

School of Electronic Engineering
and Computer Science

Final Report

Programme of study:

BSc Computer Science with
Business Management and
Accounting

Project Title:

A tool to retrieve and browse
templates of marketing material to
help the worldwide charity sector

Supervisor:

Dr. Stefan Poslad

Student Name:

Marie Desiree Cruz
100005521

Abstract

The project was designed to improve the existing practices of Br&.ish through the implementation of a database system and a back end user interface. The database system will serve as a tool to retrieve and store all the details of their marketing material templates; whereas, the back end user interface will serve as a way to categorise the templates into different categories and also define the template's information. The main challenge for this project was to produce a final working system that Br&.ish would want to use; so, usability has been identified as an important area to cover.

To meet the project's aims, background research was performed to see what tools and programming languages will be needed to be employed. Similar systems that provide marketing material templates have also been looked at and evaluated in order to identify any limitations. Such limitations include not having multiple levels of category and subcategory relationships as well as not allowing their users to search templates from a selection of predefined colours. Limitations of these existing systems have been addressed in this project as a way of providing additional benefits to Br&.ish.

This report provided a detailed requirement analysis in order to identify the complete set of requirements from the client. Since the iterative and incremental model was favoured as the software development methodology for this project, the project was divided into smaller increments. In each increment, Br&.ish was actively involved to the design and implementation by providing feedback and suggestions. A major section of this report discussed the design and the implementation of the system's different functionalities, with any issues that were encountered being highlighted. Afterwards, the system was thoroughly tested to check its robustness and to identify any errors. Part of the testing performed was to test the usability of the system; hence, a questionnaire was sent to Br&.ish to obtain their feedback.

The results from the different testing performed, the evaluation of the requirements and the result that Br&.ish provided in the questionnaire show that the project has been successful; although, further improvements can still be made. In conclusion, this project has provided a way to allow Br&.ish to store their different marketing material templates, which can be browsed by the worldwide charity sector.

Table of Contents

1. Introduction	1
1.1 Project Aims	1
1.2 Motivation and Challenges.....	1
1.3 Structure of the Report	2
2. Background	2
2.1 Br&.ish.....	2
2.2 Br&.ish's Existing Practice and Problems with their Current System	2
2.3 Proposal of New System	3
3. Literature Review	4
3.1 Database Management Systems (DBMS).....	4
3.1.1 Relational DBMS (RDBMS) model	4
3.1.2 Different Relational DBMS Products	4
3.1.3 Evaluation of RDBMS Products	5
3.2 Server Side Scripting Languages	5
3.2.1 Perl	5
3.2.2 Python	6
3.2.3 PHP	6
3.2.4 Evaluation of Server Side Scripting Languages	6
3.3 Client Side Scripting Languages	6
3.3.1 HTML5	6
3.3.2 CSS.....	7
3.3.3 JavaScript and JQuery	7
3.3.4 VBScript	7
3.3.5 Evaluation of Client Side Scripting Languages	7
3.4 Existing Work.....	7
3.4.1 Front End Systems.....	7
3.4.2 Back End Systems.....	8
3.4.3 Critical Review of Existing Works.....	9
3.5 Conclusion	10
4. Method	10
4.1 Software Development Methodology	10
4.2 Incremental Development Model	10
4.3 Iterative Development Model	11

4.4 Evaluation of Iterative and Incremental Models	11
5. Project Analysis	12
5.1 Requirements Gathering.....	12
5.1.1 Online Client Meetings (Teleconference calls)	12
5.1.2 Email Correspondence	12
5.1.3 Questionnaires.....	12
5.2 Requirements Specification.....	13
5.2.1 Use Case Diagram	13
5.2.2 Functional Requirements ('Must Have' Requirements)	14
5.2.3 Database Requirements	14
5.2.4 Non-functional Requirements	15
6. Design	16
6.1 Framework Design Overview	16
6.2 Increment 1	16
6.2.1 ER Diagram	16
6.2.2 Mapping of ER Diagram to Tables and Normalisation	18
6.2.3 Design Decisions	18
6.3 HCI Design Principles.....	19
6.4 Increment 2	19
6.4.1 Categorising Templates View	19
6.4.2 Design Decisions	20
6.5 Increment 3	20
6.5.1 Design Decisions	20
6.6 Increment 4	21
6.6.1 Improved Categorising Templates View.....	21
6.6.2 Improved Editing Templates View	21
6.6.3 Template Details View.....	21
6.6.4 Design Decisions	22
7. Implementation.....	22
7.1 Increment 1	22
7.1.1 Web Server Access	22
7.1.2 Connecting to the Database	23
7.1.3 Creating the Database Tables	23
7.2 Increment 2	23

7.2.1 Layout Structure	24
7.2.2 Implementation of Design	24
7.2.3 Adding New Categories	24
7.3 Increment 3	25
7.3.1 Improved Implementation for Categorising Functionality	25
7.3.2 Displaying Template's Details	28
7.3.3 Editing Templates Functionality (Editing Categories)	28
7.4 Increment 4	28
7.4.1 Implementation of Design Changes	28
7.4.2 Changes in the Database	29
7.4.3 Additional Feedback in Adding New Categories	29
7.4.4 Deleting Categories Functionality	30
7.4.5 Improved Editing Templates Functionality	30
7.4.6 View Details Functionality	32
7.5 Increment 5	32
7.5.1 Colour Picker Functionality	32
7.5.2 Design Implementation of Login and Logout Functionality	33
7.5.3 Edit Parameters Functionality	33
7.5.4 Final Changes in the Database	34
8. Testing	34
8.1 Increment 1 and 2	35
8.2 Increment 3	35
8.3.1 Unit Testing	35
8.3.2 Functional Testing and Integration Testing	36
8.3.3 Usability Testing	36
8.4 Increment 4	36
8.4.1 Unit Testing	36
8.4.2 Functional Testing and Integration Testing	37
8.4.3 Browser Compatibility Testing	37
8.4.4 Usability Testing	37
8.5 Increment 5	38
8.5.1 W3C HTML Validation Testing	38
8.5.2 Unit Testing	38
8.5.3 Functional Testing and Integration Testing	39

9. Integration	39
9.1 Integration Overview	39
9.2 Integration Structure	39
9.3 Integration Plan	39
9.4 Integration Issues.....	40
9.5 Best Practice for Integration	40
10. Discussion.....	40
10.1 Project Evaluation.....	40
10.1.1 Functional Requirements	41
10.1.2 Database Requirements	42
10.1.3 Non-Functional Requirements.....	42
10.2 Project Achievements	43
10.2.1 Personal Achievements	43
10.2.2 Project Achievements	43
10.3 Novelty	44
10.3.1 Comparison to Existing Works	44
10.4 Limitation	45
10.5 Further Work	45
10.5.1 Design	45
10.5.2 Drag and Drop Functionality.....	45
10.5.3 Automatic Selection of Parent Categories.....	45
10.5.4 Responsive Web Design	46
10.6 Final Feedback	46
10.7 Conclusion	46
Bibliography	48
Appendix A – Complete ER Diagram.....	51
Appendix B – ER Diagram Mapping and Normalisation	52
Appendix C – User Interface Screenshots.....	54
Appendix D – Testing Results.....	55
Increment 3 Unit Testing	55
Increment 4 Unit Testing	56
Increment 5 Unit Testing	57
Appendix E – Integration Diagram	58

Figure 1: Use Case Diagram for Br&.ish's Back End System (Project 2)	13
Figure 2: ER Diagram for Br&.ish's system.....	17
Figure 3: Wireframe Layout Structure	19
Figure 4: New Wireframe Layout Structure (Categorising templates view)	21
Figure 5: New Wireframe Layout Structure (Edit templates view)	21
Figure 6: Connecting to the Database	23
Figure 7: Creating the Customer Table.....	23
Figure 8: Basic Structure of the Back End User Interface	24
Figure 9: Displaying and Hiding the div for the Subcategory Functionality.....	25
Figure 10: Auto-complete feature for adding a category name	25
Figure 11: Validation for adding a new category	26
Figure 12: Alert box message for existing category names	26
Figure 13: User Input Sanitation	26
Figure 14: Displaying the categories hierarchically	27
Figure 15: Function for displaying the template's details and image using AJAX.....	28
Figure 16: Updating the number of characters left in adding a new category	29
Figure 17: Deleting categories functionality	30
Figure 18: Add keywords functionality.....	30
Figure 19: Alert box messages for removing keywords functionality	31
Figure 20: Editing categories functionality	31
Figure 21: Deleting templates functionality	32
Figure 22: Viewing the template's details	32
Figure 23: Colour picker functionality	33
Figure 24: Logout functionality	33
Figure 25: Edit parameters link	34
Figure 26: ER Diagram	51
Figure 27: Relational Database Model	52
Figure 28: Initial Layout for the complete categorising functionality	54
Figure 29: Initial Layout for the edit categorising functionality	54
Figure 30: Final Layout for the categorising template functionality	54
Figure 31: Final Layout for the edit categories functionality	54
Figure 32: Integration Diagram	58

1. Introduction

This section presents the aims of this final year project. In addition, the motivation and challenges in doing such a project as well as an overview of the structure of the whole report will be presented.

1.1 Project Aims

The primary aim of this project is to design and implement a fully operational database system for a design company called Br&.ish within six months in order for the worldwide charity sector to browse different marketing material templates. The marketing material templates to be browsed by the charity sector are posters, brochures, flyers, newsletters and booklets. In effect, the database system will store different information such as user's details, the marketing material templates and the category it belongs too as well as its different parameters. Parameters include the size of the template, its page orientation whether it is a landscape or portrait, its main colours, the font style, font size and font colour used.

The project's sub-aims are also as follows:

- To design a user interface system for the Br&.ish operators in order for them to categorise marketing material templates into different categories. The Br&.ish operators are the back end users who are responsible in displaying media content to the Br&.ish website.
- To design a user interface system for the Br&.ish operators that will allow them to define or edit information about the marketing material templates. This information includes adding keywords to a template, removing keywords from a template, removing the template from added categories or deleting the template itself.
- To integrate this project into the other projects proposed by Br&.ish. These other projects are further explained in section 2.3 of this report. This system integration will be vital in the deployment of a whole working system that Br&.ish will want to use.

1.2 Motivation and Challenges

Nowadays, different companies and non-profit organisations often employ a branding strategy that best represents what they have to offer to their customers. According to (Harker, 2011), *"In 2007/08, there were 171,074 general registered charities in the UK."* With this high number of general registered charities in the UK, branding and identity design, therefore, are an important factor to survive in this environment (Khan & Ede, 2009). However, small and medium-sized organisations often see branding as an additional cost and they lack resources to employ a branding professional that will help them design their marketing materials such as brochures, leaflets and business cards (Brand Channel, 2003). Br&.ish has used this opportunity to provide the marketing material resources for the worldwide charity sector to use. Therefore, the completion of this project will provide benefits not only to the company but also to their charity clients. The existing facts have motivated the author of this report to deliver a system that will be used by different charities worldwide.

The main challenge in doing this project will be creating a final working system that Br&.ish will want to use. Learning new and advanced techniques in JQuery, PHP and MySQL is also another challenge as this project will need to have a certain level of complexity and depth. Lastly, time management

and good project management planning are also big challenges in delivering the project on time as there will also be other coursework deadlines to be submitted apart from this final project.

1.3 Structure of the Report

The report will be divided into different sections. The introduction is covered in section one. Section two covers brief background information about Br&.ish, an overview about their existing system and the reasons for the need of a new system. Section three covers the literature review for this report and discusses the possible solutions of implementing this project. It also covers a review of the existing systems that are similar to this project, with its limitations being highlighted. Section four provides an overview about the chosen software development methodology that will be followed for this project. Section five contains the different requirement gathering methods that will be used for this project as well as the requirements analysis performed in order to get a complete set of functional and non-functional requirements of this project. Section six and seven discusses the different design decisions and implementation stage for this project respectively. Section eight discusses the different types of testing performed in this project. Section nine provides an explanation of the integration plan and any integration issues encountered for this project and finally, section ten provides the overall evaluation for this project.

2. Background

This section provides a brief overview about Br&.ish and their organisation. In order to have a clear understanding of the reason behind the propose development of this new system, the company's existing practice will also be presented here.

2.1 Br&.ish

Br&.ish is a multidisciplinary design company (Br&.ish, 2012) which is founded in 2010 by Elliot Cowan. He is also the Creative Director of the company. It currently employs six employees and their UK office is located at West Hampstead. Br&.ish operates globally as they also have a sister company in the USA called "*Here's My Chance*". Apart from Elliot, Br&.ish has a Chief Executive Officer, Chief Operating Officer, Intern Manager, Interns and a wide group of freelance artists who are responsible in the creative areas such as graphic designing, web designing, video production and post production. Br&.ish specialises in providing branding materials such as flyers, posters, newsletters, brochures and newsletters to Jewish non-profit organisations around the world. Apart from branding materials, the company also offers other services such as providing fundraising materials, designing clothes as well as creating logos and websites (Br&.ish, 2012). They provide their services to companies who do not have enough resources, such as money, time and people, to make their branding materials.

2.2 Br&.ish's Existing Practice and Problems with their Current System

Currently, Br&.ish offers the following type of marketing materials: Brochures, Flyers, Posters, Newsletters, Business Cards and Booklets. The templates are also provided into different sizes such as: A1, A2, A3, A4, A5, A6, A7, 5x3, 4x6, 8.5x11 inches, 13x19 inches, 17x22 inches, 22x34 inches and 34x44 inches.

At present, the designers of Br&.ish are designing marketing material templates on a customer-to-customer basis. Different designers worldwide send their designs through email, which are

individually checked by Elliot, being the Creative Director of the company. The marketing material templates that were sent contain dummy images and texts so the team at Br&.ish will be able to see how the template will actually look like. If Elliot likes the template, he will ask for only the background image of the template from the designer and his team at Br&.ish can then edit the acquired template according to what their clients want. The customised templates are sent to their customers who represent the people working in the charity sector. The charity sector can then provide their feedback and may request changes to the template such as text and/or logo positioning. In turn, the clients will decide whether they want Br&.ish to print the marketing material template and deliver it to their company address or have the template sent electronically through email.

This practice had worked for them in the past; however, there are current limitations to their practice. Firstly, their existing practice could be seen as a tedious process between the designers and customers. Response rate from designers and customers may be low which can delay the delivery of the final template. Secondly, since Br&.ish is saving their templates in their computer's local directory, these templates are prone to being deleted and moved to another directory accidentally. Thirdly, other companies, which also provide marketing material templates, have utilized technology systems and have provided users with a different interaction experience. These companies have allowed users to choose a marketing material template from a collection of different categories online and allow them to edit the parameters of the template such as uploading their own logo and adding text descriptions. In order for Br&.ish to catch up with their competitors, their existing practices must be improved.

2.3 Proposal of New System

Br&.ish has proposed the creation of a new system as a means to offer their operators an alternative way of customizing the template and also to offer their customers an alternative way of choosing different marketing material templates for their charity. Therefore, the scope of the proposed system is to design a back end user interface for the operators and a front end user interface for the charity sector. To have a distinction between these two interfaces, Br&.ish has divided the proposal into three individual projects which will be later integrated into a whole working project.

The system in project 1, which is to be developed by Virinder Singh, involves the creation of the front end user interface that will allow the charity sector to select, edit and purchase marketing material templates. The front end user interface must therefore allow different visitors to search the marketing material templates through different categories and also allow them to customise the template in a limited but flexible way. On the other hand, the system in project 3, which is to be developed by Uthayam Kumar, involves the creation of the back end user interface for the Br&.ish operators that will allow them to define the parameters in each template, which will later be customised by the charity sector. Lastly, the system in project 2, which is the system to be developed by the author, is responsible for connecting projects 1 and 3. Project 2 involves the creation of a database system which will allow the system in project 3 to upload and store the defined marketing material templates and will also allow the system in project 1 to display all the templates that were uploaded by the operators. Apart from the database system, project 2 will also involve the creation of the back end user interface to allow operators to categorise the template to different categories and also to allow them to edit the different properties of the template, such as adding or removing keywords, removing them from added categories, or deleting the template itself. The proposal of

this new system means that the author will be working closely with Br&.ish, making this a collaborative industrial project. The main contact for this project will be Elliot Cowan. He will provide any criticism and feedback throughout the project development. Apart from Elliot, the author will also work closely with Virinder and Uthayam in order to implement a final working system.

3. Literature Review

This literature review aims to briefly discuss what a relational database management system is (RDBMS) as well as presenting the different RDBMS products that have emerged throughout the years. Each RDBMS product will be discussed, highlighting their strengths and weaknesses and evaluating which of these models will be the most suitable for this project's database system. In addition, different server side scripting languages, which are programming languages that are used to connect to the database system and manipulate data, will also be evaluated in order to know which server side scripting language will be suited for this project. A similar approach will also be used with the different client side scripting languages, which are programming languages that are used to design the back end user interface for this project. Lastly, this section provides an analysis of the existing systems that already exist, with its limitation being highlighted.

3.1 Database Management Systems (DBMS)

DBMS is a software system that allows different users to manipulate and retrieve data from the database (Healey, 1991). (Connolly & Begg, 2010) explained that the DBMS allows users to define the structure of the database through the data definition language and can also allow users to add, edit, delete or view data from the database through the data manipulation language. Since Br&.ish is interested in having a system that will store different information about their marketing material templates, having a DBMS will reap benefits to the company.

3.1.1 Relational DBMS (RDBMS) model

The RDBMS model, which was introduced by Edgar F. Codd in 1970, has been the predominant choice of database systems to use. This is due to the fact that it has proven superiority amongst previous models, such as Hierarchical and Network model, in terms of addressing issues such as data independence and reducing data inconsistencies (Codd, 1970). The RDBMS uses structured query language (SQL), a programming language that uses relational algebra operations to manage the data in the database. In this model, data is structured within different tables where each table contains one or more tuples. RDBMS provides a simpler way to model data and it is its simplicity that drove this model to be the most widely used DBMS model (Codd, 1980). A RDBMS model contains a primary key which uniquely identifies each tuple and a foreign key which is an attribute from one table that refers to the primary key of another table.

Despite the widespread use of RDBMS, this model still has a disadvantage as it only handles simple data types so handling complex user-defined types such as multimedia have caused some issues (Leavitt, 2000).

3.1.2 Different Relational DBMS Products

Because of the popularity of the relational model, many RDBMS have been introduced throughout the years. These range from commercial RDBMS like Oracle to open source RDBMS such as MySQL and PostgreSQL. Commercial RDBMS will not be covered in this section because of limited access to

resources and since Br&.ish is a small company, incurring costs will be unnecessary. The following subsections will discuss the two open sources RDBMS; MySQL and PostgreSQL.

3.1.2.1 MySQL

MySQL has been viewed as the world's most popular open source database *"because of its high performance, high reliability and ease of use"* (MySQL, n.d.) MySQL offers high compatibility with different operating system platforms such as Linux, Windows and Mac. MySQL has been the popular RDBMS to use with web application development because of its compatibility with other scripting languages like PHP, Perl and Python.

3.1.2.2 PostgreSQL

PostgreSQL has been viewed as the world's most advance open source database (PostgreSQL, n.d.). It has advanced features that are not found in MySQL such as multi version concurrency control (MVCC), point in time recovery, online backup and asynchronous replication. PostgreSQL is easy to learn especially when developers have experienced in using Oracle database. It also has high compatibility with different operating systems platforms such as Linux, Windows and Mac OS.

3.1.3 Evaluation of RDBMS Products

There are loads of debates online that discuss which RDBMS is better. (Gilfillan, 2003) has stated that PostgreSQL offers advanced features that MySQL do not currently offer such as subqueries and stored procedures; however, since MySQL is more widely used, the community support is much larger than PostgreSQL. Another discussion by (Schroder, 2011) has stated that MySQL is used as a *"database back end for websites and applications, performing fast reads and numerous small queries but offering fewer sophisticated features and data integrity checks"*. PostgreSQL, on the other hand, offers a full-featured database and a lot of data integrity checks.

