Iacovos Iacovides

24128341

Web Application Development

CIS2169

Contents

[1.0 Introduction 2](#_Toc101609275)

[2.0 Diagrams 2](#_Toc101609276)

[3.0 Sketches 4](#_Toc101609277)

[5.0 Code Implementation 6](#_Toc101609278)

[6.0 Deployment planning and practice. 13](#_Toc101609279)

[**7.0** Analysis Post-mortem 18](#_Toc101609280)

[8.0 Conclusion 18](#_Toc101609281)

[9.0 References 19](#_Toc101609282)

Table of Figures

[Figure 1: Activity Diagram 2](#_Toc101609287)

[Figure 2: Class Diagram 3](#_Toc101609288)

[Figure 3: Use case diagram 4](#_Toc101609289)

[Figure 4: Sketch number 1 5](#_Toc101609290)

[Figure 5: Sketch number 2 5](#_Toc101609291)

[Figure 6: Sketch number 3 6](#_Toc101609292)

[Figure 7: Pushing in the command-line console. 7](https://d.docs.live.net/2d630ada98544848/Documents/lessons/Year%202/Semester%202/New%20folder/hary/Iacovides_I_24128341_CW2.docx#_Toc101609293)

[Figure 8: GitHub commits. 8](https://d.docs.live.net/2d630ada98544848/Documents/lessons/Year%202/Semester%202/New%20folder/hary/Iacovides_I_24128341_CW2.docx#_Toc101609294)

[Figure 9:GitHub commits. 8](https://d.docs.live.net/2d630ada98544848/Documents/lessons/Year%202/Semester%202/New%20folder/hary/Iacovides_I_24128341_CW2.docx#_Toc101609295)

[Figure 10: GitHub commits. 9](https://d.docs.live.net/2d630ada98544848/Documents/lessons/Year%202/Semester%202/New%20folder/hary/Iacovides_I_24128341_CW2.docx#_Toc101609296)

[Figure 11: Developers folder in Personal Computer. 10](#_Toc101609297)

[Figure 12: Create Module coding. 11](#_Toc101609298)

[Figure 13: Create degree coding. 11](#_Toc101609299)

[Figure 14: Create Assessment coding. 12](#_Toc101609300)

[Figure 15: Module Slots coding. 12](#_Toc101609301)

[Figure 16: Main JavaScript file coding. 13](#_Toc101609302)

[Figure 17: Netlify URL. 14](#_Toc101609303)

[Figure 18: Deployed Application. 14](#_Toc101609304)

[Figure 19: Test 1. 15](#_Toc101609305)

[Figure 20: Test 2. 16](#_Toc101609306)

[Figure 21: Test 3. 16](#_Toc101609307)

[Figure 22: Test 4. 17](#_Toc101609308)

[Figure 23: Test 5. 17](#_Toc101609309)

# 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.

A screenshot of a computer

Description automatically generated

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.

Diagram

Description automatically generated

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.

Diagram

Description automatically generated

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.

A piece of paper with writing on it

Description automatically generated

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.

Text, letter

Description automatically generated

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.

Letter

Description automatically generated with medium confidence

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 computer

Description automatically generated with medium confidence

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 fie”.*** With this code, the developer is adding an explanatory message next to the commit which will show on GitHub.
* A screenshot of a computer

  Description automatically generatedThe final line of code is ***“git push”*** which pushes the commitments to the repository in GitHub.

