



# Healthcare Provider CMS

*MSc Computer Systems and Networking Project*

Author: Georgi Butev

Student Number: 3024421

Supervisor: Dr Zhanfang Zhao

Course: MSc CSN 1024 FT

Year: 2011/2012

Faculty: ESBE

University: London South Bank University

Student Contact: [butevg@lsbu.ac.uk](mailto:butevg@lsbu.ac.uk)

Date of Submission:

This report has been submitted for assessment towards a Master of Science degree in the department of Engineering Design, London South Bank University. This report is written in the author's own words and all sources have been properly cited.

Author's Signature:

## **Abstract**

The Healthcare Provider Content Management System is software application developed by the author using open-source tools and technology. The significance of this application is the need for content management of medical products in small and medium size NHS organisations. No such solution is freely available on the internet for institutions to implement and deploy in their medical environment. Similar healthcare applications target electronic medical records content management.

The aim of this project is to create a functional and highly useful web application for medical supplies management. Another aim of this project is to deliver a complete software package containing network operating system, web server, database server, and web development tools.

The success of the implementation of the technical research and technical approach was evaluated based on testing the functionality and usefulness of the CMS. Also the project was evaluated by independent participants in a test to determine the usability of the system. Their feedback was mostly positive and favourable with few exceptions. Furthermore the system was thoroughly tested for security exploits – web application and server. The result showed that all components of the system are secure and can withstand hacker attacks.

Some of the recommendations for future research and development include – credit card payment, bank account reports, client barcode scanning, modern theme design, operation review and rollback.

Some of the CMS achievements include – secure user login, management of medical products, management of users, single page AJAX interface, generation of PDF products, creation of medical data pie and column charts, send and receive private messages.

## **Acknowledgement**

I would like to express my gratitude and appreciation to Dr Zhanfang Zhao for his invaluable help for deciding upon a suitable project, planning and time management, software development, and report structure suggestions.

I would also like to thank Dr Goran Bezanov and Dr Deb Mukherjee for their invaluable instructions and helpful suggestions for the format and contents of this report. I would like to thank Mr Ya Bao and Dr Goran Bezanov for providing me with expert knowledge in the domain of computer networks. I would like to express my gratitude towards Dr Perry Xiao for his expert knowledge in the area of network programming. Also I would like to thank Dr Stavros Dimitriou for providing me with expert knowledge in the area of UNIX operating systems.

# Table of Contents

1. Introduction	4
2. Aims and Objectives	6
3. Deliverables	7
4. Technical Background and Context	8
4.1. Requirements	8
4.1.1. Software Requirements	8
4.1.2. User Requirements	9
4.1.3. Technical Requirements	10
4.2. Design of Content Management System Model	11
4.2.1. Programming Logic	11
4.2.2. Database Schema	14
4.2.3. Mock User Interface	16
4.2.4. Search Engine Optimisation	20
4.3. Development Environment Setup	22
4.4. Software Testing	23
4.5. Security Threats	23
4.6. Maintenance	24
5. Technical Approach	25
5.1. Register	25
5.2. Login	29
5.3. Administrator	34
5.3.1 Administrator Welcome Page	34
5.3.2 Administrator Database Control	37
5.3.3 Administrator Database Control Problems	43
5.3.4 Administrator User Database Control	45
5.4. User	50
5.4.1. User Database Control	50
5.4.2. User View Product	53
5.4.3. User Update Product	55
5.4.4. User Database Control Problems	56
6. CMS Improvements for the User and Administrator	58
6.1. Messages	58
6.2. PDF Report	60
6.3. Accounting	62
7. Results and Discussion	65
7.1. Software Tests	65
7.2. CMS Interface Functionality and Usability	67
7.3. Security Threats	69
8. Conclusion and Recommendations	73
9. List of References	75
10. Bibliography	75
11. Project Planning Diagram	76
12. Appendix A (DVD-R)	77

# 1. Introduction

Healthcare Provider CMS (Medical CMS in the rest of the report) is a content management system designed to solve specific challenge for medical institutions such as pharmacies and private / public hospitals and clinics. The difficulty is managing healthcare supplies such as drugs and syringes using a unified interface part of an open source software application. There are custom software solutions which are usually requested by the medical institution. Unfortunately these professionally developed programs have very high cost which severely affects the institution's budget.

This project is open source which means that it comes free of charge and it can be further improved by the institution, professional developers, and software community. This software product is targeting the needs of small private or public hospital / pharmacies which require management of medical products but cannot afford to purchase commercial software. The software interface will be used by medical staff, physicians, General Practitioners, doctors, pharmacists, and system administrators, which means that it has to be efficient and interactive without the need for graphical enhancement. They will be able to manage medical supplies, organise them according to their needs, and generate warehouse reports.

The project should be delivered as a software suit. That is due to the fact that the hospital or pharmacy may not have much IT/ICT experience or knowledge. The suit should contain all necessary elements and components for the application's deployment without the need for third party software or IT consulting company. The suit will comprise of a virtual machine in order to run on Microsoft Windows, Apple Mac OS X, and Linux with Gnome / KDE desktop environment. The virtual machine will hold free of charge operating system, web server, and the application itself. The entire project should fit the size of a CD or a DVD optical medium.

The source code of the project should be written using a popular and open source programming language. This is due to the fact that future development will be easier because the software engineer would be already familiar with the technology and software tools associated with that language.

Examples of such are PHP (Hypertext Preprocessor) with CodeIgniter framework or Wordpress CMS, Perl with Catalyst framework, Python with Django framework, Ruby with Ruby on Rails framework etc.

Examples of suitable free database software are – MySQL, PostgreSQL, SQLite, and Microsoft SQL Express Server.

Examples of suitable free web servers are – Apache, nginx, lighttpd, Tomcat, Glassfish etc.

Examples of suitable free operating system are – Ubuntu, Ubuntu Server, Fedora (previously known as Red Hat Linux), CentOS, FreeBSD, OpenBSD etc.

Examples of suitable virtual machines which do not have to be open source but should be free are – VMWare, Oracle Virtual Box, Hyper-V, and Parallels.

This report will introduce the topic to the reader, explaining the aims and objectives associated with the software application. Also it will provide the reader with in-depth technical information specific to the project for complete understanding of the author's work. Furthermore this report provides the reader with complete and detailed explanation of the technical approach, software tests, development problems, and source code improvements. Finally the reader can consult with the conclusion and recommendations section of this document for brief and concise summary of the author's work. All software development time scheduling and organisational planning is placed in the project planning diagram section of this document.

## 2. Aims and Objectives

The aim of this project is to create a functional and highly useful web application for specific medical organisation purposes. The application should satisfy most of the potential clients' requirements and needs in terms of medical supplies management. The title of the project reflects upon the fact that the software will act as a content management system. Even though similar programs already exist they mostly deal with Electronic Medical Records. The originality of this venture is that it mostly deals with the control of clinical and pharmaceutical products in a database.

Another aim of this project is to produce software that is free and open source using open source programming languages and software tools.

Yet another aim is to supplement the development of the software application with a detailed report containing all relevant information necessary for fully understanding the work done by the author. Also to aid future testing and research based on the final, complete, and tested source code.

In order to achieve the outlined aims in this report, the following objectives would have to be satisfied:

1. Determine software, user, and technical requirements.
2. Design content management model:
  - a. Programming logic.
  - b. Database schema.
  - c. Mock user interface.
  - d. User types and privileges.
  - e. Search engine optimisation.
3. Prepare the development and production environment set-up.
4. Use features from commercial content management systems:
  - a. Perform literature research.
  - b. Collect freely available, useful, and project relevant source code.
5. Develop the application according to the already determined requirements.
  - a. Satisfy open source requirements with Linux, Apache, MySQL, and PHP.

6. Perform user and programming logic tests for software bugs.
7. Evaluate security risks and propose / implement security features.
8. Make sure that each software component is written within the time management guidelines and that the software application is finished before the deadline.
9. Produce coherent and well-structured report which details the execution of all of the above tasks. Also produce an informative conclusion section and technically valuable recommendations for further development based on the completed software application.

### **3. Deliverables**

The effort of the author should produce the following tangible deliverables which can be assessed by the supervisor and second reader:

1. Master of Science report written according to the style and format recommended by Dr Deb Mukherjee during the “Technical, Research and Professional Skills” class.
2. Healthcare Provider CMS software application source code written in PHP. Also related scripts written in JavaScript, style sheets written in CSS, standard web presentation written in HTML, empty and test database file in MySQL format, and bitmap or vector images such as avatars, icons, and web page backgrounds.
3. Ready to deploy and test platform containing a virtual machine image, configured Linux Ubuntu server operating system, configured Apache web server, configured MySQL database server, and brief README file with instructions explaining how to use the VM system and application.
4. All of the above will be recorded on a standard 4.7GB DVD-R optical media.
5. The report and logbook are hard-copy.

## 4. Technical Background and Context

This section of the report will focus and discuss the following in order to provide the reader with technical background information about the project – requirements, design of content management system, development environment, security threats, software testing, maintenance, similar projects.

### 4.1 Requirements

#### 4.1.1 Software Requirements

These requirements were determined based on the potential clients' needs and application usage.

Feature	Description
Medical Product	<ul style="list-style-type: none"><li>• Name</li><li>• Brand</li><li>• Manufacturer</li><li>• Quantity</li><li>• Colour</li><li>• Size</li><li>• Intake Type</li><li>• Purchase Frequency</li><li>• Purchase Volume</li><li>• Delivery Time</li><li>• Price</li><li>• Image</li><li>• Knowledge Base</li><li>• Scientific Base</li><li>• User Manual</li><li>• Bar Code</li><li>• Comments</li></ul>
Edit & Update Product	Change information about already existing product in the database.
Remove Product	Erase the selected entry from the database.
Purchase Product	<ul style="list-style-type: none"><li>• Email – the application would send email request to the supplier.</li><li>• PayPal – the application would negotiate a deal using PayPal's API.</li><li>• Credit Card – the application would use existing web API.</li></ul>
Catalogue Product	This would be part of the organisational features of the application.
Accounting	This would be part of the purchasing, billing, and log features.



## 4.1.2 User Requirements

These requirements were determined based on the potential clients' needs and application usage.