Since the author has experience in using MySQL in the past, MySQL would be the choice in creating the database system for Br&.ish. The design of the database system to be developed for Br&.ish should be simple so choosing MySQL will be more than sufficient. Also, MySQL has a large online documentation and different online forums, websites and books can provide technical support for different MySQL issues to be encountered.

3.2 Server Side Scripting Languages

Scripting languages are referred as dynamic languages because of its dynamic nature of interpreting data as programs and vice versa (Kanavin, 2002). Scripting languages are also called "glue language" because of its ability to glue together different tools and programs (Morin & Brown, 1999). Scripting languages can be divided into two types: Server side and Client side.

Server side scripting languages are interpreted by the web server instead of the web browser. The code written in a server side scripting language is executed on the web server first before the data is passed to the end user's browser (Bradley, n.d.). The following subsections discuss the different server side scripting languages that can be used to access the database system.

3.2.1 Perl

(Connolly & Begg, 2010, p. 993) has stated that Perl, which stands for Practical Extraction and Report Language, *"is a high level interpreted programming language with extensive, easy to use text processing capabilities"*. (Rei, et al., 2007) has mentioned that Perl is one of the most popular

scripting languages because of its text manipulation capability; however, a finding from (W3techs, 2012) has revealed that Perl is only used by 0.8% of websites.

3.2.2 Python

(Paulson, 2007) stated that Python *“is an open source, interpreted, object-oriented, dynamically typed language similar to Perl”*. Python is easy to learn because of its wide documentation online, is open source and it offers very clear and readable syntax. However, (W3techs, 2012) has revealed that Python is only used by 0.2% of websites. This is also supported by (Delorey, et al., 2007) who revealed that the usage of Python as a script language has been small but consistent.

3.2.3 PHP

PHP, which stands for PHP: Hypertext Processor, is a popular open source server side scripting language that is used to implement web applications and to access a database system (Trent, et al., 2008). PHP is normally embedded in static HTML files to produce dynamic web pages and provide developers with MySQL functions to access and retrieve data from the database. According to (W3techs, 2012) *“PHP is used by 78.5%”*. Among PHP users are Facebook, Wikipedia and WordPress.

3.2.4 Evaluation of Server Side Scripting Languages

Based from the popularity among the different scripting languages, PHP has been the dominant language used by many developers. (PHP, 2012) has stated that *“Perl can get very complicated... PHP has a less-confusing and stricter format without losing flexibility”*. The usage of Perl has been decreasing ever since due to the fact that you have to download and install a library that provides access and connection to a database (Crane, n.d.). Also, the syntax for writing Perl programs has been considered less readable (Kanavin, 2002). In comparing PHP and Python, Python offers a simple, easy to read syntax; however, Python is a more general programming language than a web development language. Python has a lot of add on libraries for web development and this can be seen as confusing to the developer whereas with PHP, it is much easier to start web development because the PHP file can just be uploaded in the web server and interpreted in the web browser.

Since the author has experience using PHP in the past, the scripting language to be used to connect to Br&.ish’s database system will be PHP. PHP’s built in database functions, its ease of integration with HTML files and its popularity among many developers are the reasons why this scripting language is chosen.

3.3 Client Side Scripting Languages

Client side scripting languages are interpreted by the web browser instead of the web server. In client side scripting, the web server lets the client computer deal with the data. The following subsections discussed the different client side scripting languages that can be used to develop the back end user interface for this project.

3.3.1 HTML5

HTML, which stands for Hyper Text Mark-up Language, is the main language used for developing the structure of websites. HTML allows users to display information in the web browser such as texts or images. The latest HTML version that has been released is HTML5. HTML5’s additional functionality allows users to embed music, videos and animations without the need of Flash player. HTML5 has proven its dominance over the use of Flash because recent study has shown that the demand for HTML5 websites continues to grow (Gallagher, 2012).

3.3.2 CSS

CSS, which stands for Cascading Style Sheet, is a language that is used to control the appearance of HTML documents and provides a way to have a distinction between the structure and design of the document. CSS allows users to be more flexible in designing a web document and is also easy to learn and understand. Research from (Lie, 2005) has concluded that *“CSS has established itself as one of the fundamental specification on the web and most web sites are using it”*.

3.3.3 JavaScript and JQuery

JavaScript is a scripting language that allows HTML pages to be dynamically changed by users and it allows developers to build dynamic HTML pages with just a small amount of programming effort (Connolly & Begg, 2010). JQuery is a scripting language that uses a JavaScript library. The purpose of JQuery is to simplify document handling in web development and provide developers with an easier abstraction to use JavaScript on their website (JQueryTutorial, n.d.). Some of the functionality of JQuery includes animation, pop-up windows and user input validation.

3.3.4 VBScript

VBScript is a language which offers identical services as JavaScript but follows the syntax of Visual Basic (Connolly & Begg, 2010). Developers who had experienced in using Visual Basic will be able to learn VBScript easily. Since Microsoft introduced VBScript, it is only supported by Microsoft's Internet Explorer or Netscape's Communicator (Haney & Vanlengen, n.d.); therefore, other web browsers do not support this scripting language.

3.3.5 Evaluation of Client Side Scripting Languages

Br&.ish has mentioned that they want the system to preferably run in Google Chrome web browser. The major drawback of VBScript is that it is only supported by Microsoft's Internet Explorer and not a lot of users are using Internet Explorer nowadays. Findings from (w3schools.com, n.d.) has shown that the use of Internet Explorer has been declining from 86.8% in 2002 to just 16.1% in 2012.

The code simplicity that JQuery features makes the code to be written in JavaScript easier and quicker. The author of this report has not used JQuery before but since JQuery is a development framework for JavaScript, learning JQuery should be an easy transition. Apart from JQuery, HTML5 and CSS will also be used as client side scripting languages as these two forms the basis of creating the structure and design or layout of any web page.

3.4 Existing Work

Existing competitor companies who offer similar services that Br&.ish wants to offer are reviewed in this section. These three companies are seen as front end systems as it allows the end users to browse different marketing material templates such as business cards, leaflets and posters. Apart from these three front end systems, other websites, which allow users to add meta-data to their products and also categorise their products in to different categories will also be discussed. These websites can be seen as back end systems as it allows users to define information about their product which can be later viewed by different users.

3.4.1 Front End Systems

The existing competitor companies who offer similar services like Br&.ish are VistaPrint, GoodPrint and Tweak. The following subsections will briefly discuss how these companies grouped their products in to different categories.

3.4.1.1 VistaPrint (<http://www.vistaprint.co.uk>)

VistaPrint is a company that provides a wide range of services to small businesses such as providing marketing materials, creating websites, designing logos and providing other office products. It allows visitors to browse all their available premade designs or visitors can also upload their own design.

VistaPrint's products are grouped in to three categories; Industry, Style and Purpose. Each category has different subcategories and it shows how many products are in each subcategory. However, there are limitations with this website. Firstly, it does not have a colours category for visitors to select a colour from a predefined colour plug-in and secondly, the subcategories do not have further subcategories to refine the template designs more.

3.4.1.2 GoodPrint (<http://www.goodprint.co.uk>)

GoodPrint is a company that provides different types of marketing material templates to users around Europe and North America. They specialise in providing business cards, Christmas cards, flyers, letterheads and brochures. GoodPrint also allows users to upload their own design and own logo to the templates.

GoodPrint's products are grouped into different categories such as by product type, by different professions and styles. However, there are disadvantages with this website. Firstly, the subcategories do not have further subcategories to refine the template designs more and secondly, designs that were returned upon selecting a category are not that accurate. For example, when the category 'Strong Colours' was selected in the styles category and 'blue' was typed in the keywords search engine, templates which have the colour pink or red in its design were also displayed.

3.4.1.3 Tweak (<http://www.tweak.com>)

Tweak is an online company that allows users to design their own marketing material templates in a much more flexible way. Some of its services include providing banners, bookmarks, business cards and flyers. Tweak also allows users to upload their own logos, add text descriptions and even change the colour of the template.

Unlike VistaPrint and GoodPrint, Tweak's products are categorised into different stages. In the first stage, users are asked to narrow down the templates by choosing which product type to customise. In the second stage, users narrow their choices by choosing their template size. In the third and fourth stages, users further narrow down their choices by choosing an industry type and a specific sector from the industry they have chosen. By categorising the products in different stages, Tweak offers its users a more specific way of searching a marketing material template. However, it shares the same limitations of VistaPrint as there is no option for grouping the templates into different colours.

3.4.2 Back End Systems

The two websites that allow users to define the information of their product are 500px and Flickr. The following subsections will briefly discuss how these two photo-sharing websites allow their users to categorise their photos in to different categories and provide meta-data information such as name, description and keywords.

3.4.2.1 500px (<http://500px.com>)

500px is an online website where users can share and sell their photos to different users worldwide. The photos are grouped into different categories and it also shows the most popular or most viewed photos on a daily basis. When a new photo is uploaded, the user can edit their photo's information such as adding its name, description, location and keywords. The user can also place their photo to predefined categories; however, the photo can only be placed to one category at a time. This can be seen as a limitation since the visibility of the photo will be limited. Because of this limitation, users can opt to create a new photo set wherein similar photos can be placed. However, this functionality is not free as a monthly fee subscription has to be paid.

3.4.2.2 Flickr (<http://www.flickr.com/>)

Flickr is another photo-sharing website that allows users to share their photos to different users worldwide. Flickr also shows the most interesting photos that were uploaded on a daily basis to their 'Explore' page. However, unlike 500px, Flickr does not allow photos to be sold as it is only a website for sharing photos.

When a new photo is uploaded, the user can edit their photo's information such as its name, description and tags. The photo can also be added to different photo groups and photo sets. The photo groups are created by other users as a placeholder for similar photos. The sets, on the other hand, are created by the user. Even though the photos can be placed to different groups and sets, Flickr does not allow the users to categorise the template to predefined categories.

3.4.3 Critical Review of Existing Works

VistaPrint, GoodPrint and Tweak have provided an insight as to how different front end users search for different templates. These systems have categorised their products into different categories. On the other hand, 500px and Flickr have provided an insight as to how different users categorise and add meta-data to their photos. However, limitations have been observed to these systems. Firstly, VistaPrint, GoodPrint and Tweak do not have a colour picker category that allows users to select from predefined colours. Although VistaPrint and GoodPrint have a category for 'Strong Colours', it does not allow users to pick from a specific colour. Secondly, the three front end systems only show one level of category relationship so a subcategory cannot contain another subcategory. Thirdly, 500px only allows users to categorise their photo in to one category and finally, 500px and Flickr only offers their full functionality if users pay a monthly subscription fee. Hence, the system that will be designed must address these limitations in order to benefit Br&.ish and their clients.

On the other hand, all the existing systems that were discussed offer their users a simple and easy to use interface. Design principles have been used by these existing systems to increase its usability. Firstly, each system provides an easy flow of information. For example, Tweak offers their users different stages of selecting a template design while Flickr and 500px allows their users to edit their photo's information straight after uploading it. Secondly, the existing systems provide users a visibility of their system's current state. For example, VistaPrint and GoodPrint display the total number of template designs that were displayed when a category has been selected by the user while Flickr and 500px display the uploading status of a photo. Lastly, the existing systems also provide feedback to the user's actions. If the user wants to delete a photo in Flickr or 500px, a confirmation box will appear to confirm their action. Since these different design principles were

observed in the existing systems, the system that will be designed must also incorporate these different design principles to increase its usability and ease of use.

3.5 Conclusion

This literature review presented the different tools and technologies that can be used to create a database and user interface system. RDBMS model has proven to be the most popular DBMS because of its higher reliability and fast performance to handle many queries. A popular RDBMS product to use is MySQL. MySQL is open source, fast, reliable, has high compatibility and easy to use. In terms of the different scripting languages that can be used, PHP has proven popular because of its built-in MySQL functions that can be used to connect to the database effortlessly to insert, delete or modify data. HTML5, alongside CSS and JQuery will be used to construct the user interface that will allow Br&.ish operators to categorise marketing material templates. HTML5 and CSS are the basic scripting languages that are needed to create the structure and design of the web page. JQuery, a language that uses the JavaScript library was favoured because it is supported by all browsers and offers functions that can be written in less code.

In addition, because of the limitations found in the existing system, the system that will be designed for this project must address these limitations such as allowing multiple levels of category relationships and allowing the template to be categorised in one or more categories at a time. Since the existing systems all provide a simple and easy to use interface, the system that will be designed must also incorporate different usability guidelines and design principles to ensure that the Br&.ish's operators will find the system easy to use.

4. Method

For this project to be successful, good project planning is needed; hence, software development methodology will be discussed in this section. This section will also discuss the chosen methodology for this project – the iterative and incremental model.

4.1 Software Development Methodology

Software development methodology is a framework that describes the overall structure and development of any software development project. Having a methodology is important as it highlights the different to-do actions as well as the order of these actions in the development process. Different software methodologies such as the waterfall cycle model, iterative model, incremental model and rational unified process, to name a few, have emerged because of the failure of most software related projects in the 1960s (Sommerville, 1996).

4.2 Incremental Development Model

The Incremental Development model breaks down the project development into a number of mini waterfall cycles rather than developing the system in one large cycle (Davis, et al., 1988). The first increment corresponds to implementing the core functionality of the system with the further increments corresponding to adding additional functionalities until the system is fully complete.

This model has several advantages. Firstly, this model supports the nature of prototyping, which is the process of delivering an incomplete working version of the system that contains a number of functionalities to the users. Secondly, this model facilitates the incorporation of adding new

requirements. Since the users can test the system in the early stages, they can provide new requirements that were not gathered by the development team at the start. However, this model still has some drawbacks. Firstly, since the system's development process is divided into increments, adding new functionality or new requirements can cause the system to deteriorate due to successive changes in the system's overall architecture (Lethbridge & Laganier, 2005). Secondly, this model requires an active involvement from users. If feedbacks from the users are not being provided in each increment, this can lead to the late delivery of the system.

4.3 Iterative Development Model

The Iterative Development model, similar with the Incremental model, delivers the project in to different cycles rather than developing the system in one large cycle. This model focuses on improving a rough draft of the system, doing a little bit of implementation, testing it, asking the users for their feedback, using these feedbacks to improve the current system and then iterating the same cycle again until the system is complete (Munassar & Govardhan, 2010).

Since the system is developed iteratively, this model possesses several advantages. Firstly, projects with unclear and incomplete requirements can benefit from using this model. Since the users are presented with a working version of the system regularly, additional requirements can be identified and implemented in the next iteration. Secondly, similar with the Incremental model, this model supports the nature of prototyping. Lastly, potential defects of the system can be identified and resolved at an early stage since testing is performed continuously. In terms of disadvantages, this model shares the same disadvantages of the Incremental model as mentioned above.

4.4 Evaluation of Iterative and Incremental Models

Software lifecycle models all have their own advantages and disadvantages which show that there is no single software lifecycle model out there that all projects use. The decision as to what model to use depends on the nature and requirements of the project itself.

The repetitive cycles of both Iterative and Incremental mean that the implementation of Br&.ish's requirements can be performed at an early stage which can lead to delivering a project on time. By choosing both models, the author can fully understand what Br&.ish really wants their system to do. For instance, if there are any unclear requirements, these can be resolved during the early increments by asking Elliot for his feedbacks. At the same time, if Elliot proposed additional requirements, these can be accommodated in the following increments. Also, since this project involves integrating this system to the other two systems proposed by Br&.ish, using iterative model means that the integration can be performed and tested earlier.

There is a risk for choosing both of these models and that is the need for active user involvement. For this system to be usable and successful, there must be a constant communication between the author and Elliot. At the same time, Elliot must be willing to provide his inputs and feedback throughout the development process. Another risk for choosing this model is that it needs active communication among the development team – the author, Virinder and Uthayam. The team needs to maintain good work ethics and time management in order for each cycle to be released on time.

5. Project Analysis

After discussing the software methodology that will be used for this project, this section will now focus as to what requirements are needed to be implemented for this project. Different forms of requirements gathering will be used to gather the requirements and a set of requirements specification, which contains the system's functional and non-functional requirements, will be produced.

5.1 Requirements Gathering

The following subsections will now look at the different types of requirements gathering methods that have been chosen for this project. These different methods will be briefly discussed and evaluated as to why they were chosen for this project.

Requirement gathering techniques range from observation, interviews, questionnaire, online client meetings and email correspondence. Observation involves observing the users in their actual setting. Conducting observations will provide more in depth details of any requirements that you might have missed before. However, observation consumes a large amount of time and Elliot is mostly unavailable as he travels to United States most of the time. Interviewing, on the other hand, is a widely used technique that is used to gather detailed requirements from the user and is useful for clearing out any unclear requirements. However, interviews can also be time consuming and Elliot is always away so he is only available to talk to online. Due to these reasons, observations and interviewing have not been chosen. Instead, online meetings, email and questionnaires were chosen as requirement gathering methods.

5.1.1 Online Client Meetings (Teleconference calls)

Online client meetings, in the form of Skype teleconference calls between the development team, Dr. Stefan and Elliot, are to be held in each project cycle demonstration. Since Elliot is always travelling, one advantage of online meetings is that this will fit to his busy schedule. During the online meetings, a demonstration of the current version of the system will be presented to Elliot through Skype's *"screen sharing functionality"*. This will allow Elliot to see the working prototype of the system even though he is miles away. Through Elliot's feedbacks, any changes or any new requirements of the system will be known and will therefore be implemented in the next project cycle. In addition, the client meetings will also serve as a way for Elliot to be informed about the development team's plans for the next cycle of the project.

5.1.2 Email Correspondence

Apart from client meetings, sending an email to Elliot will be another way to gather the requirements or to clarify inconsistencies. Since the client meetings will only be held a few number of times, sending emails will allow the author to continuously communicate with Elliot. However, email is not to be relied fully as there will be times when correspondence from Elliot might be delayed due to his busy schedule or internet connection problems.

5.1.3 Questionnaires

The last requirement gathering method to be used will be designing a questionnaire. Questionnaire is a widely use technique because it allows a person to gather large amount of data in a short period of time. Also, since the questionnaire's format is consistent, feedbacks can be easily compared. The questionnaire will be sent to Elliot and his team at Br&.ish after they have tested the system. The

questionnaire is a mixed of both close-ended and open-ended questions in order for both quantitative and qualitative data to be gathered. The link for the actual questionnaire is found at <http://www.surveymonkey.com/s/TTPS32P>.

5.2 Requirements Specification

After discussing the different requirement gathering methods that will be used, this section will now list the different requirements identified for this project. These requirements have been gathered from Elliot during the first client meeting prior to the start of this project. Any inconsistencies from the requirements have also been cleared out with Elliot through email correspondence. These requirements were analysed, refined and then modelled using a Unified Modelling Language (UML) diagram called the use case diagram. This diagram is a representation of the system from the user's point of view and describes the different interaction of the users to the system. From this diagram, a set of functional and non-functional requirements can be identified. In addition, due to the nature of changing requirements, extra requirements have also been gathered in the subsequent meetings.

5.2.1 Use Case Diagram

Figure 1 shows the use case diagram that was made for this project. The purpose of the use case diagram is to identify the functionalities of the system and also to have a clear representation of the scope of this project.

