

Team retrospective

Group 33



June 1, 2018

IFB299

QUT

**Team Member Student Number**

Dylan Vardanega N9966005

Marin Marin N9899138

Janice Vee N9427309

Callum Miller N9922679

Robert Tran N9609466

**TUTOR**

Jesse St. Germain

***Database Creation and Implementation – N9609466 (Robert Tran)***

When Creating and hosting a website, database plays an important role in storing and maintain data from all possible sites, which later helps administrators to review the information of the staffs and customers. Therefore, two databases, which are called ‘User’, ‘Music School’, were created to handle necessary data from the pages. Codes for the databases were developed in Python, which later could be viewed in MySQL, a common open-source relational database management system. The ‘User’ database contains all personal information of students such as their names, login account, addresses. Furthermore, the lesson details for each student are also included. Therefore, administrators can check whether there are any updates to the student details or a new lesson period was booked by a student. ‘Music School’ database stores a different set of information of school’s staffs such as theirs names, appropriate taught lesson and the language they are using for their classes. Moreover, a list of instruments which are able for teaching in the school is also in the database.

To check the correctness of these databases, our team tried inserting random information of students and staffs. This could be done easily as we were able to use the experience from past subject IBF130 for our work. The result met the requirement for database integration, when the databases are updated each time teachers/students create a new account or update their information. The updating process is completed by a set of PHP code included in our webpages html sources, which can receive the inserted information immediately and store it into the databases.

Although the databases and all the appropriate processes outcome were as our expectations, it could be further improved. After doing the databases using Python, we realized that we could adjust the code with MySQL, which is much easier to handle and review. By using better editor tool such as Sublime Text, we could find it more comfortable to set up the coding because this tool let us see the HTML and PHP code in a professional way.

***Server Implementation and Presentation – N9922679 (Callum Miller)***

Regarding server implementation, the website was publicly accessible, however only barely functional. Due to issues deriving after the implementation of the databases, the website itself was only functional when static HTML pages were being used. This however wasn’t desirable as it didn’t convey the full functionality of the site and outlined by Mika’s requirements.

The main issue was integrating the database SQL as a Windows based PC was used to implement the server. Therefore, third-party software available to set up a server with the product functioning on a public address was scarce. As such, Linux should have been used as the environment as it has access to a significant amount of third-party software that preforms this operation; with an even larger supporting group for those having similar issues we encountered. However, given our deadline we didn’t have time to reimplement the server on a Linux device. Due to this a, local server (LAN) using Xampp was created to allow for the presentation to go ahead while the public version was still being redeveloped. We presented Mika a document referred to as ‘Justifications and Known Errors’ that outlined the errors of the website and reasons as to why they are not functioning, including the server functionality which disused the above issues.

If we could change it from being a locally hosted server for the presentation we would implement it using a third-party software on a Linux device. Things we would keep was the use of documentation during our presentation. This allowed us to hand over things like the Justification, allowing our team to properly sell our product and present the project itself rather than covering the less interesting and irrelevant parts of the development cycle. Other than the lack of a public address, the product itself functioned seamlessly, running into no other issues that were previously outlined. LAN also allowed for better loading times (literally none) while other groups appeared to have some issues performing basic server-side actions. This allowed for a quicker presentation, for more time at the end for questions and further discussion on the project itself.

***Documentation/Formatting for Final Release – N9966005 (Dylan Vardanega)***

A final product with no errors, no flaws and no negatives is no doubt the main aim a development team strives to achieve. These include the Regression Summary, Justification and Known Errors as well as the How-To documentation (just to name a few).

For starters, the Regression Summary highlights the phases in which our regression testing would take place, and in what order. Created using our teams test case document (created by Team Members Marin and Dylan), regression testing is an important part of finalising and polishing off the final product aiding in the final release. To accompany this, a Regression Summary was created, highlighting modules (or features) in sets of 2, gauging their functionality between each other. This could be as simple as seeing if a Button redirects you to the proper page, or as advanced as initialising a User’s credentials from one page to the next with the use of a database.

A Justifications document was also created to keep Mika up to date on any known errors or issues currently present in the system. This was seen to be the most professional approach as producing a product without notifying the client of any errors or bugs may either lead to questions that could have easily been answered, or worse, not gaining client approval to continue creating the system. Its contents covered the known system issues as well as a brief server justification as to why we we’re required to use a LAN connection rather than a privately hosted domain.

