City University London

IN3007 Individual Project
Final Year Project Report

Academic Year: 2014-15

An application for access to university-related resources

by

Sergiu Tripon

Project supervisor: Dr. Vladimir Stankovic

7 May 2015

Table of Contents

Abstract

The scope of this project was to analyse, design, implement and test a web-based application that enables the students and academic staff of a university to access university related resources, and administrators to manage the resources.

The application is multi-purpose, providing its users with a wide array of functions, from checking transport status information or searching for locations around the university campus to checking the timetable, booking an event or reserving a book from the library. It was built as a prototype and is aimed at universities or other developers as a way to improve access to resources.

The current situation within City University London is analysed thoroughly in order to identify its key issues, something that the application aims to address. By establishing the problems with the current situation, requirements become easy to gather. In addition to that, it allows creativity to play an important role in coming up with additional innovative features that make the application distinctive from others.

The project report describes the methodologies used throughout the project and provides results from each phase of the project from the Requirements and Analysis and Design phase to the Implementation and Testing phase.

Conclusions and recommendations determining the success of the project considering each output delivered and the objectives met are discussed. Additionally, a reflection on the overall project is expressed while future improvements to the application delivered are identified and suggested.

1. Introduction

This report discusses the contributing factors to the production of a responsive web application designed and developed to aid students and academic staff in accessing university-related resources from a single central place.

1.1. Project overview

This project involved creating a responsive web application that enables the students, academic staff of a university to access university-related resources, and administrators to manage the resources. Some of the significant features are: allowing students to pay for course fees, booking events, reserving books from the library, checking transport status information or searching for locations within the university campus. In addition to that, various documents have also been produced, such as requirements, analysis and design and testing documents as well as user and deployment manuals. A use case diagram that showcases the entire functionality of the system and the actors involved in the functionality has been created and can be viewed on page C76 of Appendix C.

1.2. Problem to be solved

The key issue that sparked this project is the current inefficiency of City University's system when trying to perform a simple action such as paying course fees, checking a timetable or updating personal details. These actions currently require three different applications to be performed, instead of just one.

Furthermore, during the very early stages of the project, most parts of the City University website and other resource websites were not responsive. In my opinion, this is an important issue as the majority of visitors to the website will be students who are usually on the go. This means they will most likely access websites on a smart phone and if the website they are accessing is not responsive, this will decrease their control and efficiency while using it.

Thus, the scope of the project was mainly focused on solving the two main problems discussed above. The key output of the project is a responsive web application which allows users to access and perform a large array of functionality and actions. The application does not contain links to any external web applications with the exception of PayPal for payments; therefore the majority of functionality happens within the application.

Additionally, other pieces of work have been produced to accompany the web application such as Requirements, Analysis and Design and Testing documents as well as a User and Deployment manuals.

1.3. Project objectives

The objectives of the project were categorised by priority as follows:

Critical Priority

- The application will undergo an analysis and design phase based on UML.
- The application will undergo an implementation/testing phase. This will consist of UML diagrams of the system architecture and acceptance/unit/exploratory test cases.

High Priority

- The application will be fully responsive, usable on desktop/laptop, tablet or mobile phone.
- The application will provide the following functionality: pay course fees, check timetable and exam timetable, check exam results, reserve a book, book and pay for events, provide lecture feedback, check university map, message other users etc

Moderate priority

- The application will be cross-browser compatible.
- The application will undergo an evaluation phase by an end-user within the target audience.
- The application will have a user-manual.

1.4. Work plan

To ensure that the project objectives were met, a work plan was created at the start of the project and each work package has been assessed against the work plan in terms of its due date and its actual due date. By doing that, the possibility of missing deadlines by shortening or going overdue on specific work packages was prevented. In addition, supervisor meetings and the "half-way" progress review also contributed to ensuring that I have delivered the work that was expected and that the work met its objective, which subsequently indicated whether I was following the right trajectory. Before the Project Progress Review, the work plan changed while the final deadline remained the same. This change had to be made due to the fact that certain work packages were achieved quicker than initially estimated, while other work packages took longer. Therefore, the work plan was amended, with the work packages being shifted slightly, while also preventing the final deadline from being changed. The changes to the work plan can be viewed on page H45-H46 of Appendix H.

1.5. Project beneficiaries

The primary beneficiaries of the project are the students, academic staff and administrators of a university. The application may also be used to help potential developers (of the same or a similar application) due to the fact that the application was built as a prototype providing a solution to the problems identified.

2. Output summary

This chapter provides definitions of all outputs that resulted from the project. For each output, a description, type, recipient(s), usage and reference links are provided.

2.1. Student Portal web application

Description:	Web application enabling access to university-related resources. It consists of the source code and database configuration.	
Type:	Software code. Entire web application folder contains around 247 files and 79 folders; Out of this, 85 files were created and consist of code written by me. The code in 57 files was written completely by a third party, while the code in 13 files was written by a third party but also includes alterations made by me. Overall the application consists of **91.3% PHP**, **4.4% JavaScript**, **3.6% CSS** and **0.7% other** code. This data is provided by GitHub.	
Recipient(s):	• Students • Academic staff	• University administrators • Developers
Usage:	• **Students/Academic staff**: use it to access university resources • **Administrators**: use it to set up and configure the application • **Developers**: use it to further develop and improve the application	
Reference links:	**Appendix F**	**Results link**

2.2. Requirements document

Description:	The Requirements document consists of the requirements gathered prior to the analysis and design phase. Each requirement is showcased through a print screen taken from VSO, the tool used to record the requirements.	
Type:	System specification. 54 pages long.	
Recipient(s):	• Developers	
Usage:	• **Developers**: use it to view the requirements gathered which will provide an indication of the scope of the application	
Reference links:	**Appendix B**	**Results link**

2.3. Analysis and Design document

Description:	Consists of: Use case specification and diagrams, Class diagrams, ER diagram and CREATE and DROP database statements, Component and Deployment diagram.	
Type:	System specification. 126 pages long, containing 17,787 words.	
Recipient(s):	• Developers	
Usage:	• **Developers**: use it to understand the application flow and how the database backend was analysed and designed	
Reference links:	**Appendix C**	**Results link**

2.4. Testing document

Output:	Testing document	
Description:	Consists of: Unit, Usability, Cross-browser compatibility and Responsive testing documents. The preparation prior to testing, test cases, test results, resulting bugs and improvements as well as resolutions are detailed within the documents.	
Type:	Documentation. 4 documents, containing 3,511 words over 114 pages altogether.	
Recipient(s):	• Developers	
Usage:	• **Developers**: use it to find out the degree of testing that has been done on the application and the methods and tools used will provide an indication on how much further testing is required as well as a blue print of how to perform the testing phase	
Reference links:	**Appendix D**	**Results link**

2.5. Video and Audio recordings (usability testing)

Description:	The usability testing phase was recorded with video and audio detailing the events that took place. I also contacted Dr. Stephann Makri from City University's Centre of HCI design seeking some usability expertise feedback on the application. The meeting was 20

	minutes long and was audio recorded.	
Type:	Documentation. The recording of the Usability testing phase is approximately 1 hour long while the audio recording of the feedback provided by Dr. Stephann Makri is approximately 14 minutes long.	
Recipient(s):	• Developers	
Usage:	• **Developers**: use it to get a more precise look at what the end-user pointed out or suggested which may not be as clear from reading bug or enhancement records within the Usability testing document	
Reference links:	**Appendix D**	**Results link**

2.6. Manuals (user and deployment manual)

Output:	User Manuals	
Description:	Consists of two manuals, a user and a deployment manual. The user manual consists of three different manuals, one for each account type. It describes, using both text and images, how to perform every action throughout the application. The deployment manual details how to deploy the application to a web server as well as how to create the database, the database user and tables and how to connect to the database. In addition to that, it also explains a number of specific actions required to be performed prior to deployment.	
Type:	Documentation. Consists of 14,700 words over 255 pages.	
Recipient(s):	• Students • Academic staff • Administrators	
Usage:	• **Students/Academic staff/Administrators/Developers**: use it to learn how to perform actions within the application • **Developers/Administrators**: use it to learn how to deploy the application	
Reference links:	**Appendix E**	**Results link**

3. Literature review

This chapter aims to demonstrate the literature that was read and studied throughout the project in order to gain the knowledge necessary to deliver the project successfully. It includes reading and research undertaken into design approaches as well as tools and technologies integrated during the project.

3.1. Designing Multi-Device Experiences by Michal Levin

Designing Multi-Device Experiences is described as follows: *"This practical book demonstrates the variety of ways devices relate to each other, combining to create powerful ensembles that deliver superior, integrated experiences to your users." (Levin 2014, p. 2)* This book helped me to understand the primary principles of designing multi-device experiences. I learnt about a number of different design approaches such as continuous, consistent, complementary or integrated. In addition to that, I also learnt about new aspects of the technology world such as the eco-system concept. As *Michael Levin (2014, p. 3) explains, "In looking at the world of online apps and electronics today, we can see a type of ecosystem emerging. In this system—this climate of multiple devices—we see smartphones, tablets, laptops, TVs, and other connected devices all interacting with one another and wirelessly sharing data. These interactions are shaped by the different ways in which individuals use the content and services that flow between devices, in different contexts, en route to their goals"*.

Through this information, I went on to think about device interaction from a higher level perspective which made it easier to understand the interaction between multiple connected devices. Moreover, the text aided in the planning and design process of the application by helping me to ensure that appropriate design principles were followed.

3.2. Bootstrap documentation

Bootstrap is described as: *"the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web." (Bootstrap, 2015)* Bootstrap is a framework which is intended to make web development work easier while also ensuring a mobile first design approach where the focus is primarily on the mobile, with the tablet and then the desktop following. Bootstrap achieves its objectives by providing a wide array of responsive components built using HTML, CSS, and JavaScript which can be used to achieve a website design easily and efficiently. The components include buttons, panels, breadcrumbs, tables, menus, footers and many others.

Once become familiar with it, Bootstrap is extremely beneficial to any developer as it makes development easier while also implementing a responsive design at the same time. However, learning Bootstrap's syntax can take some time as every CSS class or JS function has a name given by the Bootstrap developers. By deciding to use Bootstrap, I had to learn its syntax and how to use the components correctly. Bootstrap provides three different help pages, labelled as CSS, Components and JavaScript to aid beginners in developing responsive websites. The provided Bootstrap documentation helped me to learn how to use the components in order to develop and implement the application and its responsive behaviour.

3.3. GitHub Help

GitHub is described as *"the largest code host on the planet with over 20.8 million repositories. Large or small, every repository comes with the same powerful tools. These tools are open to the community for public projects and secure for private projects." (GitHub, 2015)* Through the GitHub Help page, GitHub provides documentation which helped me by explaining how to perform certain tasks such as committing/pulling changes on files to/from version control as well as how to push commits to GitHub or create branches and merging between them.

3.4. PhpStorm Help

PhpStorm is described as: *"Enjoy productive PHP and web development with PhpStorm. Take advantage of deep code understanding, top-notch coding assistance, and support for all major tools and frameworks." (PhpStorm, 2015)* The PhpStorm manual helped me to learn how to integrate the tool with GitHub as well as how to achieve other important tasks such as setting up automatic FTP upload after committing changes or comparing code between two files. Committing to GitHub, followed by uploading the changes to the web server using FTP, has been the primary post-development deployment method. Comparing the code between two files has been used significantly throughout the "Re-use of code" document which can be accessed in Appendix G. Therefore, the help received and the knowledge gained from using PhpStorm Help has been essential in achieving critical aspects of this project.

3.5. PHP Security Cheat Sheet

PHP Security Cheat Sheet is described as follows: *"This page intends to provide basic PHP security tips for developers and administrators." (PHP Security Cheat Sheet, 2015)* The PHP

Security Cheat Sheet is a lengthy webpage which identifies a large number of security tips to counteract security threats when developing a PHP application. I read the majority of the documentation, and discovered tips which were already implemented within the application while others weren't. Using the knowledge acquired from the documentation, I implemented a number of security tips such as prepared statements, password hashing or SSL protocol.

3.6. Google Developers documentation

Google Maps is described as: *"Make places easily discoverable with interactive maps built for your users* [...] *Help your users discover places using their own personalised map, tapping into Google's rich database of over 100 million places worldwide. It's free to use, with no quotas, limits or cost to you." (Google Maps Developers, 2015)* Google provides a developers page focused on Google Maps. This page consists of various demos and information on loading maps, displaying markers and data from a database, searching for locations on the map, filtering locations by category or using geolocation to detect the user's current location. By trying the demos and reading the documentation, I gained the experience and knowledge necessary for me to develop the University Map page.