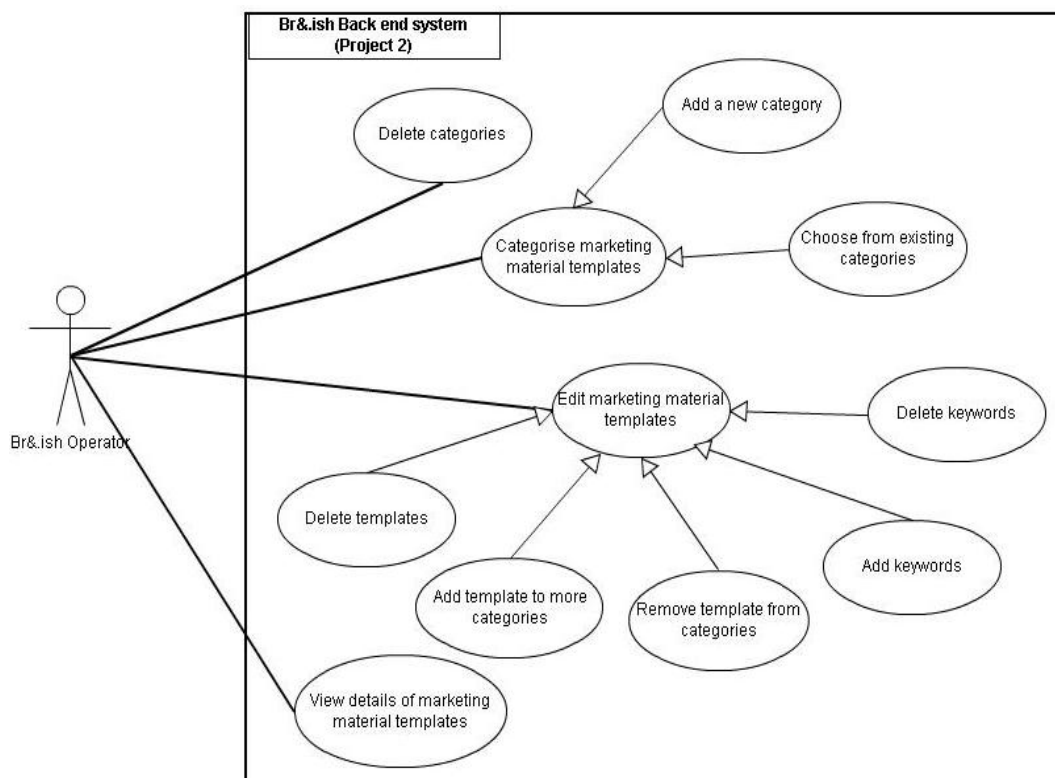


Figure 1: Use Case Diagram for Br&.ish's Back End System (Project 2)

By looking at Figure 1, the Br&.ish operator is responsible in categorising the templates into different categories. In categorising a template, the operator can either add a new category or choose from existing categories that had already been added. The operator can also delete a category if ever they made a mistake. Moreover, the operator can define or edit the template's properties. In editing a template, the operator can have the following options: they can add keywords to the template or delete keywords, add the template to more categories, remove them from added categories or

delete the template itself. Finally, the operator can also view the complete details of a marketing material template.

5.2.2 Functional Requirements ('Must Have' Requirements)

After illustrating the user's interaction with the system through the use case diagram, a set of the system's 'must have' functional requirements can now be identified. These requirements have been identified as 'must have' as these are essential in order to meet the Br&.ish's needs and the project aims. These requirements are therefore listed as follows:

- The system shall display the different templates that were uploaded by the operator.
- The system shall allow the operators to categorise a template to one or more categories.
- The system shall allow the operators to add a new category.
- The system shall allow the operators to add subcategories. Moreover, a subcategory can also have subcategories.
- The system shall allow the operators to delete a category, if ever they made a mistake.
- The system shall allow the operators to add keywords or meta-data to the templates.
- The system shall allow the operators to delete keywords from the templates.
- The system shall allow the operators to edit the template's categories. For example, the operator can add more categories to a template. At the same time, the operator can also remove added categories from the template.
- The system shall allow the operators to delete any templates.
- The system shall allow the operators to view the complete details of a template. This involves viewing the template's keywords and categories.
- The system shall allow the operators to upload another template by redirecting them to the back end user interface of Project 3.
- The system shall allow the operator to edit the defined parameters of the template by redirecting them to the back end user interface of Project 3.
- The system shall allow the operator to logout from the back end user interface by redirecting them to the back end user interface of Project 3.

5.2.2.1 Additional Functional Requirements ('Could Have' Requirements)

Additional requirements have also been introduced in this project during the different project cycle meetings. These requirements have been identified as 'could have' as these requirements are seen desirable but not essential. These requirements will be implemented if additional time or resources will be available. These requirements are listed as follows:

- The system shall automatically select a parent category if its subcategory has been selected.
- The system shall provide 'colour picker functionality' to categorise a template to different colours.

5.2.3 Database Requirements

Since this project also involves the creation of a database system for Br&.ish, a separate section has been made to list the different requirements that the database system needs to perform.

5.2.3.1 Database Requirements for Project 1

- The database system shall store the client's details such as their username, password, first name, last name and address for login or registration purposes.

5.2.3.2 Database Requirements for Project 2

- The database system shall store the different category names and category descriptions that were added by the operators.
- The database shall store the categories wherein a template has been categorised to.
- The database shall store the different keywords of a marketing material template.

5.2.3.3 Database Requirements for Project 3

- The database system shall store the operator's details such as their username and password, for login purposes.
- The database shall store the template details such as its name, description and price.
- The database shall store the filename of the template image to be uploaded.
- The database shall store the different parameters that were defined by the operator such as parameter name, font colour, font size and font style to name a few.

5.2.4 Non-functional Requirements

Non-functional requirements are requirements that explain the expected behaviour of the system. If the system has all the functional requirements but the client perceives it as hard to use, then the system will not be entirely usable. These non-functional requirements are therefore listed as follows:

5.2.4.1 User Interface Requirements

- The user interface shall be easy to use, easy to understand and easy to navigate.
- The user interface shall display the current state of the system through different page descriptions to inform the operators with their actions.
- The user interface shall use standard and direct names for buttons, links and headers.
- The user interface shall display appropriate alert and error messages to inform the operators with their actions.
- The user interface shall provide tooltips to assist the operators in using the system.
- The colour scheme, layout and style of the user interface shall be consistent with the other interfaces in Projects 1 and 3.

5.2.4.2 Browser Compatibility Requirements

The user interface system shall run and be compatible on most of the available web browsers such as Google Chrome, Mozilla Firefox, Safari and Internet Explorer.

5.2.4.3 Security Requirements

- The system shall check for 'SQL injection attacks' which are statements entered by a user that could cause unexpected behaviour in the database system.
- The system shall sanitise the operator's input to check whether a data to be inserted to the database is good data.

6. Design

After identifying the requirements for this project, this section will now present the different design processes in each project cycle starting from presenting the design overview and then describing the design choices for each cycle. This section also covers the Human-Computer Interaction (HCI) design principles that the author has implemented in making the system usable. Since this project is to be developed iteratively and incrementally, one advantage of using this methodology is that design choices can be improved in the later stages to accommodate the client's needs.

6.1 Framework Design Overview

Initially, it has been proposed and agreed that there will be 4 different cycles for this project. However, a 5th and final project cycle has been introduced on the 4th cycle meeting. This final cycle will serve as a way to further add improvements on the system as well as to obtain Elliot's final evaluation about the completed system.

Increment 1 focused on designing the database schema for this project. This involves creating an Entity-Relationship (ER) diagram, explaining the different constraints that the database should have and then normalising the tables to reduce data redundancy. Meanwhile, the subsequent increments focused on designing the back end user interface of the system. This involves producing a wireframe layout, which is a layout that illustrates how the system should look like, as well as discussing the different design decisions such as identifying the colour scheme to be used, what form of input control is to be implemented, what feedback or error messages is to be shown, and identifying the positioning of texts, buttons, links or images.

Specifically, increment 2 focused on designing the page for categorising templates while increment 3 focused on designing the page for editing the template's properties. Increment 4, on the other hand, focused on designing the page for viewing the template details as well as making improvements in the design layouts of the pages designed in increments 2 and 3. These improvements have been made in order to accommodate Elliot's suggestion of maintaining consistency between the front end and back end user interface. It is also important to note that these design increments are dependent to their preceding increment. Without a good database design, problems will arise in the subsequent increments in terms of storing or displaying information about the templates or the categories. Moreover, if the functionality for categorising the template to different categories has not been designed properly, the subsequent functionalities will also not function properly.

6.2 Increment 1

The database has been identified as the core requirement for this project because it plays a very important part in implementing the additional functionalities. Without a good database design, the database will be inefficient and will have an effect to the overall performance of the system.

6.2.1 ER Diagram

An ER diagram provides a clear illustration of the logical structure of how a system would work. The ER diagram is also helpful in terms of communicating the database design to the client because the diagram itself is written in plain English and can be easy to understand. An ER diagram has three components; entities, attributes and relationships. Entities are *"thing or object in the real world that is distinguishable from all other objects"* (Silberschatz, et al., 2002, p. 27) and are represented by a rectangle. Entities are then described by a set of attributes, which are represented by an oval shape.

Finally, relationships represent the association between entities and are represented by a triangle. ER diagram also shows the constraints between the relationships of two entities. Cardinality constraints are constraints in the ER diagram that describes how many instances of an entity are associated in a relationship. Participation constraints, on the other hand, are constraints that describes if only some or every entity is associated in a relationship. The min and max (e.g. 1...1) notation have been used for specifying these constraints where min represents the participation constraint and max represents the cardinality constraint. If the min is set to 0, not all instances of an entity are associated in the relationship whereas, if the min is set to 1, all instances of an entity are associated in the relationship. On the other hand, if the max is set to 1, only 1 instance of an entity will be associated in the relationship whereas, if the max is set to *, many instances of an entity will be associated in the relationship. Figure 2 shows the simplified ER diagram for the whole system of Br&.ish.

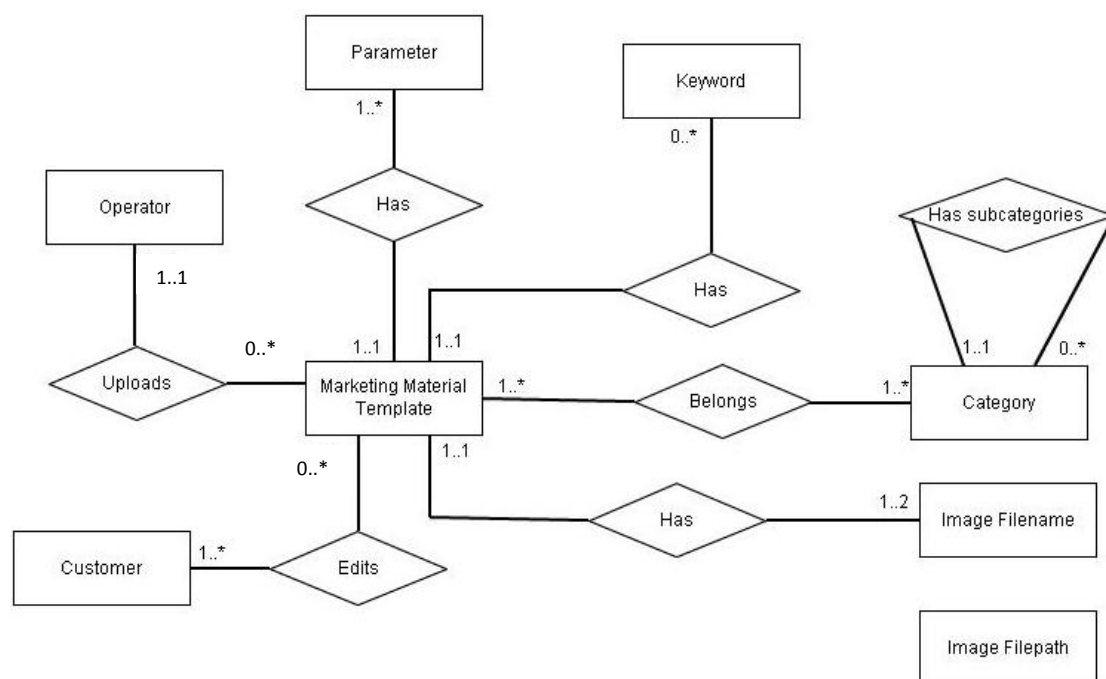


Figure 2: ER Diagram for Br&.ish's system

The different relationships that exist between different entities are explained as follows:

- A Br&.ish operator can upload many marketing material templates. Also, an operator may not upload any template hence the min for the template entity is set to 0.
- Many charity clients or customers can edit many marketing material templates. Also, a client may not edit any template (i.e. they might only browse the different templates) when they login to the system hence the min for the marketing material template entity is set to 0.
- One marketing material template has many parameters. A template that was uploaded must have at least one parameter defined hence the min for the parameter entity is set to 1.
- One marketing material template has two image filenames. A template that is to be uploaded must have at least one template image hence the min for the image filename entity is set to 1. The maximum constraint is set to two because a template can only have a maximum of two images (i.e. front side and back side)

- Many marketing material templates belong to many categories. Templates that were uploaded to the database must be categorised to at least one category hence the min for the category entity is set to 1.
- One category can have many subcategories. An operator might not add a subcategory to a category hence the min for the category entity is set to 0.
- One marketing material template has many keywords. An operator might not add any keywords to a template hence the min for the keyword entity is set to 0.

Since Figure 2 only shows the simplified ER diagram, a more detailed ER diagram has been created which includes the different attributes of the entities. While some tables are easy to provide attributes with, some tables need clarification as to what attributes are needed. For example, in the Parameter table, the author was unsure as to what kinds of attributes are needed to be stored in the table; hence, clarification was made with Elliot in order to fix the dilemma. To see the detailed ER diagram that has the complete set of attributes, see Figure 26 in Appendix A.

6.2.2 Mapping of ER Diagram to Tables and Normalisation

The ER diagram that was illustrated can now be mapped to a relational database model with the different entities mapped as different tables and relationships mapped as foreign keys, with the constraints determining the places of the foreign keys. In addition, each table will also have a primary key. The primary key of the different entities in the detailed ER diagram in Appendix A are in bold and underlined. The primary key ensures that each row in a table is unique while the foreign key ensures that integrity constraints between two tables are maintained.

Once the ER diagram has been mapped as tables, the tables still need to be normalised into an appropriate form. Database normalisation *“allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily”* (Silberschatz, et al., 2002, p. 257). However, if the ER diagram has identified all the correct entities, attributes and relationships in the first place, then if we mapped the ER diagram into a relational database, the tables will already appear normalise. Figure 27 in Appendix B shows the ER diagram mapping and the explanation for the database normalisation for this project.

6.2.3 Design Decisions

There are two options in storing images to the database. The first option involves storing the actual image to the database. The second option involves storing the image path of the template images. The latter option has been chosen because the response time will be much faster since only the path of the template image in the web server will be stored. If the actual image is stored in the database instead, the database’s performance will be slow in terms of retrieving the images and displaying it in the front end user interface for the charity sector clients to see.

By looking again at Figure 2, the ER diagram has an entity for image file path. This entity will only contain one row which corresponds to the file path as to where the template images will be placed. Prior to designing the final ER diagram, the author has initially decided that the image file path will be an attribute of the image file name entity. However, this is not efficient because if all templates are to be stored in one directory, then the image file path attribute in the image filename entity will be the same in each row. Because of this, it is much better to create a separate entity for the image file path and then manipulate the query for displaying the image through concatenation.

6.3 HCI Design Principles

A system is perceived to be a failure if users do not understand how to use it, if the system is too difficult to use or if the user has insufficient expertise or experience with a particular technology. It has been mentioned that the main challenge for this project is to develop a working system that Br&.ish will want to use. Therefore, the main goal for HCI is for systems to be usable, easy to use and to engage users so they continue on interacting with it (Poslad, 2009).

In designing the back end user interface for this project, the author has taken into account Norman's Design Principles, namely, making things visible, providing a natural mapping and providing feedbacks. Firstly, to make a system visible, the current state and information of the system must be visible to the users so they would know how to use it (Norman, 1988). The user interface should be designed in a way that it has short descriptions in different pages to inform the operator as to what each page is about. Secondly, in order to provide a natural mapping, the user interface should be designed in a way that it uses standard naming conventions to the buttons and links. For example, if an operator wants to add a new category, the button's name should be 'Add' instead of 'OK' because the standard naming conventions for adding new things is 'Add'. Finally, the third design principle is that the system should be able to provide appropriate feedbacks or error messages to the operator's actions.

6.4 Increment 2

After discussing the different design principles that the author will use, this section will now look at the design for increment 2. Increment 2 involves designing the initial wireframes of the categorising templates page. This initial wireframe has been used as the basis for implementing the layout for categorising templates page. However, during the third project cycle meeting, Elliot wanted the layout of the back end interface to be consistent with the front end interface. Therefore, this wireframe design has been updated in increment 4.

6.4.1 Categorising Templates View

Figure 3 shows the initial wireframe layout for the categorising templates page. The design layout shows a three row layout with the first row containing the logo of Br&.ish as well as the navigation links, the second row containing the main functionality of the page and the third row containing the footer. The second row of the layout will be divided into three columns with the left column containing the functionality for adding new categories, the center column containing the functionality for categorising the templates and the right column containing the template's name and images that is to be categorised into different categories.

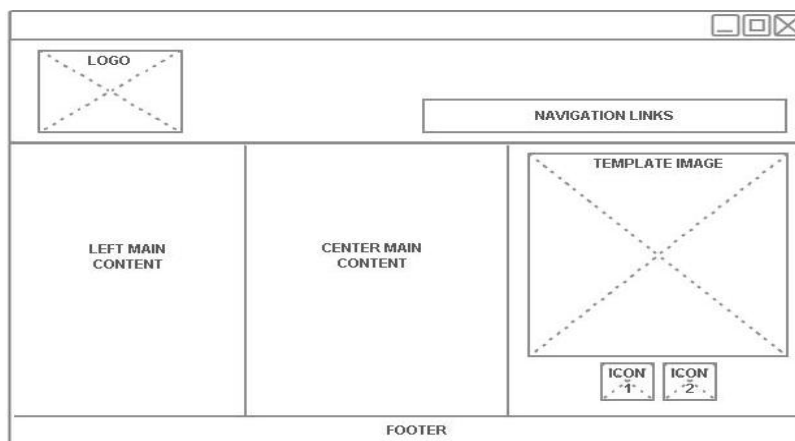


Figure 3: Wireframe Layout Structure

6.4.2 Design Decisions

The design for the categorising templates page has been similarly patterned with Br&.ish's existing website. Looking at their existing website, the design layout is very simple and minimal, with the colour scheme being mostly black, white and gray. These three colours will therefore be used as the main colour scheme to ensure consistency and to represent the branding image of Br&.ish. The colour scheme may be seen as too simple but since Br&.ish will upload different designs of templates, the hint of colour coming from the template itself will provide the additional colours in the user interface. In addition, the positioning of the logo in Br&.ish's existing website is located at the top-left corner of their page. Therefore, the positioning of the logo for the categorising templates view and the rest of the different pages in the back end user interface will also be placed at the top-left corner of the page. This ensures that consistency of the layout is observed so Br&.ish will still be familiar when they first see the back end user interface. The navigation links are all placed together so that the operators will find the system easy to navigate.

The categorising functionality is placed in the middle row to emphasize that the page is for categorising the templates. In adding new categories, text field box has been chosen for the category name while text area box has been chosen for the category descriptions since descriptions will have longer texts. Radio buttons have been chosen to ask the operator if the category to be added is a parent category or subcategory. Radio buttons ensure that only one choice will be made since a category to be added cannot be both a parent category and a subcategory. On the other hand, the lists of existing categories are modelled hierarchically with the subcategories being indented below its parent category. In terms of displaying the front side and back side of the template's image, the layout shows the front side of the image as its default image. If the operator wants to view the back side of the image, the operator can click the second icon at the bottom of the image and this will display the back side of the template's image. Designing the visibility of the images like this will save extra space in the layout.

To incorporate Norman's design principle of providing feedback, different feedbacks and error messages will be presented to the operators in the form of alert box and. For example, if the operator wants to add a new category and clicks the add button without providing a category description, the interface will be designed in a way that it will inform the operator that they need to provide a short description for the category they wish to add.

