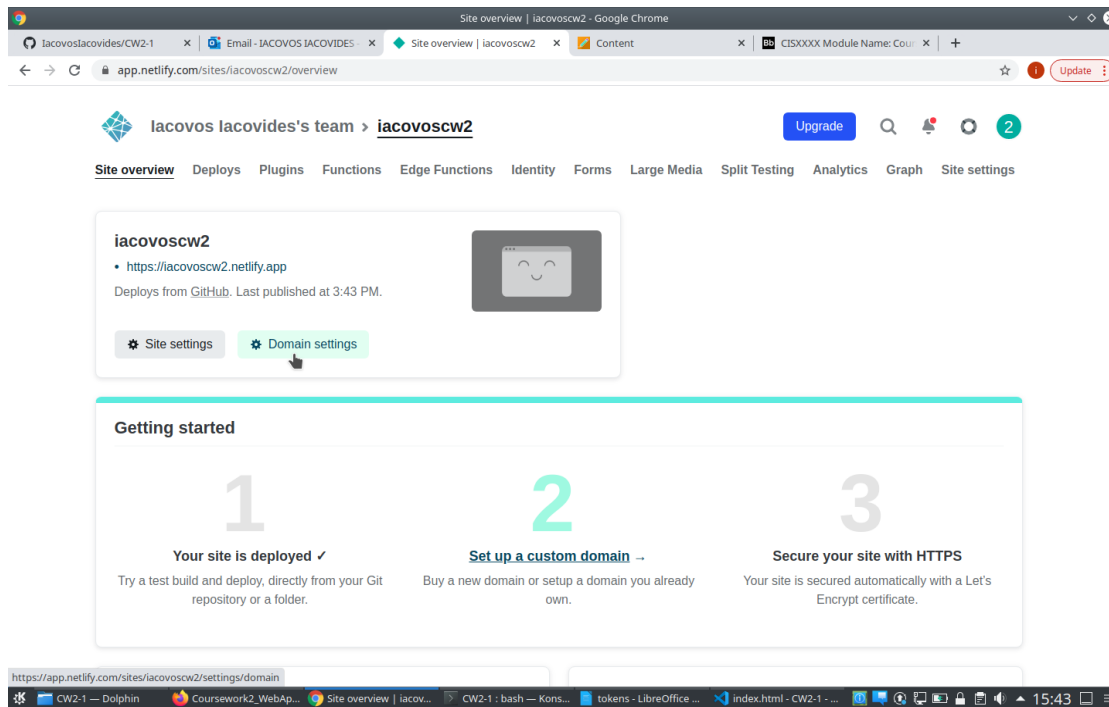


Figure 17: Netlify URL.

On the screen capture above, the home page of **Netlify** is provided. As you can see at the top of the page the developer’s name is given along with the name of the application. For the developer to deploy the application, he needs to click on the URL which is given in the first small box of the page. When that URL is clicked it will run and open the application.