3.7. DataTables documentation

DataTables is described as: *"a plug-in for the jQuery Javascript library. It is a highly flexible tool, based upon the foundations of progressive enhancement, and will add advanced interaction controls to any HTML table." (DataTables, 2015)* DataTables offers documentation that explains how to use, implement and enable every feature of the plugin. By reading the documentation, I learnt how to set the limit of records displayed within a table, enable searching and sorting on a table, as well as display custom feedback messages when the table is empty. Moreover, I learnt how to enable the pager feature which works together with the limit of records displayed feature, as it automatically creates separate pages if the number of records displayed within a table is more than the limit of records specified.

4. Method

This chapter details the methodologies and approaches used throughout each phase of the project up to its completion.

4.1. Software Development Process Methodology

Throughout the project, two different software development process methodologies were used. The Agile and Waterfall models were used concurrently for different parts of the project. I had knowledge of three development methods prior to the project: Agile, Waterfall and Scrum, with the most experience in Waterfall. Out of the three, Waterfall was chosen because of its meticulous record keeping, which would make the future enhancements of the product easier for myself or other developers. During the Implementation phase, the Agile methodology was introduced as it allowed more flexibility in terms of changes made to the initial plan. I felt that using Agile alongside Waterfall would be the best option in order to produce the highest possible standard of work considering the project was not client-based, but rather was to be implemented by an individual with moderate programming experience, and therefore likely to be affected by time constraints and errors. The Agile methodology adoption provided the ability to go back and forth until the outcome was up to standard, as long as the changes required were of a minor impact. While working on the Analysis and Design and Implementation phases, I ensured that there would be no major changes required by working diligently and always thinking ahead, weary of the potential impact that every change could have on different stages of the project.

Traditional to the Waterfall model, documentation such as Use case specification, Use case diagrams, Class diagrams and ER diagrams has been produced. In addition to that, a Component and Deployment diagram have also been created. However, the documentation had not been produced entirely prior to the start of the Implementation stage, as it would usually in the Waterfall model. Each stage of the implementation was first planned and analysed, and stage-specific documentation was produced before the implementation. If the stage involved database driven functionality, then the tables would be designed, created and added to the database, as well as documented on the ER diagram. Once implemented, a brief unrecorded unit test of that specific unit of development work was performed. If minor bugs were found, I would go back to the Implementation stage and fix the bugs, before performing the unit test again. If a major bug was found during the unrecorded unit test, a bug record would be created within VSO, the tool used for recording the Testing phase. Once the bug was created, I would not go back to the Implementation phase; instead I would move onto developing and documenting the next unit. The bugs would then be fixed during the bug fixing stage of the Testing phase. If the unit test passed, the Implementation phase would end and I would move onto the next unit. At the end of the Implementation phase, a complete, recorded unit test was performed against the application, while test suites, cases and results were created and documented thoroughly.

4.2. Project Communication

Throughout the project, I communicated to my supervisor and the module leaders through email, in person or occasionally Skype. I had a meeting with my supervisor at least once every month. Meeting minutes were produced during each meeting. The entire email communication and meeting minutes were compiled to form a Project Communication document, which can be accessed in Appendix H.

4.3. Requirements phase

In a client-based project, the requirements phase is very different to that of a non client-based project. A client-based project usually involves an individual who has no substantial experience of the project’s scope. This means the requirements would be primarily gathered from the client, as they’ve directly experienced the issues and identified their requirements. In contrast, in a non client-based project, requirements must be gathered from a group of people that share mutual problems with the current situation. In this specific project, the scenario is different since I, myself, fit within the target audience of the project as a student who uses the City University website and other university-related resources regularly. During my first two years of university, I struggled to understand why the university provided so many resources on several different web applications. I was constantly being redirected to a variety of different web applications in order to achieve a minor task such as updating personal details or paying course fees. An analogy to the issue could be calling BT and constantly being redirected to a number of different departments, when all I want to know is why the Internet is not working. Therefore, I came up with the majority of the application requirements. Other requirements were identified through sessions with a number of other individuals who were within the target audience and expressed similar difficulties. Additional requirements were later added to further improve the current situation such as transport updates, messenger, calendar etc. All requirements were recorded within a web-based tool called VSO.

4.3.1. Use case specification and diagrams

In order to showcase and demonstrate the flow and functionality of the system, I took advantage of my UML experience and created a use case specification. A template was designed and used to create a specification for each use case an actor can perform. This provided extensive details about the preconditions, basic flow of events to perform a certain action, alternative flows, key scenarios as well as post-conditions. To represent the use case specification visually, use case diagrams were created, which aid in understanding the general navigation, functionality and flow of the system as well as identifying the actors involved. The use case specifications and diagrams were useful as they helped me to spot inconsistencies in the initial thought process. This led to preventions of major errors at the design stage, which could have had a large impact on the project if identified at a later stage.

4.4. Design phase

During the design phase of the application, I had to analyse the application from a number of different angles such as the application and database backend and the GUI. The application backend represents the functions developed to perform the actions within the application such as manipulating the database, sending emails or retrieving/sending data to/from external sources. In comparison, the database backend represents the database, its tables, the data types and the relationships between the tables. The GUI represents the interface involved in the end-user interaction. During this stage, the documentation produced became more technical while also being more specific and clear in regards to how the application will be designed and developed. At this stage, the backend database was designed and consisted of the database tables, their relationship, each column’s data type and possible constraints.

4.4.1. Class diagrams

Initially, I started with the idea that the entire application will be covered by one class diagram. Soon after starting to design the diagram, I realised that was not enough to clearly represent the large amount of different areas and functionality involved. Additionally, most of the areas within the application consist of multiple functions, which in theory make them separate sub-systems. For example, the Account area can be considered as one of the main sub-systems within the application. However, the Account sub-system itself allows the user to perform various actions such as: update account, change password or delete account. Another essential factor to be considered is the fact that the functions within a sub-system are completely internal to that sub-system, therefore it does not involve or depend on any functions outside of it. For example, the action of paying course fees, reserving a book, or booking an event do not have any impact on each other or on any other actions. Considering the factors mentioned above, I decided that creating a single class diagram would result in a cluttered and unclear diagram, rendering it unfeasible. Thus, I decided to create separate class diagrams to represent each sub-system within the application. By doing that, each sub-system is showcased with more focus and the overall understanding of each class diagram is improved. The class diagrams were used throughout the Implementation phase, while implementing and developing the application, sub-system by sub-system.

4.4.2. CREATE and DROP database statements

Prior to defining the CREATE database statements, I spent some time analysing the use case specification and diagrams, as well as the class diagrams, and composed a list of table names to form the database backend. I took each table and listed the columns and their data types in a file within the application folder, essentially building the CREATE statement structure for each table. Additionally, within the file, a list of DROP statements is present, which deletes the tables. The file was often used during the development stage as it was occasionally required for the database tables to be deleted and re-created.

The statements became extremely useful because they served as great help during the design process of the ER diagram, as they contain essential information, such as table and column names, column data types, primary and foreign keys as well as column constraints such as *NOT NULL*, *AUTO INCREMENT* or *UNIQUE*. NOT NULL means that the column cannot be NULL, while *AUTO INCREMENT* increases the ID column by 1 every time a new record is created. *UNIQUE* enforces unique data within that specific column. It is worth mentioning that the statements created were not used to create the database tables until the Implementation phase started. However, I found it easier to envisage the tables and their layout by typing them into a file prior to representing them through the ER diagram. In addition to that, I also believe that creating the statements before designing the ER diagram made its creation more efficient due to the fact that the text-based statements were a lot easier to manage and alter than the ER diagram.

4.4.3. ER diagram

In comparison to the class diagram, I was able to create an ER diagram covering all database tables, because the number of tables required was less than the pieces of functionality involved. Before creating the ER diagram, I produced a text-based file consisting of CREATE and DROP statements, which helped me to create the ER diagram. The ER diagram provided a visual overview of the database tables and the relationships between them. I believe that the ER diagram helped to spot critical, logical or accidental errors resulting in a change to the database design and the CREATE statements. For example, initially, the table that stores the user's details was missing a nationality column. This mistake was discovered during one of the reviews of the diagram. Other logical mistakes involved the tables which form the database backend of the Timetable and Library sub-systems. The mistakes are extensively discussed in the Decisions sub-chapter of the Results chapter (Chapter 5).

4.5. Implementation phase

The implementation phase involved designing and developing the web application using the requirements gathered and the foundation built during the Design phase of the project.

4.5.1. MySQL/phpMyAdmin

The RDBMS System chosen was MySQL. Firstly, the choice was influenced by my experience with the technology during the first year of university when I used both MySQL and phpMyAdmin to create the database backend of a web application. Secondly, due to the fact that the chosen programming language was PHP, the obvious choice was MySQL as the two technologies, fit and interact well together. Thirdly, considering the length of the project, I required an easy and efficient to install tool, while at the same time being accessible on the web through a GUI interface, which would enable me to work efficiently. In order to find out which tools matched my requirements, I carried out a brief research phase. Regarding the installation aspect, *Robin Schumacher, MySQL's Director of Product Management states that "Just the .NET framework install took more than 5 times the installation time of the MySQL Server on one of my test machines." (MySQL, 2015)*. The previous experience and the research carried out made me choose MySQL due to its quick configuration, ease of use and web-based aspect as well as being the most suitable for the programming language chosen to develop the application. The query language used within the application was SQL. The tool used to handle the administration of the MySQL database was phpMyAdmin.

4.5.2. PHP

The main programming language used within the application is PHP. *"PHP is a popular general-purpose scripting language that is especially suited to web development." (PHP,* *2015)* Firstly, I chose PHP because I had more experience using it than using any other programming language. This assured that the project would be completed successfully and on time. Secondly, considering the application is web-based, and PHP is a "web-focused" programming language, I always considered it as a strong candidate when I researched ways to develop the application. Other candidates included C# together with ASP.NET, but due to the amount of learning involved and knowledge required, I considered this option too risky and impactful on the project's success. If I had come up with the project idea earlier, I may have tried to learn C# and ASP.NET in order to determine their suitability. Finally, PHP's ability in regards to speed, pre-configuration as well as my previous experience using it influenced my decision to select it. Other programming languages were used such as HTML, CSS and jQuery. Both HTML and CSS are essential when developing a web-based application. Certain aspects such as form validation and posting form data cannot be achieved using HTML alone, therefore jQuery was necessary. jQuery is a JavaScript library and the difference between the two is explained as follows *"The difference is that jQuery has been optimized to perform many common scripting functions and it does so while using fewer lines of code."* The main reasons I chose to use jQuery were because I found it easier to write and understand than JavaScript, and because it is lightweight and cross-platform compatible.

4.5.3. GitHub/PhpStorm

During the first stages of the Implementation phase, I used Notepad++ as an IDE to develop the application, while being aware that by definition, it is not an IDE. Further, I researched alternatives to hosting the application's code rather than relying on the Windows file system. I found a solution called GitHub, and implemented it as the application's code host and version control. The entire application code was hosted on GitHub, which improved certain aspects of the development process, but also increased the difficulty of others. Notepad++ is not GitHub compatible; hence it does not provide any GitHub features. GitHub offers its own desktop application, which detects changes to files and allows users to commit and push the changes. Although this process was still achievable using Notepad++, switching between the two applications constantly was not effective. I required an alternative that would meet a set of requirements, without depending on any other external applications, such as being GitHub compatible or including FTP functionality. Through my research, I found an IDE specifically built for PHP called PhpStorm which became the obvious choice. PhpStorm allowed me to commit and push changes to the GitHub server from within the application, which eliminated the time that was previously wasted with the initial set-up of the GitHub desktop app and

Notepad++. In addition to that, PhpStorm also enabled me to configure it so that for every file that was committed, it would also be automatically uploaded to the web hosting server using FTP, which allowed me to see how changes affected the application in real time. I had no previous experience in using GitHub or PhpStorm; therefore I needed some time to learn both technologies. The documentation provided by GitHub and PhpStorm Help was used to help me perform certain tasks within both applications, such as establishing the connection between GitHub and PhpStorm or setting up the FTP connection to the web server. The documentation used is specified within the Literature review (Chapter 2).

4.5.4. External third party tools and APIs

Throughout the implementation phase, I used a number of external third party tools and APIs such as: Bootstrap, Bootstrap Calendar, DataTables, Bootstrap Date Time Picker, TFL Transport API and the Google Maps API. I believe that the use of the tools enabled me to work efficiently while also improving the look and quality of the application developed. The Results chapter (Chapter 5) provides more information on the use of these external third party tools and APIs.