6.5 Increment 3

The design in Increment 3 involves designing the editing templates page of the back end user interface. The layout design is consistent with the design layout made in increment 2, with the same three row layout being used as the design layout. However, this layout has been changed in increment 4, as mentioned previously. Although the final layout for the editing templates page is now different, this section will still describe the initial design choices that the author has made.

6.5.1 Design Decisions

The functionality for editing the templates has been designed in a way that if the operator selects a template to edit, the page will automatically display the template's details without the operator clicking a button. Radio buttons are used in selecting a template to ensure that only one template at a time can be edited by the operator. If the user interface allows the operator to edit multiple templates at the same time, the operator might be overwhelmed because of too much information

and might make the wrong changes. On the other hand, checkboxes are used in displaying the list of categories from which the template can be added or removed. Checkboxes are used so that multiple changes can be made when the operator edits the template's categories.

6.6 Increment 4

Increment 4 involves improving the designs of increments 2 and 3 as well as designing the template details view. As mentioned previously, the layout has been patterned from the design layout of the front end user interface as Elliot wanted to maintain consistency in both front and back end interface. The changes that have been made will be therefore discussed in the following subsections.

6.6.1 Improved Categorising Templates View

Figure 4 shows the revised layout of the categorising templates view. By comparing this design layout with the original layout of the categorising templates in Figure 3, the navigation links are now placed in line with the logo. The new layout also displays an additional column underneath the logo and navigation links. This additional column will hold the page description that would serve as guidance to the operators so they would have an overview as to what the page is about.

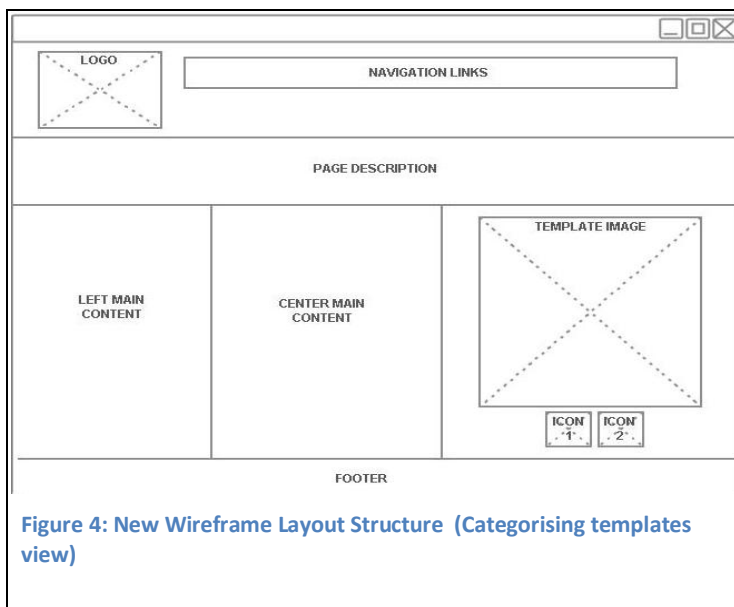


Figure 4: New Wireframe Layout Structure (Categorising templates view)

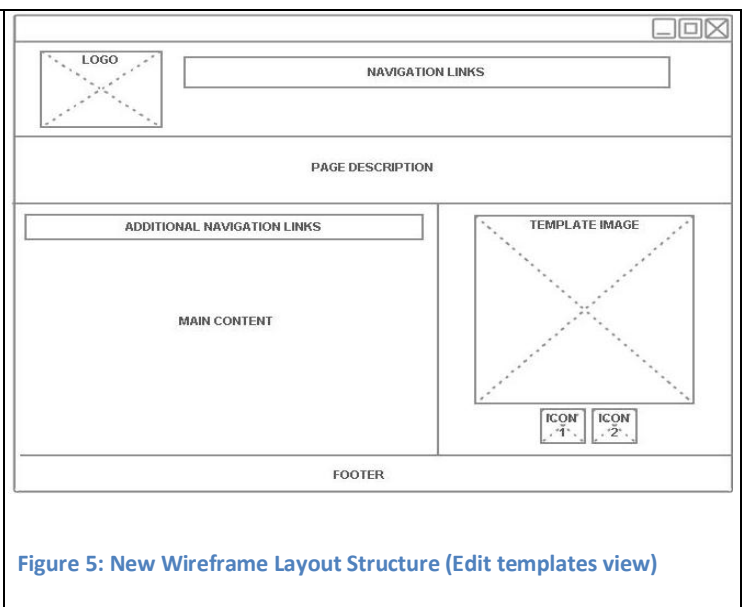


Figure 5: New Wireframe Layout Structure (Edit templates view)

6.6.2 Improved Editing Templates View

On the other hand, Figure 5 shows the revised layout of the editing templates view with additional links added to the new layout. These additional links represent the links for editing the properties of the template. Apart from editing the template's categories, the operator will also be given the functionality to add or remove keywords from the template and delete the template itself.

6.6.3 Template Details View

Finally, a page for displaying all of the template's details will also be needed for this project. The design layout is very similar to the editing templates view without the additional links in the second row layout. If an operator selects a template, the template's image, the template's list of keywords and the categories it belongs to will be displayed.

6.6.4 Design Decisions

The navigation links, even though they are still grouped together, are now placed beside the Br&.ish logo as this is where the navigation links from the front end user interface are placed. To incorporate Norman's design principle of making things visible, the font colour of the link for an active page will be in black while the other links will be in the colour gray. By distinguishing the colour of the active link, the user will have a very clear idea that they are on the correct page.

In designing the functionality for deleting categories if ever the operators made a mistake, the author has decided to use a pop-up box because if the delete categories functionality is to be displayed on the same page as the categorising templates, there will be too much information on the screen. With a pop-up box, their attention will be focus on the pop-up box and can then concentrate on the task of deleting a category instead.

Additional error messages are also to be displayed to the operators. For example, if the operator wants to delete a category but has not selected at least one category, the interface will be designed in a way that it will inform the operator that they need to select at least one category.

Moreover, in terms of designing the improved version of the editing templates view, the template's images are to be displayed whenever the operator wants to edit its properties. The reason behind this is that if the operator wishes to add keywords in the template, the first thing that they would look at is the actual template's image because they can think of what keywords to add since they are looking at the visual representation of the template. If they are only presented with the template's name, they might not be able to remember what that template looks like since they look at different template designs on a day to day basis.

7. Implementation

This section covers the implementation stage in the project development process. Since this project is developed iteratively and incrementally, different parts of implementation have been performed in all the different cycles. After each increment, the latest version of the system is shown to Elliot to get his feedback. The first cycle focused on implementing the database system for this project while the subsequent cycles focused on implementing the back end interface for this project. Screenshots of some of the implementation code will be provided for reference and will be explained in non-technical terms for easy understanding. The different issues that have been encountered in implementing some of the functionalities will also be discussed in order to see how these issues were tackled.

7.1 Increment 1

As first mention in sections 3.1.2.1 and 3.2.3, MySQL alongside with PHP, will be used to create the database tables for this project. To implement the database system, a connection to the database server has to be invoked, create the database tables if they do not exist yet, and then close the invoked connection to the database.

7.1.1 Web Server Access

Since this project is to be integrated with the other projects proposed by Br&.ish, a web server is needed for the development team to start implementing the individual projects and then later integrating each one of them. The author had asked Elliot whether he would be able to provide a

server space for this project; however, there was no confirmation from him as to whether or not he would be able to grant the server space. Because of this, the development team has decided to use Virinder's web server instead.

7.1.2 Connecting to the Database

Initially, issues have been encountered in connecting to the database server because the author has not used PHP's built-in functions in MySQL such as *mysql_connect* and *mysql_select_db* before. However, after doing some background readings on how to invoke a connection to the database, this problem was fixed. To establish a database connection, the database credentials have to be explicitly provided. These credentials include the host name, which is normally 'localhost', the username and the password. Figure 6 shows the snippet of code for connecting to the database. If the credentials provided are correct, the database name, to which the tables will be created, will be selected.

```
//Establish a connection to the database
$conn = mysql_connect($db_host, $db_user, $db_pw);

//If connection is succesful, open the database
if($conn){
    mysql_select_db($db_name) or die ('Could not open $db_name' . mysql_error());
}
else{
    die('Cant establish a connection' . mysql_error());
}
```

Figure 6: Connecting to the Database

7.1.3 Creating the Database Tables

Once the connection to the database has been established, the tables that have been identified in Appendix B can now be created. In total, there are 10 tables to be created for this project. Creating these tables involve specifying the data type for the different attributes, the length of characters allowed in the attribute and then specifying the different constraints such as primary keys, foreign keys and NULL constraints. The most common data types that were used for this project are 'int' for numbers and 'varchar' for texts. Figure 7 shows the snippet of code for creating the Customer table. The NOT NULL constraint simply means that an attribute needs to have a value. In addition, to ensure that each primary key of the table's row is unique, AUTO INCREMENT feature has been used. The AUTO INCREMENT feature will automatically generate a unique identifier once a row has been inserted to the table.

```
/**A table that stores details about the customers. The primary key of this table is the Cust_ID**/
$CustomerTable = "CREATE TABLE $table1(
    Cust_id int(3) NOT NULL AUTO_INCREMENT,
    Fname varchar(10) NOT NULL,
    Lname varchar(10) NOT NULL,
    Username varchar(10) NOT NULL,
    Password varchar(8) NOT NULL,
    Last_login datetime NOT NULL,
    Reg_date date NOT NULL,
    Contact_num varchar(11) NOT NULL,
    Email varchar(20) NOT NULL,
    DOB date NOT NULL,
    PRIMARY KEY (Cust_id))
ENGINE=InnoDB";
$createTable1 = mysql_query($CustomerTable);
```

Figure 7: Creating the Customer Table

7.2 Increment 2

Increment 2 focused on implementing the design layout for categorising the templates. Even though Elliot had mentioned on the third project cycle meeting that he wanted the design of the back end

user interface to be similar with the front end user interface, this section will still describe as to how the layout for the categorising templates view has been initially implemented.

7.2.1 Layout Structure

The layout structure for the back end user interface is to be designed with a three row layout. To separate these contents, HTML's `<div>` tag has been used. Div tags are used in HTML when different contents of similar properties need to be grouped together. The positioning or placement of these div elements in the page is then controlled using CSS.

Figure 8 shows the basic structure of the back end user interface. The whole content of the interface is placed in one div tag called 'main_container'. The main container then contains three separate div tags for the 'header', 'body_container' and 'footer', with the 'body_container' containing more div tags for the body's left content and right content.

```
<body>
  <div id = "main_container">
    <div id = "header">
      <div id = "logo_container"></div>
      <div id = "top_links"></div>
    </div>

    <div id = "body_container">
      <div id = "left_content">
        <div id = "new_category"></div>
        <div id = "existing_category"></div>
      </div>
      <div id = "right_content"></div>
    </div>

    <div id = "footer"></div>
  </div>
</body>
```

Figure 8: Basic Structure of the Back End User Interface

7.2.2 Implementation of Design

CSS has been used to control the look and feel of the layout. The predominant colours used for implementing the layout were black, gray, white and sky blue. The style of the cursor was also implemented in a way that if the operator hovers on a link, the cursor will change from an arrow style to a pointer style to let the operators know that they can click on that link. Apart from the cursor style, the colour of the link also changes from white to sky blue when hovered. This will make the hovered link stand out from the other links found in the user interface. However, the sky blue colour was later dropped since it does not conform to the Br&.ish's colour scheme.

7.2.3 Adding New Categories

In implementing the functionality for adding a new category, HTML's `<form>` tag has been used, which is used to gather different inputs from the user. The `<form>` tag contains an `<input>` tag of type 'text' for the name of the category to be added and a `<textarea>` tag for the description of the category. Radio buttons have been implemented for specifying whether the category to be added is a parent category or a subcategory. The author wanted to implement hidden divs so that only when the radio button for subcategory has been selected, a hidden div that displays all the different categories will be displayed. This functionality has been implemented by using JQuery's `show` and `hide` functions. Figure 9 shows how JQuery was used to control the visibility of the 'parent_categories' div tag. The div tag is displayed when the radio button for subcategory is clicked and hidden when the radio button for parent category is clicked.

```
$('#child').change(function() {  
    $('#parent_categories').show();  
});  
  
$('#parent').change(function() {  
    $('#parent_categories').hide();  
});
```

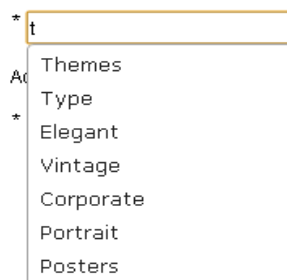
```
<div id = "parent_categories" style = "display:none">  
    //code to display the parent categories  
</div>
```

Figure 9: Displaying and Hiding the div for the Subcategory Functionality

Once the operator finished filling out the forms for adding a new category, the operator can either click the 'Clear' button or the 'Add' button. Clicking the 'Clear' button will remove the details while clicking the 'Add' button will result in data to be submitted for form validation. In addition, an auto-complete feature has also been implemented by using JQuery's *autocomplete* function. This has been implemented in order to provide suggestions to the operator if they type a category name. Figure 10 shows the screenshot of the actual output from the user interface if the operator types a letter in the category name.

Add a new category:

(Note: HTML tags are ignored)



* t

- Themes
- Type
- Elegant
- Vintage
- Corporate
- Portrait
- Posters

Figure 10: Auto-complete feature for adding a category name

7.3 Increment 3

In this increment, the functionality for categorising the templates is now fully functional, with interaction to the database occurring. In addition, the functionality for editing the template's categories has also been partly implemented through the use of AJAX. The use of AJAX has not been mentioned in section 3.3 because this language was only decided to be implemented during the project's third cycle. AJAX is a client side scripting language that gathers data from the web server and displays it to the web browser, without the need for a page reload. An advantage of displaying the data automatically is that the operator's workload in interacting with the user interface will be lessen as they do not need to click a button if ever they want to view some data.

7.3.1 Improved Implementation for Categorising Functionality

When the operator adds a new category, an alert message will appear to inform the operator that it has been added. Once added, the category will appear on the list of existing categories. Different forms of validations have also been implemented in this increment. In addition, displaying the categories and its subcategories hierarchically has also been implemented. To see the actual initial design layout for the complete categorising functionality, see Figure 28 in Appendix D.

7.3.1.1 Form Validation

Since one of Norman's design principles is to provide the operators with feedback to their actions, different types of form validation have been implemented in the forms of error messages and alert box message. This section will present examples of the different types of validation that the author has implemented. Figure 11 shows the actual output from the user interface if ever the operator tried to click the 'Add' button without providing a category name and category description. The error messages are displayed in red to emphasise to the operators that they need to provide those details.

The error messages will disappear once the operator typed something in the two fields. This has been implemented through the use of JQuery's *validate* function. Figure 12, on the other hand, shows the alert box message that is displayed to the user interface if ever the operator tried to add a new category name that already exists in the database. When the operator clicks on the 'Add' button, a function for checking duplicates gets executed. If the query that is returned by the function returns one row, that means the category name already exists in the database.



Figure 11: Validation for adding a new category

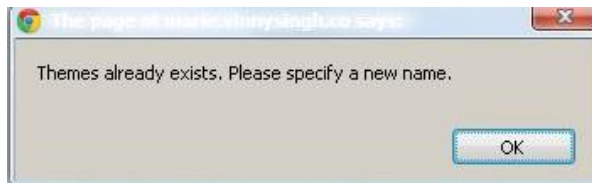


Figure 12: Alert box message for existing category names

7.3.1.2 Sanitising User Input

Initially, no sanitisation of user input has been implemented by the author. However, when the system was tested by Virinder, he entered HTML tags and special characters in the category's name and description. In return, these special characters were stored in the database table. In order to avoid random characters to be stored, inputs coming from the operators are sanitised to ensure that the category names and category descriptions only contain alphanumeric characters. Figure 13 shows the snippet of code that was used to sanitise the operator's input. The category's name and description are first checked whether they contain any special characters. If they contain special characters, the special characters will be removed. Moreover, if there are trailing whitespaces before the category name and description, this will also be removed to save memory space in the database. Lastly, to observe consistency, all category names should be written with the first letter as uppercase and the rest as lowercase. If the operator types a category name in lowercase, the system will convert the first letter as uppercase.

```
//Remove any special characters
$new_category = preg_replace('/[^A-Za-z0-9\-\_]/', '', $new_category);
$description = preg_replace('/[^A-Za-z0-9\-\_]/', '', $description);

//Removed any whitespaces at the beginning of the variable
$new_category = ltrim($new_category);
$description = ltrim($description);

//Convert first letter to uppercase
$new_category = ucfirst(strtolower($new_category));
```

Figure 13: User Input Sanitation

7.3.1.3 Displaying Parent Categories and Subcategories

One of the main functionality for this system is to display the categories and its subcategories. This would be easily achieved if a subcategory does not have further subcategories, meaning there is only

one level of relationship between a category and subcategory. However, this project requires that a subcategory can also have subcategories so multiple levels of subcategory relationship will exist. The author initially had difficulty in implementing this functionality as only the first level of category and subcategory relationship was displayed. After extensive online readings on PHP and MySQL, this functionality has been successfully implemented in the end.

```
function loopCategories($sqlQuery, $root, $depth){
    $numRows = 0;
    $x = 0;
    while ($displayCategories = mysql_fetch_array($sqlQuery)){
        if($displayCategories['Parent_Category_ID'] == $root){
            echo "<option value = \"" . $displayCategories['Category_ID'] . ">";

            while($x<$depth){
                echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~";
                $x++;
            }

            echo $displayCategories['Category_name'] . "</option>";

            if(mysql_num_rows($sqlQuery) > 0){
                mysql_data_seek($sqlQuery, 0);
                loopCategories($sqlQuery, $displayCategories['Category_ID'], $depth+1);
            }
        }
        $numRows++;
    }

    if(mysql_num_rows($sqlQuery) != $numRows){
        mysql_data_seek($sqlQuery, $numRows);
    }
}
```

Figure 14: Displaying the categories hierarchically

Figure 14 shows the snippet of code as to how existing categories are to be displayed. In order to display all the levels, a recursive function is needed to be implemented. This function gets executed iteratively until all level of category and subcategory relationships have been displayed. The first time the function is executed, the variable `$root` will have the value `NULL` and `$depth` will have the value `0`. If the category's *Parent_Category_ID* is `NULL`, the category is a parent and will be printed without any indentations. The function will be executed iteratively to check if the parent category has any subcategories. If the category is a subcategory, the subcategory will be printed with indentations by printing whitespaces.

7.3.1.4 Categorising the Template

When a template has been categorised to different categories, the *Category_ID* of the selected categories as well as the *Template_ID* of the template will be stored in the *Template_Category* table using the '*insert statement query*'. After clicking the 'Submit' button, an alert pop-up box will inform the operator that the template has been categorised to the selected categories.

However, an issue was encountered if templates are categorised in to two categories – a category that the template has been categorised already and a category that the template has not been categorised into yet. The system will inform the operator that the template has not been added into the selected categories; however, this is incorrect. In order to fix this problem, the list of categories (existing categories), from the *Template_Category* table, that the template belongs to and the list of chosen categories that was selected by the operator, should be stored in separate arrays and be compared. PHP's *array_diff* function was used in order to get the difference between the existing categories and the chosen categories. If the function returned a difference, this difference will be added to the *Template_Category* table.

7.3.2 Displaying Template's Details

Part of the implementation is to integrate this system to the back end user interface in project 3. Uthayam's system in project 3 needs to store the template's details to the database system. Once they are inserted, these data needs to be retrieve by using the '*select statement*'. The select query is used to select a particular row in the table. The template's details such as its name and image need to be displayed in the author's system so it can be categorised into one or more categories. The JQuery code for displaying the template's image has been implemented in project 3 so the same code has also been used in this system.