Feature	Description
Web Based	The software application would be web based because there are no restrictions in terms of the user operating system. All modern OSs support some sort of a web browser capable of rendering HTML tagged content and JavaScript interaction. In addition the user has no direct access to the executable programme and cannot temper or potentially crash the Healthcare Provider CMS. This is due to the fact that the application runs on a dedicated secure server.
Simple and easy user interface.	The user interface is essential because all interaction with the database and supplies organisation is done using the web browser. The user interface design would have to be formal in accordance to modern web design standards. In addition the usage learning curve would have to be moderate. Preferably JavaScript and AJAX would be used instead of the popular and feature rich MS Silverlight and Adobe Flash technology.
Access control and privileges.	Such medical application demands proper user authentication and authorisation. Users would be divided into groups with varying degree of access and control over the database. The most restricted users would be medical staff – they would only view records. The only unrestricted users would be the CMS administrators. Certain supplies such as mild and severely addictive drugs would have to be approved by a senior staff member.
Barcode Scanner	This would be the unique identifier for each supply. Entering each and every code would be an extremely tedious task to accomplish. That is why the application would have to support some sort of barcode scanning.
Email System	The user would be able to compose emails using the Healthcare Provider CMS user interface. The email would include the product name and description as part of the title and body of the email.
Bulletin System	All users would be able to read important notices and news from other users with correct privileges.

Different Views	According to user authorisation he / she will see different web layout. Some will not be able to interact with the database because of their low level privileges.
-----------------	---

### 4.1.3 Technical Requirements

The following section discusses the target computer station hardware and software requirements. These build upon the previously listed user and CMS software requirements.

Requirement	Comment
Server OS	<ul style="list-style-type: none"> <li>Windows Server 2003, 2008, 2008 R2</li> <li>Ubuntu 10.4 LTS, FreeBSD 8</li> <li>Mac OS X 10.7 and 10.6</li> </ul>
Server Service	<ul style="list-style-type: none"> <li>For Microsoft Windows – Internet Information Service 7.5.</li> <li>For Linux and UNIX based distributions – Apache 2.3 or 2.2.</li> <li>For Mac OS X – MAMP (Mac, Apache, MySQL, PHP).</li> <li>Cloud services such as ‘Google App Engine’ and ‘Ubuntu One’.</li> </ul>
Server Hardware	<ol style="list-style-type: none"> <li>Shared web hosting services from the following providers: <ul style="list-style-type: none"> <li><a href="http://www.godaddy.com">www.godaddy.com</a></li> <li><a href="http://www.networksolutions.com">www.networksolutions.com</a></li> <li><a href="http://www.pixelinternet.co.uk">www.pixelinternet.co.uk</a></li> <li><a href="http://www.rackspace.co.uk">www.rackspace.co.uk</a></li> </ul> </li> <li>Dedicated web hosting server: <ul style="list-style-type: none"> <li>CPU – Intel Pentium 4 at least 2GHz clock speed.</li> <li>RAM – 2GB DDR2 or DDR3 memory at least.</li> <li>Storage – the application requires no more than 10GB.</li> <li>Network – 100Mbps Ethernet or similar network.</li> </ul> </li> </ol>
Server Software	<ol style="list-style-type: none"> <li>PHP 5.2</li> <li>MySQL 5.5</li> <li>Apache PHP module – mod_php.</li> <li>IIS 7 – application development features, CGI module, FastCGI module, and PHP_via_FastCGI module.</li> </ol>

## **4.2 Design of the Content Management System Model**

### **4.2.1 Programming Logic**

Programming logic is needed to bind all the elements of the content management system, such as database access (e.g. queries, responses), displaying DB records to the web browser, building a consistent user interface, sending/receiving messages, sending/receiving emails, buying medical supplies, authorisation, security, etc. Various programming languages are used to achieve these tasks, such as PHP, Java, and C#. Using these languages the developer could create a meaningful, useful, and structured plan or strategy for the content management system operations. This plan is represented in the figure #1 below with a flow chart.

This project relies on PHP version 5 for the programming logic. It is worth to note that Java with Springs, Struts and C# with .NET and Ruby on Rails are also suitable development technologies.

PHP is a general purpose programming language much like Java and C#. It is high level object oriented language well supported by the major operating systems. However PHP is an interpreted scripting language. This means that there is no need for the script (i.e. the source code) to be compiled after each iteration. The server runs the script whenever there is a new request, line by line, and serves the result to the user via the common gateway interface.

According to [11] PHP's popularity is largely due to its ease of development, open source nature (tutorials and documents are very easy to find), low learning curve (compared to Java), and light weight in terms of required hardware resources (e.g. CPU power and memory size) compared to Java and .NET which require a lot more processing power and RAM.

Also according to [11] Ruby (another interpreted general programming language) and the more popular web development framework Ruby on Rails are praised for their undoubted benefits over PHP and web development frameworks such as Zend, CakePHP, CodeIgniter etc. Unfortunately Ruby shared web hosting is hard to find and often expensive (compared to PHP).

The following figure #1 represents a flow chart process of the application's execution. Before the user lands on the universal (for all level users) home page, the application checks whether the installation has been configured and finalised.

Afterwards the user could authenticate using username and password. There are three types of user privileges – limited, unlimited, and administrator. In short the limited user can only view the database and search based on some criterion.

Unlimited users can also make changes to the database, compose text messages, and purchase medical supplies. The administrator has the ability to control all aspects of the content management system.

Unlimited users using the unrestricted view can perform the following tasks.

Task	Comment
View and search database.	Display results and print records.
Organise and edit records.	Select, edit, remove, and update records. Review and commit changes.
Search for supplies.	Search for supplies using a keyword. Display the search results. Compare the supplies in a table by certain criteria. Purchase the selected product using MySQL transaction, commit, rollback functions and PayPal or other third party API.
Write and send text messages. Compose and publish news.	Compose messages and news articles. Upon successful submission, store the text in the database.
Logout Request	The user can terminate a session with a button or upon exiting from the web browser.

The administrator would be using the administrator view and can perform the following tasks.

Task	Comment
View database and purchase.	Display results and print reports.
Manage purchases, manage database, view messages, manage messages, view news, and manage news.	Add, edit, update, and remove database records. Also the user can print recent changes, all DB records, or selected DB records.
View and manages users.	Add, edit, update, and remove database records. Also the user can print DB records reports.
Manage CMS configuration.	View the configuration file, edit the file, and confirm changes.
Generate Statistics	Used to print history reports about users and their activities.
Logout Request	The user can terminate a session with a button or upon exiting from the web browser.



## 4.2.2 Database Schema

The database engine and data aggregation method is essential to this project. There are three issues to be considered – database software, engine, and planning. It is also imperative for the database to be relational, multi-access, encryption support (for passwords), and be SQL (Sequential Query Language) compliant. There are plenty of available commercial and free of charge DB software products suitable for this project such as Oracle 11g, PostgreSQL, IBM DB2, and MS SQL Server.

The table below compares some of the features to consider between MySQL and the rest.

Commercial Database Engines	Considerations
Oracle DB and IBM DB2	Typically used with Java and Java web development frameworks such as Springs and Struts. Advantages over MySQL according to [11]: <ul style="list-style-type: none"><li>• Superior auditing ability in terms of transactions to the DB.</li><li>• Plenty of security features appropriate for the enterprise.</li><li>• Many built-in scalability and reliability functionality.</li><li>• MySQL number of joins is limited to 61 unlike Oracle.</li><li>• RAC (Real Application Cluster) is used for failover management.</li></ul>
Microsoft SQL Server	Typically used with C# and .NET framework. There are two versions – one for enterprise development and express for web development. Advantages over MySQL according to [11]: <ul style="list-style-type: none"><li>• Better performance for busy and demanding web applications.</li><li>• High availability and reliability.</li><li>• Auto tuning performance depending on circumstances.</li><li>• Fewer security exploitations.</li><li>• The Express edition is free of charge.</li></ul>
PostgreSQL	Typically used with Perl, Python, and Ruby rather than PHP. This software product is much more enterprise in its functionality compared to MySQL and it is more similar to Oracle DB. It is also open-source, free of charge, and ACID SQL compatible unlike MySQL.
MongoDB	Is a popular “No SQL” database software which allegedly outperforms SQL database software by a factor of ten because it does not use the standard SQL joins functionality. However it has been criticised because it of its immaturity, unreliability, and security flaws.

The list below outlines the benefits of MySQL compared to the rest:

1. Popular amongst web developers scripting with PHP.
2. Concurrent operations (e.g. insert, update, remove records) unlike a flat file database.
3. Several storage engines are available – MyISAM and InnoDB (for safe transactions).
4. Full-text indexing of table fields for faster searching of records in the database.
5. The memory engine stores data in the RAM and uses hash indexing for better performance.

There are several MySQL engines to choose from. The default one is MyISAM which is perfect for majority of operations required by dynamic web sites. However since the Medical CMS features buying medical supplies, there is a need for transaction MySQL engine. That is the InnoDB engine which supports transaction, commit, and rollback for crucial and important DB operations. The following figure #2 represents the database schema for the Healthcare Provider CMS.