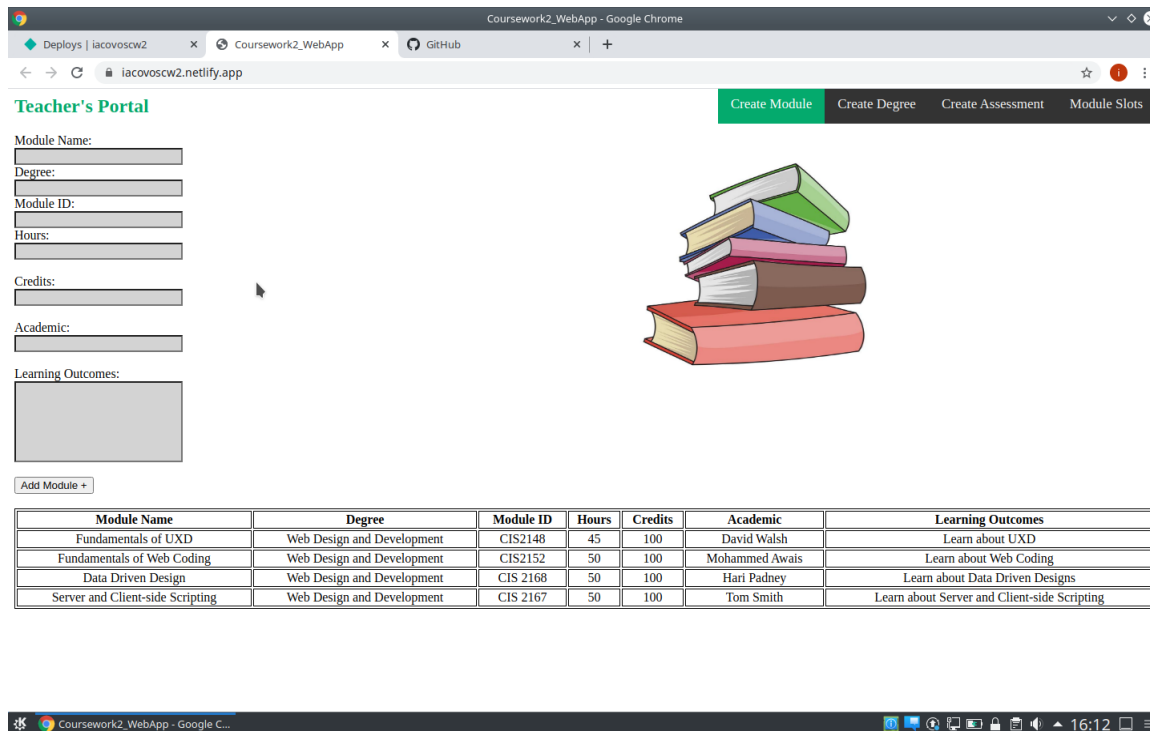
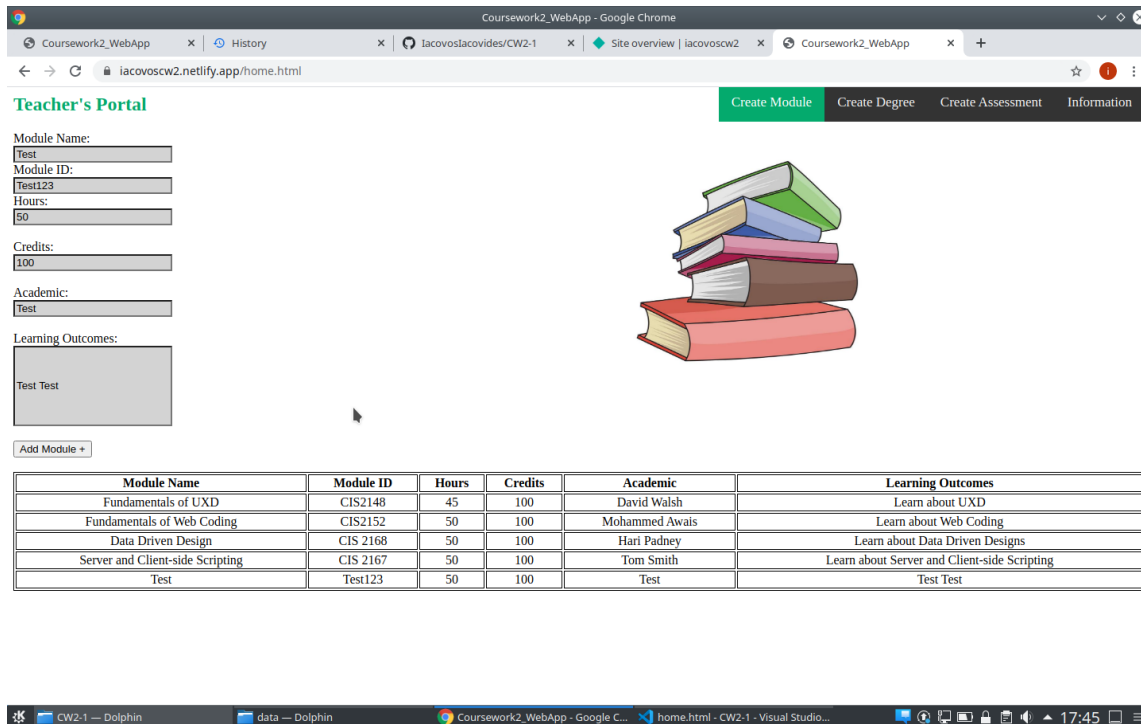


Figure 18: Deployed Application.