4.6 Testing phase

This chapter details the testing approaches carried out on the application while also describing the methodologies and tools used during the phase.

4.6.1 Unit testing phase

The purpose of the unit testing phase was to assess and analyse any potential flaws within the system in order to eliminate them prior to the system going live or a demo of the application. Passing the unit testing phase would also mean that the application has met the requirements gathered during the Requirements phase. The test cases used as part of the Unit testing phase were created using a tool called Visual Studio Online. Visual Studio Online is a cloud-based application derived from Visual Studio that *"provides testing tools that help you adopt testing practices such as manual, automated, exploratory, and load testing." (Visual Studio Online, 2015)* I chose to store and document the test cases using VSO because it is a web application within a cloud-based environment. I also believe it to be a safer option than others, such as Excel, where managing large documents proves to be more difficult and data may be lost easily. VSO allowed me to create requirement-based test cases as the requirements were also recorded using the same tool, which meant that I did not have to create any test cases from scratch. The Unit testing phase was performed in a side-by-side testing environment where I had both a test case and the application on the same screen. When a test case was considered as passed, I would simply mark it as passed, and then move on to the next one. If a test case failed, the tool allowed me to fail the test case while also creating a bug for it. At the end of this phase, a document was produced detailing the preparation, test cases, results as well as bug fixing. In addition to that, I also performed responsive and cross-browser compatibility tests. The responsive testing helped me to determine the level of responsiveness of each page, while the cross-browser compatibility testing helped me to determine whether each page is compatible with the '*three most used browsers' (Browser market share, 2015)*. The results formed two further documents which were produced once testing was completed.

4.6.2 Usability testing phase

In order to complement the Unit testing phase carried out, a lengthy usability testing phase was undertaken with an end-user within the application's target audience. The participant was Pratiksha Vekaria, a Spanish and Italian undergraduate student from UCL. The user was asked to take on a number of different tasks within each section of the application. During the first semester of this academic year, I chose HCI as one of my electives. The theoretical and practical experience gained enabled me to produce the usability test cases while also conducting the overall process of the phase. The usability test cases provided the end-user with the bare minimum on how to perform a specific task such as "Starting on the Home page, how would you create a task?" This enabled them to truly evaluate and navigate through the interface as they would if the application was not a prototype. The usability test cases were also stored on VSO. A similar side-by-side testing environment was used, with the participant passing or failing the test cases, as well as creating bugs containing their comments when necessary. The entire Usability testing phase was video and audio recorded. The audio consists of the verbal interaction between myself and the participant, as well as the participant's feedback. The feedback provided by the user was also recorded in a bug record format. I believe that this phase helped me to understand the participant's thought process, and which areas or actions they had issues with. Similarly to the Unit testing phase, at the end of this phase, a document was created which details its events. Furthermore, I contacted Dr. Stephann Makri from the Centre of HCI design seeking expert usability feedback. We had a meeting where he evaluated the application's interface, and provided feedback. The meeting was audio recorded and the recording can be accessed on the media submitted.

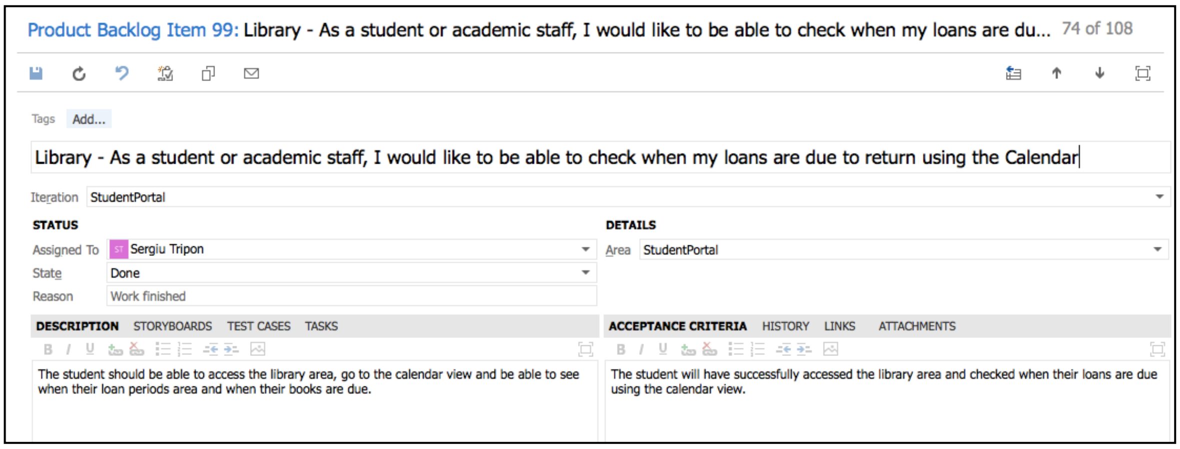
5. Results

The results section of this report will detail the outcomes of using the methods described in the Method chapter (Chapter 4). This chapter will also showcase the outputs that resulted from the project, which were outlined in the Output summary (Chapter 2).

5.1. Requirements phase

Initially, the requirements were based on City University London's current problems in regards to accessing university-related resources. However, I later considered that other universities may have the same problems, and would require a similar application. Thus, the application and its requirements were built as a prototype that could potentially be used by any university to bring their university-related resources into a single place.

Once a full list of requirements was composed, it was transformed into an electronic format through an application called VSO. The terminology for requirements used within VSO is Product Backlog. A product backlog record was created for each requirement. By opening up a product backlog record, I was able to specify a number of different aspects regarding the requirement such as who it is assigned to, its state and description as well as its acceptance criteria. An example of the format and content of the requirements can be viewed in Figure 1.



Product Backlog Item 99: Library - As a student or academic staff, I would like to be able to check when my loans are du... 74 of 108

Tags Add...

Library - As a student or academic staff, I would like to be able to check when my loans are due to return using the Calendar

Iteration StudentPortal

STATUS

Assigned To Sergiu Tripon

State Done

Reason Work finished

DETAILS

Area StudentPortal

DESCRIPTION STORYBOARDS TEST CASES TASKS

The student should be able to access the library area, go to the calendar view and be able to see when their loan periods area and when their books are due.

ACCEPTANCE CRITERIA HISTORY LINKS ATTACHMENTS

The student will have successfully accessed the library area and checked when their loans are due using the calendar view.

Figure 1. Example of the format and content of a requirement regarding the Calendar area of the application

At the end of the Requirements phase, a document was produced consisting of each requirement categorised by the account type that has the privileges to perform it. The Requirements document can be accessed in Appendix B.

5.1.1. Use case specification and diagrams

Following the requirements phase, I started to think about designing a use case specification template and I found one by *Eclipse.* A reference to the template can be accessed in the References section of this report. Using the use case specification template, I started to write the specification for each requirement. This included the requirement's description, its actors, preconditions, basic flow of events, alternative flows as well as its key scenarios and post conditions.

Once the specification was completed, I transformed it into a use case diagram. Initially, I started to create an all-in-one use case diagram which would have consisted of representations of the entire use case specification. This proved to be a bad decision due to the fact that the system is made out of 12 separate sections while each includes extensive functionality which would make the use case diagram unclear and untidy. To avoid having a badly designed diagram, I decided to create a separate diagram for each section of the system. That way, anyone analysing the diagram would clearly understand what each actor does and can do, and the flow of actions within the application. In addition to this, I created an overview diagram which shows a high level view of how the areas of the system are used by the actors. The use case diagrams were created using Visual Paradigm. The creation of the use case specification and use case diagram aided the design of the class diagram as well as the Implementation phase. An example of a use case diagram can be seen in Figure 2. The use case specifications (p. C1-C75) and diagrams (p. C76-C84) created can be accessed in Appendix C.

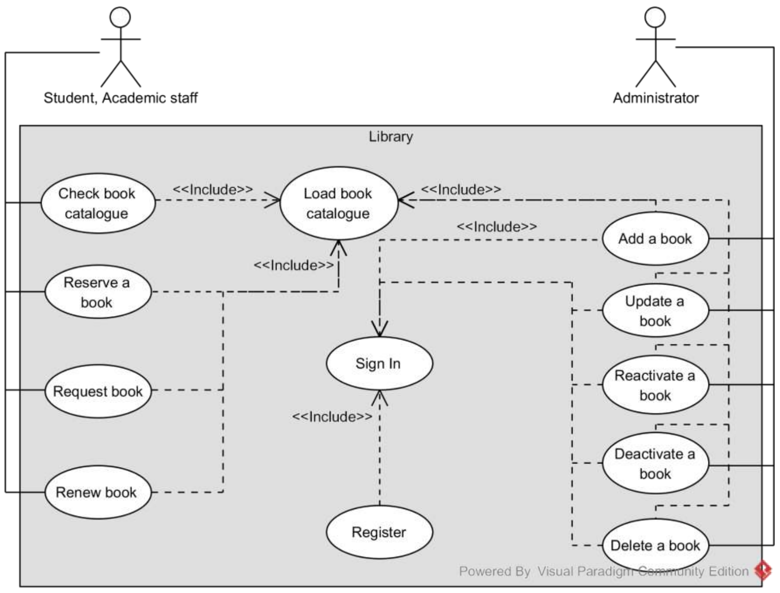


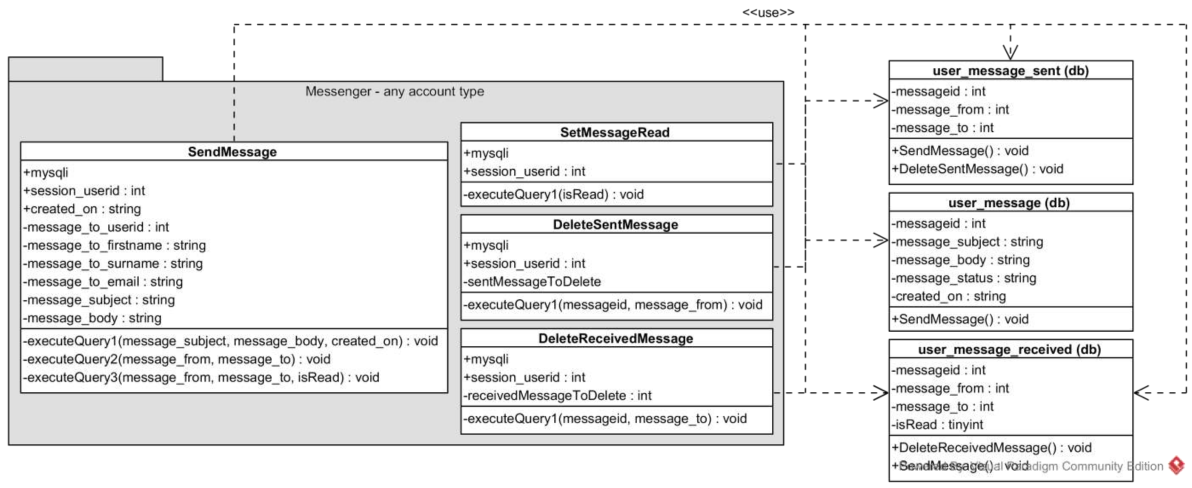
Figure 2. Examples of use case diagrams for the Transport and Library sections

5.2. Design phase

The design phase of the project involved the creation of several pieces of documentation. Firstly, based on the foundation built by the Requirements phase, I created a number of class diagrams showcasing the design and flow of each sub-system of the application. Secondly, the database was designed by initially thinking of the tables required and building the table structure in a text-based format. This was later transformed into the ER diagram which provided a graphic representation of the database design. Finally, I also created a Component and Deployment diagram, which can be accessed (p. C123- C124) in Appendix C.

5.2.1. Class diagrams

The creation of the use case specification and diagrams built a foundation for the class diagrams. I created a number of class diagrams representing each area of the system. This enabled me to analyse and envisage the structure of the classes more clearly and specifically. Each class within a diagram represents a function, its input and outputs as well as the operation and constructor within the application. Additionally, the class diagrams also provided an insight into the connectivity, dependability and inheritance between the classes. An example of a class diagram can be seen in Figure 3. The Class diagrams (pg. C85-C97) created can be accessed in Appendix C.