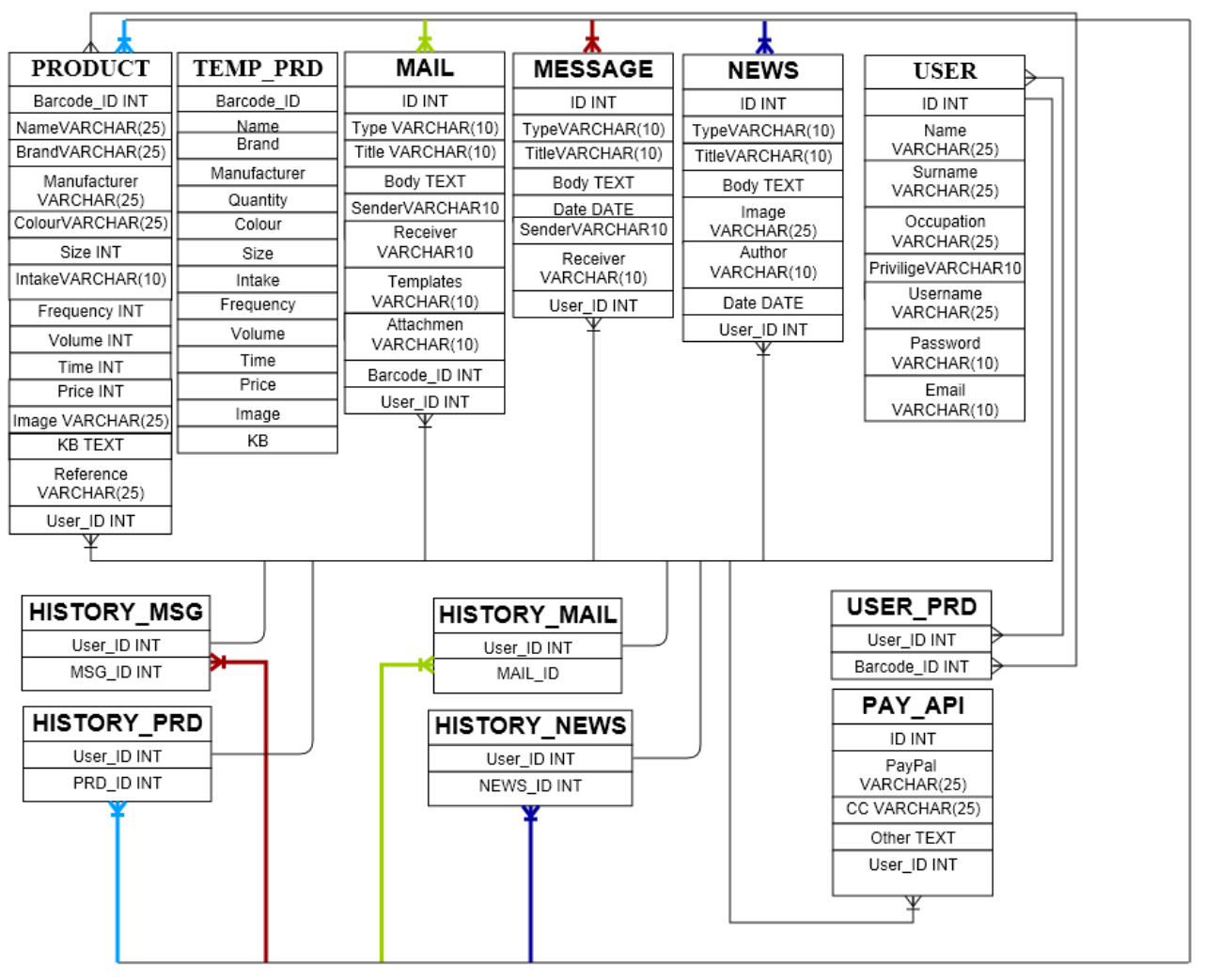


Figure 2: Database Schema

There are no one-to-one relationships in the database because there is no such uniqueness. The one-to-many relationships are mainly between the user identification and product, message, mail etc. That is because one user could be related to many products or messages and emails. The many-to-many relationships are part of the match tables. For example the table USER\_PRD or user to product matching table relates the user and product tables. Other such tables part of the history report data aggregation are – HISTORY\_MSG, HISTORY\_PRD, HISTORY\_MAIL, and HISTORY\_NEWS.

### 4.2.3 Mock User Interface

The user interface is part of the web design effort for this project. Its planning is no less important than the programming logic and database structure. There are three issues that should be considered when designing the Medical CMS – style, accessibility, and technology.

The style of the user interface should match the client’s requirements of simplicity and ease of use. Possible web design ideas are available as templates from commercial web services. The keywords that describe these templates are:

- Product – e-commerce and content management system.
- Category – medical and business.
- Style – clean, minimalist, corporate, neutral.

Based on the research of web design templates from [www.templatemonster.com](http://www.templatemonster.com) the following preliminary web design was drafted for the Medical CMS. The figures below represent the wireframe of the content management system.

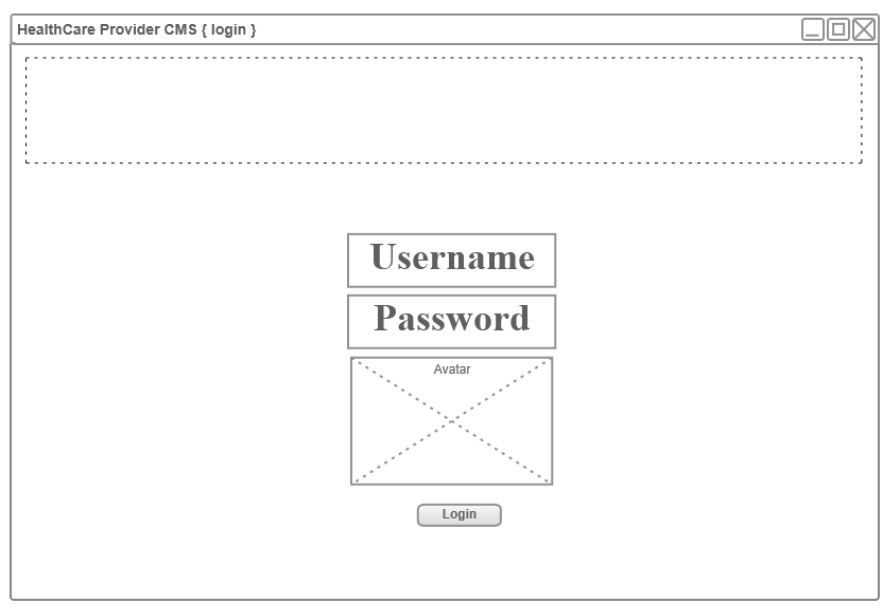


Figure 3: Login Screen



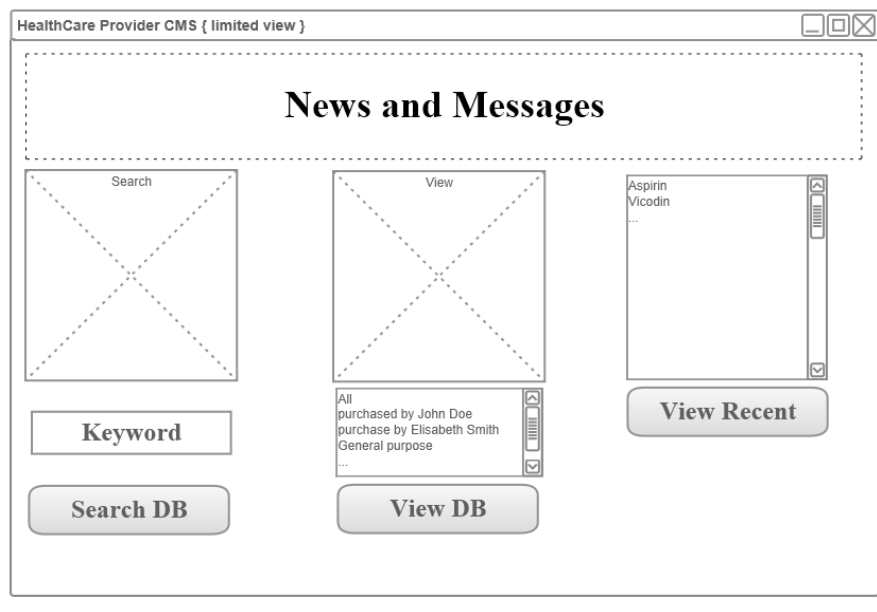


Figure 4: News and Messages

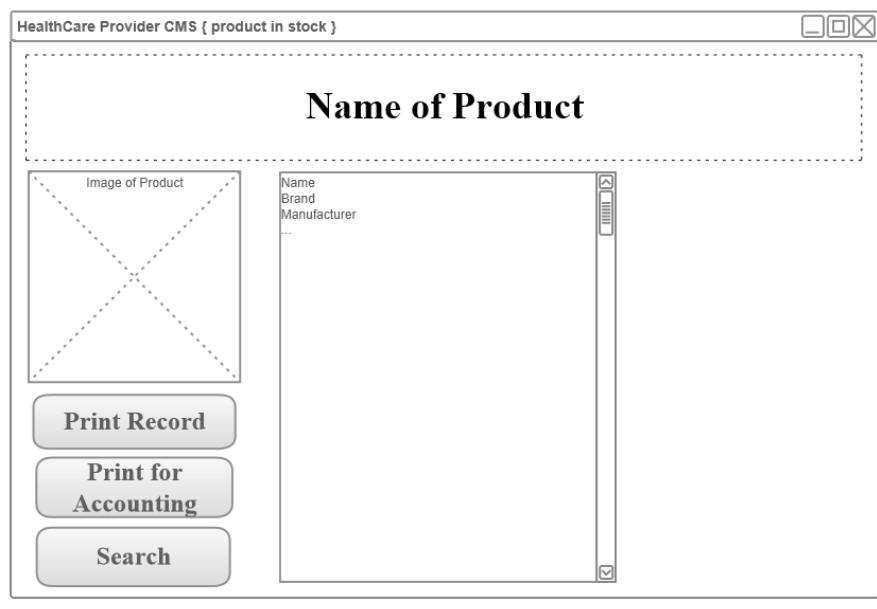


Figure 5: Restricted View of Product

Since the Medical CMS uses the web browser as the main interaction point between the user and the database, standard output is necessary. Otherwise the output display in the web browser would be damaged or even unintelligible.

There are three web standards considered for this project that are supported by all modern web browsers – HTML, CSS, and JavaScript. Software technologies such as Adobe Flash and Microsoft Silverlight were not considered because web browsers such as Internet Explorer and Firefox require additional plug-ins.

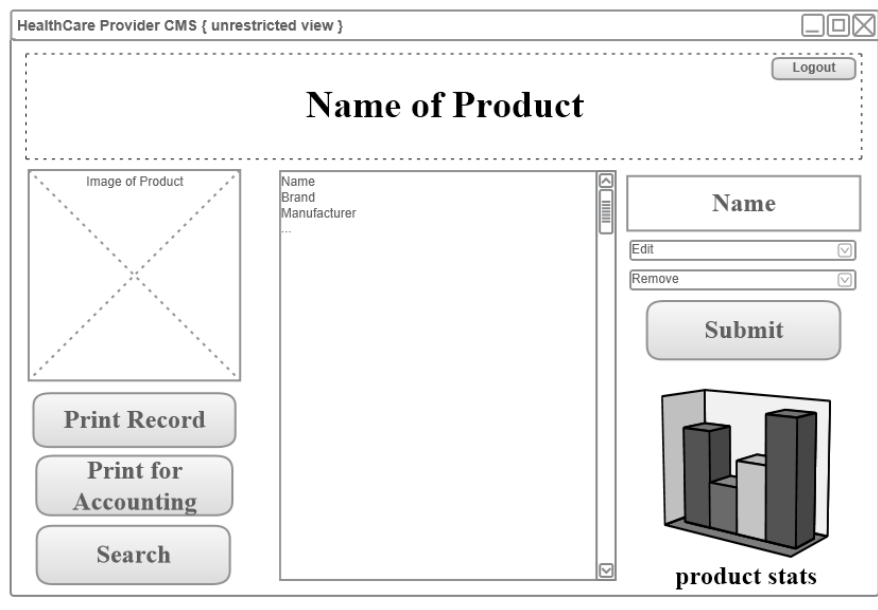


Figure 6: Unrestricted View of Product

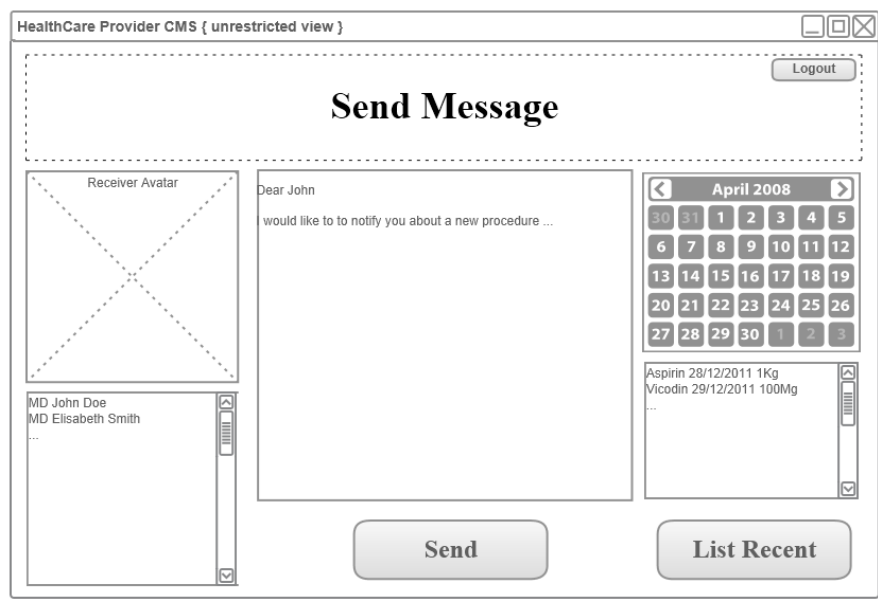


Figure 7: Unrestricted View Send Messages

It is worth considering that HTML 5 and CSS 3 would become well accepted standards in the near future. Thus building the Healthcare Provider CMS using the newest possible technologies would be beneficial. However such strategy might be potentially problematic because enterprise type organisations update and upgrade core software applications only when it is necessary. There are instances of companies running 10 and more years old software as their core technology and have little plans of changing it. For example Red Hat Linux Enterprise latest release was back in 2002. Also FreeBSD 4 (released early 2000) remains the core of many web hosting service providers, banks, and the military.