On the screenshot above, the deployed application is portrayed. As you can see there is a navigation bar with all of the major features that the application offers. Additionally, you can see that on the

left-hand side of the page there are some text boxes where the user can input data and then return them in the table below.

The developer also tested where all the code works the way it should. Testing can eliminate the chances of getting errors on the final product (testing, 2022). On the screenshots below, various tests will be given, concerning the table and whether the data from the input text boxes are returned the correct way.



The screenshot shows a web browser window with the URL `iacovoscw2.netlify.app/home.html`. The page title is "Teacher's Portal". On the left, there are input fields for "Module Name" (Test), "Module ID" (Test123), "Hours" (50), "Credits" (100), "Academic" (Test), and "Learning Outcomes" (Test Test). To the right is an illustration of a stack of books. Below the inputs is an "Add Module +" button. A table below the button displays the data entered:

Module Name	Module ID	Hours	Credits	Academic	Learning Outcomes
Fundamentals of UXD	CIS2148	45	100	David Walsh	Learn about UXD
Fundamentals of Web Coding	CIS2152	50	100	Mohammed Awais	Learn about Web Coding
Data Driven Design	CIS 2168	50	100	Hari Padney	Learn about Data Driven Designs
Server and Client-side Scripting	CIS 2167	50	100	Tom Smith	Learn about Server and Client-side Scripting
Test	Test123	50	100	Test	Test Test

Figure 19: Test 1.

As you can see on the screen capture above, the developer had inputted some data in the text boxes on the left hand, and when he clicked on the "Add Module +" button, the data were returned as a new row in the table below.