7.3.3 Editing Templates Functionality (Editing Categories)

Elliot had mentioned during the second project cycle demonstration that it would be beneficial if the user interface will allow them to edit the actions that they have initially made. To incorporate this feedback, the editing templates functionality had been partly implemented in this increment using AJAX. There was an issue however because AJAX was a scripting language that was not used by the author in the past so background readings have to be made in order to implement this functionality. Figure 15 shows the snippet of code for displaying the template's information depending on what select option was click. Every time the operators select a template name, the function *showInformation* will be called. This function will call two other functions, *showInfo* and *showImage*. The first function is responsible for displaying the list of categories the template belongs to and does not belongs to while the latter function is responsible for displaying the template's image. To see the actual initial design layout for the editing templates functionality, see Figure 29 in Appendix C.

```
<select id = "view_details_templates" name = "templates" onchange = "showInformation(this.value)">  
    function showInformation(string){  
        showInfo(string);  
        showImage(string);  
    }  
</select>
```

Figure 15: Function for displaying the template's details and image using AJAX

7.4 Increment 4

In this increment, additional improvements have been added in the categorising functionality and editing templates functionality is now also fully functional with new features added such as allowing operators to add keywords to the template, remove keywords from the template or delete any templates that were uploaded. Deleting categories functionality as well as viewing the complete details of a template have also been implemented in this increment. Moreover, additional changes have also been introduced to the database and finally, the design for the back end user interface has also been updated to look similar with the front end user interface.

7.4.1 Implementation of Design Changes

The changes in the CSS means that the design of the back end user interface is now more minimal and simple, with white and gray as its predominant colours. Black was use as the default font colour for all texts whereas gray was used as the default font colour for pages that are not active. Also, when a link or a button is hovered, the colour of the hovered link or button has now been changed from sky blue to black in order to be consistent with Br&.ish's colour scheme.

To implement the additional column for the page description that was mentioned in section 6.6.1, a new HTML's div tag has been added in the script file to hold this content. Different pages all have different descriptions to guide the operator as to what he or she needs to do with the interface. In addition, tooltips have also been implemented whenever a link is hovered to provide additional instructions to the operator.

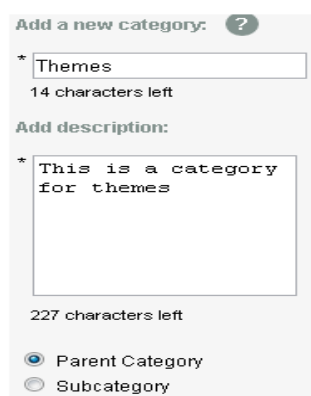
7.4.2 Changes in the Database

If we look back in Appendix B, all the tables have been normalised from first normal form up until third normal form. This means that all attributes only contain one value. However, during this increment, the system in project 3 needed to store multiple values in some attributes. For example, in defining a text parameter, the operator can defined multiple font colours and font types for the text that is to be displayed in the template. This means that the *Font_Colour* attribute of the *Parameter* table can have 'blue, red, green' as its values and the *Font_Style* attribute can have 'Arial, Calibri, Times New Roman' as its values.

By normalisation rules, separate tables should be created for any attributes containing multiple values; however, this means that more 'table joins' would be performed in order to retrieve the data and this can slow down the reading process. Also, more rows will be created since one template can have multiple colours and this can increase the size of the database. Therefore, a decision among the development team has been made to denormalise the *Parameter* table to allow multiple values to be stored in one attribute. This would benefit the system in project 1 as Virinder will only need to write a very simple query to display the different font colours and font types of a template, without the need of an additional join query.

7.4.3 Additional Feedback in Adding New Categories

Since there is a specified length of characters in each attribute in the database tables, the operators need to be informed about the number of characters they are allowed to type in for the category's details in order to incorporate Norman's design principle of system visibility. The limit for the category's name is 20 characters while the limit for the category's description is 256. Figure 16 shows the actual output for displaying the number of characters left.



Add a new category: ?

* Themes
14 characters left

Add description:

* This is a category for themes
227 characters left

☒ Parent Category
☐ Subcategory

Figure 16: Updating the number of characters left in adding a new category

The number for displaying the characters gets updated every time the operator releases a key in the keyboard. This is achieved by using JQuery's *keyup* function to bind the subtracting of the number of characters type in each key release from the actual character limit. By providing this feedback to the operator, they will get a clear idea if the category name and description that they will provide is still less than or equal to the character limit.

7.4.4 Deleting Categories Functionality

As first mentioned in section 6.6.4, the delete categories functionality will be implemented as a pop-up box. When the operator clicks on the 'delete categories' link, the pop up form for deleting categories will be displayed while the main page for categorising the templates will be hidden temporarily. This is achieved by using JQuery's *fadeIn* and *fadeOut* function. Figure 17 shows the snippet of code and the actual output for displaying the pop-up form for deleting categories. The design layout of the pop-up box has been modified using CSS. The main page will be displayed again once the operator clicks on the 'x' button or the 'OK' button in the confirm dialogue box. If the operator wants to delete a category and that category has subcategories, the subcategories will also be deleted. This has been implemented by using the *ON DELETE CASCADE* constraint of MySQL.

```
//open the popup for deleting categories
$("#popup").click(function(){
    $("#main_container").hide();
    $("#popup_form").fadeIn(1000);

    //position form at center
    $("#popup_form").css({
        left: ($(window).width() - $('#popup_form').width()) / 2,
        top: ($(window).height() - $('#popup_form').height()) / 2,
        position:'absolute'
    });
});

//close popup for deleting categories
$("#closeButton2").click(function(){
    $("#popup_form").fadeOut(500);
    $("#main_container").show();
});
```

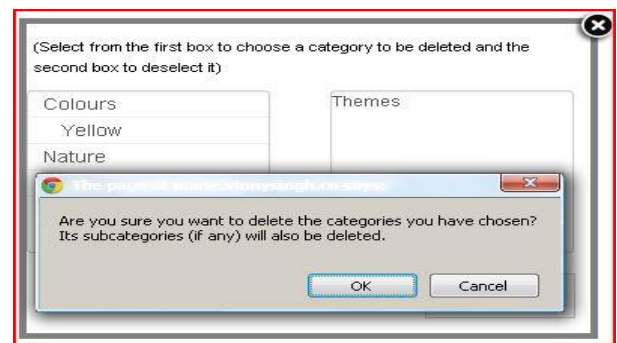


Figure 17: Deleting categories functionality

7.4.5 Improved Editing Templates Functionality

Apart from allowing the operators to edit the template's categories, additional functionalities for editing the templates have been further implemented in this increment. These additional functionalities will be further described in the following subsections.

7.4.5.1 Adding Keywords Functionality

Since the front end user interface will have a search function, keyword functionality needs to be implemented. The keywords are helpful if the charity sector client wishes to search for specific templates. This functionality has also been seen in the existing systems that were discussed in section 3 namely; 500px and Flickr. Figure 18 shows the actual output from the user interface if the operators click on the 'Add keywords' link and chooses a template. Once the operator has finished typing the template's keyword, he or she can click the 'Add' button and the system will display an alert box message as a confirmation of the operator's action. These keywords will be added to the database system through the use of 'insert statement query'.

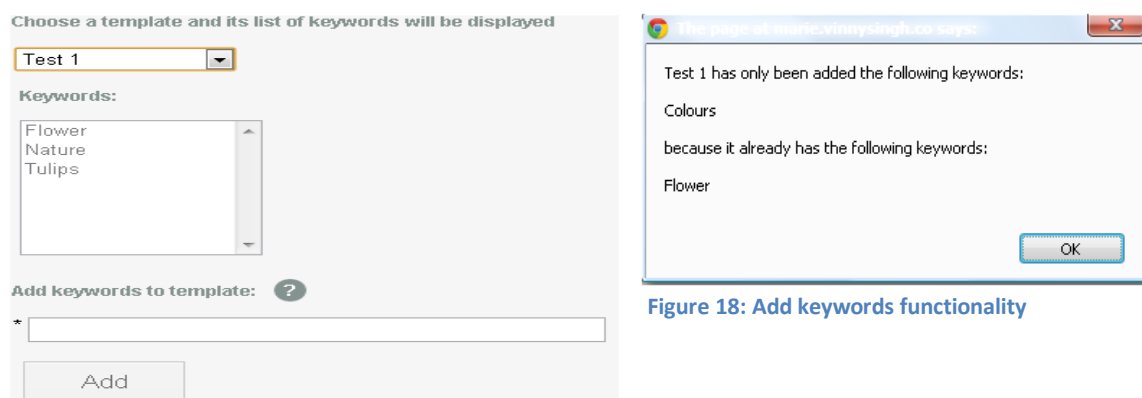


Figure 18: Add keywords functionality

To avoid duplication in the *Template_Keyword* table, the system will check if the keywords that were typed match the one that were already stored in the database. If the keyword that the operator typed already exists in the database, the alert box message will display a confirmation that the keyword already exists and this keyword will not be added in the *Template_Keyword* table. PHP's *array_diff* function was also used in order to get the difference between the existing keywords that the template already has and the new keywords that the operator will add. The difference that this function will return will be inserted in the *Template_Keyword* table.

7.4.5.2 Removing Keywords Functionality

At the same time, if the system allows the operator to add keywords to the template, it should also allow the operators to delete any keywords if ever they want to make some changes. These keywords are deleted from the database system through the use of '*delete statement query*'. Figure 19 shows an alert box message that displays a message to the operator that the keyword that they have selected has been deleted.

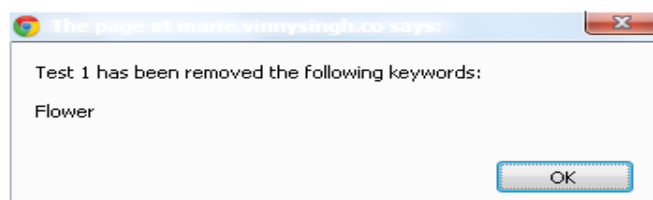


Figure 19: Alert box messages for removing keywords functionality

7.4.5.3 Improved Editing Categories Functionality

The implementation for the editing categories functionality has also been fully implemented in this increment. Figure 7Q shows a screenshot of the actual output from the interface if the operator clicks on the 'Edit Categories' link. Once the operator has finished in editing the template's categories, the 'Submit' button can be clicked. Clicking the 'Submit' button will cause an alert box to be displayed that confirms the operator's action about the changes that he or she has made. If we compare the final design layout in figure 20 from the initial design layout found in Figure 29 of Appendix C, we can see that the two choices for editing the categories are now placed side by side instead of displaying it in one column. Placing them side by side makes it more readable and in addition, if ever the number of categories stored in the database increases, an automatic scrollbar will also be displayed in the page.

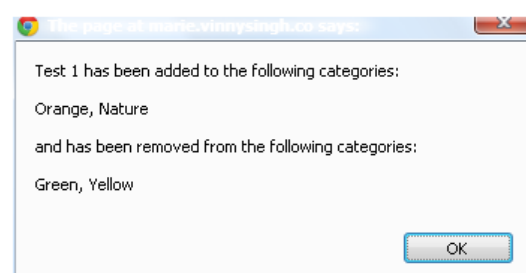
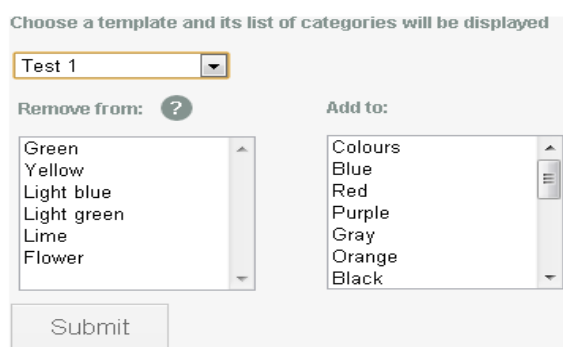


Figure 20: Editing categories functionality

7.4.5.4 Deleting Templates Functionality

The system also allows the operators to delete any templates that they have uploaded to the database. Figure 21 shows the screenshot of the confirm dialogue box that double checks with the operator if they really want to delete the template that they have selected. Once a template has been deleted, the database will check the other tables that have an 'ON DELETE CASCADE'

constraint. If we look back again at Figure 27 in Appendix B, several tables have the template's primary key as their foreign key. The template's keywords, parameters and images will also be deleted since there is no point in keeping these details from the database once the operator deletes that template.



Figure 21: Deleting templates functionality

7.4.6 View Details Functionality

Finally, functionality for viewing the details of the template has been implemented in this increment. This functionality allows the operators to view the details of the template they uploaded such as its keywords and categories. Figure 22 shows the screenshot of the actual user interface when the operator clicks on the 'View Details' link. If the template has not been categorised to any templates, the page will display 'No categories added'. On the other hand, if the template does not have any keywords, the page will display 'No keywords added'. This has been implemented using PHP's *mysql_num_rows* function. If the function returned 0, the query did not return any result and the appropriate message will be displayed.

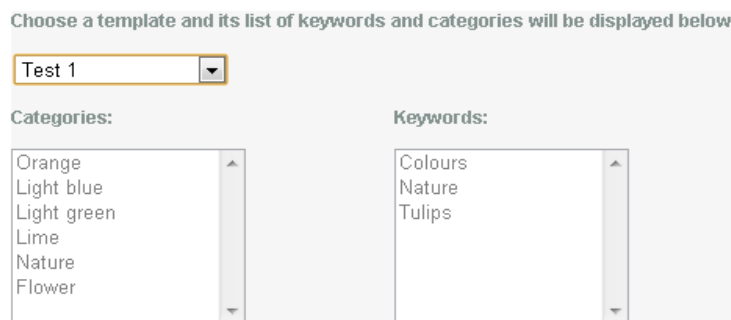


Figure 22: Viewing the template's details

7.5 Increment 5

In this final increment, an additional functional requirement has been implemented as part of Elliot's suggestion in improving the categorising functionality of this system during the fourth project demonstration. Final changes have also been introduced to the database and an additional link has been added in the edit templates functionality in order to allow the operators to edit the defined parameters of the template by redirecting them to the back end user interface of Project 3. Finally, the CSS design of the login and logout functionality, which was implemented by project 3, has also been implemented in this increment.

7.5.1 Colour Picker Functionality

During increment 4, Elliot had suggested the possibility of implementing a colour picker functionality to categorise a template into different colours instead of adding new colours to the category list. This functionality would be desirable to the operators as they will not have to manually add colours to and will therefore speed up their working process. Since there is a wide resource available to implement this functionality, the author has decided to implement this. An issue that was encountered in implementing this functionality is that numerous JQuery colour plugins that can be

downloaded are now available to use. However, this lacks complexity as the author wanted to create the colour picker functionality from scratch. Upon doing different background readings, the author stumbled upon html's `<area>` tag. This tag is responsible for specifying the different coordinates of a specific area, to where a user can click on, from an image. The image that was used for the colour picker has been created from scratch using Macromedia Fireworks software. Figure 23 shows the screenshot of the colour picker functionality that was created for this system. The operators can choose from 20 different colours, which have been predefined and inserted into the *Category* table in the database. If the operator clicks on a colour, the colour's name will be printed to inform the operator of their choice.



Figure 23: Colour picker functionality

7.5.2 Design Implementation of Login and Logout Functionality

The system in project 3 has implemented a login and logout page for Br&.ish in order to authorise the access to the back end user interface. If the username and password that were provided by the operator match the username and password that were stored in the *Operator* table in the database, the operator will be redirected to the back end user interface. Initially, it was thought that this functionality will be implemented by Virinder in project 1 since he also had to implement the login functionality for the charity clients. However, there has been a misunderstanding since the login for the clients are independent from the login of the operators. In order to implement this on time, the author has decided to design the login and logout functionality through the use of CSS while the validation is being implemented by Uthayam at the same time. By doing the design and implementation concurrently, the login and logout functionality has been completed at a much faster pace. The first image in Figure 24 shows the screenshot of the design for the login functionality while the second image shows the design for the logout functionality. If the operator clicks on the logout button, a confirmation dialogue box will pop up to double check their actions, as seen in the third image in Figure 24. Clicking 'OK' will log out the user whilst clicking 'Cancel' will not log out the user. To see the final design layout for the categorising templates functionality, see Figure 30 in Appendix C.

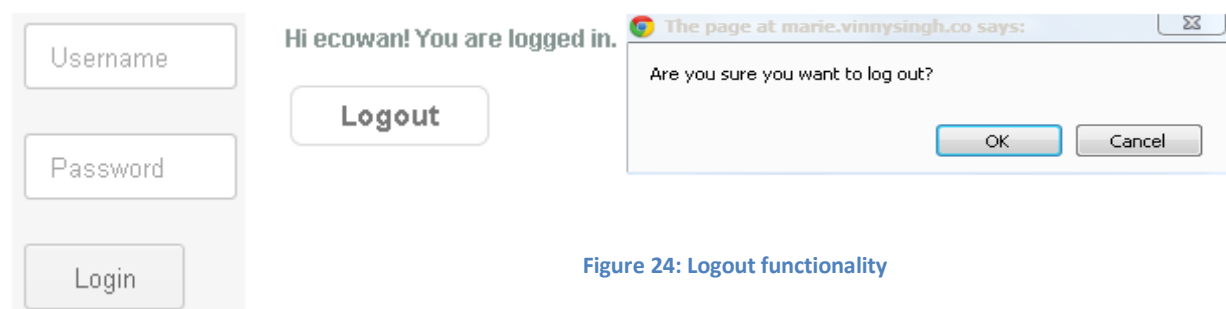


Figure 24: Logout functionality

7.5.3 Edit Parameters Functionality

Apart from allowing the operators to add keywords to the template, remove keywords from the template, edit the template's categories or deleting a template itself, it has also been agreed by Uthayam and the author that the operators should be allowed to edit the parameters of the

different templates that they have uploaded. To implement this additional requirement, a new link for editing the template's parameters has been added. Figure 25 shows the screenshot of the actual user interface when the operator clicks on the 'Edit Parameter's link. If the operator selects a template and clicks on the 'Edit' button, the operator will be redirected back to the system in project 3 wherein they can perform the parameter changes. To see the final design layout for the editing templates functionality, see Figure 31 in Appendix C.

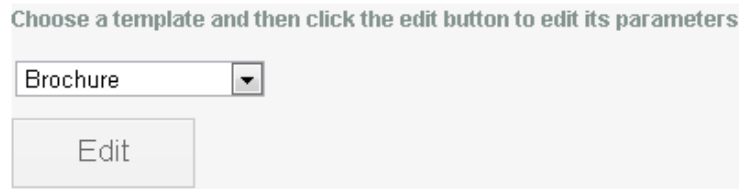


Figure 25: Edit parameters link

7.5.4 Final Changes in the Database

In order to support the colour picker functionality described in section 7.5.1, an additional attribute called '*Category_type*' has been added in the Category table. This attribute determines the type of the category – whether it is of 'text' type or 'colour' type. This was added in order to have a clear distinction between the manually added categories and the predefined colour categories. Looking back at section 7.4.4, the functionality for deleting categories has been implemented and in order to avoid deleting the predefined colour categories, the categories that are only available to be deleted are the categories of type 'text'.

Moreover, the system in project 3 needed an additional attribute to hold information about the different pricing options (i.e. whether it is 'bronze', 'silver' or 'gold') for a template; hence, '*Price_options*' attribute have been added in the Template table. Also, since a template can have multiple price options, the attributes *Price_options* and *Basic_price* have been denormalised in order to handle multiple values.

8. Testing

This section will now discuss the different testing performed in this project. Since implementation was broken down into different increments, the testing stage has also been performed iteratively in each cycle. This section will cover the different types of testing that were used such as unit testing, functional testing, integration testing, browser compatibility testing and usability testing. If errors or bugs in the system have been identified, appropriate changes in the code have been made in order to fix the errors. Elliot has been involved in some of the testing such as functional and usability. The different functional requirements that have been tested in the functional testing have been presented to him through the project cycle demonstration meetings. Meanwhile, a working link of the latest version of the system has been sent to him in order to test its usability.