Figure 8: Buy Product

**HealthCare Provider CMS { administrator view }**

[ ] [ ] [X]

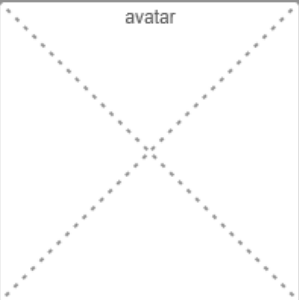
---

**Logout**

## Manage Users

**User Details**

- Name
- Surname
- username
- password
- privilege



**username**

**Submit**

**Remove**

**keyword**

User, Privileged User or Admin

List of Users

**View User History**

**Print User History**

Figure 9: Manage Users

Web Standard	Comment
HTML 4	The Hyper Text Markup Language is the de facto standard for creating web pages and web sites. The developer could use tags to markup headings, paragraphs, links, images, forms, tables etc. These would be displayed to the user via the web browser. It is important to note that this version of HTML is supported by all modern web browsers unlike HTML 5.
XHTML	It is mostly identical to HTML version 4.1 but it is in fact cleaner and “proper” HTML syntax. That is because the developer would have to use the following to validate against the XHTML certification checker: <ul style="list-style-type: none"> <li>• Properly nested tags.</li> <li>• Text, image, hyper link elements should be enclosed with start and ending tag.</li> <li>• Specific so called empty elements should be closed (e.g. <code>&lt;br /&gt;</code>).</li> <li>• All tags should be written in lower case (e.g. <code>&lt;body&gt;</code>).</li> <li>• Root tags such as <code>html</code>, <code>body</code>, <code>title</code>, <code>head</code> cannot be repeated in a file.</li> </ul>
HTML5	It is the new draft standard developed by W3C. Amongst various tag syntax innovations it includes several new tags, common for the internet. These are the video, audio, offline applications, content editable attributes, and drag and drop.
CSS2	CSS formatted files are used as a style sheet for HTML documents. Using descriptive IDs and Classes the developer could style the background, text, fonts, links, lists, tables etc. CSS 2 is supported by all modern web browsers.
CSS3	Builds on top of CSS2 and introduces new functionality such as transform and transition, linear and radial gradients, all new CSS3 selectors, border-radius, box-shadows, text-shadows, various font-face support.
JavaScript	It is used to create dynamic HTML web pages and control the HTML API also known as the DOM (Document Object Model). Using JS programming logic the developer could control all HTML document objects such as text, images, and links as well as to take care of sessions, forms validation, and browser cookies.

#### 4.2.4 SEO

Search engine optimisation aims to normalise the contents of a web site in such a way that it is easier for a search engine to crawl, index, and consequently display relevant search results to the inquiring user.

It is important to target few engines for best results. That would be Google, Yahoo!, and Bing. The developer could optimise the site using the following attributes.

- Anchor text.
- Site popularity.
- Link context.

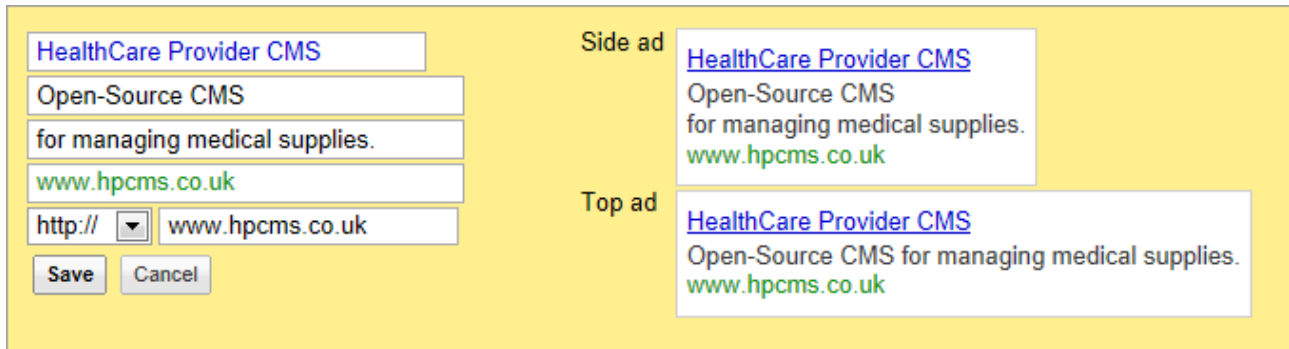
- Title tags.
- Keywords.
- Site language.
- Content and site map.

Finally the success of the SEO strategy could be assessed using web server statistics (e.g Webalizer) and Google Analytics. Furthermore the developer could use specific search engine campaigns to promote the site to potential customers. For example Google's AdWords, Yahoo's Search Marketing, and Bing's adCenter.

Potential issues for SEO could be usage of anything else than pure HTML and CSS, for example – JavaScript, Flash, ASP.NET, PHP all fall into that category. Search engines are known not to index all Flash web sites, which is disastrous for SEO strategies. The following is an example of a search engine optimisation strategy.

Elements	Example
Title tags and meta text should be descriptive and should not obscure.	<pre>&lt;head&gt;&lt;title&gt;Healthcare Provider CMS&lt;/title&gt; &lt;meta name="description" content="Healthcare content management system for organising and purchasing medical supplies"&gt; &lt;meta name="keywords" content="healthcare, medical, hospital, clinic, CMS, content management system"&gt; &lt;/head&gt; &lt;body&gt;&lt;h1&gt;Healthcare Provider CMS Home Page&lt;/h1&gt;</pre>
Anchor Text	<pre>&lt; a href="purchase_medical_supplies.html"&gt;Purchase Medical Supplies&lt;/a&gt;</pre>
Alternative Text	<pre>&lt;img width="250" height="250" src="aspirin.jpg" alt="Aspirin"&gt;</pre>
Short and neat URLs.	<a href="http://www.hpcms.co.uk/index/database/aspirin/">http://www.hpcms.co.uk/index/database/aspirin/</a>
Style and Format	<pre>&lt;strong&gt; and &lt;emphasis&gt; tags are preferred to &lt;b&gt; and &lt;i&gt;</pre>
No Frames	Search engine crawlers have difficulties parsing HTML frames and it is very possible that only the landing page would be indexed and the rest of the web site would be disregarded.
Google AdWords	<p>The following steps are required to sign-up for the service:</p> <ol style="list-style-type: none"> <li>1. Everyone with a Gmail account can sign-up.</li> <li>2. Set time zone and currency.</li> <li>3. Choose the budget.</li> <li>4. Choose territory and city where the ad will be shown.</li> <li>5. Choose between text ad and image advertisement.</li> <li>6. Decide upon headline, description line, display URL, destination URL, and keywords.</li> <li>7. Link the newly created ad to Google Analytics for further monitoring.</li> </ol>

For example the images below will be displayed in Google to user queries with relevant keywords.



The screenshot shows a Google AdWords campaign setup interface. On the left, there is a form with the following fields: "HealthCare Provider CMS" (text), "Open-Source CMS" (text), "for managing medical supplies." (text), "www.hpcms.co.uk" (text), and a URL field with "http://" and "www.hpcms.co.uk". Below the form are "Save" and "Cancel" buttons. On the right, there are two ad preview boxes. The top box is labeled "Side ad" and the bottom box is labeled "Top ad". Both boxes display the same ad text: "HealthCare Provider CMS" (link), "Open-Source CMS" (text), "for managing medical supplies." (text), and "www.hpcms.co.uk" (text).

Figure 10: Google AdWords Campaign

### 4.3 Development Environment Setup

Developing the Medical CMS using an external server would be inconvenient, slower, and more expensive. That is why it is recommended for developers to setup a pre-production environment for developing web applications.

There is one appropriate software suite that satisfies the requirements and it is free to use – XAMPP. There are plenty more software applications available on the Internet which target specific operating systems and all of them have Apache, PHP, and MySQL in the product's abbreviation. For example WAMP, MAMP, and LAMP respectively for Windows, Mac OS X, and Linux based distributions. However only XAMPP supports all three platforms.

Furthermore the software package comes with additional built-in applications common to web development, such as:

- PEAR – PHP applications repository.
- ProFTPD – file transfer.
- phpMyAdmin – web user interface control of the MySQL server.
- OpenSSL – secure socket layer used to encrypt data sent from the user to the server and vice versa.
- GD, libjpeg, libpng – libraries for manipulating images.
- zlib, libxml, pdf class – libraries for manipulating text documents.
- Webalizer – used to monitor network traffic and produce content and resource statistics.

The installation of XAMPP is simple because the developer has to only copy and paste the XAMPP directory structure.

Another possible development solution is the combined usage of a virtual machine and industry standard web services server. This project would use Oracle's Virtual Box which can be installed on Windows XP / Vista / 7 and run Linux / UNIX based server distribution. The network operating system would be Ubuntu Server 12.04 LTS. The official CD from Canonical comes with pre-installed daemons which require simple configuration via nano or vim of configuration files placed in the /etc directory. As a result the virtual machine will perform and function similarly to web hosting data centre service. It is worth to note that the following had to be configured in order to debug the web application. This is necessary because unlike Java and C# which come with compiler / linker programs, PHP and JavaScript do not have compilers – they are parsed by Apache and Internet Explorer respectively. For PHP the `/etc/apache2/apache2.conf` and `/etc/php5/apache2/php.ini` had to be altered as such to enable server response error reporting.

```
display_errors = on
error_reporting = E_ALL & ~E_NOTICE
php_flag        display_errors    on
php_value       error_reporting    2039
```

For JavaScript modern web browsers offer developer consoles which can monitor and record script events based on user interaction. Notable examples are Internet Explorer 9 and Google Chrome 11.