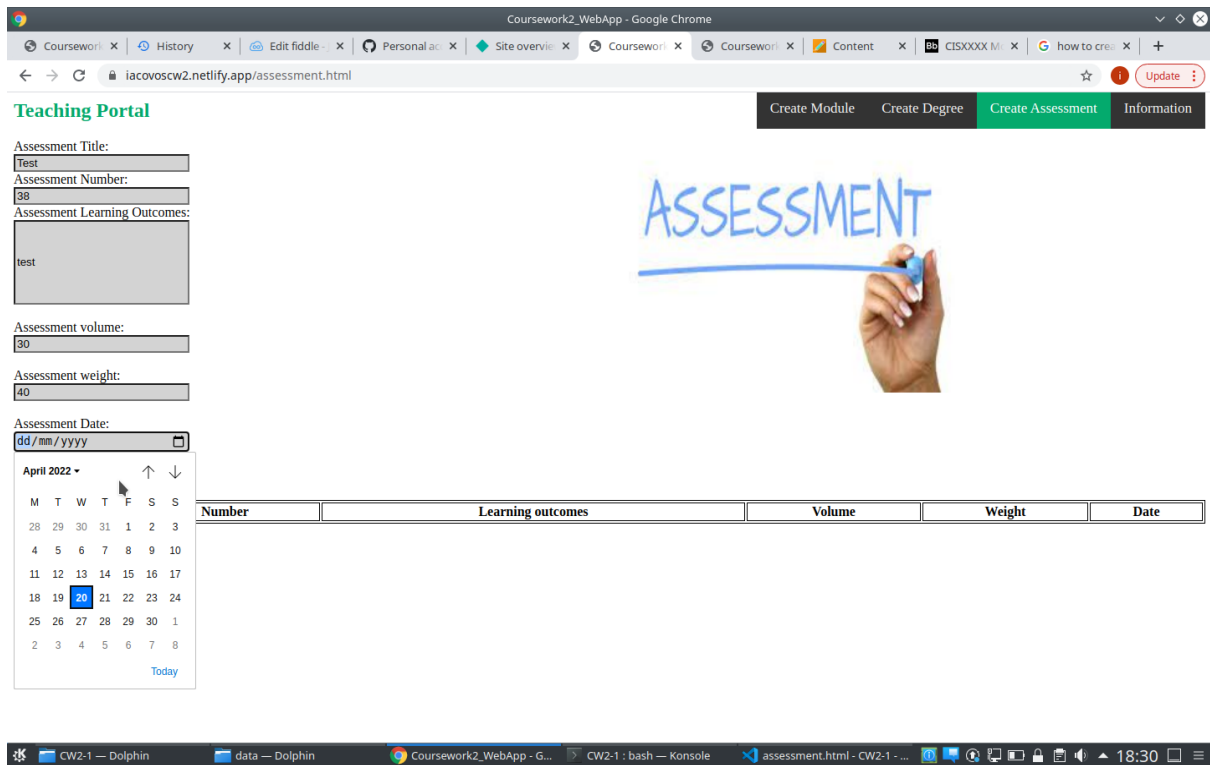


Figure 20: Test 2.

On the screenshot above the developer wanted to check whether the “date” feature was working the correct way and it did.

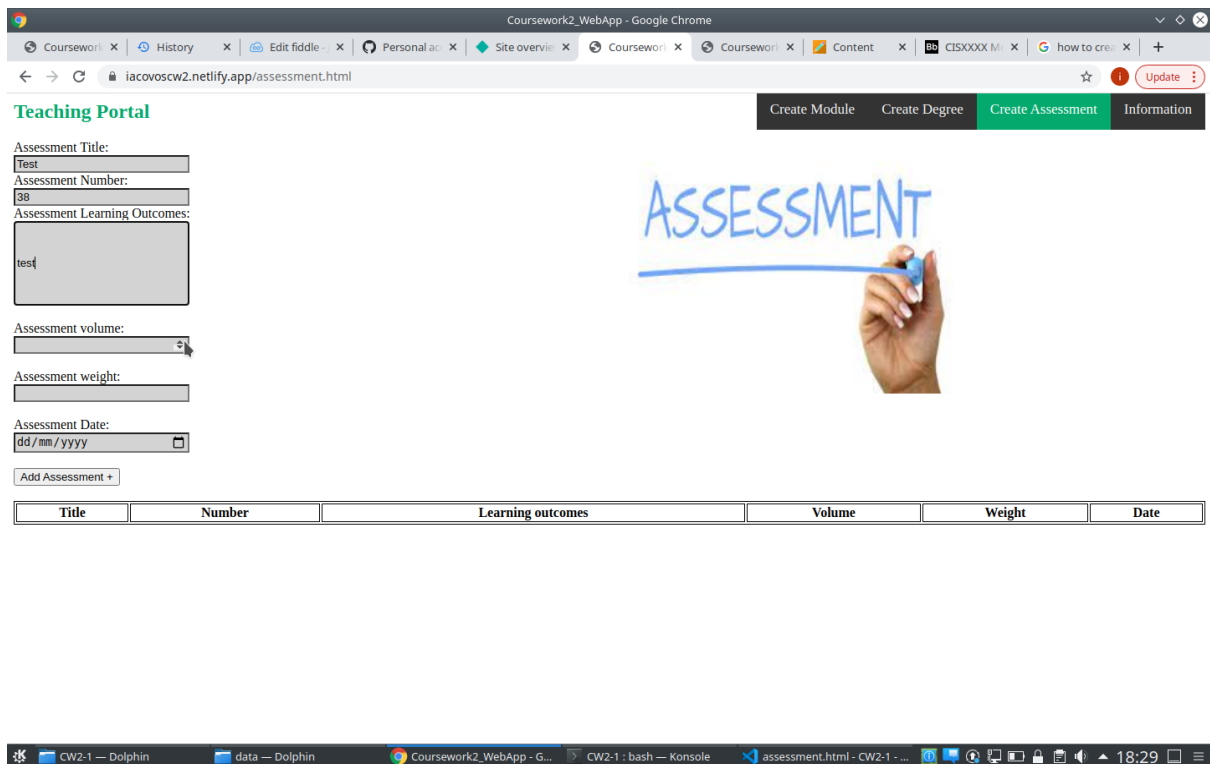


Figure 21: Test 3.

On the screenshot above the developer wanted to check whether the “number” feature was working the correct way and it did.