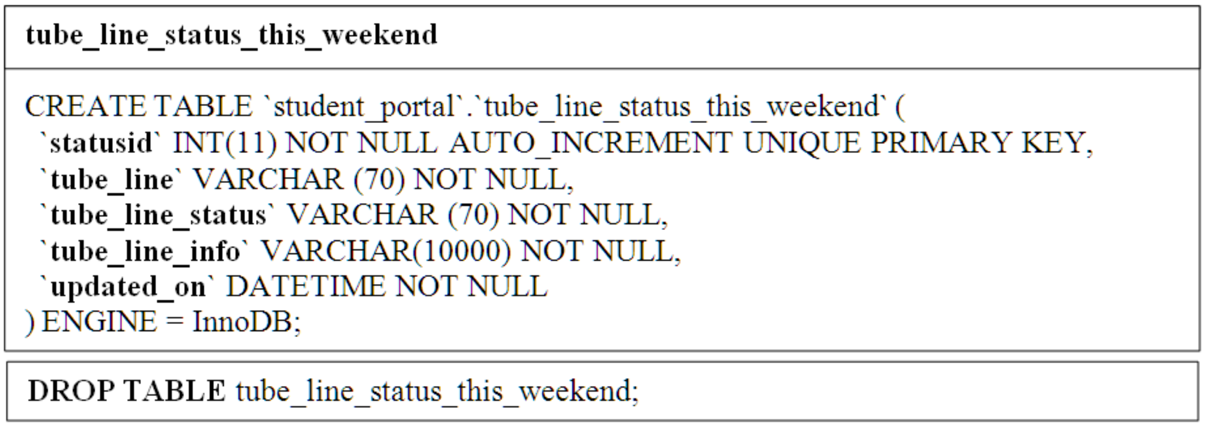


<<use>>
Messenger - any account type
SendMessage
+mysqli
+session_userid : int
+created_on : string
-message_to_userid : int
-message_to_firstname : string
-message_to_surname : string
-message_to_email : string
-message_subject : string
-message_body : string
-executeQuery1(message_subject, message_body, created_on) : void
-executeQuery2(message_from, message_to) : void
-executeQuery3(message_from, message_to, isRead) : void
SetMessageRead
+mysqli
+session_userid : int
-executeQuery1(isRead) : void
DeleteSentMessage
+mysqli
+session_userid : int
-sentMessageToDelete
-executeQuery1(messageid, message_from) : void
DeleteReceivedMessage
+mysqli
+session_userid : int
-receivedMessageToDelete : int
-executeQuery1(messageid, message_to) : void
user_message_sent (db)
-messageid : int
-message_from : int
-message_to : int
+SendMessage() : void
+DeleteSentMessage() : void
user_message (db)
-messageid : int
-message_subject : string
-message_body : string
-message_status : string
-created_on : string
+SendMessage() : void
user_message_received (db)
-messageid : int
-message_from : int
-message_to : int
-isRead : tinyint
+DeleteReceivedMessage() : void
+SendMessage() : void
Visual Paradigm Community Edition

Figure 3. Example of a class diagram for the Messenger area of the application

5.2.2. CREATE and DROP database statements

The next phase following the class diagrams was the database design phase. Before starting the design of the ER diagram, I spent some time analysing both the use case specification and diagrams and the class diagram in order to build the table structure required to serve the functional requirements identified. I created a file entitled *db_statements.sql* and within this file I typed a list of table names. Furthermore, I started to think about the table names with a criterion in mind. The criterion was to not have any words in a plural format within any table name. The reason for the criterion was because in previous experiences, having plurals within table or column names tends to cause a large number of mistakes where the trailing letter of a name is missed out. I did not want to risk making such mistakes, as they are extremely difficult to detect, although easy to solve once detected. Once I finalised the list consisting of the table names, I built upon it by adding the MySQL syntax as well as the table columns, their data types and constraints. An example of a table's CREATE and DROP statements can be seen in Figure 4. The CREATE and DROP database statements (pg. C98-C121) created can be accessed in Appendix C.



tube_line_status_this_weekend

```
CREATE TABLE `student_portal`.`tube_line_status_this_weekend` (
 `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `tube_line` VARCHAR (70) NOT NULL,
 `tube_line_status` VARCHAR (70) NOT NULL,
 `tube_line_info` VARCHAR(10000) NOT NULL,
 `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

```
DROP TABLE tube_line_status_this_weekend;
```

Figure 4. Example of a CREATE and DROP statement for the *tube_line_status_this_weekend* table

5.2.3. ER Diagram

Based on the CREATE database statements, I created an ER diagram. In comparison to the use case and class diagrams, the ER diagram consists of all the tables required for the application in a single diagram. The reason behind it is because a table can serve multiple functions; therefore the number of tables required is not as large as the number of functions. The diagram consists of a number of different entities that represent each table. Each entity consists of several different columns as well as column attributes. The column attributes within the ER diagram were identified while building the CREATE database statements.

Specific columns in a table are linked to other columns in other tables and this creates the relationship between two or more tables. The relationships are of different types such as: one to one, one to many, many to many etc. The relationships between tables are used extensively when writing queries in order to select and display data. For example, there is a table that stores the users of the system called *user_signin* with a column called *userid* and another table that stores the lectures called *system_lecture* with a column called *lectureid*. In order to match a user with a lecture, another table called *user_lecture* was created which contains two columns, *userid* and *lectureid* and has a one-to-many relationship to both the *user_signin* and the *system_lecture* table. In order to select lectures that are allocated to a specific user within the *user_lecture* table, the query would primarily return data from the *system_lecture* table but will also take advantage of the relationship with the *user_signin* table through the *user_lecture* table in order to only return the lectures allocated to that specific user. The ER diagram helped me by providing a visual overview of the logic and flow of the database which then helped me to understand how the application will be implemented and work. The ER diagram also helped me in both designing and creating the database. I often went back to the ER diagram during the Implementation phase in order to check specific table relationships or structures. The ER diagram created can be accessed on page C122 of Appendix C.

5.3. Implementation phase

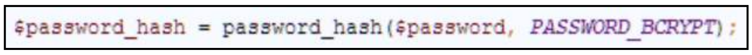
This sub-chapter details the tools I integrated within the application, the challenges I faced and the new skills I learnt throughout the process.

5.3.1. PHP

Prior to the start of the project, I had basic experience in PHP. In order to become accustomed to the way PHP works, I started to search for the documentation that would help me gain the knowledge through a fast-paced learning process. PHP is represented on the web through its own website, which includes a vast document library detailing all its functions and features. Below, I will describe how I used the documentation to implement certain functions and improve previously written code.

password_verify function: I read a large amount of documentation and articles regarding security, focusing specifically on authentication security. This taught me that the recommended security practice regarding passwords is to hash them prior to inserting the data into the database. *"Unlike secret key and public key algorithms, hash functions, also called message digests or one-way encryption, have no key. Instead, a fixed-length hash value is computed based on the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered." (Hash functions, SANS Technology Institute, 2015)*

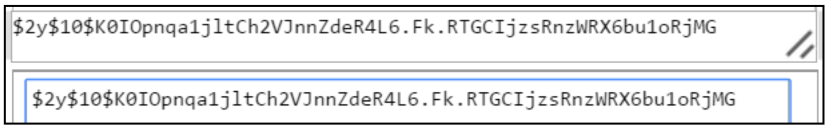
I also learnt that there are numerous encryption techniques that have been broken, and their usage is no longer recommended. The broken encryption algorithms that I found out about are MD5 and sha512. I decided to avoid these, and started to research into encryption algorithms incorporating hashing functions. The research carried out allowed me to find out about bcrypt, which is currently the best and most secure encryption algorithm. *"Besides incorporating a salt to protect against rainbow table attacks, bcrypt is an adaptive function: over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power." (bcrypt, 2015)* Following the discovery of bcrypt, I started to encrypt passwords using the following snippet of code:



```
$password_hash = password_hash($password, PASSWORD_BCRYPT);
```

Figure 5. Example of the encryption process of a password using bcrypt within PHP

The outcome of a plain-text password, "Password1", encrypted with bcrypt looks like this:



```
$2y$10$K0IOpnqa1jltCh2VJnnZdeR4L6.Fk.RTGCIjzsRnzWRX6bu1oRjMG
$2y$10$K0IOpnqa1jltCh2VJnnZdeR4L6.Fk.RTGCIjzsRnzWRX6bu1oRjMG
```

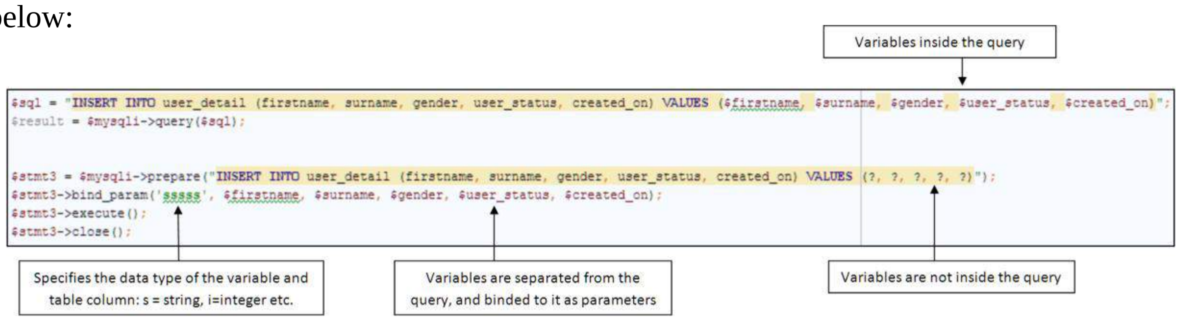
Figure 6. Example of the final outcome of a password "Password1" encrypted using bcrypt

Once the encryption algorithm choice was made, I shifted my attention to the authentication process. The system was required to match the normal password entered by the end-user with its matching hashed counterpart found within the database. Initially, my understanding indicated that gathering the normal password entered by the end-user, encrypting it and then comparing it with the password in the database would work. However, it did not, due to the fact that the hashed output of a password will always be different. This is when I understood how hash functions work. By carrying some further research, I discovered the *password_verify* function which is the other half of the *password_hash* function, and the two are to be used together. The *password_verify* function is able to check and identify a plain text password's matching hashed equivalent efficiently and securely.

prepared statements: While reading about security, I also gained substantial knowledge on MySQL queries. The most important and vulnerable types of queries are the INSERT and UPDATE statements, as they directly manipulate the tables, in comparison to SELECT statements which only return data. The documentation studied recommended using prepared statements when performing queries. A prepared statement can *"be thought of as a kind of* *compiled template for the SQL that an application wants to run, that can be customized using variable parameters. Prepared statements offer two major benefits:*

- *The query only needs to be parsed (or prepared) once, but can be executed multiple times with the same or different parameters. When the query is prepared, the database will analyze, compile and optimize its plan for executing the query.* [...] *By using a prepared statement the application avoids repeating the analyze/compile/optimize cycle. This means that prepared statements use fewer resources and thus run faster*
- *The parameters to prepared statements don't need to be quoted; the driver automatically handles this. If an application exclusively uses prepared statements, the developer can be sure that no SQL injection will occur (however, if other portions of the query are being built up with unescaped input, SQL injection is still possible). (PHP Prepared statements, 2015).*

An example of a normal query and a query using a prepared statement can be seen in Figure 7 below:

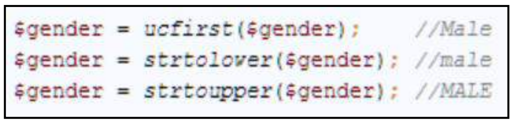


```
$sql = "INSERT INTO user_detail (firstname, surname, gender, user_status, created_on) VALUES ($firstname, $surname, $gender, $user_status, $created_on)";
$result = $mysqli->query($sql);


$stmt3 = $mysqli->prepare("INSERT INTO user_detail (firstname, surname, gender, user_status, created_on) VALUES (?, ?, ?, ?, ?)");
$stmt3->bind_param('sssss', $firstname, $surname, $gender, $user_status, $created_on);
$stmt3->execute();
$stmt3->close();
```

Variables inside the query
Specifies the data type of the variable and table column: s = string, i=integer etc.
Variables are separated from the query, and binded to it as parameters
Variables are not inside the query

Figure 7. Comparison between a normal query and a query using the *prepare* function.

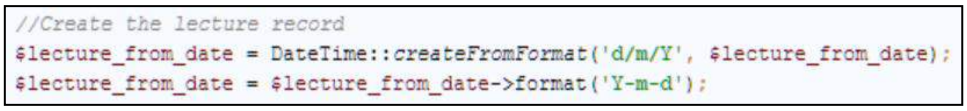
First letter capital, **all capitals, all lowercase**: Similarly to the no plurals in the table names constraint that I introduced, another constraint was to have no capital letters in the database as typing a letter lowercase rather than uppercase would cause the same sort of mistakes as missing the trailing "s" on a word. On the client side, certain words had to be in a first letter uppercase format, and a function I found, called *ucfirst*, enabled me to achieve that. Other scenarios required an all letters capital format and that was achieved by using another function named *strtoupper*. An example of how these functions were used throughout the application can be seen in Figure 8.