## **4.4 Software Testing**

Module, functionality, and user testing is recommended in order to test the Medical CMS for software bugs. Test scenarios would be part of the final application functionality assessment. The scenarios would be performed using manual testing by the developer, automated testing by a third-party software product, and user testing.

The manual and user test cases would be developed later on to reflect the project's maturity. The user test would require independent users (preferably LSBU students) who will go through several instructions, performing a well defined task. Afterwards they would have to provide feedback about the usability of the application and problems they have encountered.

## **4.5 Security Threats**

Web application vulnerability scanner software products can be used as part of the CMS security threats assessment. Uniscan and Acunetix are vulnerability scanning tools for common malicious web application attacks, such as RFI (Remote File Inclusion), LFI (Local File Inclusion), RCE (Remote Code Execution), XSS (Cross Site Scripting), and SQL Injection.

In addition Acunetix also scans for AJAX and SSL based application vulnerabilities. Furthermore it is a tool used by IT professionals and big IT companies, such as NASA, Disney, Fijitsu, and Credit Suisse. The developer could also use Metasploit and NexPose frameworks which are allegedly used by information security specialists and ethical hackers.

The following table lists security threats for the Medical CMS.

Software Application	Description
Apache	Distributed Denial of Service and remote session hijacking.
PHP	SQL injection via HTML forms and buffer overflow.
MySQL	DDOS and unauthorised access to the database.
JavaScript	Cross Site Scripting and Session Hijack.

## 4.6 Maintenance

The maintenance functionality is important part of the software application to safeguard the Medical CMS from complete failure and gridlock. The two options that were considered are MySQL server database backup and Apache error log.

Database backup means that the DB structure, all of its tables, fields, and entries will be saved into an SQL file. If there is a problem with the database the developer could simply use the “backup.sql” file. This could be easily achieved with the PHPMyAdmin user interface and the export functionality. Furthermore the developer could use the MySQL server command line.

For example the following code uses mysqldump to backup the database:

```
// Lock the table to prevent further modification of records.  
// The READ argument allows users to read the records in spite of the lock down.  
LOCK TABLES medical READ;  
// Flush the tables so that all active index pages are written to disk before the backup.  
FLUSH TABLES medical;  
// Export the database structure to an .sql file.  
mysqldump medical > medical.sql  
UNLOCK TABLES;
```

The Apache web server stores three log files by default “access.log”, “error.log”, and “ssl\_request.log”. They may be useful for determining errors and bugs with the CMS application.



## 5. Technical Approach

This section of the report details the approach undertaken by the student to achieve the project's objectives using software engineering techniques and tools.

### 5.1 Register

In order for the medical personnel to use all features of the content management system they have to register using the web site's interface and later the administrator has to assign them privilege roles according to their occupation in the organisation and operational needs. The administrator's role to manage privileges is very important because anyone with access to the web site can register thus their access to the application should be restricted appropriately. This is even more valid when new administrator accounts are created – the administrator has to authorise them manually to preserve the CMS security and prevent malicious activities.

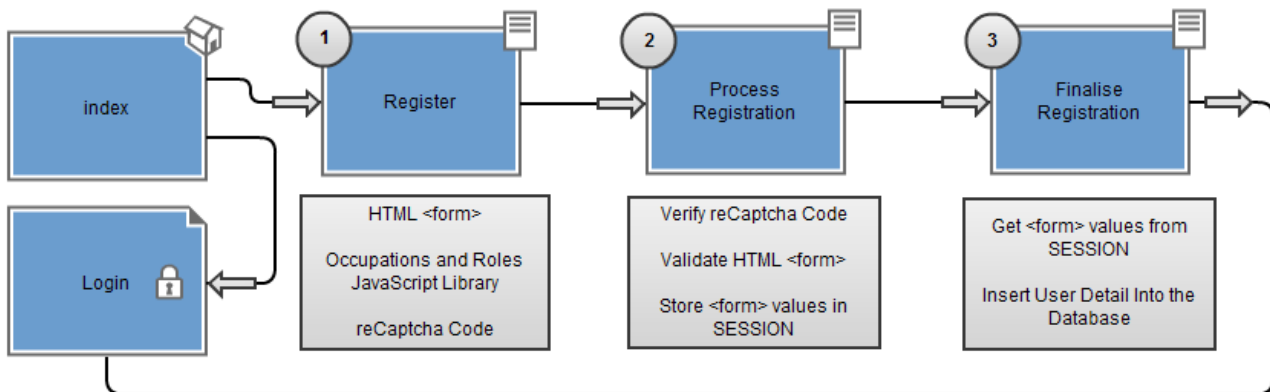


Figure 11: Registration Procedure

The figure above illustrates the process undertaken by the user to register for the Medical CMS. He or she will be unable to access any other web pages and will be redirected to the Login page to authenticate as genuine users. The **first step** is Register.php will display an HTML form to the user containing occupation and role drop down options, forename, surname, username, password text fields, and gender drop down option. In addition Google's solution against spam (i.e. reCaptcha) and site bots will be displayed. This form would require the user or web crawler to type correctly the two words as shown by the widget. Since the image is actually a scan from an old book where letters are smudged and have inconsistent and irregular form, it is difficult for malicious programs to proceed. After several unsuccessful guesses the reCaptcha widget will restrict access to the system to the hacker's IP address. However this task is relatively easy for a human who is used to reading and writing text containing letters that deviate immensely from the standard print.

Google is offering this solution for free because this software can transcribe large quantities of textual information with 99.5% accuracy compared to the much inaccurate OCR method of digitising books. According to Google [1] their software can also be used for trusted online polls, preventing dictionary attacks guessing passwords for authentication, search engine bots unwanted indexing of web site content.

The reCaptcha software requires to be activated using valid Google Mail account. The developer would receive a public key for displaying the HTML/JavaScript widget, private key which is used during the Captcha code verification, and confirmation of the domain name associated with the web application. Even though this project does not use DNS services, the Ubuntu Server hostname or alternatively “localhost” can be used. Google's reCaptcha API can be accessed using PHP, Perl, Java, ASP.NET, and even WordPress. Upon code verification the registration script would pass the private key as an argument and the function will return either “success” or “incorrect-captcha-sol”.

The HTML form displays two special drop down lists – medical occupation and roles. The two were linked together so that when the user selects an occupation (e.g. Doctor) then he/she has to select a related role (e.g. General Practitioner or Surgeon) rather than looking at a long list of all possible roles. To achieve this a JavaScript file was included which contains all occupations, roles, and a function for the drop down lists. The usage of JavaScript was necessary to prevent the web page from being reloaded.

The JS file contains several arrays with medical role titles and two functions – `getRole( )` which reloads the role drop down list depending upon the user's occupation selection and `changeTheme( )` which sets the reCaptcha visual properties. The code snippet below shows one of the arrays and one of the functions. First the `registration.role.option` HTML element is reset from its previous state to allow for modification. Second the switch control loads the appropriate array into the drop down list by iterating over it and adding each array value as new option element part of the `<select>` element.

```
var dental = [ "Dentist",
               "Dental Nurse",
               "Dental Hygienist",
               "Dental Therapist",
               "Dental Technician"];
function getRole(selected){
    document.registration.role.options.length = 0;
    switch (selected){
        case 'dental':
            for (var i = 0; i < dental.length; i++){
                document.registration.role.options[i] = new Option(dental[i], dentalValue[i], false, false);
            }
            break;
```

In the **second step** `ProcessRegistration.php` will display exactly the same HTML form to the user which will contain his / her details. It is important to note that the text fields and drop down lists are

greyed-out and disabled so that the user will not get confused and start filling the form all over again. This is accomplished using the <input> element parameter – disabled=”disabled”. The purpose of this page is to verify whether the user has supplied suitable user details data.

If that is true the user will be notified and he / she can proceed further. However if there is unsuitable data entry the “submit” button will be disabled and the user will not be able to continue with the registration. The interface will notify the user of any issues and he / she will have to return back and start all over again. The validation class (check\_registration.php) has several functions shown in the class diagram below.

Validation	
+ correct: boolean	+ response: array
- message: String	- forename: String
- surname: String	- username: String
- password: String	- repeat_password: String
- __construct(): void	+ checkForename(): array
+ checkSurname(): array	+ checkUsername(): array
+ checkPassword(): array	+ checkAll(): String

The checkForename() and checkSurname() functions will return true only if the forename is longer than 3 characters, it is a single word, it contains numbers, and it contains uppercase letters. The checkPassword() function is similar with the exception that it requires at least 7 characters long password. The checkRepeatPassword() function will return true only if the two passwords supplied by the user have a binary match that equals zero. The checkAll() function will return a disable parameter for the “submit” button if any of the previously listed functions return false. This is used in order to prevent the user from registering with unsuitable user details. The following code snippet shows how the checkForename() function validates the length of the string and uppercase letter condition. The second code snippet shows that if the function returns a boolean true, visual green icon is displayed next to the text field. If the return is boolean false, visual red icon is displayed.

```
function checkForename(){
    if(!empty($_POST['forename'])){
        if (strlen($_POST['forename']) < 3) {
            $message = "Forename is less than 3 characters long.";
            $response[1] = $message;
        }
        if (!preg_match('/[A-Z]/', $_POST['forename'])) {
            $message = "Forename should contain uppercase letters.";
            $response[1] = $message;
        }
    }
}
```

```

if($checkForname[0]) {
    echo "<img src=\"media/correct.ico\" />";
} else {
    echo "<img src=\"media/incorrect.ico\" />";
    echo "<br/>";
    echo $checkForname[1];
}

```

In addition to the form validation, the reCaptcha code has to be verified. This is achieved using the verification class (checkrecaptcha.php) which makes an inquiry to the server passing two arguments – the private key and user supplied code. Both responses (e.g. error and success) are stored in message variable used to notify (e.g. green / red icon) the user. The verification class is detailed below using a class diagram.

Verification	
+ message: String	+ path: String
- response: array	
- __construct(): void	+ check(String): String

Finally the correct user details from the form array are stored in the super global PHP SESSION associative array for further manipulations. It is worth to note that each web page using this array should contain session\_start( ) function which creates a unique identification session for each new client connected to the server. When necessary small text files (cookies) are stored on the client's hard disk drive which contain permanent session variables associated with an IP address.