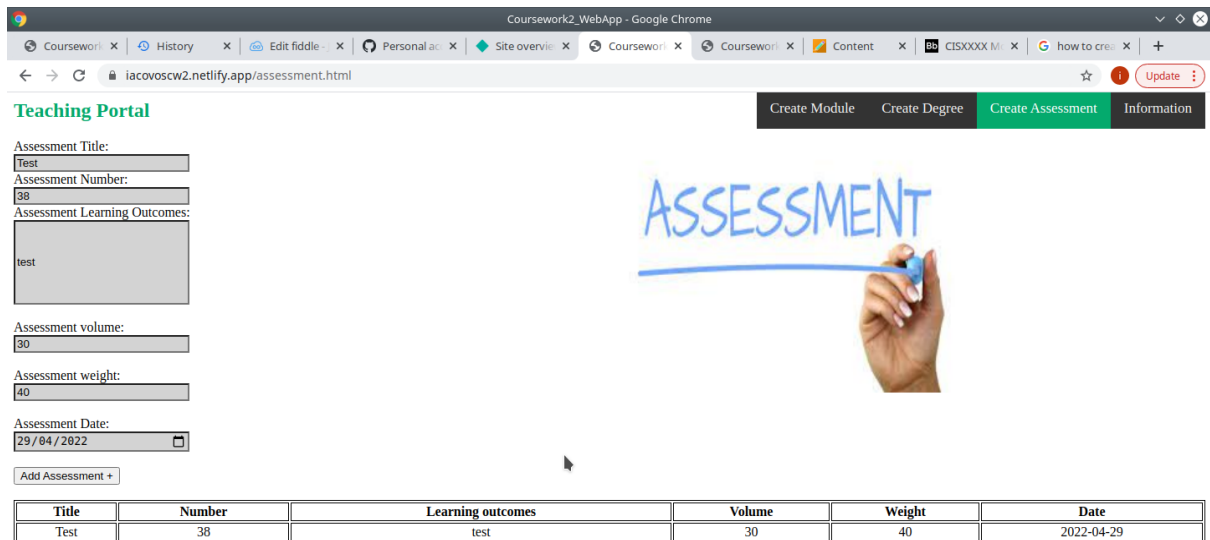


Figure 22: Test 4.

As you can see on the screen capture above, the developer had inputted some data in the text boxes on the left-hand and when he clicked on the "Add Assessment +" button, the data were returned as a new row at the bottom of the table.