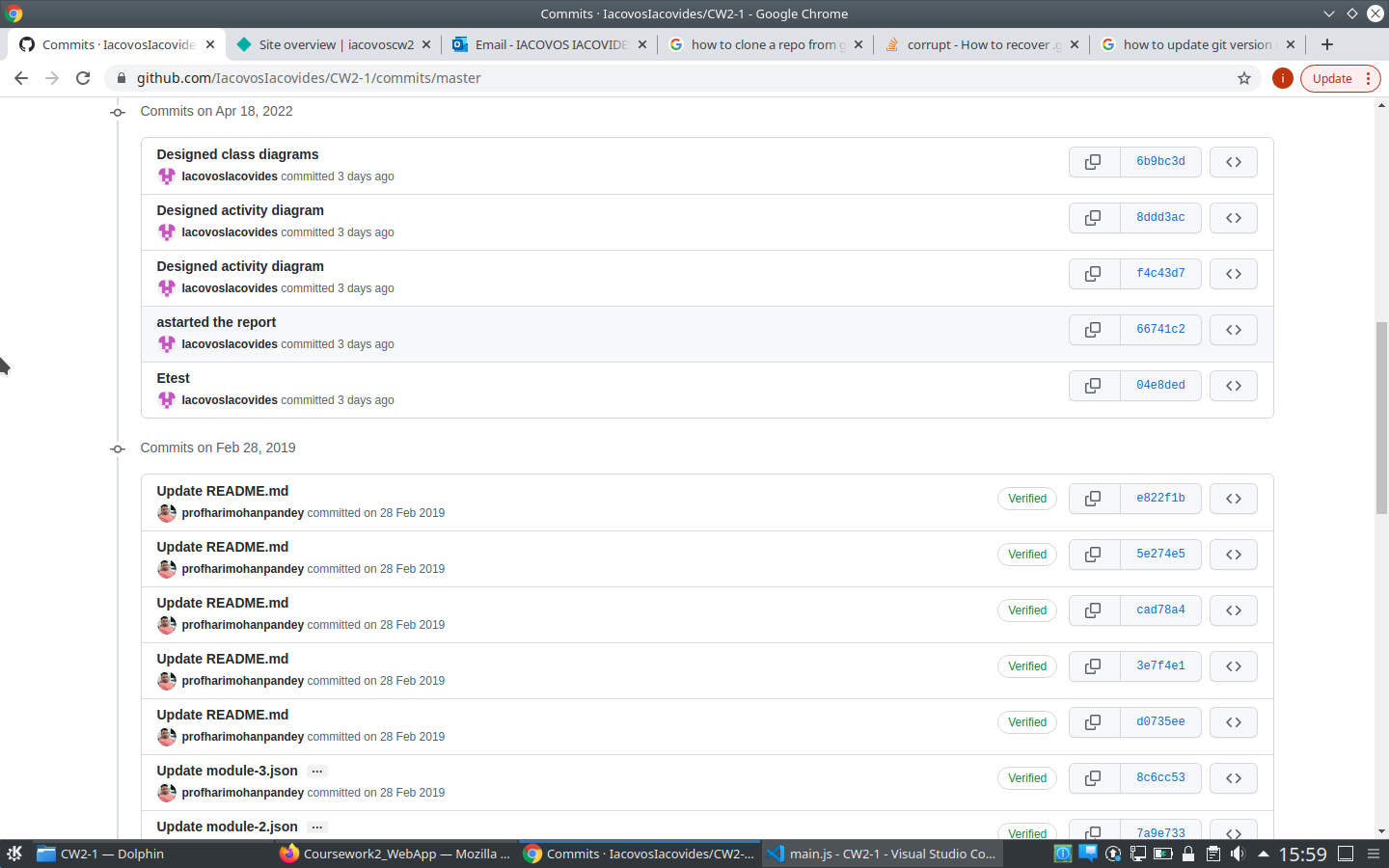