In the **third step** FinaliseRegistration.php connects to the MySQL database and sends the registration form details which are stored into the “user” table. This is achieved using the databaseOperations class used for SQL commands, such as – SELECT, INSERT, UPDATE, and DELETE. The method returns either an error message using PHP's native mysql\_error( ) function or an array which contains SQL result set using PHP's native fetch\_array\_assoc( ) function. The following code snippet shows how user data is stored in the appropriate table in the database.

```

function insertIntoUserTable($form) {
    $query = sprintf("insert into user (id, occupation, role, forname, surname, username, password,
repeat_password, gender, date) values ('', '%s', '%s', '%s', '%s', '%s', SHA2('%s', 224), SHA2('%s', 224),
'%s', '%s')", mysql_real_escape_string($form[0]),
mysql_real_escape_string($form[1]), mysql_real_escape_string($form[2]),
mysql_real_escape_string($form[3]), mysql_real_escape_string($form[4]),
mysql_real_escape_string($form[5]), mysql_real_escape_string($form[6]),
mysql_real_escape_string($form[7]), mysql_real_escape_string($form[8]));

    $result = mysql_query($query);
    if(mysql_affected_rows() == 0){
        $errorMessage = mysql_error();
    }
    return $errorMessage;
}

```

The SQL query is constructed using a function string (i.e. `sprintf( )` method) because it supports additional manipulation of data outside the string literal. Each user form data is interjected into the statement using “%s” from the array. It is important to note that each form string is passed through `mysql_real_escape_string( )` method in order to clear any SQL injection data manipulation.

SQL injection attacks are successful when user data input is not validated to prevent malicious SQL commands. According to the official PHP documentation an attacker can easily acquire all users' data such as passwords and confidential information such as credit card number. In addition an attacker can perform a DROP command which can delete an entire database. To prevent this from happening the Medical CMS makes queries using lower database privileges and filters all data inputs using the `mysql_real_escape_string( )` function. MySQLi and PDO\_MySQL are recommended alternative APIs which use special prepared SQL statements to deal with SQL injection issues. However they offer no actual benefit for the project and were neglected.

It is important to note that passwords in the user table are encrypted using the SHA2( ) method. This was necessary to prevent from data misuse during database exploitation. It is also possible for someone with limited access to the whole database (e.g. USE and SELECT commands only) to view all passwords in the user table. However if the text is hashed or data is obfuscated while preserving its integrity, then the hacker will only be able to see a long string of meaningless alphanumeric characters. Even though hashing is considered a cryptographic method it does not use a password or secret key during encryption. For example `SHA2('lsbu', 224)` will produce the following string which will be stored in 64 bit variable character column – `d5f83df2bf5454376ede5de10b3a1c454e04af29dc28c341cf24ad31`. The second argument decides the number of bits used for hashing. According to MySQL's official documentation the number of bits could be – 256, 384, and 512. MySQL offers other hashing algorithms such as SHA1 and MD5.

The function string query is sent to the database engine using the `mysql_query( )` method. This is followed-up by `mysql_affected_rows( )` method which returns the number of rows affected by the query. If the number equals zero this means that there was a problem and the `mysql_error( )` method should be called to determine the issues and notify the user or administrator.

## **5.2 Login**

The login script is used to authenticate users to the system. Only registered users can access the content management system. Depending on their privileges assigned by the administrator they can perform simple tasks or manage the CMS.

The login script checks whether the user supplied username and password match the records in the database. If that is true the user is redirected to the privilege appropriate interface otherwise the user is redirected to the error page. Upon successful authentication the SESSION array is loaded with user details which are useful for their specific home page. In addition it is possible for the medical staff to login into the system quickly without username and password. This was necessary because some personnel may need the system immediately. In this case they would use the “quick login” module which requires only a PIN (Personal Identification Number). In total there are three levels of access to the system. The first level authenticates users by “quick login” and allows medical staff to view, search, and navigate the web site without being able to add, update or remove any information. The second level authenticates users by “login” and allows the medical staff to also modify data in the content management system. The third level of authentication is only for administrators who can perform database and medical users modifications as well as general web site configurations. The figure below demonstrates the login procedure.

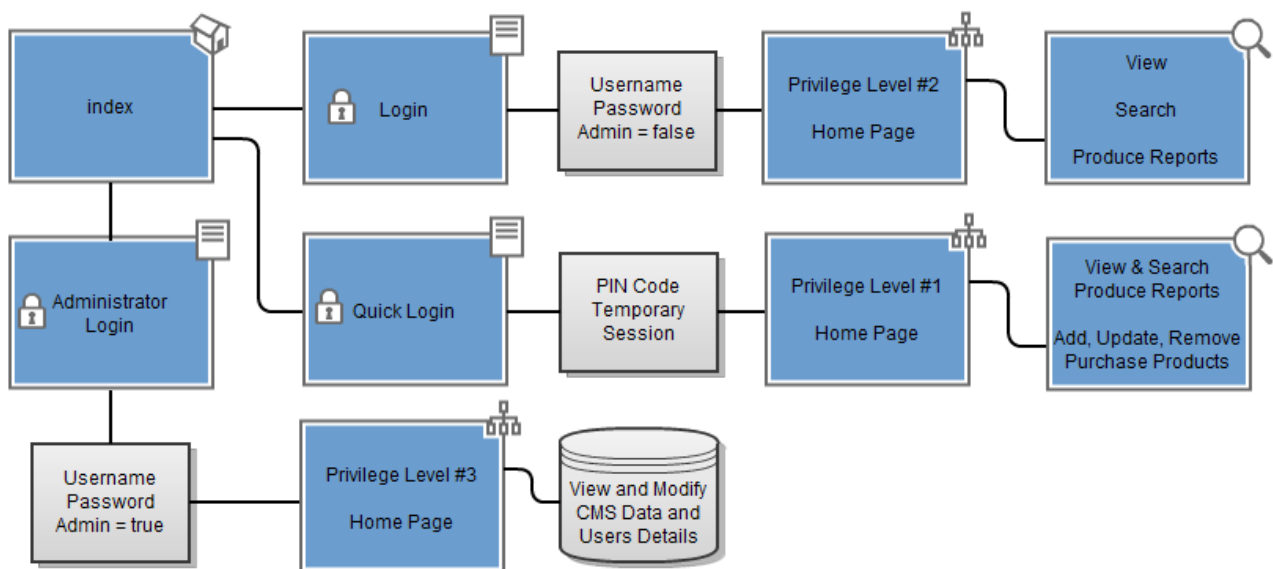


Figure 12: Login Procedure

Quick login procedure is similar to mobile phone screen lock protection. It avoids unwanted telephone calls while the device is placed in a pocket. Also if the mobile device was lost or stolen no one will be able to make expensive call / message charges on the owner's account. It is no more than 4 digits long because it is easy to remember and the owner can quickly start using the telephone. However for more advanced operations with the device it requires a secure alphanumeric password such as installation of mobile applications from the manufacturer's software store or change of important system settings. For example Apple iPhone iOS and Google Android OS lock the screen when the device is idle for more than a minute with PIN code. When the user is installing applications from the App Store or Google Play or changing his / her payment account,

the device will require a secure password. One major drawback of PIN codes is obviously the length – it would take only  $10^4$  combinations to crack the code. To fix this issue the device allows only three incorrect consecutive tries after which the mobile phone remains locked disallowing further trial and error. In this project the quick login module uses JavaScript to catch user input on the go. It allows only four numbers as PIN code, disallows further trial and error for the current SESSION id for 10 minutes after three consecutive failures, and prevents the usage of common PIN codes as described by [2] - 1234, 0000, 2580, 1111, 5555 etc.

The following code snippet from “quick\_login.js” is used to capture user input from the 0 to 9 dial, part of the quick login interface. There are two additional buttons – one for reset of the currently selected PIN code and another for code submit of the four digits long number. Every form input button calls the `getDigit()` function upon button click. The function also sends the value of the input form. The function adds each received digit into the `pinCode` array. When the number of digits equals 4 which is the standard and recommended PIN code length, the function displays the aggregated value of digits to the user. This value is also submitted as part of the form to the “QuickLogin.php” script. Only then the user can proceed and authenticate against the server database. There is another function which is used to reset or rather refresh the web page.

```
var pinCode = new Array();
var asterix = " *";
var i = 0;
function getDigit(digit) {
    pinCode[i] = digit;
    i++;
    if(pinCode.length == 4){
        document.quick_login.ten.value = pinCode[0] +
        pinCode[1] + pinCode[2] + pinCode[3] + asterix;}
}
function reseto() {
    location.reload();}
```

The following code snippet from “Login.php” is used to prevent three consecutive trial and errors which maybe part of a brute force PIN code cracking method. First it sets the super global PHP variable SESSION “trial\_error” to zero. The global variable is very useful because it remains in the SESSION until it is manually destroyed by the `session_destroy()` function. A local variable would not have been suitable because it will be reset each time the script is parsed by the server which would make it impossible to count the number of unsuccessful PIN code trials. The session variable is automatically incremented when the user decides to click on the submit button. If the value equals three, the quick login interface will remain locked for 10 minutes. Immediately the “trial\_error\_lock” variable is set to the current time – only minutes. To check whether 10 minutes have elapsed, the difference between the current minute and recorded minute is calculated. To prevent negative results the IF condition uses logical OR method to compare the difference.

```

if(!isset($_SESSION['trial_error'])){
    $_SESSION['trial_error'] = 0;
}

if($_SESSION['trial_error'] >= 3) {
    $disabled = "disabled=\"disabled\"";
    $message = "Try again after 10 minutes to Quick Login.";
    $message .= " Recorded at " . date('i') . ".";
    if((date("i") - $_SESSION['trial_error_lock'] >= 10) ||
        ($_SESSION['trial_error_lock'] - date("i") >= 10)) {
        $_SESSION['trial_error'] = 0;
    } else {
        $_SESSION['trial_error_lock'] = date("i");
    }
}
}

```

The following code snippet from “QuickLogin.php” is used to check whether the PIN code matches any entries in the database. If that is true the user is authenticated and provided with appropriate home page link according to their privilege and a greeting according to the “gender” column. Also this script is used to increment the value of the “trial\_error” variable. Then unset the “trial\_error” and “trial\_error\_lock” variables in the session only when a match has been identified.

```

if(isset($_POST['submit'])) {
    $_SESSION['trial_error'] += 1;
    $query = sprintf("select * from user where pin='%s'",
        mysql_real_escape_string($_POST['submit']));
    $result = mysql_query($query);
    if(!$result) {
        $errorMessage = mysql_error();
    } else {
        if(mysql_affected_rows() == 0) {
            $errorMessage = "No match was found!";
        } else {
            while ($row = mysql_fetch_array($result)) {
                unset($_SESSION['trial_error']);
                unset($_SESSION['trial_error_lock']);
            }
        }
    }
}

```

The following code snippet from “QuickLogin.php” is used to welcome the user. The proper honorific title (i.e. Mr / Mrs) is determined by the entry in the database for gender and the message is composed with the entry for surname.

```

if(isset($_url)) {
    if($_SESSION['gender'] == "Male"){
        echo "Welcome Mr " . $_SESSION['surname'] . "!";
    }
    if($_SESSION['gender'] == "Female") {
        echo "Welcome Mrs " . $_SESSION['surname'] . "!";
    }
}

```

```

$query = sprintf("select username, password from user where username='%s' and password=SHA2('%s', 224)",
    mysql_real_escape_string($_POST['username']),
    mysql_real_escape_string($_POST['password']));
$result = mysql_query($query);

if (mysql_affected_rows() == 0) {
    echo "<img src=\"media/incorrect.ico\" />";
    echo "<br/>";
    echo "Password is incorrect.";
}

```



The code snippet above from “ProcessLogin.php” is used to check for username and password matches in the database and notify the user if either is incorrect. If both are correct the user will be authenticated. If that is not the case the submit button will remain blocked.

