Alexandria Griffiths - n9458816

TEAM 111, IFB299

Personal Portfolio

# Overview

I was the team leader for Team 111 and one of two team members that worked on developing the product.

I have written the entire website dynamically using PHP and HTML. There were no frameworks used in the development of this project. The **114 pages** of code (**10000+ words**) I have contributed have been self-taught over the semester and written by hand from scratch without any scaffold or guidance.

Coming into this unit, I had only a basic understanding of HTML. However, in order to complete this project I had to learn PHP and continually teach myself new functionality by using online tutorials and browsing online forums. The websites I used most to teach myself the concepts necessary to complete this project are **w3cschools** and**stackoverflow**.

My contribution extends to writing all of the website’s functionality (such as signup/login/logout, user permissions/restricted access, retrieving from and sending to the database, form submission, site navigation, displaying the results of SQL queries, an automatic email notification system and more) and creating/maintaining the database - all from scratch.

All of my code is extensively tested over 10s of hours of development and is bug-free.

*My complete code contribution can be found in the document “Total Code Contribution to IFB299.docx”.*

# Summary of Artefacts:

## Release 1

Artefact 1 - *Remotely hosted and accessible database (PHPmyAdmin)*

Artefact 2 - *Creating a dynamic product with PHP*

Artefact 3 - *Inserting data into the database using $\_POST forms*

Artefact 4 - *PHP functions in .inc files to control $\_SESSION permissions*

Artefact 5 - *Testing documentation and Selenium suite*

## Release 2

Artefact 6 - *Researching the format of the Business Letter (Peer Review 2)*

Artefact 7 - *Editing the Business Letter (Peer Review 2)*

Artefact 8 - *Displaying data from an SQL query in a table*

Artefact 9 - *Using PHP $\_GET form submission, retrieving $\_GET data from new page*

Artefact 10 - *Automatic email notification system*

# Artefacts for Release 1

## Artefact 1 - Remotely hosted and accessible database (PHPmyAdmin)

The submission criteria for Releases 1 and 2 stated that tutors required portable access to the project and, if implemented, its database. Although I began creating the database locally using MySQL I realised that production needed to stop until a portable solution was found. I was unable to export the database efficiently or reliably between team members using MySQL so an alternative solution had to be found. I downloaded XAMPP, a self-contained portable application with access to a portable database. However; upon testing its portability with other team members the database service was inaccessible and we found XAMPP to have further compatibility issues with ports. I also set up an EC2 Amazon remote server although this service required too much access to my personal email account and bank account with too many hidden fees for me to feel comfortable using it.

Lastly, I set up a remotely hosted domain with its own database using X10hosting services. These services used the program PHPmyAdmin. I proceeded to access and edit this database with the appropriate tables, primary keys, fields, field restrictions and test case data.

*See Artefact 1 documentation for evidence of this work:*

*>attached screenshot of MySQL  
>attached screenshot of XAMPP  
>attached screenshot of Amazon EC2 web browser history and confirmation email  
>attached screenshot of X10hosting  
>attached screenshots of PHPmyAdmin database + exported database .sql file*

Creating a remotely hosted database was crucial to store data and ensure portability for the product at all stages of development.

## Artefact 2 - Creating a dynamic product with PHP

My team didn’t possess enough programming skills, so I learned the PHP programming language by following the online tutorials outlined in the artefact evidence.

*See Artefact 2 documentation for evidence of this work:*

*>attached PHP pages I wrote  
>a browser history of websites I used to teach myself PHP*Most helpfully:  
[*http://www.w3schools.com/php/*](http://www.w3schools.com/php/)[*http://stackoverflow.com/questions/11312316/how-do-i-add-php-code-file-to-html-html-files*](http://stackoverflow.com/questions/11312316/how-do-i-add-php-code-file-to-html-html-files)

Learning PHP was vital in ensuring our website was dynamic and could access functionality that was required for our product and more advanced than basic HTML such as $ variables, post/get forms and PDO/SQL queries.

## Artefact 3 - Inserting data into the database using $\_POST forms

Our product had to connect to the remotely hosted database to insert, update and retrieve data regarding users, permits, violations and health and safety reports in order to deliver the functionality requested in the specification document. As no-one in my team possessed the programming skills to do this, I had to teach myself using online forums how to post values using a form and insert those values into the database using PHP, PDO and SQL queries.

*See Artefact 3 documentation for evidence of this work:*

*>a browser history of websites I used to teach myself login functionality  
>attached document demonstrating $\_POST, PHP function, PDO and SQL using code screenshots  
>attached code (index.php, signup.inc)*

Learning how to save PHP $ variables from values posted through a form and learning how to use those values in PDO and SQL queries in PHP allowed me to access the database from any page on the website, with any variable filter for any purpose.

## Artefact 4 - PHP functions in .inc files to control $\_SESSION permissions

Part of the specifications for our product included the creation of user accounts that granted (or restricted) access to different pages of the site. As no-one in my team possessed the programming skills to do this, I had to teach myself how to write PHP functions, assign values to and retrieve values from $\_SESSION variables and how to access .inc files using online forums.

After much research, I retrieved the user’s account information from the database using SQL, implemented $\_SESSION variables to ‘store’ the user’s permission and then accessed these $\_SESSION variables using PHP functions I created in .inc files.

*See Artefact4 documentation for evidence of this work:*

*>a browser history of websites I used to teach myself login functionality  
>attached document demonstrating admin $\_SESSION permissions using code screenshots  
>attached code (index.php, adminPermission.inc, menuAd.php)*

These $\_SESSION variables and .inc files allowed me to restrict access to the site according to the user’s account permissions.