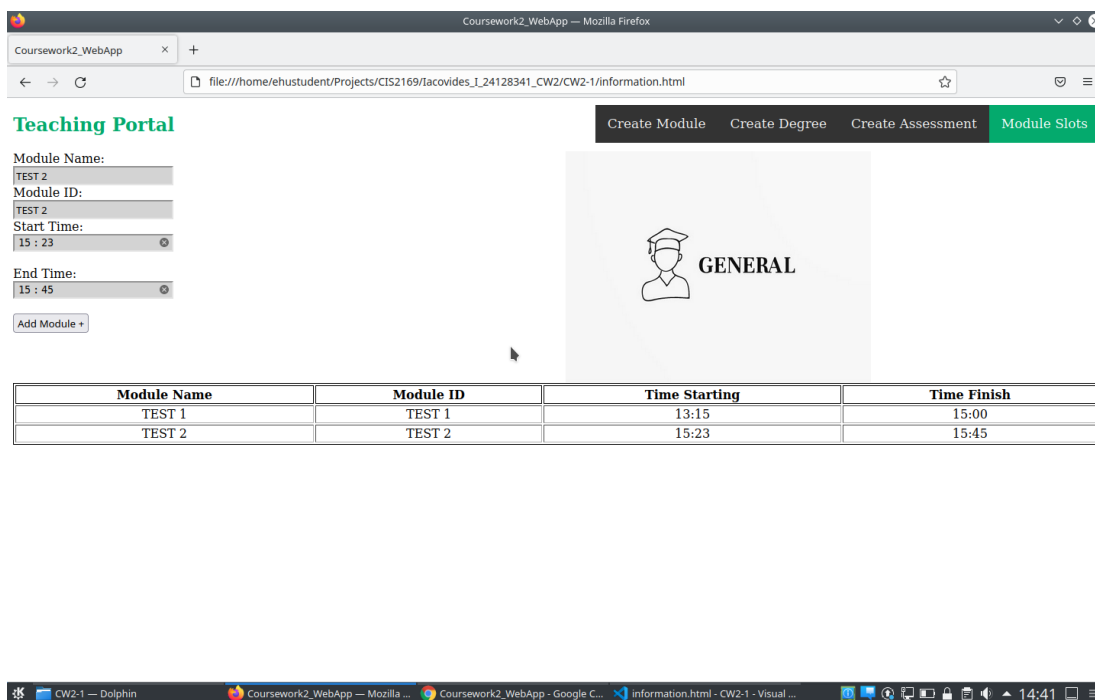


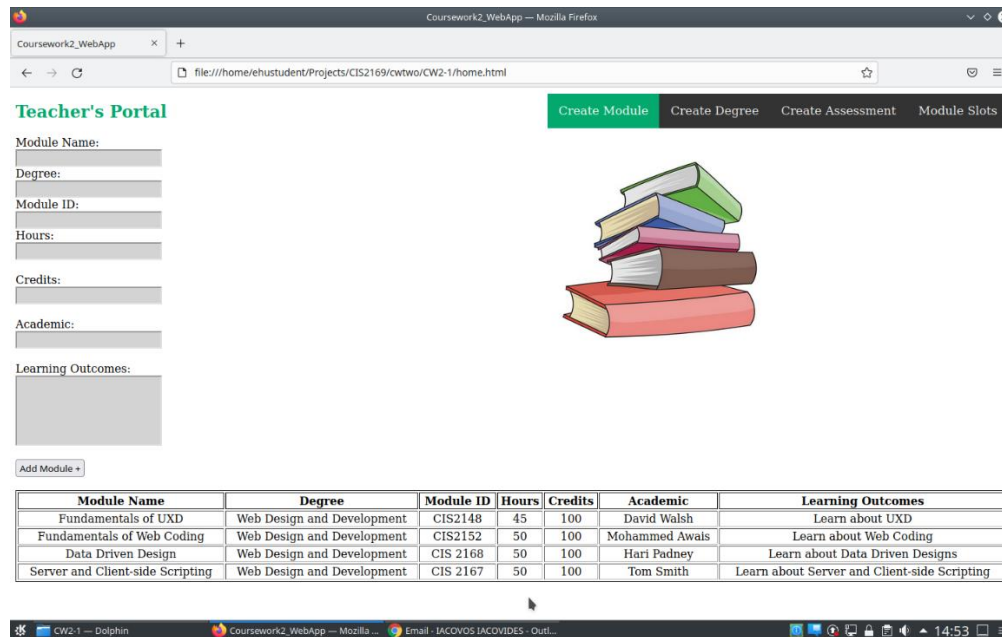
Figure 23: Test 5.

On the screenshot above, the developer wanted to check whether the time features were working the way they should. He did two tests and both of them work the way they should.

As mentioned above, the URL link for viewing and using the application is

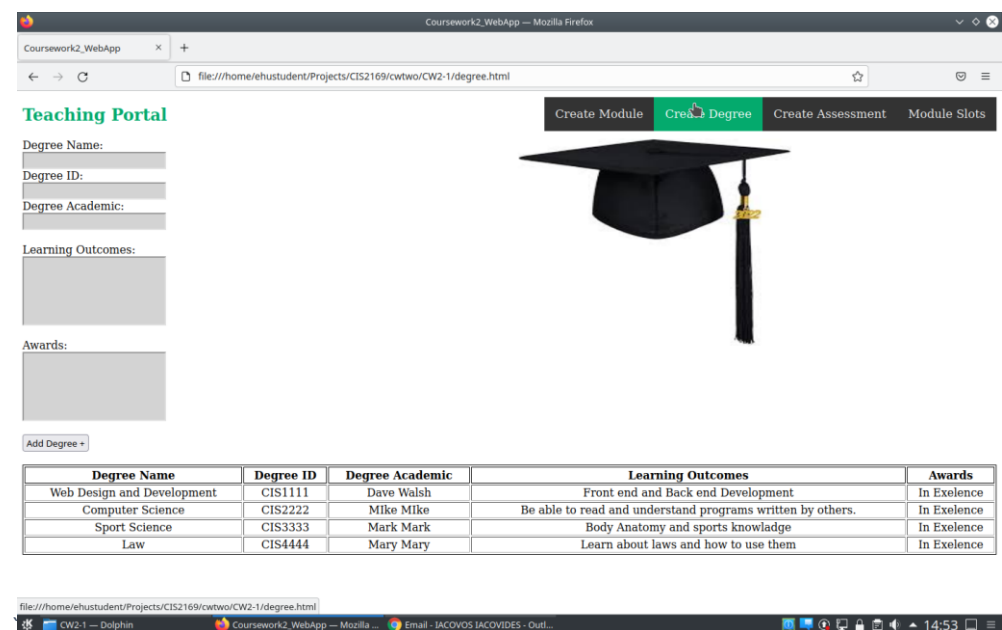
<https://iacovoscw2.netlify.app/>.