In addition, the complete testing results have not been attached in this report as it will take up a lot of space since the testing performed was extensive. These testing results have been submitted as a different document. Instead, only the important test results will be shown, highlighting any errors that were found.

8.1 Increment 1 and 2

Since the implementation for the first increment involves the creation of the database tables, the testing that was involve is to double check the different integrity constraints of the tables such as its primary keys, foreign keys, NOT NULL constraints and ON DELETE CASCADE constraints. Each attribute was double checked if it has the correct data type and length as well as checking whether or not it is ok to have a NULL value.

The testing in Increment 2 only involves validating the design layout that was implemented by using an online mark up validation service provided by W3C (World Wide Web Consortium). The validation service checks the HTML code whether it conforms to the valid and correct way of writing. Validating the HTML code is important because it checks whether you are conforming to the HTML5's writing style and not to the older versions such as XHTML. Most of the errors found were the usage of the same ID selector to different HTML elements instead of using class selector. In HTML, selectors are used to identify a HTML element that is needed to be styled using CSS. However, the id selector only works for a single HTML tag. In order to validate this error, the id selector was changed into a class selector since class selectors are use for styling a group of HTML elements. After correcting the other errors that were found by W3C and then validating the HTML file again, W3C will acknowledge that the page has been successfully validated and will returned no errors.

8.2 Increment 3

Since more implementation has been made in Increment 3, this means that unit testing and functional testing will be performed to check if the added functionalities are correct. The main testing that has been performed is testing the categorising functionality of the system since this functionality has been completed in this increment. In addition, integration testing has also been started since the third increment involves integration with the system in project 3. Usability testing was also performed to check if the back end interface is easy to use and navigate. The following sections describe the different results from the different types of testing that has been performed.

8.3.1 Unit Testing

The main aim of unit testing is to test the system from the author's point of view. Unit testing is more technical and involves testing the individual functions that were written in the implementation code. In order to test the categorising functionality, different test or sample data have been provided. Table A in Appendix D shows some of the unit testing results in this increment.

The different testing actions when adding a new category range from straightforward actions such as providing a category name and description, specifying if the category is a parent or subcategory and then clicking the 'add' button afterwards. Apart from these, other actions such as clicking the add button without completing the required fields, adding a category name that already exists and also providing category names with special characters have been performed. All of the testing actions performed have provided correct results so no further corrective implementation was made.

Meanwhile, the different testing actions when choosing from existing categories range from selecting one or more categories in categorising the template and categorising the template to a previously selected category. As mentioned in section 7.3.1.4, an error was found if the template has been categorised to two categories. For example, if a template has been categorised to 'Themes' and 'Vintage' but the template has already been categorised to 'Themes', the system displayed an

alert message informing the user that the template has already been added in those two categories. To fix this error, PHP's *arraydiff* function has been used. This function compares two arrays and takes the difference. The difference that this function returns is therefore inserted to the database.

8.3.2 Functional Testing and Integration Testing

If unit testing is testing the system from the author's point of view, functional testing is testing the different functional requirements from the client's perspective. To test the system's categorising functionality, each functional requirement related to the categorising functionality has been tested thoroughly. These functional requirements include displaying the template details that were uploaded by the operator, allowing the operators to add a new category or subcategory and also allowing the operators to categorise a template to one or more categories. These different functional requirements were shown to Elliot and have all been met so no further improvements have been made to these requirements.

Apart from functional testing, integration testing was also performed in order to test if the integration with Uthayam's system runs smoothly. The integration testing that was performed was mostly to check if correct values were inserted to the different database tables. The only minor issue that was found is the length of some of the attributes. For example, the length for the attribute *File_name* in the *Template_Filename* table has been adjusted to 50 from 20. This was changed because if the template's filename is longer than 20 characters, the image will not be displayed as only the first 20 characters of the filename will be stored to the database.

8.3.3 Usability Testing

In order to check if the user interface that was implemented in the third increment is easy to use, a questionnaire has been sent to Elliot in order to get his feedback. The questionnaire was sent a couple of days prior to the actual third project cycle demonstration meeting in order for Elliot to play around with the system before presenting it on the project demonstration. However, no feedbacks were received from Elliot so the author had to ask a couple of students to test the system instead. The students were observed as to how they think the system would work and both of them have actually found the user interface to be confusing because of the user interface's lack of description. In addition, there were no clear instructions as to what they should do next so the author had to actually guide them in using the system. These feedback results have been noted down and were implemented in the fourth increment in order for the system to be more usable and easy to use.

8.4 Increment 4

Increment 4 covers most of the testing stages as more functionality was tested in this increment. Browser compatibility testing was performed to check if the system behaves the same in other browsers. The following subsections describe the different results of the different testing processes that have been performed.

8.4.1 Unit Testing

Table B in Appendix D shows some of the unit testing results in this increment. The different testing actions in this increment include checking if the number of characters left in typing a category name and description is correct and checking if correct messages are displayed in the alert box depending on the action performed. Additional checks have also been performed to see if the template contains the correct keywords and categories. Most of the testing actions performed have provided

correct results; however, an error was found when a category is to be deleted. If a category has been selected and the 'delete' button has been clicked, the deleted category will still appear if the 'delete categories link' has been clicked again because the page did not load properly. Once the page has been refreshed, the deleted category will be removed. To fix this error, an *onLoad* listener tag has been attached to the `<body>` tag to ensure that the page gets loaded properly. In addition, an error was found if the template does not have any keywords. If a template does not have any keywords, an empty space was displayed in the page. However, this is incorrect because it does not inform the users. To fix this error, PHP's *empty* function was used. This function checks if the array, to which the SQL query is assigned to, is empty. If the array is empty, the template does not have any keywords. Therefore, the message 'No keywords added' is to be echoed. The same fix was performed if the template has not been categorised to different categories.

8.4.2 Functional Testing and Integration Testing

Most of the functional requirements that were listed in section 5.2.2 have been implemented in the fourth cycle so each of these requirements was fully tested. These requirements include allowing the operators to add keywords to a template, deleting keywords from a template, editing the template's categories, deleting any templates or categories, viewing the complete details of a template and allowing the operators to upload another template by redirecting them back to the user interface of project 3. These different functional requirements were again shown to Elliot and have all been met so no further improvements have been made to these requirements.

Meanwhile, the integration testing that was performed in this increment was checking if correct values were still inserted to the different database tables. In terms of the user interfaces of the different projects, consistency was observed to check if each interface is similar with each other in order to perceive a uniform integrated system. This involves checking if the same colours, font styles and sizes were used as well as checking if the layout structure and links positioning are the same.

8.4.3 Browser Compatibility Testing

The back end user interface for this project has been tested in four different browsers; namely, Google Chrome, Mozilla Firefox, Safari and Internet Explorer. Initially, the editing templates functionality of the system only works in Google Chrome. If the operator clicks on 'Add Keywords' link, type the keywords and then clicks the 'Add' button, the page is not being submitted properly. The same error was observed when the operator also tries to delete a keyword or edit a template's categories. This error occurred because the submit button was in a different form from the textbox input for the keywords. After doing fixes in the code and placing the submit button in the same form, the editing templates functionality is now working in all major browsers. Moreover, in terms of the interface's appearance, the layout is consistent and similar in all the browsers.

8.4.4 Usability Testing

During the fourth project cycle meeting, the link for the latest working integrated system of both the front end and back end interface was sent to Elliot. Through Skype's "screen sharing functionality", the development team were able to observe as to how Elliot will use the system. No instructions have been provided to him in order to know if he would be able to navigate the system easily. Upon observing, Elliot initially provided the template's description in the description box for adding new categories; however, he changed this after realising that the description is for new categories. In terms of categorising the template, he was able to complete the task easily. Moreover, he tested the

functionality for adding keywords to a template and was also able to complete the task easily. One suggestion that Elliot provided was the ability to drag and drop subcategories to parent categories rather than adding them hierarchically. Since this suggestion was only introduced in the fourth cycle, the author has informed Elliot that due to time constraints, this functionality would not be completed on time.

A week after the fourth cycle meeting, Elliot and his team at Br&.ish tested the system again and sent back the results for the questionnaire that was sent to him. The feedbacks were mostly positive and he mentioned that categorising and deleting the templates were very easy to do. However, when asked about the ease of use in navigating the system, Elliot answered 'Slightly Easy'. To further improve the usability of the system, question mark icons beside the functionality were displayed in the interface. When these icons are hovered, instructions related to that functionality is printed. In addition, Elliot also suggested the possibility of having a colour picker functionality so he can just pick from predefined colours rather than adding the colours manually. Therefore, this functionality was implemented in the next increment, as seen in section 7.5.1. In terms of editing the categories of a template, Elliot had mentioned that there are a lot of blank spaces in the page; so, the layout for this functionality has been improved to take into account his feedback. In terms of the overall design of the system, Elliot had mentioned that the system looks exactly like a back end system, which is what the author was aiming; however, he mentioned that the font styles used were basic.

8.5 Increment 5

Increment 5 covers the final testing procedures performed for this project. The following subsections describe the different results of the different testing processes that have been performed.

8.5.1 W3C HTML Validation Testing

All of the HTML files that were created have been validated by using W3C's online validation service. As mentioned previously, this was first performed in increment 2. This has been performed to ensure that correct HTML syntax has been used in coding. Some of the errors that were found include missing closing HTML tags and not including the 'src' attribute in the `` tag. These errors have all been fixed and W3C had acknowledged that all files have been successfully validated.

8.5.2 Unit Testing

Since the colour picker functionality was implemented in this increment, additional unit testing was performed to test this functionality. Table C in Appendix D shows the unit testing results in this increment. Most actions related to the colour picker functionality return correct results; however, deselecting functionality was not provided if the colour was clicked again. In order to fix this error, a new variable that increments every time the colour was clicked, was added. If the variable is equal to 1, the colour was clicked for the first time and therefore, the colour's name will be displayed in the page. However, if the variable is equal to 2, it means that the colour was clicked again. The colour's name is therefore replaced with an empty space to display that the colour has been deselected. Also, if the variable is equal to 2, the variable will also be set back to 0 so that the next time the colour is to be clicked again, the colour's name will be printed.

Apart from the colour picker functionality, the logout button and the 'Edit Parameters' link have also been tested. As seen in table C in Appendix D, the testing actions returned positive results.

8.5.3 Functional Testing and Integration Testing

Three more functional requirements have been completed and tested in this increment. These requirements include providing a colour picker functionality to categorise the template, allowing the operators to edit the template's parameters and upload a new template by redirecting them back to the user interface of project 3. These different functional requirements were again shown to Elliot and have all been met so no further improvements have been made to these requirements.

Similar with the previous integration testing from the last increment, the integration testing that was performed in this increment was to check if correct values were still inserted to the different tables. In addition, the size of the attributes *font_style* and *font_colour* of the Parameter table has been increased from 20 to 50 to accommodate the length for multiple values.

9. Integration

This section covers the overview of the system integration since this project does not only cover individual work but also, it covers aspects of system integration to combine all three projects into one that will be delivered to Br&.ish. The following subsections also discuss the design choices that were made in doing the integration, the actual integration plan that was proposed and lastly, the different integration issues that were encountered throughout the project development.

9.1 Integration Overview

Figure 32 in Appendix E shows the overview of the integration for Br&.ish's whole system. The main functionalities of these three individual projects have also been provided in Appendix E. For this system integration to work, the individual systems each have its own dependencies to the other systems. Firstly, for Uthayam's system to work, the author's database system has to be designed efficiently. The tables need to have the correct primary key, foreign key constraint, correct length and data type for the different tables' attributes. Secondly, for the author's system to work, Uthayam's system has to store the data to the database correctly so it can be displayed in the author's system. Lastly, for Virinder's system to work, his system needs to access the database of the author and write different queries to display the values that were stored in the database.

9.2 Integration Structure

Initially, Virinder has proposed to use a PHP framework called 'CodeIgniter' in integrating the three systems. However, because of its simplicity, the development team did not go ahead in using the said framework. In addition, Virinder also proposed the use of a version control system in order to track the changes that have been made in our individual systems once it has been integrated. However, the idea of using a version control system was never discussed again; hence, it was not used. Finally, Virinder proposed a new file directory structure again to the development team after the third project cycle. His proposal involves storing all of our codes into different folders in one directory. The separation of folders means that the back end user system and the front end user system will stay separated. The development team went ahead with this plan and successfully completed the integration to this new directory structure on the 6th February 2013.

9.3 Integration Plan

The author's integration plan was to do the integration after each project cycle. The first increment did not involve any integration as the development team focused on developing the core

requirements of the individual systems. The second increment did not involve any integration too as it focused on designing the basic layout of the interfaces and its individual functionalities. Increment 3 involved an integration plan between the author's system and Uthayam's system. This integration focused on Uthayam's system to store the uploaded template information to the author's database such as its name, description, file name and parameter values as well as allowing the operator to categorise the recently uploaded template to different categories. Virinder was not involved in the integration as his third increment focused on implementing his high fidelity designs. The fourth increment involves a complete integration plan among the development team. Since all the codes of the three individual systems are all placed in the new directory structure, the integration should have run smoothly. However, several issues have arise which lead to the integration plan not being followed. These integration issues are discussed in the following section.

9.4 Integration Issues

Even though an integration plan has been proposed, this plan was not followed as strictly as expected because the integration encountered several issues. Firstly, Virinder's system was integrated on increment 4 which meant that any issues that he encountered with his system were known at a very later stage. Secondly, Virinder also encountered issues in displaying the data to his system because of his SQL query skills. This should not be a problem in the first place since the development team studied Database Systems previously and had a few similar database coursework to work on. Looking back on the prerequisite skills needed in this project, Stefan has also mentioned that one of the key skills needed is to be able to use SQL.

9.5 Best Practice for Integration

To prevent any integration issues in the future, the development team must have integration meetings regularly and must check that the integration plan is followed strictly. If the integration was not followed according to plan, a contingency plan must be made. The development team must also have the set of technical skills that are needed in order for the whole project to be completed. In addition, since the development team will also have other coursework deadlines, priorities and time management have to be made to ensure that the project's schedule will be met.

10. Discussion

This final section will discuss the result and evaluation of the project. Different requirements that have been implemented will be summarised, with important results highlighted. The different achievements that have been achieved by the author will also be covered in this section. Novel ideas that have been implemented, which are not found in the existing works that were researched, as well as the system's limitation and future developments will also be discussed. Finally, this section will also discuss the final evaluation from Elliot in order to know if this project has been a success.

10.1 Project Evaluation

The different requirements that have been proposed will be evaluated in this section. It is important to evaluate these requirements because this will serve as an indicator as to how well the project has met the user's requirements, which in turn, will indicate how well the project has achieve its aims.

10.1.1 Functional Requirements

The functional requirements that have been listed in section 5.2.2 and 5.2.2.1 will be evaluated in this section to see how successful the system has implemented them.

10.1.1.1 Adding new categories

The functionality for adding new categories and subcategories has been implemented in this project. Through the use of HTML's `<form>` tag, the author is able to provide a way for operators to input the details of the category they wish to add. As forms are susceptible to SQL injection attacks, the inputs provided by the operator have been sanitised to ensure that only alphanumeric characters will be inserted in the *Category* table. If the category to be added is a subcategory, this option has been provided through the use of HTML's radio button and JQuery. To ensure that the subcategory to be added will have the parent category's ID in its *Parent_Category_ID* attribute, unit testing was performed to check if this is correct. Moreover, the functionality for deleting a category has also been implemented. The SQL query that was used for deleting a category is correct as it also deletes any subcategories if the category to be deleted is a parent category.

10.1.1.2 Categorising the template

The functionality for categorising the template to one or more categories has been implemented in this project. By normalising the database into 3rd normal form, the table *Template_Category* has been created to hold information as to which categories a template belong to. Once the operator has selected the categories, different SQL queries were performed in order to retrieve the selected categories' ID and the template's ID and then later inserting these IDs to the *Template_Category* table. Also, the 'could have' requirement for implementing the colour picker functionality for categorising the template to different colours has also been met by using HTML's `<area>` tag. The unit testing that was performed in this functionality ensures that no errors are to be found. If there are any errors, corrective measures have been performed, as seen in table A in Appendix D.

In terms of displaying the existing categories, this has been a success as the author was able to display the categories hierarchically through the use of recursion. The whitespaces that were printed before the subcategory's name provided a clear idea that it is a subcategory.

However, a 'could have' requirement that was not been able to be implemented by the author was the automatic selection of parent categories due to limited resources. The only available online resource that was found for this functionality was to use WordPress' automatic selection of parent categories; however, the use of WordPress was not in the list of Br&.ish's solutions because of its simplicity.

10.1.1.3 Editing and Viewing the template's information

The functionalities for editing the template's information have been implemented in this project. Firstly, the system allows the operators to add keywords to a template. To ensure that the keywords to be added are not duplicates, checks were performed to see if each keyword to be added exists in the table. If it already exists, this will not be inserted and the operator will be informed. Secondly, operators are also able to delete any keywords. To lessen the operator's workload, multiple keywords can be deleted at a time. Once keywords are deleted, these will be successfully removed from *Template_Keywords* table. Thirdly, the functionality for editing the template's categories has also been implemented. The system allows the operator to remove a template from any of its categories and also allows them to further add a template to more categories. The categories that

the operator wants to be removed are deleted from the *Template_Category* table while the categories that are further added are inserted to the same table. Finally, the functionality for deleting the template has also been implemented. Through the use of *ON DELETE CASCADE*, deleting a template will cause other related information to be deleted such as its keywords and parameters.

On the other hand, the functionality for allowing the operators to view the complete details of a template has also been successfully implemented. The system displays all the keywords and categories to which a template belongs to. If the template has not been added any keywords or categories, appropriate messages will be printed to the system to inform the operator.

10.1.1.4 Integrating with project'3 back end user interface

The functionality for displaying the template details that were uploaded in project 3 such as its name and images has been implemented. These details have been retrieved by writing different SQL queries. The latest template that was uploaded is displayed in the categorising page so that the operator can add them into different categories. If the operator wants to view all the templates that have been uploaded, the system is also able to provide this functionality.

Moreover, the system has also implemented the functionalities for integrating with the system in project 3. These functionalities include allowing the operators to upload another template and editing its parameters by redirecting them back to the system in project 3. In addition, the logout functionality that was added in this system has also been successful as it was able to redirect the operator to the login page in project 3. The integration that was performed between project 2 and 3 can be seen as a success as there is a clear flow of information between these two systems. Also, since consistency in the design layout has been observed, the two systems appear to look like one.

10.1.2 Database Requirements

The different requirements for the database that were listed in section 5.2.3 have all been met. The design of the different database tables has ensured that different information related to the charity sector clients, the operators, the templates and categories, to name a few, are stored correctly. Different constraints such as primary key, foreign key, *NOT NULL* and *ON DELETE CASCADE* have been used to ensure that any insertion, deletion or updates that will be performed will be correct.

10.1.3 Non-Functional Requirements

As mentioned in section 5.2.4, non-functional requirements are important to any projects as these measures how usable and secure the overall system is. In terms of the different user interface requirements, the user interface uses standard naming conventions for the buttons and links. Moreover, the user interface also displays appropriate alert and error messages, depending on the operator's actions. This has been evident when different actions that were performed resulted in different alert messages to be printed. Tooltips have also been implemented to guide the operators in navigating the page.

To see the result for the overall usability of the system, the link for the system has been sent to Elliot for usability testing and feedback has been provided during the fourth project cycle meeting. Based from the questionnaire results, it is apparent that the system that was implemented did not lack usability. Most of the results are positive especially in terms of categorising the template into different categories, deleting a category or template, or adding keywords to the template. On the

other hand, those questions that have 'Slightly Easy' as its answer have been looked at in order to be improved. These usability issues have been addressed in the final increment, as seen in section 8.4.4.