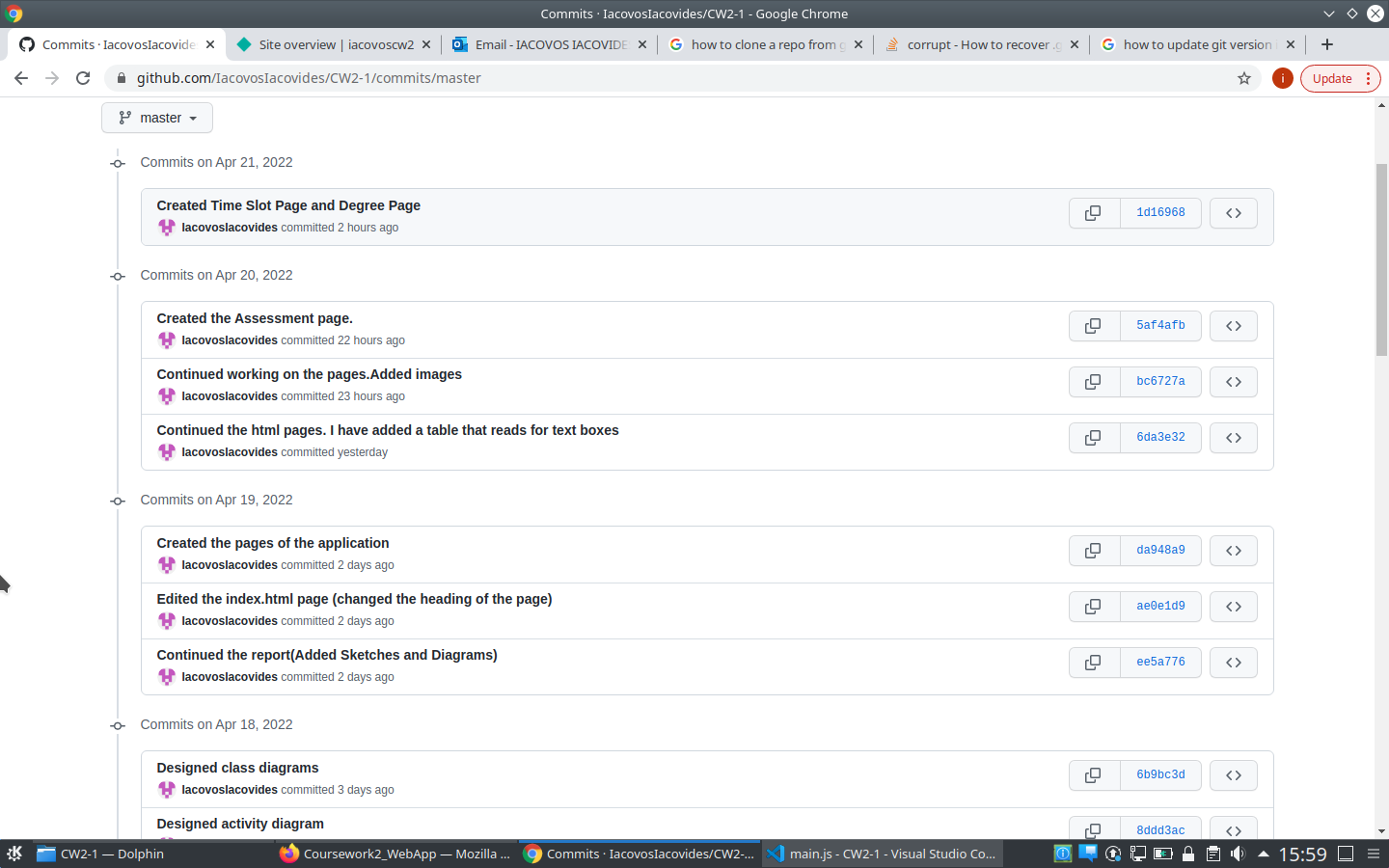
Figure 8: GitHub commits.