Moreover, the How-To-Guide allowed for further explanation of the project itself, highlighting each of the features and their function. Seeing Mika was absent, its inclusion proved to be even more effective, meaning that Mika herself could review the website without needing to be guided on how to do so. Colour coded text relating to similarly coloured rectangles were to highlight the feature in question. A How-To-Guide, although sometimes simple in its design, is an imperative piece of documentation that should accompany any functional product, therefore being included with the final release.

Although these documents proved useful and essential to the Final Release, consistency and communication between members to create these final documents could have been improved. Mainly in the department proof reading, a minimum of 2 members were selected to review final pieces of documentation. However, the documents themselves were always explained to each member, and even though they hadn’t directly influenced its wording, the topics discussed were always reviewed, with suggestions and guidance being considered. In future, proof reading needs to be heavily enforced, with meeting goals of double checking documentation possibly being incorporated. In doing so, the quality of the work produced will no doubt increase while also involving team members, keeping them up to date on all current work being produced and an insight into who is doing what.

***Test Cases – N9899138 (Marin Marin)***

A test case is a document which has a set of test data, conditions or variables under which a tester will determine whether a system being tested satisfies requirements or works as expected.

The test cases that we have produced had the following elements: Test case ID, Test Objective, Precondition, Steps, Test data, Expected result / Acceptance criteria, Post-condition and Test Status.

The process of developing a test case has aided us in finding problems in the requirements and the design of Pinelands Music School website. One of the major benefits that we have achieved using test cases is that they can be referred to at any point of the development stage allowing the reader/team member to easily understand the end-to-end functionality of any feature within the application. Essentially, when writing a test case, we needed to make sure that we only test one thing at a time which has allowed us to find any potential defects in our product. Additionally, to verify that the software meets the end user requirement we had to follow one of the major rules when writing our test cases and that is to keep them ‘atomic’. Making sure that we do not overlap, and complicate the criteria contained within the document resulted in delivering excellent test coverage. Additionally, during our testing we tried to cover all positive and negative outcomes. As previously mentioned, to ensure that none of the key functionality is missed in the testing process we had to write in simple and easy to understand language. Meaningful and consistent names were used when naming forms, fields, tests cases, etc. assuring an easier readability as well as clear understanding of our test cases.

There were not many aspects we did not accomplish well for this objective, however there may be room for improvement when it comes to optimizing the quality of the test cases. Due to the requirement to adhere to stringent timelines, we may have lacked the opportunity to conduct more thorough testing of some of the important functionalities and aspects of our website. Additionally, we may consider including more team members to participate in writing test cases. The main benefit of this is the prevention of being stuck by losing the ability to create new test cases due to the lack of new testing scenarios.

One of the main takeaways to improve upon next time is certainly to have a focus on writing better quality test cases by covering the “high priority bugs”, simply because these bugs have a greater impact on the system and are more significant to the end user. Furthermore, we could involve the testing team to start working on the project from the beginning which would result in identification and elimination of potential errors in the early stages and even in the pre-development phase.

***Website Integration – N9427309 (Janice Vee)***

A website needs to have a continuous flow of theme which reaffirms to the user that they are still on the same website. Consistency also conveys professionalism and credibility. Our team has created a website with a clearly identifiable theme to ensure that this criteria and expectation is fulfilled. We delegated the task of creating the individual webpage framework codes by division between team members. This meant that the final product was an amalgamation of code which no continuous theme.

The functionality and skeletal elements were present however the fluidity and interconnected links were absent. To create a fully functional interconnected website we introduced links and button navigation. The links were implemented via the employment of ‘hrefs’ and the code was written primarily in html (with database functionality enabled by php and general functionality via javascript). The work done by individual team members was to a high quality and therefore integration and website continuity was achieved in an efficient manner. Moreover, everyone could work autonomously during developmental stages and collaborated when there was a tangible product.

Overall, the team worked together well to produce a satisfactorily viable product. There wasn't much that did not go well with website integration other than the occasional wait for response. Although there were weekly group meetings for the second sprint, there were some absences as exam time was looming, unavoidable illnesses and other commitments. This hindered the group's productivity as discussions were limited to online correspondences and this could be somewhat inefficient for certain tasks. It did slow the group down but not significantly. Of course, things cannot run in optimal state all the time. So, creating a contingency plan based off unexpected absences can be implemented to mitigate these negative implications.

To address this identified issue, we will gather everyone's availabilities for the following week during the team meeting of that week; to ensure that we know when everyone is available. This would be the "back up" plan. We will be able to work more efficiently by not waiting on others indefinitely as we know when their availabilities are. This will also allow for better planning and allocation of tasks as we will be able to better grasp team member's capabilities to complete certain task with the consideration of their available time.is available.