In terms of the browser compatibility, the testing that was performed in section 8.4.3 has shown that the system works in different browsers. Initially, the system only works in Google Chrome but after doing some fixes in the code, the system now also works in other web browsers.

Finally, security requirements have also been addressed by this system. Checks for any SQL injection attacks have been implemented by sanitising the user's input through the use of different PHP functions. In addition, different security measures such as using sessions instead of cookies and avoiding the usage of global variables have all been employed.

10.2 Project Achievements

This section will now discuss the different achievements that have been achieved by the author in implementing this final year project. This will be divided into two subsections: personal and project. The personal achievements will discuss the different achievements of the author in terms of the new skills and lessons learned. On the other hand, the project achievements will discuss how successful the project is in terms of achieving the different aims listed in section 1.1.

10.2.1 Personal Achievements

Prior to starting this project, the author has not used JQuery, AJAX and PHP's built in function in MySQL to insert or manipulate data in the database; hence, this whole project served as a new learning experience in terms of programming. During the increments, different issues have been encountered in terms of implementing the different functionalities for this project. These issues include implementing the colour picker functionality from scratch, showing the multiple levels of relationship between categories and subcategories as well as using AJAX to display the information automatically. However, through background readings, different online resources and programming practices, the different functionalities that the author initially perceived as complex were implemented. After the completion of this project, the author is now more confident in using the different languages used in web and database development. Moreover, completing this project has also improved the author's existing skills such as problem solving, time management and communication. For instance, since the author has to work closely with Uthayam to implement the whole back end user interface, team work and communication skills have been improved in order to integrate the two separate back end systems into one and also to maintain the whole design uniform. Finally, the author has experienced how to work with real life project clients. This experience has not been easy; however, it has been rewarding since this project has provided the author with the essential skills and knowledge that will be useful in the future.

10.2.2 Project Achievements

Looking back at section 1.1, the aim of this project is to design a database system and a back end user interface for Br&.ish. The database system must store different information about the templates while the back end user interface must allow Br&.ish to categorise the templates into different categories and define its properties such as its keywords. Another aim of this project is to perform system integration with projects 1 and 3. Overall, all the different aims that were stated have been achieved. The database system has been designed efficiently and different information about the templates has been stored successfully. The different SQL queries that have been written

have been successful in retrieving the different data from the database. Moreover, the back end user interface allows the operators to perform their tasks such as categorising the template and editing or defining the template's information. This is evident through the completion of the functional requirements.

Moreover, the iterative and incremental model that was used to structure the development process of this project has helped tremendously in completing the different stages for this project on time. Using this model has been the best solution as it ensures that the system will be implemented and verified with the client, even at the early stages. Moreover, this model also incorporates the addition of new requirements that have been proposed by Elliot. Although this project still encountered some issues in terms of system integration, as mentioned in section 9.4 and feedback from Elliot was also not regularly received on time, the project is still a success as it implemented all the proposed aims and had been delivered on time.

10.3 Novelty

This section will present the different functionalities that this system has implemented that are not found in the existing works that the author has researched at section 3.4. However, this section presents a limitation as the comparison will only be made with the existing systems that have been looked at.

10.3.1 Comparison to Existing Works

If we look back at section 3.4.3, the existing systems that offer similar services with Br&.ish have shown limitations with their functionalities. The first limitation is that the front end systems do not support the idea of subcategories having subcategories. This might be seen as a limited way of searching the templates. This limitation has been addressed by this system as it allows the operators to add multiple levels of subcategories. By designing the back end user interface like this, the templates will be specifically categorised; therefore, the front end users (the charity sector clients) will be able to see more subcategories and can search for the templates more specifically.

The second limitation is that the front end systems that were researched do not support the idea of having a colour picker category. VistaPrint and GoodPrint have a category for 'Solid Colours' but as mentioned previously, this category returns all the templates with solid background colours and not specific ones. This limitation has been addressed by this system by implementing a colour picker functionality that will allow the operators to categorise the template to different predefined colours. By having this functionality, the front end users will be able to see the different colours to which the template has been categorised. Clicking on a specific colour category will therefore display all the specific templates on that selected colour category.

Finally, 500px only allows their user to categorise a photo to one category. This might be seen as a limited way of searching for the photo that was uploaded by the user because it is only placed in one category. This limitation has been addressed by allowing the operators to categorise the template to one or more categories at a time. By having this functionality, the visibility of the template will be higher as it will appear in different categories.

10.4 Limitation

This section discusses the limitation found in this project in its functionality. Looking back at one of the system's functional requirement, the system must allow the template to be categorised into one or more categories. This has been implemented as operators can select one or more text categories; however, the colour picker functionality that has been implemented does not allow multiple colours to be selected. This is due to the limitation of HTML's `<area>` tag. This limitation been consulted with Elliot and he mentioned that selecting one colour at a time will be ok for the mean time.

Even though this limitation exists, the overall functionality of the system still works. If not for the time constraint, this limitation could have been implemented; so, this would be considered as a scope for future developments.

10.5 Further Work

The system that has been implemented in this project can still be further enhanced in the future. This has been evident in some of the feedback and suggestions that were given by Elliot during the project cycle meetings. The following subsections will briefly describe what these enhancements are and how could these additional functionalities benefit Br&.ish.

10.5.1 Design

From the questionnaire that was sent to Elliot, he mentioned that the design of the system is: *"Not bad, It's very basic with basic fonts, and it looks just a like backend system. It could be more 'fun' but it does the job."* Because of this comment, the look and feel of the whole system can be improved in the future by using more artistic font styles and by also creating a customised header or template design using Photoshop. In addition, since different templates are to be uploaded, the system can also benefit if it has a custom slideshow that will display some of the templates that have been uploaded. The custom slideshow will add more colours and will make the website more interactive. By incorporating these visual improvements in the future, the operators will feel more excited in using the system.

10.5.2 Drag and Drop Functionality

As first mentioned in section 8.4.4, Elliot suggested the functionality of being able to drag and drop subcategories to its correct parent category instead of adding it hierarchically. This could have been implemented; however, this functionality was suggested during the later stages of the development. This would benefit the operators as their work for adding subcategories will be lessened. For instance, if the operator added a subcategory to a wrong category, the operator has to delete that subcategory and add it again to its correct parent category. With drag and drop, the operator can just drag the subcategory and drop it to the correct parent category, without the need to delete it.

10.5.3 Automatic Selection of Parent Categories

As mentioned, the functionality for the automatic selection of parent categories has not been implemented because of the limited resources that were found. If the operator wants to categorise a template and selected a subcategory, the parent category (ies) of that subcategory should also be selected. For instance, if the operator selected the subcategory 'Vintage', its parent category 'Themes' should also be automatically selected since 'Vintage' is a 'Theme'. This would lessen the workload of the operator as they will not have to select both 'Vintage' and 'Themes'.

10.5.4 Responsive Web Design

Responsive web design has been identified as the most sought technology that most businesses will use in the year 2013. (Natda, 2013) defined responsive web design as an approach of delivering optimal viewing experience to different users who are using different platforms. As more and more users are using their smart phones and tablets in browsing online contents, more and more businesses are therefore providing a mobile or tablet version of their website. Having a mobile version of the system will benefit the Br&.ish operators because this will allow flexibility and mobility as they will be able to access the back end user interface whenever they want.

10.6 Final Feedback

In order to assess the level of success of this project, a final feedback has to be obtained from Elliot. This feedback is vital as this will indicate whether or not Br&.ish will want to use the system or use this as a prototype that will be the basis for further developments and implementation.

As mentioned in section 6.1, the 5th increment was proposed to serve as a way to obtain this final evaluation. However, the proposed meeting was cancelled by Elliot due to an unexpected family emergency which led him to travel abroad. As a contingency plan, the author had sent an email to Elliot as a way to gather his final feedback. Unfortunately, Elliot has not responded to the email so his input was not received. This is part of the risk of implementing this project as most of the time, inputs or feedback from Elliot was not received regularly. For example, the request for a web server access has not been granted and the online questionnaire has to be sent to him a couple of times before it was sent back.

Since there was no final feedback provided, the feedback that was evaluated was his usability feedback from the fourth project cycle meeting. As mentioned in section 8.4.4, most of the answers that were provided from the questionnaire are positive. Any suggestions that he provided with the design and implementation were accommodated in the 5th increment. This is evident in implementing the colour picker functionality and fixing the blank spaces in the template's categories. One question in the questionnaire was about the satisfaction with the author's overall progress and Elliot said that he is very satisfied with the progress presented to him.

10.7 Conclusion

This project focused on improving the existing practices of Br&.ish in order to provide their charity sector clients a more efficient and advanced way of searching for different marketing material templates. By looking at their existing practice in section 2, it is evident that improvements have to be made as their designers are currently designing the templates on a customer-to-customer basis. Moreover, different competitors that have been looked at such as VistaPrint, GoodPrint and Tweak allow their users to search different products through different categories and keywords. Other systems such as 500px and Flickr have also allowed their users to define pieces of information of a photograph that they have uploaded. In order for Br&.ish to keep up with their competitors, this project has introduced the implementation of a database system and a back end user interface for the operators.

The successful implementation of the database and the back end user interface has been achieved by dividing the whole project into smaller project cycles. Since the time to complete this whole project is only 6 months, dividing the workload in the different stages of the development process

ensures that this project will be delivered on time. By using the iterative and incremental model, a working version of the system is always available and presented to Elliot. This meant that in each cycle, Elliot was involved throughout the development process as he was able to provide important inputs, feedback or suggestions. Any feedbacks have been taken into account and this has been apparent through the implementation of additional requirements. In addition, the project cycle meetings also acted as a way to address any issues or ambiguity in the project.

Apart from the technical side of doing the implementation for this project, great importance has also been placed in making the system usable and easy to use. It has been mentioned that the main challenge for this project is to make a final working system that Br&.ish will want to use. To tackle this challenge, Norman's HCI design principles have been utilised and has been backed up by the results that Elliot provided in the questionnaire. Areas that did not meet the guidelines for usability have been looked at and improved.

The evaluation that was performed has proved that all the 'must have' requirements that have been proposed as well as one 'could have' requirement have all been met. The different types of testing that were performed have been extensive in order to cover all aspects of the different functionalities as well as to address error handling. The testing results have shown that the system is robust enough to handle different kinds of data. In terms of requiring an active involvement from Elliot, there were times wherein this was not met due to different emergencies and work related issues that came up from the client's side. Nevertheless, by following the different advices that the author's supervisor has given and by maintaining a good work ethic and time management, the project has still been delivered on time and has been successfully integrated with the other systems.

In conclusion, this project has been a success as it was able to achieve all the different aims that were stated in section 1.1. The system that was implemented has provided a way for Br&.ish to categorise their templates into different categories and to define and edit their marketing material templates' information. Even if this project has met its aims, there are still room for further improvements if additional time and resources have been available. As a final remark, this project has served as a way to allow Br&.ish to store their different marketing material templates that can be browsed by the worldwide charity sector.

Bibliography

Br&.ish, 2012. *About*. [Online]

Available at: <http://brand-ish.co/>

[Accessed 2 December 2012].

Bradley, A., n.d. *Server Side Scripting*. [Online]

Available at: http://php.about.com/od/programingglossary/g/server_side.htm

[Accessed 2 December 2012].

Brand Channel, 2003. *Down to the Core: Branding Not-For-Profits*. [Online]

Available at: http://www.brandchannel.com/features_effect.asp?pf_id=140

[Accessed 25 November 2012].

Codd, E., 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), pp. 377-387.

Codd, E., 1980. Data Models in Database Management. *ACM SIGMOD Record*, 11(2), pp. 112-114.

Connolly, T. & Begg, C., 2010. *Database Systems: A Practical Approach to Design, Implementation and Management*. 5th ed. Boston, Massachusetts: Pearson Education, Inc.

Crane, A., n.d. *Experiences of Using PHP in Large Websites*. [Online]

Available at: <http://www.ukuug.org/events/linux2002/papers/pdf/php.pdf>

[Accessed 2 December 2012].

Davis, A., Bersoff, E. & Comer, E., 1988. A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering*, 14(10).

Delorey, D., Knutson, C. & Giraud-Carrier, C., 2007. *Programming language trends in open source development: An evaluation using data from all production phase sourceforge projects*. [Online]

Available at: <http://dml.cs.byu.edu/~cgc/pubs/WoPDaSD2007.pdf>

[Accessed 3 December 2012].

Gallagher, B., 2012. *HTML5 Keeps Growing: Users Have Built One Million HTML5 Sites On Wix In The Past Three Months*. [Online]

Available at: <http://techcrunch.com/2012/07/11/html5-keeps-growing-users-have-built-one-million-html5-sites-on-wix-in-the-past-three-months/>

[Accessed 3 December 2012].

Gilfillan, I., 2003. *PostgreSQL vs MySQL: Which is better?*. [Online]

Available at:

<http://www.databasejournal.com/features/postgresql/article.php/3288951/PostgreSQL-vs-MySQL-Which-is-better.htm>

[Accessed 1 December 2012].

Haney, J. & Vanlengen, C., n.d. *Server-Side Scripting in JavaScript/JScript And VBScript*. [Online]

Available at: <http://proc.isecon.org/2000/100/ISECON.2000.Haney.pdf>

[Accessed 2 December 2012].

Harker, R., 2011. *Voluntary Sector Statistics*. [Online]
Available at: www.parliament.uk/briefing-papers/SN05428.pdf
[Accessed 25 November 2012].

Healey, R., 1991. Database Management Systems. *Geographic Information Systems: Principles and Applications*, Volume 1, pp. 251-267.

JQueryTutorial, n.d. *What is JQuery?*. [Online]
Available at: <http://www.jquery-tutorial.net/introduction/what-is-jquery/>
[Accessed 2 December 2012].

Kanavin, A., 2002. *An overview of scripting languages*. [Online]
Available at: <http://sun.sensi.org/~ak/impit/studies/report.pdf>
[Accessed 2 December 2012].

Khan, H. & Ede, D., 2009. How do not-for-profit SMEs attempt to develop a strong brand in an increasingly saturated market?. *Journal of Small Business and Enterprise Development*, 16(2), pp. 335-354.

Leavitt, N., 2000. *Whatever happened to Object-oriented databases?*. [Online]
Available at: <http://www.leavcom.com/pdf/DBpdf.pdf>
[Accessed 29 November 2012].

Lethbridge, T. & Laganier, R., 2005. *Object-Oriented Software Engineering*. 2nd ed. Berkshire: McGraw-Hill Education.

Lie, H., 2005. *Cascading Style Sheets*. [Online]
Available at: http://www.cn-cuckoo.com/css/Hakon_Wium_Lie_css_2005.pdf
[Accessed 2 December 2012].

Morin, R. & Brown, V., 1999. *Scripting Languages*. [Online]
Available at:
<http://www.mactech.com/articles/mactech/Vol.15/15.09/ScriptingLanguages/index.html>
[Accessed 2 December 2012].

Munassar, N. & Govardhan, A., 2010. A Comparison Between Five Models Of Software Engineering. *International Journal of Computer Science*, 7(5), pp. 94-101.

MySQL, n.d. *Why MySQL?*. [Online]
Available at: <http://www.mysql.com/why-mysql/>
[Accessed 1 December 2012].

Natda, K., 2013. Responsive Web Design. *An International Referred Journal of Business, Accounting, Information Technology & Law*, 1(1).

Norman, D., 1988. *The Design of Everyday Things*. New York: Basic Books.

Paulson, L., 2007. Developers Shift to Dynamic Programming Languages. *Computer*, 40(2), pp. 12-15.

PHP, 2012. *PHP and other languages*. [Online]
Available at: <http://uk3.php.net/manual/en/faq.languages.php>
[Accessed 2 December 2012].

Poslad, S., 2009. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. West Sussex: John Wiley & Sons Ltd..

PostgreSQL, n.d. *Advantages*. [Online]
Available at: <http://www.postgresql.org/about/advantages/>
[Accessed 1 December 2012].

Rei, L., Carvalho, S., Alves, M. & Brito, J., 2007. *A Look at Dynamic Languages*. [Online]
Available at: <http://lrei.wdfiles.com/local--files/feup/A Look at Dynamic Languages.pdf>
[Accessed 2 December 2012].

Schroder, C., 2011. *PostgreSQL vs. MySQL: Which Is the Best Open Source Database?*. [Online]
Available at: <http://www.openlogic.com/wazi/bid/188125/>
[Accessed 1 December 2012].

Silberschatz, A., Korth, H. & Sudarshan, S., 2002. *Database System Concepts*. 4th ed. New York: McGraw-Hill.

Sommerville, I., 1996. Software Process Models. *ACM Computing Surveys*, 28(1), pp. 269-271.

Trent, S. et al., 2008. Performance Comparison of PHP and JSP as Server-Side Scripting Languages. *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pp. 164-182 .

w3schools.com, n.d. *The Internet Explorer Browser*. [Online]
Available at: http://www.w3schools.com/browsers/browsers_explorer.asp
[Accessed 2 December 2012].

W3techs, 2012. *Usage statistics and market share of PHP for websites*. [Online]
Available at: <http://w3techs.com/technologies/details/pl-php/all/all>
[Accessed 2 December 2012].

Appendix A – Complete ER Diagram

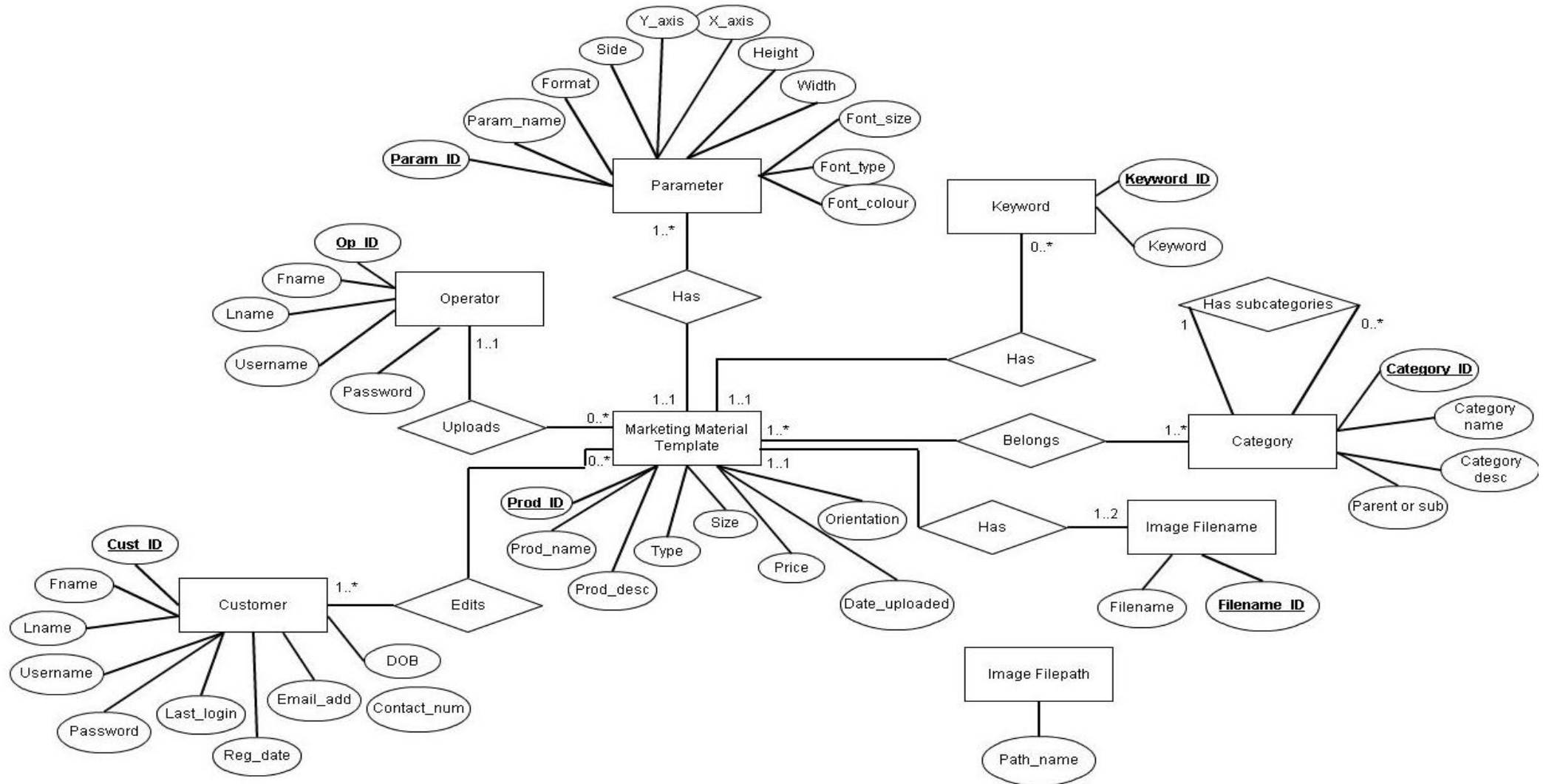


Figure 26: ER Diagram

Appendix B – ER Diagram Mapping and Normalisation

ER Diagram Mapping

The ER diagram from Appendix A has been mapped to a relational model as seen below in Figure A. Each entity has been mapped as different tables, with each table containing a primary key, with the exception of the Image_Filepath table. The Image_FilePath table only contains one value – the path name from which the images are to be stored.