```
$gender = ucfirst($gender);    //Male
$gender = strtolower($gender); //male
$gender = strtoupper($gender); //MALE
```

Figure 8. Example of how *ucfirst*, *strtolower* and *strtoupper* functions were used

UK to MySQL date format: One of the obstacles that I had to overcome was controlling how the date was being inputted on the client side and how it was being inserted into the database. This obstacle was created by MySQL's requirement of a specific date format as stated in the manual: *"Although MySQL tries to interpret values in several formats, date parts must always be given in year-month-day order (for example, '98-09-04'), rather than in the month-day-year or day-month-year orders commonly used elsewhere" (MySQL Date and Time types, 2015)* A UK date format is DD/MM/YYYY. This meant that initially the date had to follow a UK format as most visitors to the website would be from the UK. However, when the data had to be inserted into the database, it had to be converted to the format required by MySQL. After some research, I learnt about the *CreateFromFormat* and *format* functions. By using these two functions the date was converted as shown below:



```
//Create the lecture record
$lecture_from_date = DateTime::createFromFormat('d/m/Y', $lecture_from_date);
$lecture_from_date = $lecture_from_date->format('Y-m-d');
```

Figure 9. Example of how the *CreateFromFormat* and *format* functions were used to convert the date to a different format

5.3.2. GitHub/PhpStorm/dploy.io

GitHub provides *"Powerful collaboration, code review, and code management for open source and private projects."(GitHub, 2015)* I acquired a GitHub micro account which allows five private repositories. One of the repositories was used to store this project. By using GitHub, it meant I required an IDE which was compatible with it, in order to continue developing efficiently. After carrying out a research phase, I decided to use PhpStorm. PhpStorm is fully compatible with GitHub and allowed me to perform GitHub-related actions such as Commit, Push, Pull or Merge.

The majority of the development process of the application was structured as follows:

- **Making changes within PhpStorm**: I carried out the development work within the IDE and PhpStorm tracked all the changes made and prepared them for a Commit.
- **Committing to GitHub**: Once enough changes were made to one section of the application, I would perform a Commit action, which created a GitHub commit record, locally. I would then continue to work, and commit again at a later stage.
- **Pushing to the GitHub server**: Usually after the development work was completed, I would push all changes to the version control.

- **Deploying with dploy.io**: After every major development or before the testing phase, I would erase the entire application folder from the web hosting server, and deploy a brand new, fresh version from the GitHub repository. dploy.io is a web-based tool which enabled me to deploy the application directly from the GitHub version control which was efficient and it assured me that the project's version control is working as required.
- **File history and Code compare**: If needed at any time, I was able to view the history of any file by selecting a file, and right clicking on the "Show history" button. A list of versions of that specific file would load. By double clicking on one of the versions, I could compare the code changes that were made on that specific date.

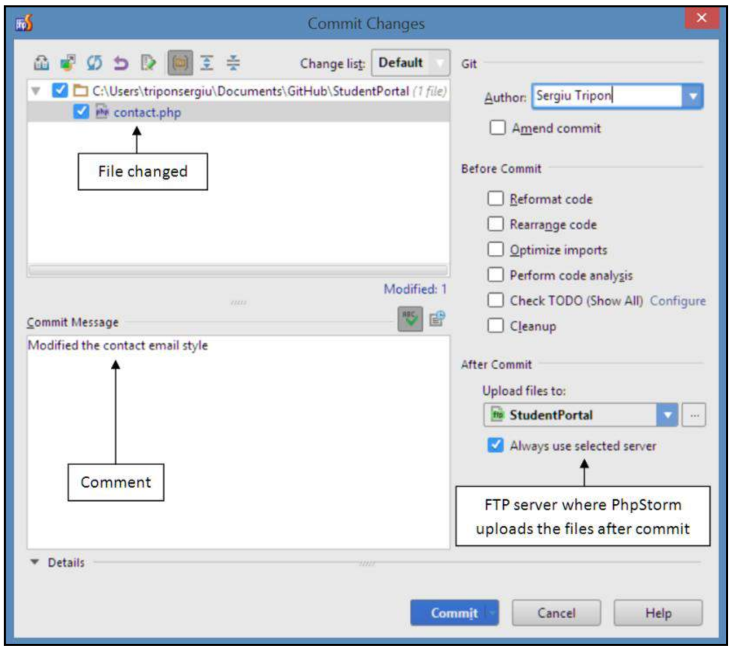

Change list: Default
C:\Users\triponsergiu\Documents\GitHub\StudentPortal (1 file)
✓ contact.php
Modified: 1
Commit Message
Modified the contact email style
Author: Sergiu Tripon
☐ Amend commit
Before Commit
☐ Reformat code
☐ Rearrange code
☐ Optimize imports
☐ Perform code analysis
☐ Check TODO (Show All) Configure
☐ Cleanup
After Commit
Upload files to: StudentPortal
✓ Always use selected server

Figure 10. Example of the Commit process to the GitHub server

5.3.3. External third party tools and APIs

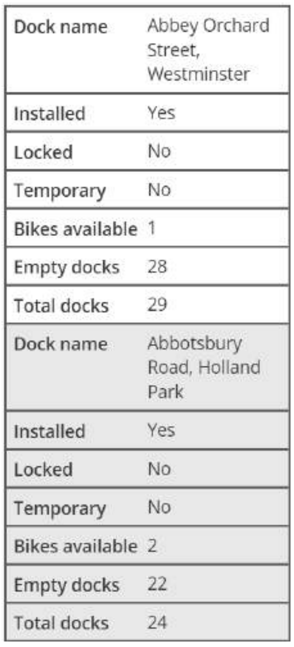
Throughout the implementation, a wide array of third party tools and APIs were used in order to provide or improve certain functionality. Some of the tools and APIs integrated include: Bootstrap, Bootstrap Calendar, DataTables, Google Maps, TFL Transport API, and PayPal. Furthermore, I created a "Re-use of code" document which provides extensive details and explanations of which code was written by me, which code was written by a third party but consists of changes added by me as well as which code was completely written by a third party. The document also includes code comparisons and references. The "Re-use of code" document can be accessed in Appendix G.

5.3.3.1. Bootstrap

Bootstrap was the focal tool behind the responsive concept of the project. Bootstrap is a library of HTML, CSS and JavaScript classes and components. I had previous knowledge of the tool which made it an easy choice. With the exception of a number of other plugins that added or improved certain components such as tables, date time pickers, or dropdowns, the application solely uses Bootstrap's components throughout. Two scenarios where difficulties were faced in achieving responsive design are described below.

Home page: I envisaged how the application's home page will look like aesthetically. The concept was "dashboard-like", similar to the Metro style used within Windows 8, linking the user to the resources available to them based on their account type. Additionally, the home page had to be responsive, so that it would be easily accessible and aesthetically pleasing on any screen size. This proved to be quite difficult since a wide range of screen sizes had to be satisfied. Bootstrap enabled me to overcome this challenge by using the pre-configured grid classes which address and focus on each screen size specifically.

Tables: The table is the application's most used component. Tables can consist of a large number of columns. These columns can also contain lots of information, which increases the column's width. On a standard sized desktop screen, this does not cause any problems. However, when it comes to a mobile's screen, table sizes tend to become a lot more problematic. This issue was solved when I found a snippet of CSS code which turns the normal table with the columns displayed horizontally, into a table with the columns displayed vertically when accessed using a mobile. This way the information displayed is preserved on the smaller screen size. The source of this technique can be accessed in Appendix G. An example of the implementation can be seen in Figure 11.



Dock name	Abbey Orchard Street, Westminster
Installed	Yes
Locked	No
Temporary	No
Bikes available	1
Empty docks	28
Total docks	29
Dock name	Abbotsbury Road, Holland Park
Installed	Yes
Locked	No
Temporary	No
Bikes available	2
Empty docks	22
Total docks	24

Figure 11. Responsive table

5.3.3.2 DataTables

DataTables was another tool that was used to further improve and enhance the user's experience. It is a JavaScript plugin which adds a number of functions to the table component within the application such as allowing columns to be sorted and adding a search box to the table which enables the filtering of data. The tool also allowed me to enable the pager feature which means users can navigate through the different pages displayed. Figure 12 shows how the tool was used to enhance the application’s tables.

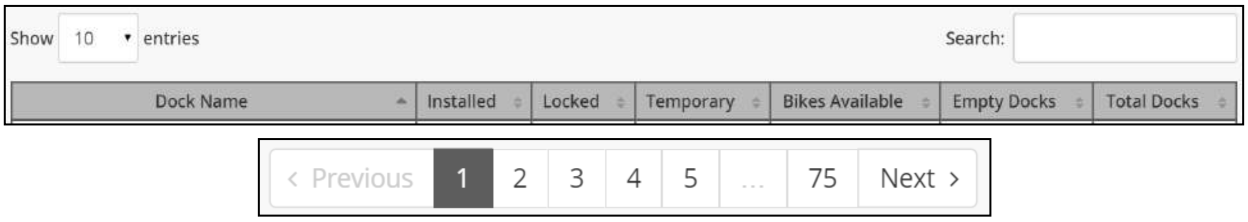


Figure 12. Example of how DataTables was used to add extra functionality to the tables

5.3.3.3 select2

In terms of design, one of the biggest challenges encountered was enforcing the aesthetic consistency of the dropdown component throughout the application. By default, every browser interprets the HTML drop-down component in a specific aesthetic way, which affects the application’s consistency from browser to browser. *“select2 gives you a customizable select box with support for searching, tagging, remote data sets, infinite scrolling, and many other highly used options.” (select2, 2015)* I used the select2 plugin in order to preserve the appearance of the dropdowns regardless of the browser.

5.3.3.4 Bootstrap Date Time Picker

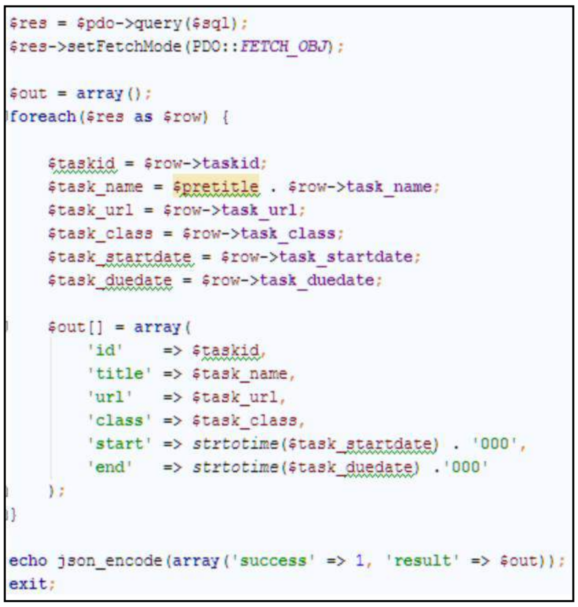
The application consists of several inputs where the user is required to either enter a date and/or time. I found it extremely difficult to find a tool to meet those requirements. After trying several plugins, I found Bootstrap DateTimePicker, which met my requirements of being responsive, compatible with Bootstrap and providing both date and time functions.

5.3.3.5 Bootstrap Calendar

Another important component that is widely used throughout the application is the calendar component. The calendar is used within the Library, Calendar and Events areas. I conducted research in order to find a calendar component which had to be responsive and compatible with Bootstrap and I chose to implement a tool called Bootstrap Calendar. Although I encountered some challenges in displaying data from the database on the calendar, the plugin’s developer provided the necessary documentation for me to overcome this. An example of the code that returns records from the database and binds them to the calendar can be seen in Figure 13.

5.3.3.6 Google Maps API

During the writing process of the PDD, one of the standout features defined was the ability of viewing locations on a map while also being able to search for a location in order to view nearby locations surrounding the university campus.

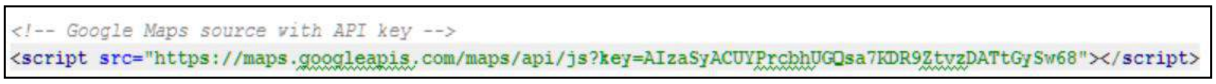


```
$res = $pdo->query($sql);
$res->setFetchMode(PDO::FETCH_OBJ);

$out = array();
foreach($res as $row) {

    $taskid = $row->taskid;
    $task_name = $pretitle . $row->task_name;
    $task_url = $row->task_url;
    $task_class = $row->task_class;
    $task_startdate = $row->task_startdate;
    $task_duedate = $row->task_duedate;

    $out[] = array(
        'id'    => $taskid,
        'title' => $task_name,
        'url'   => $task_url,
        'class' => $task_class,
        'start' => strtotime($task_startdate) . '000',
        'end'   => strtotime($task_duedate) .'000'
    );
}

echo json_encode(array('success' => 1, 'result' => $out));
exit;
```

Figure 13. Example code of how records are binded onto the calendar

I decided to use Google Maps as the map service due to its popularity which means the visitors will not require training. Firstly, I began by reading articles on Google Developers which recommended acquiring a Google Maps API key. I achieved this by creating an account and acquiring the key which was implemented within the system. Secondly, the objective was to be able to load a Google Maps within the application. Thirdly, I discovered a tutorial which detailed the process of displaying data from the database onto the map. The knowledge and experience gained from the tutorial was used to develop the location overview section on the University Map page. Other achievements included: custom icons for markers, filtering markers by category and custom controls which were used to detect the user's current location through the use of HTML5 geolocation. The location search page consists of a search box and a radius drop-down, which are used to calculate and identify nearby university-related locations surrounding the user's search criteria. These locations are stored within the database and are displayed on the map.

Most of the functionality of the University Map area was achieved with the help of the documentation provided by Google due to my limited knowledge of Google Maps development. However, I believe that by implementing a vast amount of Google Maps-related functionality, I have gained significant knowledge and experience in the domain.