## Artefact 5 - Testing documentation and Selenium suite

In order to ensure that the system could meet the requirements set by the given specification document, I wrote a suite of test cases in Selenium and integrated System Integration Testing and User Acceptance Testing by taking on the role of a user to test user stories in the system. I also wrote up the scrum testing techniques used throughout Release 1 in a word document.

*See Artefact 5 documentation for evidence of this work:*

*> word document with the testing techniques and processes we adhered to throughout Release 1  
> a folder of Selenium test cases*

It was through my testing that we confirmed our completion of each user story outlined in Release 1 and confirmed that the functionality of our product was operating as it was supposed to.

## Other non-artefact contributions to note (evidence can be provided upon request):

* I arranged the user stories for Release 1 & 2 and contributed heavily to the description of each user story and writing the tasks for each user story.
* I contributed heavily to MOSCOW documentation
* I contributed heavily to writing the Team Agreement
* I contributed heavily to the feature recommendation document for the client team
* I wrote the description and acceptance criteria for most of the user stories in the Story Card PowerPoint
* I wrote the ‘Interactions with Client Team’ documentation for Release 1

# Artefacts for Release 2

## Artefact 6 - Researching the format of the Business Letter (Peer Review 2)

Although we had researched the format of business letters for our first Peer Review, I was concerned by the feedback we received regarding the format and content of our submission and endeavoured to find an example of the precise business letter format our tutors desired to ensure an improved mark for the second Peer Review. To do this, I consulted our unit coordinator and used his feedback to provide recommendations to my team.

*See Artefact 6 documentation for evidence of this work:*

*> screenshots of timestamped email correspondence  
> screenshots of shared recommendations and PDF files to group page  
> copies of the attached PDF files.*

As well as passing on the PDF files he provided as example documentation to my team, I made the following recommendations:

* use correct spacing for ‘sender’, ‘date’, ‘recipient’ and ‘from’ fields
* align ‘sender’ and ‘from’ fields to the right of the page
* break information into smaller, more readable paragraphs according to content
* address Team 113 directly (e.g. ‘your team did…’ instead of ‘Team 113 did….’)
* focus on Team 113’s product and delivered business value over their presentation
* justify feedback and recommendations

I included this artefact because it was instrumental in improving the quality of our second Peer Review business letter.

## Artefact 7 - Editing the Business Letter (Peer Review 2)

Our initial draft of the business letter for the second peer review was an amalgamation of the bullet points we had brainstormed in Google Docs. I heavily edited this document to ensure that our submission would be consistent with the standards expected of a business letter, met the Blackboard criteria for the Peer Review assessment and was consistent with my research and recommendations from artefact 6 - notably I focused on recommendations and feedback justification.

*See Artefact 7 documentation for evidence of this work:*

*>word document showing the original draft and my final submission after editing, with my contribution colour-coded*

My editing ensured that we delivered a business letter to Team 113 with the correct formatting and with comprehensive, relevant feedback.

## Artefact 8 - Displaying data from an SQL query in a table

Our user stories specified that data from the database be presented according to specific SQL queries restricting the results according to the user. As no-one in my team possessed the programming skills to do this, I had to teach myself how to display PDO SQL results cell by cell by row by table using $\_SESSION variables, HTML, SQL and PHP.

*See Artefact 8 documentation for evidence of this work:*

*>a browser history of websites I used to teach myself PDO SQL retrieval, tables, row looping and separating data from raw result output  
>attached document demonstrating table display using code screenshots  
>attached code (readpermits.php)*

Implementing this system allowed for data to be filtered and displayed according to status, table and the user accessing it.

## Artefact 9 - Using PHP $\_GET form submission, retrieving $\_GET data from new page

Part of the specifications for our product included the ability for admin accounts to access and edit Health and Safety reports from the website front-end (without going directly into the remote database). As no-one in my team possessed the programming skills to do this, I first had to figure out how this might be achieved in CS logic. I deduced that I had to use the internet and teach myself how to save the id of the report the user wished to edit, how to open a new page with prefilled input fields according to that id and how to update the existing records for that report id in the database using the new information.

I found that I had to use $\_GET to access a value between pages, and that to use $\_GET I had to submit a get form with the value I wanted saved. Once I retrieved this $\_GET value of the report id, I could set the ‘value’ of each input field to the appropriate value from the database with SQL queries and PHP variables. Updating the existing records required learning how to use UPDATE queries in SQL rather than INSERT INTO.

*See Artefact 9 documentation for evidence of this work:*

*>a browser history of websites I used to teach myself $\_GET forms, hidden fields and PHP forms  
>attached document demonstrating $\_GET using code screenshots  
>attached code (writereports.php, editreports.php)*

Posting these $\_GET variables allowed me to retrieve data sent by a form on a previous page in order to send an SQL query from a new page using the $\_GET to display the correct data.

## Artefact 10 - Automatic email notification system

Our user stories specified that our product needed to include an automatic email notification system that is triggered by certain events in the system. As no-one in my team possessed the programming skills to do this, I had to teach myself the relevant skills by searching online forums, learning about the email() function in PHP, learning how php.ini worked on remotely hosted servers and how to access a placeholder email address using x10hosting to send the emails.

*See Artefact 10 documentation for evidence of this work:*

*>a browser history of websites I used to teach myself email setup and sending emails in PHP  
>attached document demonstrating email functionality using code screenshots  
>attached code (parking.php)*

Implementing this system allowed for emails to be sent automatically to the user upon permit approval or a new violation against their name, and for emails to be automatically sent to all admin accounts upon new violation, permit or health and safety report lodgements to prompt followup action.

## Other non-artefact contributions to note (evidence can be provided upon request):

* I contributed heavily to Sprint Plan 3
* I contributed heavily to Sprint Plan 4
* I contributed to the testing documentation