There are different rules in mapping the relationships from the ER diagram to become foreign keys in the database tables. The two relationships that exist in the ER diagram are one-to-many and many-to-many relationships. If a relationship of one-to-many exists, the primary key of the entity having the one side will be the foreign key of the entity having the many side. For example, since a relationship of one-to-many exists between Template and Parameter tables, the primary key of Template, which is Prod_ID is the foreign key in the Parameter table. On the other hand, if a relationship of many-to-many exists, a new table, called the bridge table has to be created. The bridge table will have both the primary keys of the two entities involve in the many-to-many relationship as its foreign keys. At the same time, these two keys will also be the joint primary keys of the bridge table. For example, since a relationship of many-to-many exists between Operator and Template, a new bridge table called Operator_Template table has been created.

Operator (Op_ID, Fname, Lname, Username, Password)

Customer (Cust_ID, Fname, Lname, Username, Password, Last_login, Reg_date, Contact_num, Email_add, DOB)

Template (Prod_ID, Op_ID, Prod_name, Prod_desc, Type, Size, Price, Date_uploaded, Orientation)
FK Op_ID references Operator (Op_ID)

Customer_Template (Cust_ID, Prod_ID)
FK Cust_ID references Customer (Cust_ID)
FK Prod_ID references Template (Prod_ID)

Parameter (Param_ID, Prod_ID, Param_name, Format, Side, Y_axis, X_axis, Width, Height, Font_size, Font_colour, Font_type)
FK Prod_ID references Template (Prod_ID)

Keyword (Keyword_ID, Keyword, Prod_ID)
FK Prod_ID references Template (Prod_ID)

Image_Filename (Filename_ID, Filename, Product_ID)
FK Prod_ID references Template (Prod_ID)

Image_Filepath (Path_name)

Category (Category_ID, Category_name, Category_desc, Parent_or_sub, Parent_Category_ID)
FK Parent_Category_ID references Category (Category_ID)

Template_Category (Prod_ID, Category_ID)
FK Prod_ID references Parameter (Prod_ID)
FK Category_ID references Category (Category_ID)

Figure 27: Relational Database Model

Normalisation

After mapping the ER diagram into a relational database model, we must ensure that the model has been normalised in order to avoid duplications or redundancies of data. For this project, the database will be normalised up to third normal form (3NF) only. These forms will be briefly discussed in the following paragraphs. Other forms beyond 3NF such as Boyce-Codd, fourth and fifth normal form will not be considered as normalising the database to 3NF is already at its optimal level.

First Normal Form (1NF)

A relation is said to be in 1NF if all relations have atomic attributes and does not contain repeating values (Silberschatz, et al., 2002). Atomic attributes are attributes that cannot be broken down into smaller attributes because it is already in its simplest form. All relations in Figure A are already in 1NF as all attributes are already simple attributes. Also, all attributes of a relation do not contain any repeating values. For example, in the Operator table, an operator will only have one operator id, one name, one username and password.

Second Normal Form (2NF)

A relation is said to be in 2NF if the relation is already in 1NF and if all the non-primary key attributes are fully dependent on the primary key attribute. All relations in Figure A are already in 2NF as all non-primary key attributes depend on the primary key attribute. For example, in the Operator table again, all non-primary key attributes namely, Fname, Lname, Username, Password and Last_login depends on the operator id. This means that just by looking at the operator id, we can easily give information about the operator's details since the operator's id determines the operator's details.

Third Normal Form (3NF)

A relation is said to be in 3NF if the relation is already in 2NF and if no transitive dependencies exist. Transitive dependency means that an attribute depends on the primary key through another attribute. The 3NF is the same as the 2NF but it is stricter as 3NF ensures that the non-primary key attributes solely depends on the primary key and not on any other attributes. All relations in Figure A are already in 3NF as all non-primary key attributes only depend on the primary key attribute. Again, if we look at the Operator table, all the non-primary key attributes only depend on the primary key.

Since Figure A is already normalised, these tables can now be created in the implementation stage of this project.

Appendix C – User Interface Screenshots



Figure 28: Initial Layout for the complete categorising functionality

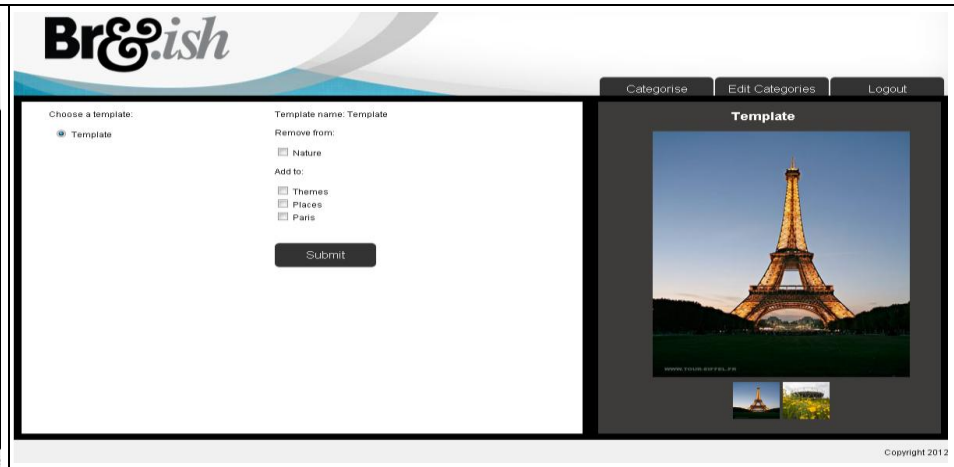


Figure 29: Initial Layout for the edit categorising functionality

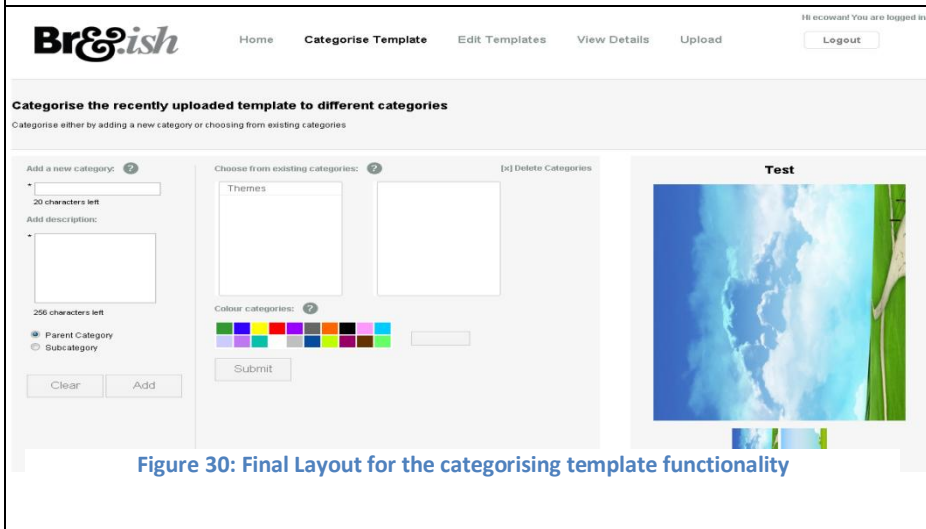


Figure 30: Final Layout for the categorising template functionality

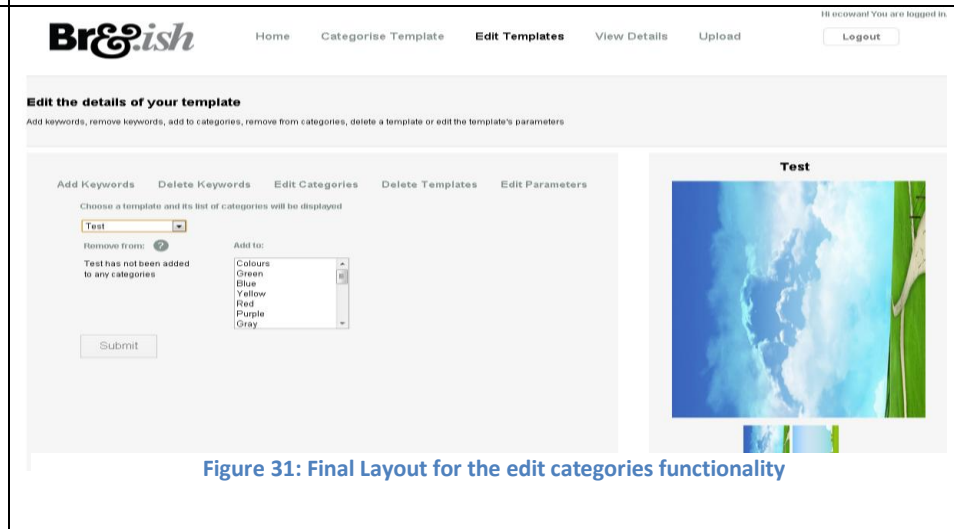


Figure 31: Final Layout for the edit categories functionality

Appendix D – Testing Results

Increment 3 Unit Testing

Description	Interface Action	Database Action	Result	Corrective Measure
Adding a new category – the author clicks the ‘Add’ button without specifying a category name and description.	The JQuery code for displaying the error message will be executed and will appear below the category name field and description field.		Correct	
Adding a new category – the author adds Themes, its description and specified that it is a parent category and then clicks the ‘Add’ button.	An alert box message appears and informs that the category ‘Themes’ has been added. After the page refreshes, ‘Themes’ has been added to the list of existing categories.	The category name and description are inserted successfully to the database. Since the category is a parent category, ‘NULL’ value is inserted to the attribute parent_category_id. Also, an automatic category ID has been inserted because of its AUTO INCREMENT feature.	Correct	
Adding a new category – the author adds a category name, description, specified that it is a subcategory and clicks the ‘Add’ button. (E.g. Minimal is the subcategory while Themes is the parent category)	An alert box pops up and informs the author that the category ‘Minimal’ has been added as a subcategory of ‘Themes’. After the page refreshes, the subcategory ‘Minimal’ is displayed with indentations below the category ‘Themes’.	The category name and description are inserted successfully to the database. Since the category is a subcategory, its parent’s category ID is inserted as its parent_category_id. Also, an automatic category ID has been inserted because of its AUTO INCREMENT feature.	Correct	
Choosing from existing categories – the author selects Themes and Minimal and clicks the ‘Submit’ button.	An alert box informs the author that the template has been added to the following categories: Themes and Minimal	The function for retrieving the template ID and the chosen category ID gets executed followed by the function for inserting these IDs to the Template_Category table.	Correct	
Choosing from existing categories – the author selects Themes and another category called Vintage and clicks the ‘Submit’ button.	An alert box message pops up and informs that the template has already been added to those categories	The function for inserting the IDs in the Template_Category table is not executed	Incorrect – The template should have been categorised to Vintage	Fixed by using PHP’s <i>array_diff</i> function.
Sanitising user input – the author types	The function <i>preg_replace</i> gets executed and replaces	The function for inserting the category details in the	Correct	

'Th+e_m???es' as the category name and clicks the 'Add button'	special characters with an empty character. An alert box message then pops up and informs that Themes already exist.	Category table is not executed		
--	--	--------------------------------	--	--

Table A: Unit Testing Results for Increment 3

Increment 4 Unit Testing

Description	Interface Action	Database Action	Result	Corrective Measure
Adding a new category - the author types Themes as a new category name to be added.	The JQuery code for counting the remaining characters left allowed gets executed. The system prints '14 characters left'.		Correct	
Deleting categories functionality – The author selects Themes and clicks the 'Delete' button	A confirmation box gets displayed to confirm if the author really wants to delete the category. If ok button is clicked, the category and its subcategories are deleted. If cancel button is clicked, the confirm box disappears and the pop up box for deleting categories is active again.	If ok button is clicked, the query for deleting categories gets executed. ON DELETE CASCADE has been used as a constraint so deleting a parent category causes its subcategory to be deleted as well.	Correct	
Deleting categories functionality – The author clicks the 'Delete Categories' link again	The category that was deleted (i.e. Themes) still appeared in the list of existing categories. However, once the page was refreshed or reloaded, the deleted category is now gone.		Incorrect – The page does not reload once a category has been deleted.	Fixed by attaching an onLoad listener to the <body> tag that reloads the page every time.
Editing templates functionality (Adding keywords) – the author selects a template.	The function <i>showInformation</i> gets executed.	The queries for displaying the template's images and the template's keywords get executed	Incorrect – If the template does not have any keyword, a message should be printed.	Fixed by using the empty function of PHP.
Editing templates functionality (Editing categories) – the author selects a template	The function <i>showInformation</i> gets executed.	The queries for displaying the template's categories and the template's images get executed	Incorrect – If the template has not been categorised to any categories, a message should be printed.	Fixed by using the empty function of PHP.

Table B: Unit Testing Results for Increment 4

Increment 5 Unit Testing

Description	Interface Action	Database Action	Result	Corrective Measure
Choosing from colour categories - the author selects the colour blue.	The user interface displays the word blue in the text field box.		Correct	
Choosing from colour categories - the author selects the colour blue and selects the colour blue again to deselect.	The user interface does not display any changes		Incorrect – the word blue in the text field box should disappear	Fixed by adding a variable that increments every time the selected area has been clicked. If the variable is equal to 1, the colour name will be displayed. If it's equal 2, an empty space will be displayed
Choosing from colour categories - the author selects the colour blue and clicks the 'Submit' button.	An alert pop up box is displayed to inform the author that the template has been categorised to the colour blue.	The function for inserting the IDs of the template and the colour blue category in the Template_Category table is executed.	Correct	
Choosing from existing categories and colour categories - the author selects 'Themes' from existing categories, the colour blue again and clicks the 'Submit' button.	An alert pop up box is displayed to inform the author that the template has only been added to 'Themes' as it has already been categorised to the colour blue.	The function for inserting the IDs of the template and the 'Themes' category in the Template_Category table is executed.	Correct	
Logout functionality – the author clicks the logout button	A confirmation box appears to confirm whether or not the author really wants to logout. If ok button is clicked, the code for resetting the session variable is executed and the author is redirected back to the login page.		Correct	
Editing templates functionality (Edit Parameters) – the author selects a template and clicks the 'Edit ' button	The php page 'imgDisplay.php' for processing the form gets executed which redirects the author to the edit parameters page of project 3		Correct	

Table C: Unit Testing Results for Increment 5

Appendix E – Integration Diagram

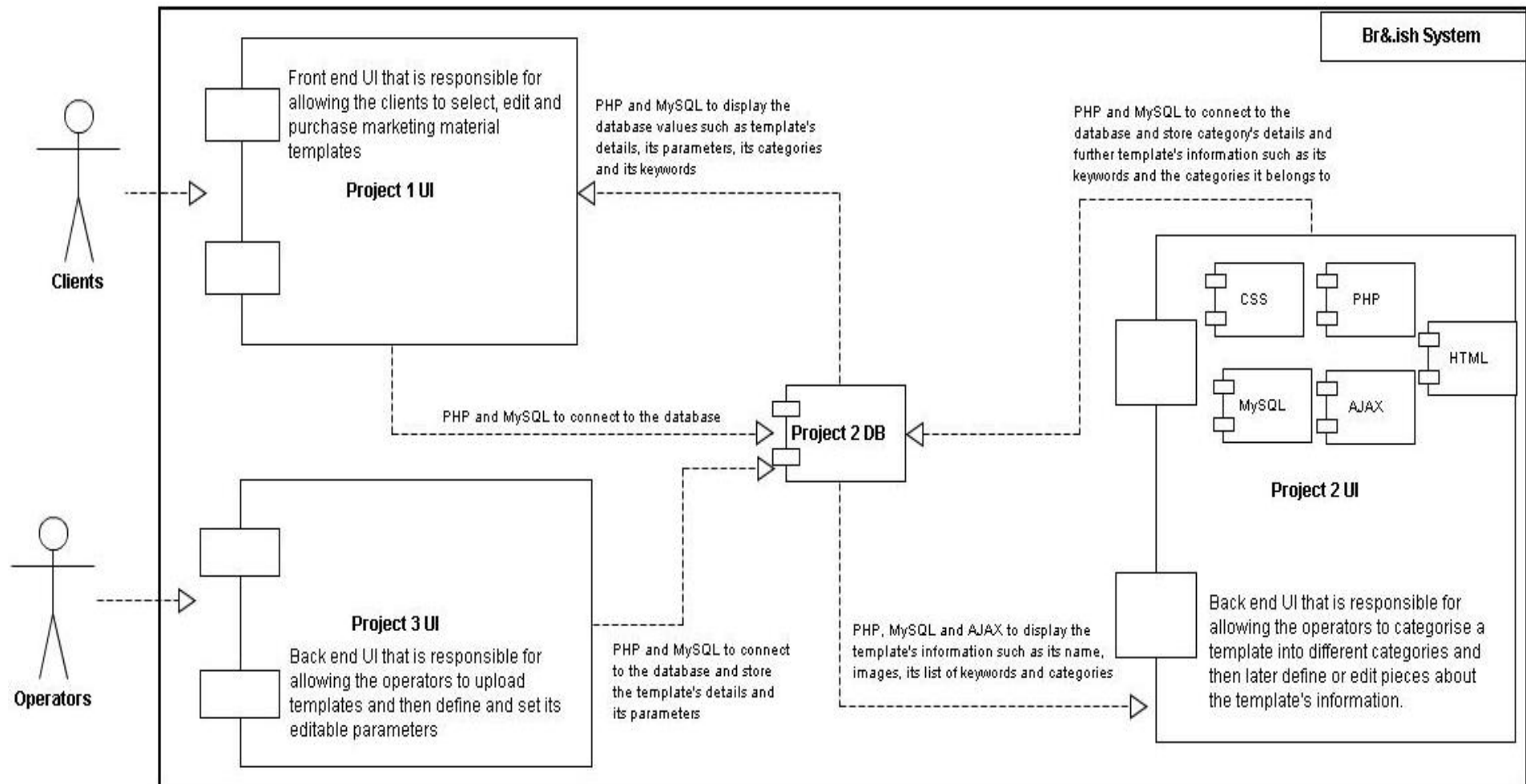


Figure 32: Integration Diagram