After the user has been authenticated with username and password that match the records in the database a session variable is set which can be checked by other scripts (e.g. search, create report, send PM) to verify the user's authenticity. Also a query is made to the database to collect all information about the currently logged user.

The session will be valid for a certain amount of time. This is configured by the /etc/php5/apache2php.ini file which contains the “session.gc\_maxlifetime = 1440” parameter. It means that after 24 minutes the server will reset the array if the remote workstation is idle. Unfortunately PHP does not support garbage collection for optimal memory management which means that either shell script, cron job, or web server automation mechanism has to manually free-up the memory associated with a specific session id and associated pages and segment in the physical memory.

The session array also stores user privileges for the Medical CMS. All users have personal web pages with “Interface.php” for general purpose access, “Control.php” for advanced access, and “Admin.php” for administrator / moderators. The following code snippet demonstrates how this is achieved.

```
if($_SESSION['privilege'] = 1) {  
    $url = "Home/" . $_SESSION['forename'] . $_SESSION['surname'] . "/Interface";  
} elseif ($_SESSION['privilege'] = 2) {  
    $url = "Home/" . $_SESSION['forename'] . $_SESSION['surname'] . "/Control";  
} elseif ($_SESSION['privilege'] = 3) {  
    $url = "Home/" . $_SESSION['forename'] . $_SESSION['surname'] . "/Admin";  
}
```

It is important to note that the user specific directories already exist. They are created during the registration process and are inspired by the common Linux directory structure. All directories and files are “branches” mounted to the root file system (in this case the index.php location). Each new user requires a multi-level directory structure which contains web forms, interfaces, and controls. Also better organisation and easier security audit – generated reports, purchases reports, email / PM reports, and images. The following code snippet demonstrates how that structure is created. The PHP native system method is used to make direct kernel calls and return formatted responses.

```
function createNewUser($forename, $surname) {  
    $path = $forename . $surname;  
    system("mkdir -p Home/" . $path);  
    system("mkdir -p Home/" . $path . "/Images");  
    system("mkdir -p Home/" . $path . "/Reports");  
    system("cp Interface.php Home/" . $path . "/" );  
}
```

## 5.3 Administrator

The administrator of the content management system has the highest privileges compared with the other users that can access the database. The administrator's interface allows to view, add, insert, update, delete, manage products, users, messages, and reports.

The following sections discuss the administrator interface modules. It is worth to note that the administrator has the greatest control over the web application, and that restricted users will inherit similar interface with significantly fewer control options. To make sure that limited privileges users cannot access the master interface each page performs a simple SESSION check to verify the user's identity. This is achieved with the following source code placed at the very beginning of all admin web pages.

```
if(($_SESSION['authenticated'] == true) && ($_SESSION['privilege'] == 3)) {  
} else {  
    header("Location: Error.php");  
    exit;  
}
```

### 5.3.1 Administrator Welcome Page

The successfully authenticated administrator is redirected to a page which contains useful information about server and web application statistics. These environmental variables can be directly accessed using PHP and JavaScript. For example PHP's \$\_SERVER array contains valuable information about the server. However JavaScript can acquire valuable information about the client. This is possible because each web browser has to send workstation “headers” to the server. This data is used by the Apache web server to parse the contents of the file accordingly. Also the administrator can consult with MySQL database's status. Unfortunately the MySQL server status contains a lot of useless information and few variables were included. Finally each time the administrator or privileged user is authenticated and redirected to his/her personal interface, the application makes sure that this event is logged in the database with information about the user, date, and time. In fact that is the significant reason to include all these variables in the interface.

The following table details which global variables were used for the interface and their description.

PHP Variable Name	Description
\$_SERVER['SERVER_ADDR']	The web server's IP address.
\$_SERVER['SERVER_NAME']	The web server's host name.

<code>\$_SERVER['SERVER_SOFTWARE']</code>	Web server software – server operating system (Ubuntu) and web server application (Apache).
<code>\$_SERVER['REMOTE_ADDR']</code>	The workstation's IP address.
<code>\$header['Accept-Language'];</code>	Default language and region locale.

The following table details the “user\_info.js” JavaScript file which is used to acquire user specific information as sent by the workstation's web browser (e.g. Internet Explorer and Safari).

User Information JavaScript	
+ allHeaders: String	+ enabledCookies: boolean
+ browserVersion: String	+ result: String
+ ini( ): void	+ filterResults( ): void
+ getBrowserVersion( ): String	+ getOSVersion: String
+ getOSBits( ): Integer	+ displayBrowserVersion( ): void
+ displayOSVersion( ): void	+ displayOSBits( ): void
+ displayCookiesSupport( ): void	+ displayJavaSupport( ): void

In order to access the headers the DOM object “navigator” was used. The result is as follows:

Navigator Object – Property and Method	Description
appVersion	String associative array that contains web browser version, OS version, CPU bits, web browser engine – MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0; .NET CLR 2.0.50727; SLCC2; Media Center PC 6.0; CPNTDF.
appCodeName	The code name of the web browser – Mozilla.
appName	The actual name of the web browser – Microsoft Internet Explorer.
cookieEnabled	Some web browser prevent the storage of cookies for enhanced security.
javaEnabled( )	Java is not part of the Microsoft Windows operating system. However the user can install it in order to used fast performance real time applications.

Unfortunately the “appVersion” property returns rather unintelligible information. In order to understand the meaning behind the acronyms this MSDN resource [3] offers valuable guide into deciphering various Microsoft Windows tokens (e.g. MSIE and Trident).

The following code snippets show how the application determines the client's web browser version.

```
function getBrowserVersion() {
    if (allHeaders.match(/MSIE 10.0/i) != null) {
        result = "Internet Explorer 10";
    } else if(allHeaders.match(/MSIE 9.0/i) != null) {
        result = "Internet Explorer 9";}
}
```

```
function getOSVersion() {
    if (allHeaders.match(/Windows NT 6.2/i) != null){
        result = "Windows 8";
    } else if (allHeaders.match(/Windows NT 6.1/i) != null){
        result = "Windows 7";
    }
}
```

The variable “allHeaders” contains the result from the navigator object. It is matched using regular expression statement against browser identification code words as listed at [3]. When using Regex the global argument (i.e. 'g') is not necessary. This is because only the first occurrence of the string is needed. However the Regex case insensitive argument (i.e. 'i') is necessary. The procedure to determine the version or name identification of the operating system is similar as evident by the second code snippet. Again the identification reference is from [3].

The following code snippet demonstrates how the result from the header filtering is displayed to the administrator. The table data HTML element identified with “javaInstalled” will receive the value.

```
function displayJavaSupport() {
    if(javaInstalled){
        document.getElementById("javaInstalled").innerHTML = "Yes";
    } else {
        document.getElementById("javaInstalled").innerHTML = "No";
    }
}
```

Finally the MySQL database status command has to be executed, the result to be formatted, and stored in the log table. The console command has two arguments (i.e. GLOBAL and SESSION) both of which produce a table with 310 rows in total. Global means status form all users and databases dating back to the database's first run. Session means status from the current user and currently used database. It is reset when the user is no longer active – logged off. Since both return extreme amounts of information, the result was filtered specifically for the user. The few selected are listed in the table below along with brief description.

Variable Name	Description
Bytes_received	Total incoming database traffic in bytes.
Bytes_sent	Total outgoing database traffic in bytes.
Com_select	Total number of performed select statements.
Max_used_connections	Total number of currently active database users.
Uptime	The amount of time the database server has been running continuously.

It is important to note that some results are formatted for the administrator. For example the “Bytes\_received” variable is difficult to comprehend. The following code snippet demonstrates how the SQL query result is formatted for Kilo, Mega, and Giga bytes. Specific for this query is the MYSQL\_NUM argument. The result will not be an associative array but rather a numeric one.

```
$result = mysql_query("show global status");
while($row = mysql_fetch_array($result, MYSQL_NUM)) {
    if($row[0] == "Bytes_received") {
        $dbIncommingTraffic = round($row[1]/1024 , 3) . " KBytes";
        if($dbIncommingTraffic >= 999) {
            $dbIncommingTraffic = round($dbIncommingTraffic/1024, 3) . "MB";
            if($dbIncommingTraffic >= 999) {
                $dbIncommingTraffic = round($dbIncommingTraffic/1024, 3) . "GB";}}}

```

The log in the database is used for all administrators and authenticated members of the CMS. It contains the workstation's IP address, language locale (e.g. bg-BG), browser version, OS version, OS bits, username, and date/time. The latter is very important for accurate logging. The MySQL TIMESTAMP type was used which stores data in the following format → 2012-07-11 21:30:29. This is achieved using the following MySQL query. The time is recorded using the MySQL NOW() function.

```
$query = sprintf("insert into log (id, ip, language, browser, os, osbits, username, date)
values (null, '%s', '%s', '%s', '%s', '%s', '%s', NOW())",
$_SERVER['REMOTE_ADDR'], $language, $headers[1], $headers[2], $headers[3], $username);

```

### 5.3.2 Administrator Database Control

The administrator can manipulate data associated with the products in the database using four pages. They are **view**, **add new**, **update**, and **remove**. Their usage is necessary because the administrator has to be able to access the database indirectly through the interface rather than directly through the MySQL command line tool.

The view page displays all product records in the database. The user can sort the results by type (i.e. product name, manufacturer, retail price, date) and range (i.e. 5, 25, 50, 100, all).

The add page displays a form with which new products can be added to the database. The form combines text fields, drop down boxes, text area (for prescription details), and files upload (for knowledge base document and product cover image).

The update page displays limited list of products (the latest in the database). The user can only change single product information at a time.

The remove page displays limited list of products (similar to update page). The user can make single or multi selection of products to be erased permanently from the database. Unlike the update page the remove page interface allows for the administrator to remove several products in one go.

The following image below illustrates with a flowchart the links between the four admin controls.