```
<!-- Google Maps source with API key -->
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyACUYPrcbhUGQsa7KDR9ZtvzDATtGySw68"></script>
```

Figure 14. Example of the Google Maps API key usage

5.3.3.7 TFL Transport API

The second API integrated within the application was the TFL API. This was integrated within the Transport area where the application displays “Live” and “This Weekend” tube line and station status information, as well as “Live” cycle hire availability. Firstly, I acquired the TFL Transport API access by creating an account on TFL’s developers page. The API provided consists of a number of XML files, which are hosted and updated regularly by the TFL staff and contain various transport information. The next objective was finding out how to read and extract the data from the XML files provided. A research phase was carried out which resulted in the discovery of the *SimpleXML* class. This enabled me to read data from the XML files. By achieving this, the main functionality of the Transport area was complete. Additionally, I did not like the idea of reading files from TFL’s server and relying solely on it, and therefore decided to insert the data into the database prior to displaying it. Having an *updated_on* column as part of the database table meant that the system was able to let the end-user know when the data displayed was last updated. I created a script that would read the data from the XML and insert it into the database. The data that TFL sends differs in size and may not contain some previous data and/or may contain some new data. This made it difficult to update the tables, so instead the script was built to erase the tables, and then re-insert the data. This became an issue because, if the TFL server was down, the tables would get erased without any new data being inserted, which would result in the end-user seeing empty tables. I solved this issue by implementing a URL validation check before performing any other actions. An overview summary of how the final script behaved is as follows:

1. Check if URL is valid
2. If URL is invalid, stop
3. If URL is valid, erase tables
4. Gather new data
5. Insert new data into the tables

In order to complete the functionality of the Transport page, I had to overcome another obstacle. The script updating the transport status was required to be run automatically on a regular basis; otherwise the data would not be updated. I did some further research and learnt about cronjob, which is a tool within the web hosting server that allowed me to set-up a “job” that runs automatically according to the frequency specified. A cronjob can run a multitude of files including PHP scripts. I chose to set-up a cronjob which runs the script every 5 minutes.

5.3.3.8 PayPal

Another integral tool within the application is the PHP PayPal class. PayPal is integrated twice throughout the application allowing users to book and pay for events and students to pay their course fees. In order to achieve the integration, I had to do some research into PayPal and the first step was to make a PayPal developer account which provided a PayPal sandbox environment. A sandbox environment is identical to a usual PayPal environment, except that the money and transactions are artificial. This provided the flexibility required to fully test the application while avoiding setting up bank accounts etc. PayPal provided me with two email addresses, a facilitator and a buyer email address. The facilitator address *"represents the merchant, or receiver, in a transaction" (PayPal developer, 2015)*, while the buyer address *"represents the buyer, or sender, in a transaction" (PayPal developer, 2015).*

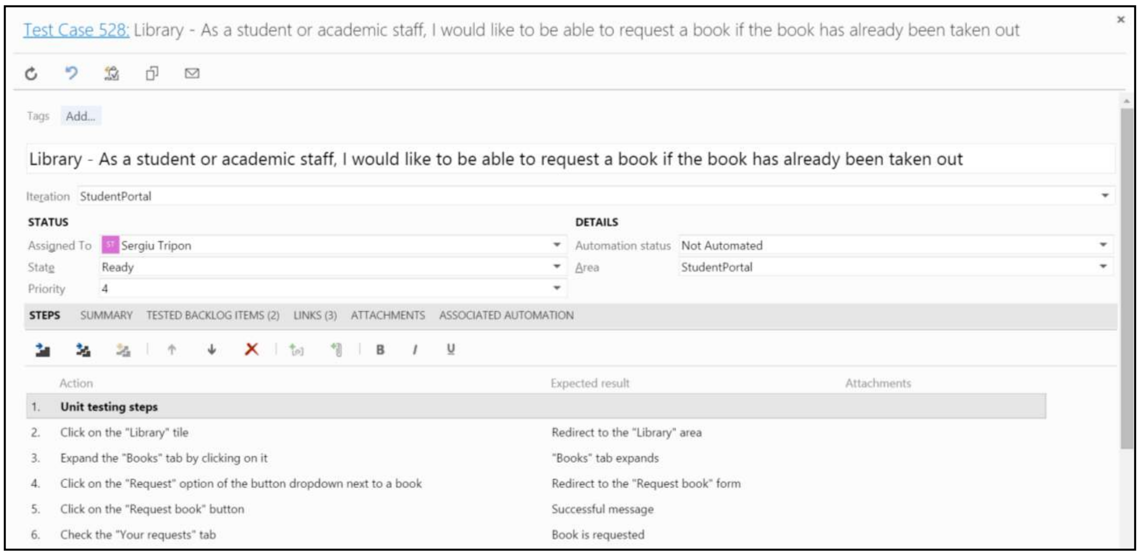
On the PHP side, through the research carried out, I found out that a PHP PayPal class is required, which takes pre-configured data as well as data entered by the end-user and sends it to PayPal. I implemented this class within the application, and added certain alterations to the events occurring before and after a payment is completed or cancelled. The PHP PayPal class required some of the details provided by PayPal such as the facilitator email address in order to identify which PayPal account it needs to redirect the end-user to. The PayPal class also features IPN which *"is a message service that automatically notifies merchants of events related to PayPal transactions. Merchants can use it to automate back-office and administrative functions, like automatically fulfilling orders and providing customers with order status." (PayPal developer, 2015)* I used the IPN to perform the necessary table manipulations in order to complete the payment on the back-end while also notifying the end-user through email notifications.

5.4 Testing phase

Various testing methods were used to test the application such as: Unit, Usability, Cross-browser and Responsive testing. This chapter details the events of each testing phase.

5.4.1 Unit testing

Test cases were created using a GUI form in VSO. The form allowed me to specify a number of important details of a test case such as its state, priority and test steps as well as linking bugs and attachments to it. Test steps provide an indication of the navigations and actions required to perform a test. VSO allowed me to create shared steps for the steps which remained identical for each test case. Actions such as navigating to the website address and signing in were combined together into a shared step which was then added to every test case. Once the test cases had been created, I started to test each test case against the application. Tests would be passed or failed based on whether the action within a specific requirement worked correctly when tested. Bug records were created for failed tests. If the action worked correctly, but I noticed something wrong aesthetically or logically, a bug would still be created and the test case would be failed, as I wanted to record every single imperfection. At the end of the unit testing phase, a document was composed, consisting of print screens of each test case, as well as the test results and resulting bugs. The document also includes tests done after bugs were fixed. The document can be accessed (pg. D1-D62) in Appendix D. An example of a unit test case can be seen in Figure 15 below.


Test Case 528: Library - As a student or academic staff, I would like to be able to request a book if the book has already been taken out

Tags: Add...

Library - As a student or academic staff, I would like to be able to request a book if the book has already been taken out

Iteration: StudentPortal

STATUS
Assigned To: Sergiu Tripon
State: Ready
Priority: 4

DETAILS
Automation status: Not Automated
Area: StudentPortal

STEPS | SUMMARY | TESTED BACKLOG ITEMS (2) | LINKS (3) | ATTACHMENTS | ASSOCIATED AUTOMATION

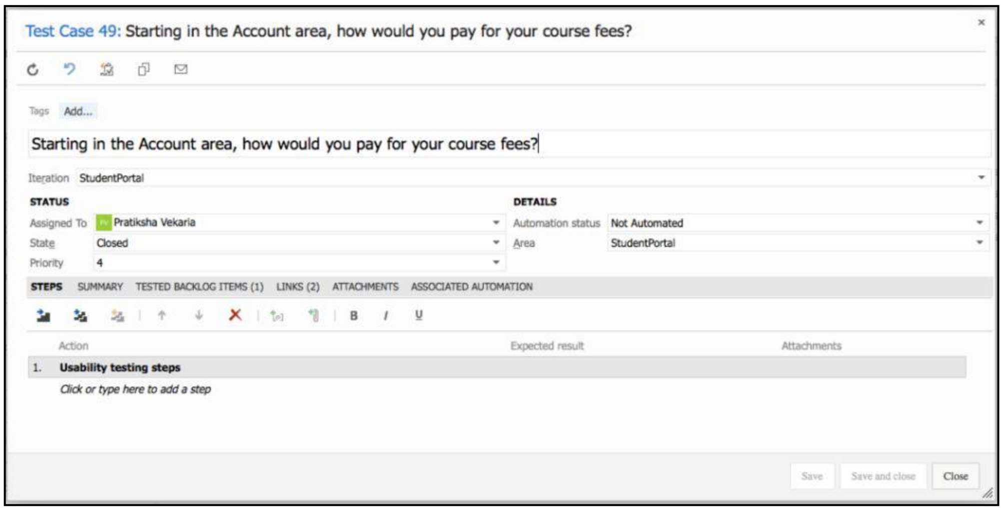
	Action	Expected result	Attachments
1.	**Unit testing steps**		
2.	Click on the "Library" tile	Redirect to the "Library" area	
3.	Expand the "Books" tab by clicking on it	"Books" tab expands	
4.	Click on the "Request" option of the button dropdown next to a book	Redirect to the "Request book" form	
5.	Click on the "Request book" button	Successful message	
6.	Check the "Your requests" tab	Book is requested	

Figure 15. Example of a unit test case for the Library area of the application

5.4.2 Usability testing

After the unit testing phase was completed, the usability testing phase followed. It consisted of a number of test cases which an external end-user tested as tasks. The purpose of a usability test case is to find out the end-user's thought process and logic and how they interact with the application. There's a big difference between how the developer of an application assumes end-users will use it, and how end-users actually use it. I was present at all times during the usability testing phase, which meant that I was able to see all aspects of the end-user interaction such as confusion or facial expressions. The entire usability testing phase was recorded both video and audio. The recording involves a screen and audio recording of the user interaction which would enable me to look back at the usability testing phase and check the feedback provided in the future. The end-user contributed with feedback on aspects of the application that they had issues with while also identifying potential improvements that could be made. Bug records were created for each piece of feedback. In fact, most of the bug records created were enhancements that the end-user felt would improve the application, rather than issues experienced during the testing phase.

The bugs that came out of the usability testing phase were also created within GitHub. GitHub calls "bugs", "issues" and it allowed me to assign "bug" and "enhancement" tags to them. After the usability phase was completed, I analysed the list of issues on GitHub and decided whether they were viable, necessary or significant to the application. As I went through the bug fixing phase, specific issues were updated by either being solved or tagged as "not applicable". Some issues were tagged as "not applicable", as I considered them unnecessary or requiring too much implementation effort, taking into account the length of the project. At the end of the usability testing phase, I created a document consisting of the test cases, results and resulting bugs. Additionally, I organised a meeting with Dr. Stephann Makri requesting usability feedback. He suggested changes such as the renaming of some labels and removal of unnecessary table columns within the GUI. He also recommended making the "Pay course fees" function one of the main areas and naming it "Fees", rather than providing it within the Account area. The Usability testing document can be accessed (pg. D63-D108) in Appendix D. An example of a usability test case can be seen in Figure 16.


Test Case 49: Starting in the Account area, how would you pay for your course fees?

Tags: Add...

Starting in the Account area, how would you pay for your course fees?

Iteration: StudentPortal

STATUS		DETAILS	
Assigned To	Pratiksha Vekaria	Automation status	Not Automated
State	Closed	Area	StudentPortal
Priority	4		

STEPS · SUMMARY · TESTED BACKLOG ITEMS (1) · LINKS (2) · ATTACHMENTS · ASSOCIATED AUTOMATION

	Action	Expected result	Attachments
1.	**Usability testing steps**		
	Click or type here to add a step		

Figure 16. An example of a usability test case for the Account area of the application

5.4.3 Cross-browser compatibility testing

Each page's functionality and GUI within the application was tested against the '*three most used browsers' (Browser market share, 2015)*, and the results were documented and can be accessed (pg. D109-110) in Appendix D.

5.4.4 Responsive testing

In addition to the unit and usability testing phases carried out, I also conducted a Responsive testing phase which consisted of testing each page within the application using a smart phone in order to determine its responsiveness. The results of this testing phase can be accessed (pg. D111-D112) in Appendix D.

5.5 Manuals

At the end of the project, I created a user and deployment manual to be used by the users and administrators of the system.

5.5.1 User manual

The user manual created consists of three documents, one for each account type containing instructions on how to perform actions within the application. The document consists of both text-based and image-based instructions. The reason behind creating separate documents was due to the fact that each account type holds different privileges which dictate what each account type can do. The user manual can be accessed (pg. E1-E239) in Appendix E.

5.5.2 Deployment manual

The deployment manual provides instructions to aid administrators and other developers in deploying the application. The deployment manual can be accessed (pg. E240-D254) in Appendix E.

5.6 Decisions

Throughout the project, a few critical decisions had to be made. Decisions may not always be right, but there is always potential to learn from them in order to impact future decisions. Some of the most important decisions I had to take are detailed in this chapter.

5.6.1 Timetable area