A screenshot of a computer

Description automatically generatedAs 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 application is stored in a folder on the developer’s personal computer. Graphical user interface, application

Description automatically generated

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

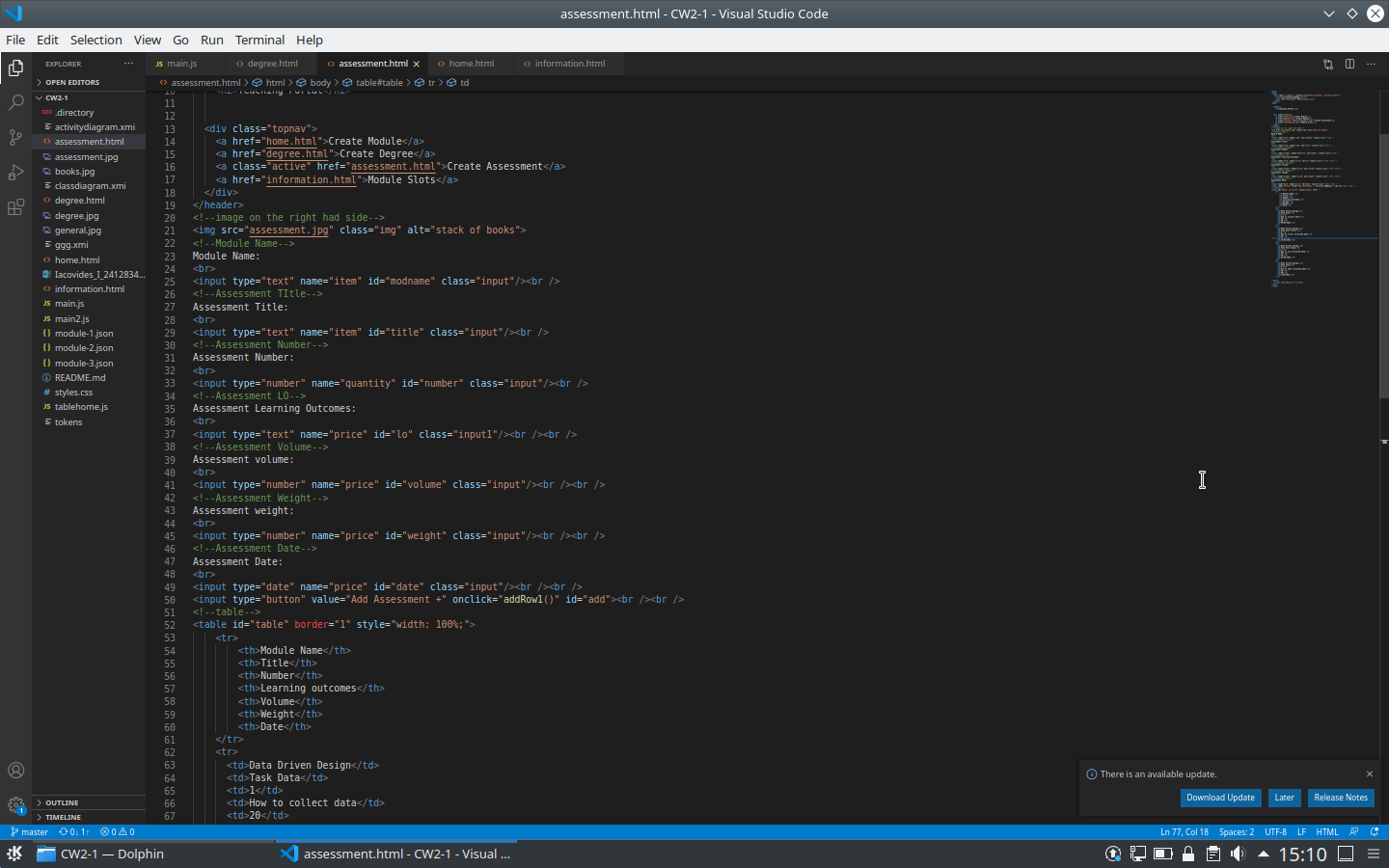


Figure 14: Create Assessment coding.

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.

A computer screen capture

Description automatically generated with medium confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 16: Main JavaScript file coding.

# 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/**](https://l.facebook.com/l.php?u=https%3A%2F%2Fiacovoscw2.netlify.app%2F%3Ffbclid%3DIwAR2aYamFVozKyYh4EIAidklfyaeoTcvcFGwdXdXi44CtGyPAyhzL367K3Jg&h=AT2BsSz9ZkN_l7WQjnl4DvXbH3LfJ8QWfptGBreotAEVzbxOo7sz6RS_Euvi-J6AuMEyrMrrcTIxCU-spDnalEIIa7V6vll31b7xfDC87SGN6V37GE4Yq4Ma-2Zvthz0ZMaIWg)**.**

Graphical user interface, application, email, website

Description automatically generated

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.

Graphical user interface, diagram

Description automatically generated

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.

Graphical user interface

Description automatically generated

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

Table

Description automatically generated

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 [**https://iacovoscw2.netlify.app/**](https://iacovoscw2.netlify.app/)**.**

# **7.0** 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.

# 8.0 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 end result is a basic fully working application that helps instructors and tutors add modules, assessments, degrees, and time slots to the modules.

# 9.0 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].