## 7.0 Final Application (final output when opening the application)



Module Name	Degree	Module ID	Hours	Credits	Academic	Learning Outcomes
Fundamentals of UXD	Web Design and Development	CIS2148	45	100	David Walsh	Learn about UXD
Fundamentals of Web Coding	Web Design and Development	CIS2152	50	100	Mohammed Awais	Learn about Web Coding
Data Driven Design	Web Design and Development	CIS 2168	50	100	Hari Padney	Learn about Data Driven Designs
Server and Client-side Scripting	Web Design and Development	CIS 2167	50	100	Tom Smith	Learn about Server and Client-side Scripting

Figure 24: Final Create Module Page.



Degree Name	Degree ID	Degree Academic	Learning Outcomes	Awards
Web Design and Development	CIS1111	Dave Walsh	Front end and Back end Development	In Excellence
Computer Science	CIS2222	Mike Mike	Be able to read and understand programs written by others.	In Excellence
Sport Science	CIS3333	Mark Mark	Body Anatomy and sports knowledge	In Excellence
Law	CIS4444	Mary Mary	Learn about laws and how to use them	In Excellence

Figure 25: Final Create Degree Page.



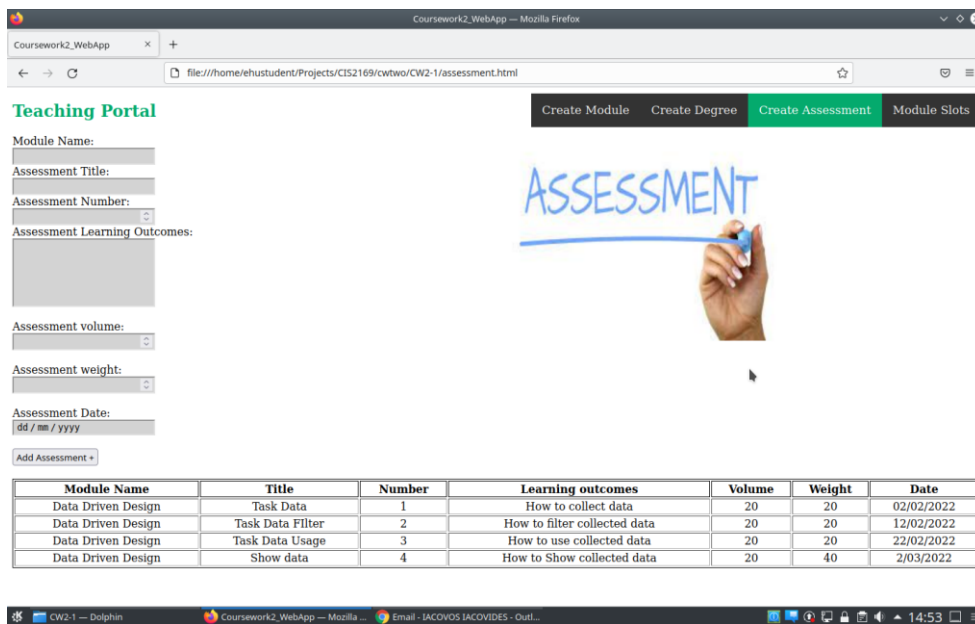


Figure 26: Final Create Assessment Page.