Before: The module, lecture, tutorial and exam database table relationship started out a lot differently to how it ended. Each table related to the Timetable and Exams areas had a table-specific primary key column: *moduleid*, *lectureid*, *tutorialid* and *examid*. The lecture, tutorial and exam tables also had a column *moduleid* which was a foreign key in order to establish the relationship to the module table. In addition to that, a relationship to the user table was required in order to be able to allocate modules, lectures, tutorials and exams to different users with a student account type. The relationship was established by using a lookup table *user_timetable*. The *user_timetable* table had two columns: *userid* and *moduleid,* therefore establishing the relationship between the module and user table. The relationships between the six tables are represented through the ER diagram below. Please note that the unnecessary table columns were removed from the tables to avoid confusion.

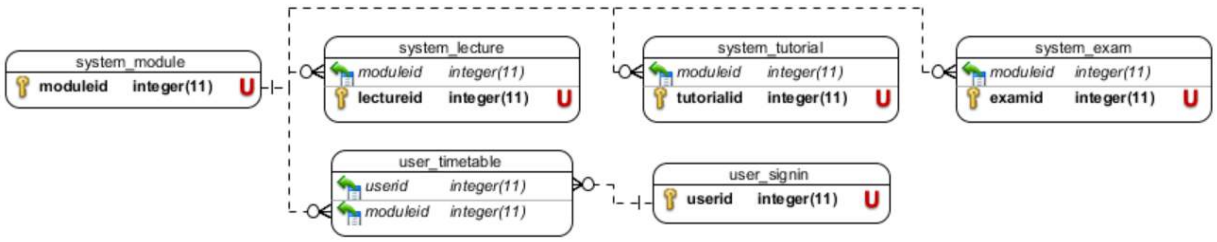


Figure 17. ER diagram representing the initial logic and thought process behind the Timetable area

The initial set-up was flawed because it wouldn't have been able to serve the scenario where more than one lecture, tutorial or exam needed to be allocated to different users. This was because of the fact that the lecture, tutorial and exam tables had no direct relationship to the user table, as the relationship was established solely through the module table. This meant that technically, even though it was a one to many relationship, the relationship between the module table and lecture, tutorial and exam tables was one-to-one.

After: The issue was mitigated by creating separate look-up tables to serve the relationships between the lecture, tutorial and exam tables and the user table. The *user_timetable* table was renamed to *user_module*. The revised structure of the relationships is shown below:

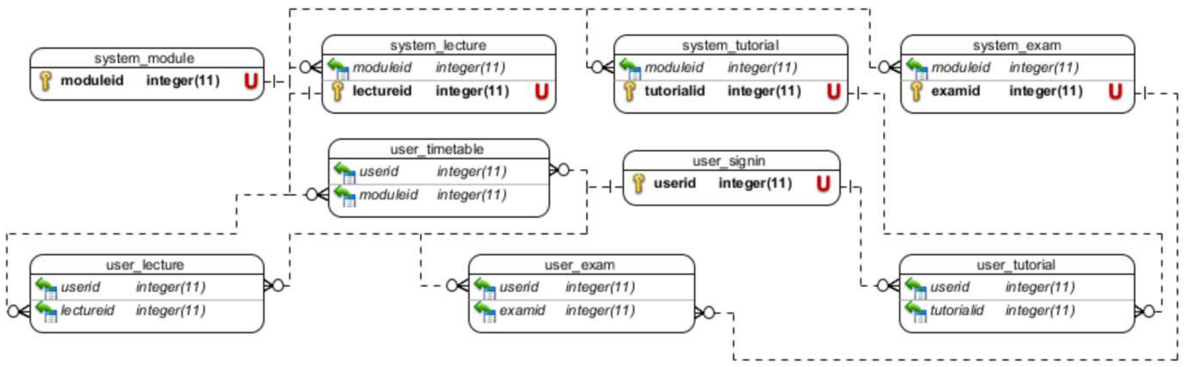


Figure 18. ER diagram representing the new, revised structure of the database back-end

By taking the decision to change the relationship structure, it enabled me to implement administrator-specific functions for creating multiple lectures, tutorials and exams that have a relationship to a single module and are allocated to multiple users as required.

5.6.2 Library area

During the implementation of the Library area, the area's original flow was as follows: an end-user would reserve a book. By reserving a book, the end-user would also loan the book, however the loan part was not defined and the user was not correctly informed during the interaction with the application. A concise summary of the issue would be the lack of a clear difference between the processes of reserving a book, collecting a book, and loaning a book. The problem was solved by creating additional tables to hold reservations, loans and requests and the flow of the process was also altered into a full-fledged library-specific process where the workflow is as follows: the end-user reserves a book, which the administrator marks as collected once it has been physically collected. Once physically collected, the loan period starts. When the book is physically returned, the administrator marks the book as returned and the book becomes available to the other end-users once again. During the loan period of a book, other end-users can request the book. When returned, the end-user that requested the book gains exclusive access to it and a new reservation is set up for them to collect the book.

6. Conclusions and Recommendations

This chapter will look back at the project objectives and determine the overall success of the project as well as detail potential improvements to the existing application. I will also discuss any issues encountered while reflecting on the knowledge acquired throughout the project. Initially, the PDD defined the main objectives and sub-objectives of the project. The objectives remained the same throughout the project. In addition to that, a work plan was also created during the initial stages of the project. This, on the other hand, changed as the project progressed. A revision of the work plan was produced and presented during the first Project Progress Review. The initial estimations were inaccurate due to the fact that I didn't have enough information on how long certain technologies or methods would take to learn. Considering the reasons, the work plan had to be changed before the Project Progress Review. The before and after versions of the work plan can be accessed (pg. H45-H46) in Appendix H. Although there were deviations from the initial work plan, the project was completed at the same time as originally planned.

6.1 Project objectives

Through the delivery of the PDD, a number of project objectives of different priorities were outlined. The objectives are used as the form of measurement in determining the project's success.

6.1.1 Critical priority

"The application will undergo an analysis and design phase based on UML"
"The application will undergo an implementation/testing phase. This will consist of UML diagrams of the system architecture and acceptance/unit/exploratory test cases."

Firstly, during the writing process of the PDD, I defined two critical priority objectives. Both objectives were achieved and completed successfully. I created an Analysis and Design document which consists of the following:

- Use case specification
- Use case diagrams
- Class diagrams
- CREATE and DROP database statements
- ER diagram
- Component diagram
- Deployment diagram

Secondly, through the delivery of the Analysis and Design document, I achieved the first objective and parts of the second objective. In order to achieve the testing aspect of the second objective, I carried out a usability testing phase with an end-user while I also organised a meeting with a university academic, who evaluated the application's interface and provided valuable feedback. Due to the project not being client-based, I consider the usability testing phases to have achieved the acceptance testing aspect of the second objective as the participant fit within the application's target audience. Exploratory testing was performed throughout the project in order to pre-test newly built functionality. *"Exploratory testing means testing an application without a set of tests defined in advance, and without a script of predetermined steps." (MSDN, 2015)* This testing was unrecorded and only incorporated during the Implementation phase, in order to provide me with an efficient indication on whether a specific functionality worked correctly. If the exploratory test passed, the specific functionality would go on to be tested thoroughly during the Unit testing phase.

6.1.2 High priority

"The application will be fully responsive, usable on desktop/laptop, tablet or mobile phone"
"The application will provide the following functionality: pay course fees, check timetable and exam timetable, check exam results, reserve a book, book and pay for events, provide lecture feedback, check university map, message other users etc."

The objectives stated above were considered of High priority. I believe that these objectives were achieved by the end of the project. A mobile responsive web application consisting of the functionality defined in the second objective was developed and a demo was presented to the project supervisor and a second marker. Responsive testing was undertaken in order to determine whether the application's pages are responsive. Additionally, several other features were added to the initial functionality defined at the start of the project. These include: additional account types such as student, academic staff and administrator, transport status information and calendar functionality where a user is able to create personal tasks and view them on a calendar. The functional objectives defined at the start of the project can be accessed in Appendix A. The main obstacle in achieving these objectives was the lack of knowledge in regards to the domains, programming languages and technologies involved. However, the literature review and the constant research undertaken throughout the project enabled me to overcome the challenges while also acquiring substantial knowledge and experience.

6.1.3 Moderate priority

"The application will be cross-browser compatible"

"The application will undergo an evaluation phase by an end-user within the target audience"

"The application will have a user-manual"

I considered three objectives of Moderate priority. All three objectives were achieved by the end of the project. The application was determined as cross-browser compatible by the end of the Cross browser compatibility testing. The application underwent a Usability testing phase carried out by an end-user, which produced a number of outputs such as a Usability testing document and video and audio recordings of the user interaction. Furthermore, three separate user manuals targeting each account type of the application were created. The user manuals guide the users through every action with the aid of both text and images.

6.2 Literature review

Throughout the project, I did a substantial amount of reading. Little to no experience in the main technologies used within the project meant I had to carry out extensive reading in order to acquire this as quickly as possible. One of the focal points that helped me achieve that was the literature read and studied throughout the project. I conducted a significant amount research in order to learn a variety of programming languages, technologies and methodologies. The literature outlined in Chapter 2 of this report enabled me to improve my understanding of programming languages and tools such as PHP, Bootstrap, GitHub, Google Maps etc. Furthermore, through the literature studied I also gained experience in designing multi-device experiences as well as implementing preventions against security threats.

6.3 Research

Moreover, I performed research in terms of the suitable software, tools, plugins and methodologies that should be used throughout the project. In terms of software development methodologies, I studied, researched and analysed Waterfall, Agile and Scrum and decided to use a combination of Waterfall and Agile. In regards to building a responsive application, I read a number of articles and researched a number of tools that would help to achieve the objectives identified, and I decided to use Bootstrap. In addition to that, I also wanted to know whether there were other applications available that provided similar functionality to the application developed as part of this project. I found that there are a number of other similar applications in the market. However, none of the other applications' array of functionality compared to the application developed throughout this project.

I also underwent substantial research when I decided to use an IDE instead of Notepad++ due to the transition to GitHub and version control. Several software tools were trialled such as: Sublime Text, Dreamweaver, NetBeans, PhpStorm and others. In the end, I chose PhpStorm as it met all the requirements such as GitHub and FTP compatibility. I believe that without using and transitioning to PhpStorm, I wouldn't have been able to develop the application as efficiently while also maintaining a high standard of development.

6.4 Reflections on the overall project

If in the future I were to be given the opportunity to conduct and manage a similar project again, I would make a few adjustments, specifically to the planning of the project, which I would spend more time on. Also, more time would be spent on the analysis of the work packages and the work plan. This would have led to fewer changes being made during the project. I would also make additions to the Analysis and Design document, such as a state machine diagram, activity diagram or sequence diagram. Moreover, I also believe that conducting the usability testing phase with more than one end-user would help me in terms of better understanding how to provide a well designed user experience.

6.5 Potential future improvements

During the usability testing phase, the end-user suggested an enhancement to the Messenger area of the application involving the ability to attach files to messages. I chose not to implement this, as the development effort required was considered too much at that stage in the project. Other improvements to the current application could be:

- Bus transport updates within the Transport area
- Indoor maps within the University Map area
- Add to Google Calendar option within the Calendar and Events areas

6.6 Personal assessment

I believe this project was extremely beneficial from both an academic and personal perspective. I wanted this project to be as challenging as possible and I think this is represented through the complexity of the project. Completing such a wide range of tasks, while incorporating multiple tools, integrations, and methodologies helped me to discover new abilities and skills. It also gave me a new sense of belief and confidence. I believe that the knowledge and experience acquired throughout the project will aid future work, whether academic or professional. From a technical perspective, my skills and experience in programming languages such as PHP, MySQL or jQuery have greatly improved. At the beginning of the project, my knowledge of the domain was basic; however by the end of it, my understanding of the domain reached new heights. Overall, I consider this project to be my most complex and difficult. In addition to this, having the opportunity to make use of the knowledge acquired from other modules such as HCI during the Usability testing phase was extremely satisfying. I consider the knowledge and experience acquired during the project truly irreplaceable and I am pleased to have taken the decision of making the project objectives challenging and complex as it ensured focus, hard-work and determination.

6.7 Conclusion

In conclusion, I believe that this project was successful since the project objectives have been completed and met accordingly. In addition to the completion of the project objectives, the Final Project Progress Review and the Unit testing and Usability testing phases are the basis used in order to justify and determine the success of the project. Furthermore, I believe that I have gained invaluable knowledge and experience throughout the project and I hope that the application will be used, re-used or will go on to inspire future developments.

7. Glossary

Acronym	Full form
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
RDBMS	Relational Database Management System
PHP	PHP: Hypertext Preprocessor
FTP	File Transfer Protocol
VSO	Visual Studio Online
ER diagram	Entity Relationship Diagram
HCI	Human Computer Interaction
SSL	Secure Socket Layer
IDE	Integrated Development Environment
API	Application Programming Interface
TFL	Transport for London
UCL	University College London
XML	Extensive Markup Language
MD5	Message-digest algorithm
sha512	Cryptographic hash function
PDD	Project Definition Document
IPN	Instant Payment Notification
GUI	Graphical User Interface
BT	British Telecommunications

8. Web application details

URL: https://student-portal.co.uk

Login details

Email: admin@student-portal.co.uk
Password: Password1
Account type: Administrator

The Administrator account type is able to create and manage all required entities in order to use or test the application.

The user manual provided explains using text and images how to perform every action available within the system.

The deployment manual provided explains using text and images the process of deploying the application.

9. Appendices

Appendix	Contents	Submission type
A	Project Definition Document	Online on Moodle
B	Requirements document	Online on Moodle
C	Analysis and design document • Use case specification • Use case diagrams • Class diagrams • CREATE and DROP database statements • ER Diagram • Component diagram • Deployment diagram	Online on Moodle
D	Testing document • Unit testing • Usability testing • Responsive testing • Cross-browser compatibility testing	Online on Moodle
	Usability testing video and audio recordings	On media (USB)
	Audio recording of meeting with Dr. Stephann Makri from Centre of HCI design	On media (USB)
E	Manuals • User manual • Deployment manual	Online on Moodle
F	Web Application - source code	On media (USB)
G	Re-use of code document	Online on Moodle
H	Project Communication document	Online on Moodle

10. References

The following is a list of references used throughout the Project Report:

1. Levin, M.L., 2014. Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices. 1st ed. US: O'Reilly Media.

2. Bootstrap · The world's most popular mobile-first and responsive front-end framework. . 2015. Bootstrap · The world's most popular mobile-first and responsive front-end framework. . [ONLINE] Available at: http://getbootstrap.com/. [Accessed 30 April 2015].

3. Features · GitHub. 2015. Features · GitHub. [ONLINE] Available at: https://github.com/features. [Accessed 30 April 2015].

4. PHP IDE :: JetBrains PhpStorm. 2015. PHP IDE :: JetBrains PhpStorm. [ONLINE] Available at: https://www.jetbrains.com/phpstorm/?fromMenu. [Accessed 30 April 2015].

5. PHP Security Cheat Sheet - OWASP. 2015. PHP Security Cheat Sheet - OWASP. [ONLINE] Available at: https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet. [Accessed 30 April 2015].

6. Google Developers. 2015. Google Developers. [ONLINE] Available at: https://developers.google.com/maps/. [Accessed 30 April 2015].

7. Using PHP/MySQL with Google Maps - Google Maps API — Google Developers. 2015. Using PHP/MySQL with Google Maps - Google Maps API — Google Developers. [ONLINE] Available at: https://developers.google.com/maps/articles/phpsqlajax_v3. [Accessed 30 April 2015].

8. Creating a Store Locator with PHP, MySQL & Google Maps - Google Maps API — Google Developers. 2015. Creating a Store Locator with PHP, MySQL & Google Maps - Google Maps API — Google Developers. [ONLINE] Available at:

https://developers.google.com/maps/articles/phpsqlsearch_v3. [Accessed 30 April 2015].

9. DataTables | Table plug-in for jQuery. 2015. DataTables | Table plug-in for jQuery. [ONLINE] Available at: https://www.datatables.net/. [Accessed 30 April 2015].

10. MySQL :: Why Move to MySQL from Microsoft SQL Server?. 2015. MySQL :: Why Move to MySQL from Microsoft SQL Server?. [ONLINE] Available at: https://dev.mysql.com/tech-resources/articles/move_from_microsoft_SQL_Server.html. [Accessed 30 April 2015].

11. PHP: Hypertext Preprocessor. 2015. PHP: Hypertext Preprocessor. [ONLINE] Available at: http://php.net/. [Accessed 30 April 2015].

12. JQuery vs. JavaScript: What's the Difference Anyway?. 2015. JQuery vs. JavaScript: What's the Difference Anyway?. [ONLINE] Available at: https://blog.udemy.com/jquery-vs-javascript/. [Accessed 30 April 2015].

13. Testing tools| Visual Studio . 2015. Testing tools| Visual Studio . [ONLINE] Available at: https://www.visualstudio.com/features/testing-tools-vs. [Accessed 30 April 2015].

14. Browser market share. 2015. Browser market share. [ONLINE] Available at: https://www.netmarketshare.com/browser-market-share.aspx?qprid=2&qpcustomd=0. [Accessed 30 April 2015].

15. Example: Use-Case Specification. 2015. Example: Use-Case Specification. [ONLINE] Available at: http://epf.eclipse.org/wikis/openup/core.tech.common.extend_supp/guidances/examples/use_case_spec_CD5DD9B1.html. [Accessed 01 May 2015].

16. Hash Functions. 2015. Hash Functions. [ONLINE] Available at: http://www.sans.edu/research/security-laboratory/article/hash-functions. [Accessed 01 May 2015].

17. bcrypt.js/README.md at master · dcodeIO/bcrypt.js · GitHub. 2015. bcrypt.js/README.md at master · dcodeIO/bcrypt.js · GitHub. [ONLINE] Available at: https://github.com/dcodeIO/bcrypt.js/blob/master/README.md. [Accessed 01 May 2015].

18. PHP: Prepared statements and stored procedures - Manual . 2015. PHP: Prepared statements and stored procedures - Manual . [ONLINE] Available at: http://php.net/manual/en/pdo.prepared-statements.php. [Accessed 01 May 2015].

19. MySQL :: MySQL 5.7 Reference Manual :: 11.3 Date and Time Types. 2015. MySQL :: MySQL 5.7 Reference Manual :: 11.3 Date and Time Types. [ONLINE] Available at: http://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html. [Accessed 01 May 2015].

20. Select2 - The jQuery replacement for select boxes . 2015. Select2 - The jQuery replacement for select boxes . [ONLINE] Available at: https://select2.github.io/. [Accessed 01 May 2015].

21. About Sandbox Accounts - PayPal Developer. 2015. About Sandbox Accounts - PayPal Developer. [ONLINE] Available at: https://developer.paypal.com/docs/classic/lifecycle/sb_about-accounts/. [Accessed 01 May 2015].

22. Performing Exploratory Testing Using Microsoft Test Manager. 2015. Performing Exploratory Testing Using Microsoft Test Manager. [ONLINE] Available at: https://msdn.microsoft.com/en-us/library/vstudio/hh191621%28v=vs.110%29.aspx. [Accessed 04 May 2015].

The following is a list of references that were used throughout the life cycle of the report:

1. Levin, M.L., 2014. Designing Multi-Device Experiences: An Ecosystem Approach to User Experiences across Devices. 1st ed. US: O'Reilly Media.
2. PhpStorm 8.0.2 Help :: PhpStorm. 2015. PhpStorm 8.0.2 Help :: PhpStorm. [ONLINE] Available at: https://www.jetbrains.com/phpstorm/help/phpstorm.html. [Accessed 19 March 2015].
3. PHP: PHP Manual - Manual . 2015. PHP: PHP Manual - Manual . [ONLINE] Available at: http://php.net/manual/en/index.php. [Accessed 13 February 2015].
4. CSS · Bootstrap . 2015. CSS · Bootstrap . [ONLINE] Available at: http://getbootstrap.com/css/. [Accessed 19 December 2014].
5. Components · Bootstrap . 2015. Components · Bootstrap . [ONLINE] Available at: http://getbootstrap.com/components/. [Accessed 19 December 2014].
6. JavaScript · Bootstrap . 2015. JavaScript · Bootstrap . [ONLINE] Available at: http://getbootstrap.com/javascript/. [Accessed 19 December 2014].
7. GitHub Help - GitHub Enterprise Documentation . 2015. GitHub Help - GitHub Enterprise Documentation . [ONLINE] Available at: https://help.github.com/. [Accessed 7 January 2015].
8. Font Awesome Icons. 2015. Font Awesome Icons. [ONLINE] Available at: http://fontawesome.io/icons/. [Accessed 19 December 2014].
9. Manual. 2015. Manual. [ONLINE] Available at: https://www.datatables.net/manual/index. [Accessed 18 December 2014].
10. Examples - Select2 . 2015. Examples - Select2 . [ONLINE] Available at: https://select2.github.io/examples.html. [Accessed 20 January 2015].
11. Getting Started - Google Maps JavaScript API v3 — Google Developers. 2015. Getting

Started - Google Maps JavaScript API v3 — Google Developers. [ONLINE] Available at: https://developers.google.com/maps/documentation/javascript/tutorial. [Accessed 06 February 2015].

12. Using PHP/MySQL with Google Maps - Google Maps API — Google Developers. 2015. Using PHP/MySQL with Google Maps - Google Maps API — Google Developers. [ONLINE] Available at: https://developers.google.com/maps/articles/phpsqlajax_v3. [Accessed 06 February 2015].
13. Creating a Store Locator with PHP, MySQL & Google Maps - Google Maps API — Google Developers. 2015. Creating a Store Locator with PHP, MySQL & Google Maps - Google Maps API — Google Developers. [ONLINE] Available at: https://developers.google.com/maps/articles/phpsqlsearch_v3. [Accessed 06 February 2015].
14. Open data users - Transport for London. 2015. Open data users - Transport for London. [ONLINE] Available at: http://www.tfl.gov.uk/info-for/open-data-users/. [Accessed 05 September 2014].
15. Home | PayPal Developer. 2015. Home | PayPal Developer. [ONLINE] Available at: https://developer.paypal.com/. [Accessed 17 December 2014].
16. PHP Security Cheat Sheet - OWASP. 2015. PHP Security Cheat Sheet - OWASP. [ONLINE] Available at: https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet. [Accessed 20 December 2014].
17. Simple PayPal integration code using PHP - Ready to use script. 2015. Simple PayPal integration code using PHP - Ready to use script. [ONLINE] Available at: http://www.asif18.com/15/php/simple-paypal-integration-code-using-php---ready-to-use-script/. [Accessed 14 December 2014].
18. Serhioromano/bootstrap-calendar · GitHub. 2015. Serhioromano/bootstrap-calendar · GitHub. [ONLINE] Available at: https://github.com/Serhioromano/bootstrap-calendar. [Accessed 06 May 2015].

19. dploy.io – Continuous deployment for everyone. 2015. dploy.io – Continuous deployment for everyone. [ONLINE] Available at: http://dploy.io/. [Accessed 19 February 2015].