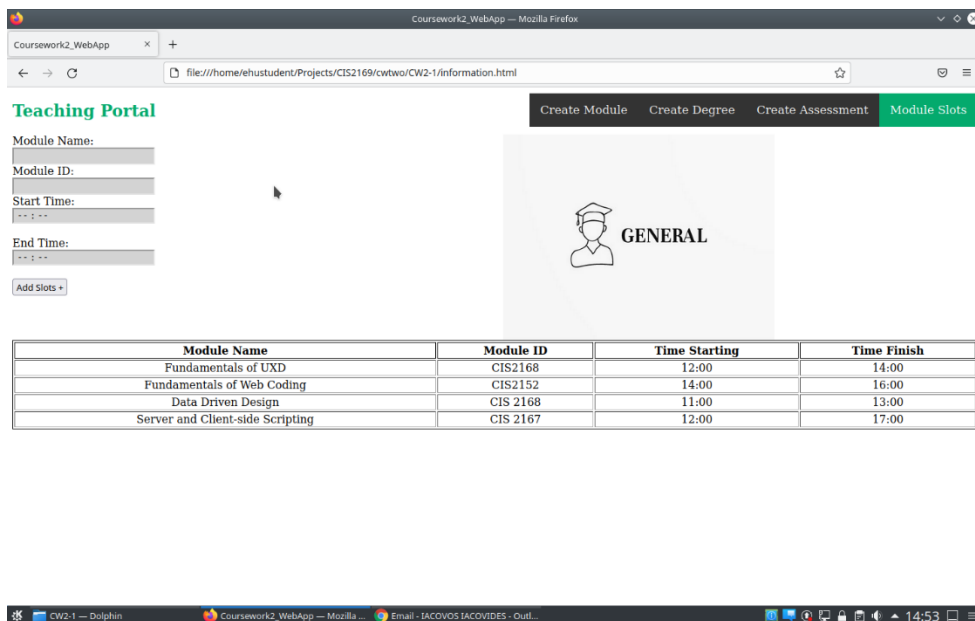


Figure 27: Final Module Slots Page.

## 8.0 Analysis Post-mortem

In this section what went well during the development of the application and what went wrong will be specified. Moreover, if there was anything that the developer would do differently next time will be specified as well.

At the beginning of the code designing and implementation, the developer was very confident about what he was doing. The application was coming out the way it should. There was one major aspect that was hard for him though. When he started reading the case study, he could not understand what diagrams he should create. However, after some more reading and researching, he finally understood what to do. After designing the diagrams, the coding was easy to do except for some areas where the use of JavaScript files made it a bit confusing. With some appropriate research, he managed to finish the code resulting in a properly and accurately working system.

When the code was almost done, the repository found a corrupt file which made it impossible for the developer to push the changes into GitHub. This meant that, even though he further developed the code, he could not commit to any changed because of that file. After talking to the tutors, he then created a clone of his local repository into another folder which in the end fixed the error. This obstacle though wasted a lot of time for the developer. In the end, he managed to fix it and commit to changes again and end up pushing into GitHub and automatically into Netlify for deploying the website. Next time the developer will be more careful of what he is doing to assist in eliminating the chances of such a corrupt file emerging again and damaging the repository.

For the next time, the developer would surely do more research, since better research means that the designing of diagrams, sketches, and the code will be much more accurate and easy to do.

## 9.0 Conclusion

After taking into consideration everything that has been detailed above, along with the screenshots and explanations, the developer has done honest work when it came to developing this project. He has done some research that helped him to design diagrams that were needed and also, he managed to implement documents of code that led to a full, accurate working application that has no errors. As mentioned at the beginning of this report the developer should enhance an already existing application by improving several things that the application is lacking. The result is a basic fully working application that helps instructors and tutors add modules, assessments, degrees, and time slots to the modules.

## 10.0 References

Bluescape, 2019. *BlueScape*. [Online]

Available at: <https://www.bluescape.com/>

[Accessed 22 April 2022].

Contributor, C., 2020. *SmallBussines.Chron*. [Online]

Available at: <https://smallbusiness.chron.com/>

[Accessed 22 April 2022].

testing, S., 2022. *ToolsQA*. [Online]

Available at: <https://www.toolsqa.com/>

[Accessed 22 April 2022].