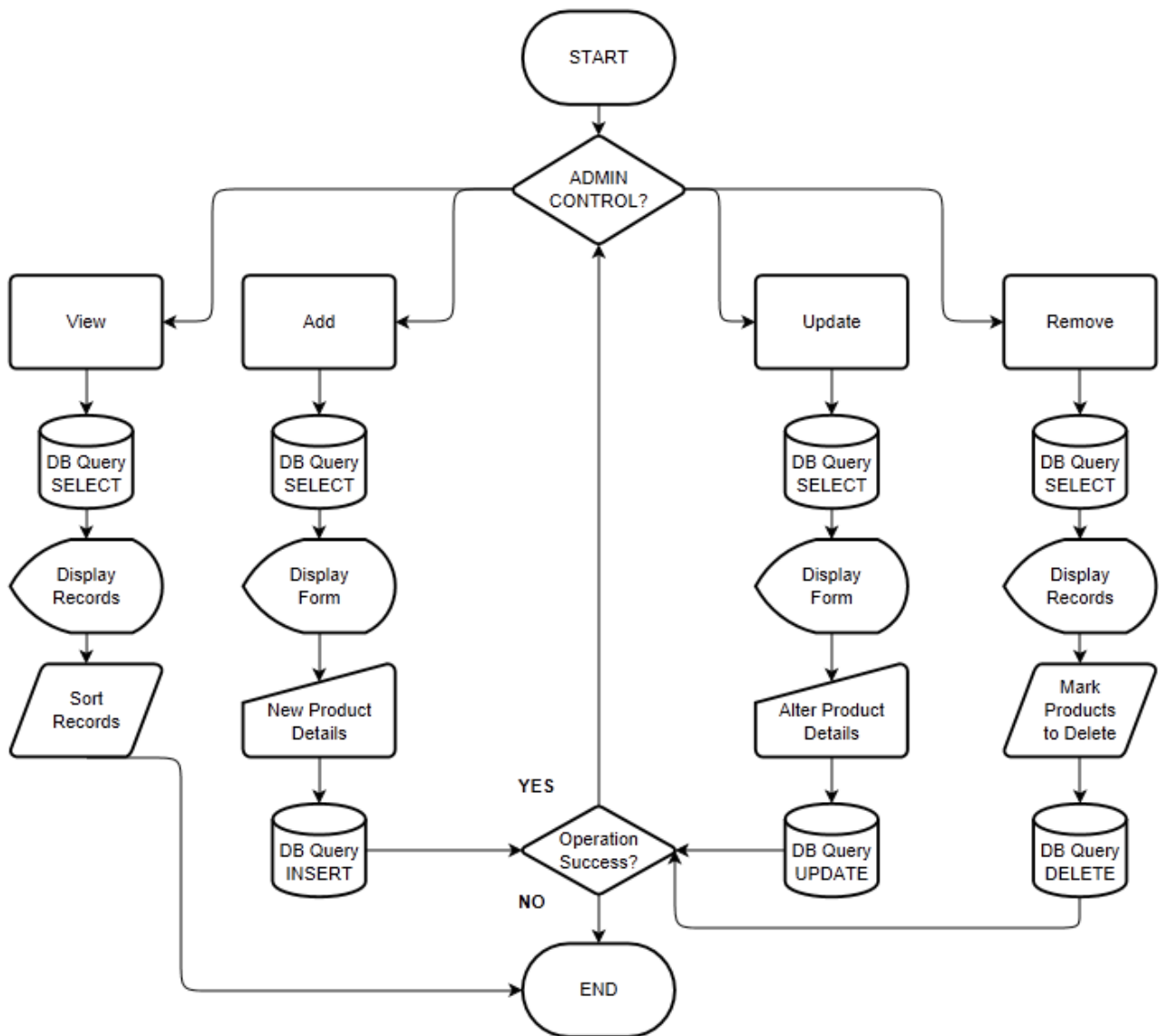


Figure 13: Administrator Database Control

The following code snippet demonstrates how product information fetched from the database is displayed using HTML elements and PHP iteration. The while loop iterates over the array containing the database records. Afterwards each table data HTML element will contain data from the array corresponding to the database table structure – in the example below the retail price and name of the product.

```

while ($row = mysql_fetch_assoc($result)) {
    echo "<tr>";
    echo "<td><a href=" . $row['kb'] . ">" . $row['name'] . "</a></td>";
    echo "<td>" . $row['retail_price'] . "</td>";
}
  
```

The product information array is generated using the selectAllProducts( ) database class function. It is worth to note that not all rows from the “product” database table are selected for a simpler and optimised display.

The sort variable is determined by a switch statement and the user's choice, for example in alphabetical order ascending or by date descending. Similarly the limit is also determined by a switch statement and the user's choice from a drop-down menu, for example 5 or all.

```
$query = "select name, manufacturer, retail_price, kb, cover, modified from product order by " . $sort .
$limit;

switch ($_POST['sort']) {
    case 'name':
        $sort = "name";
        break;

switch ($_POST['limit']) {
    case 5:
        $limit = " limit 0, 5";
        break;
```

To **add new product to the database** the user needs to manually type information about the product into the provided HTML forms. Especially useful was the possibility to group drop-down menu choices using the select – optgroup – option construct. Afterwards the submitted product data is processed by accessing the global \$\_POST PHP array which contains the form data. Then a query is sent to the MySQL server to add data to the “product” table in the relevant table rows. It was also important to perform simple form verification using JavaScript to prevent empty field data being submitted to the database.

The following input type text element calls the checkIfEmpty( ) JavaScript function on blur event. This means that the user has reached the element and then moved to another element with his / her mouse or using the keyboard tab key. The function sends a single argument “this” which is used to identify which of the many HTML elements the user left blank. The JavaScript code will check whether the value of the HTML element is null or empty. In that case the script will alert the user of the error and will notify the user which element was left blank.

```
<td><input type="text" id="name" name="name" value="Aspirin" onblur="checkIfEmpty(this)"></td>

function checkIfEmpty(HTMLInputElement) {
    var element = HTMLInputElement;
    if(element.value == 0){
        alert(element.id + " field cannot be left blank!");
    }
}
```

The form action leads to the same web page unlike previous implementation for the user's convenience. He / she will be able to immediately continue adding new products to the database. Also the user will get notified whether the operation was successful. This is achieved by checking the number of affected rows in the database. If their number is zero this would mean that the query failed and the database table was unaffected.

The SQL query for adding products to the database is using the INSERT statement and PHP's sprintf( ) function which allows for string and integer substitutions – %s and %d. Nonetheless there were three unusual arguments, part of the statement. First was the last modified date for each product. Using the MySQL NOW( ) function each time the query is run, the corresponding table row will receive time and date information. This is very useful for tracking product changes in the medical content management system. Second was uploading files to the system. There were two possibilities – use of MySQL's binary file data type for storage known as BLOB or use of MySQL's variable character data type where only the link to the file will be stored.

The benefit of using BLOB is that the image will be stored in binary format in the database from where it can be easily retrieved, there will be no need to rely on external storage. The drawback is that writing binary files (i.e. images and PDF documents) to the MySQL engine is more difficult than simply uploading the image or document file to predetermined storage space. Another drawback is the fact that it is much more difficult to manage images in a database which could lead to massive performance slow-down.

That is why the global \$\_FILES array was used to transfer the image file or document from the user's to the server's hard disk drive. The array can store multiple binary files depending on the number of input file type HTML element forms. The following code snippet represents part of the upload procedure. The foreach loop iterates over the \$\_FILES array looking for an error as defined by PHP's online manual [4]. Some errors that can be prevented are – exceeding file size, partially uploaded or corrupted file, blank field error, missing temporary file configuration on the server, and failure to gain write access to a specified folder designated for file uploads. In addition there are two necessary string constructs for the address / location of the temporary file and permanent file address on the server. Finally the move\_uploaded\_file( ) function moves the file placed in the server's temporary folder to the final destination where the CMS application stores images and documents. The permanent address of the file is stored in the database for future reference.

```
foreach ($_FILES["product_file"]["error"] as $key => $error) {
    if($error == UPLOAD_ERR_OK) {
        $temporary = $_FILES["product_file"]["tmp_name"][$key];
        $final = $_POST['name'] . "_" . $_POST['barcode'] . "_" . ucfirst($_FILES["product_file"]["name"][$key]);
        move_uploaded_file($temporary, "Documents/$final");
    }
}

$kbLink = "Documents/" . $_POST['name'] . "_" . $_POST['barcode'] . "_" . ucfirst($_FILES['product_file']
['name'][0]);
$coverLink = "Documents/" . $_POST['name'] . "_" . $_POST['barcode'] . "_" . ucfirst($_FILES['product_file']
['name'][1]);
```

The **update product page** displays all products using selectAllProductsUpdate( ) and allows the administrator to change the product details and immediately commit the changes to the database.



The method is similar to view product page where `mysql_fetch_assoc()` function is used to iterate over the results fetched from the database. There were three specific difficulties with the update form. First the file size limit was lowered from 1MB to quarter that value. The reason is that the user will be only change the cover of the product which should not exceed 262144 bytes or 256KB. Second is the hidden id for each product. It is required when querying the database to make sure the exact product is modified. However the following input arguments construct was used to make sure the administrator is not bothered by useless information to him/her such is the product id.

```
echo "<input type=\"hidden\" name=\"id\" value=\"\" . $row['id'] . \"\" />";
```

Third the number of details for each product was restricted. It would be unreasonable to display all possible information about medical products because it will fill the entire screen – very user unfriendly interface. The following snapshot represents the update product form.

Name:	Quantity:	Bulk Price:	Retail Price:	Barcode:	Last Modified:	Cover:
Aspirin	10 x Tablettes	5.99	9.99	1234999	2012-08-21 20:58:48	Z:\Documents\aspirin.jpg <input type="button" value="Browse..."/> <input type="button" value="Update"/>
Aspirin	10 x Tablettes	9	13	4321	2012-08-21 17:43:53	<input type="button" value="Browse..."/> <input type="button" value="Update"/>

Figure 15: Update Product Form

Upon user submission the `updateProduct()` function is executed which substitutes the product cover image and alters the records based on the table product id. The following code snippet checks whether the file size is within the acceptable range, moves the temporary file, performs the SQL query for product update, and checks for an error using `mysql_affected_rows()` function.

```
if($_FILES['cover_file']['size'] != 0) {
    move_uploaded_file($_FILES['cover_file']['tmp_name'], "Documents/$final");
    $queryFileUpload = sprintf("update product set cover='%s', cover_file_type='%s',
cover_file_size='%s' where id='%s'",
    $coverLink, $cover_file_type, $cover_file_size, $_POST['id']);
    $resultFileUpload = mysql_query($queryFileUpload);
    if(mysql_affected_rows() == 0) {
        $errorMessage = "Error!";}}}
```

The **delete product page interface** displays all products to the user and allows him / her to make multi selection of products using check boxes in order to perform swift and easy remove product operations. The functionality of this page is very similar to update page.

The major difference is the form which encompasses all retrieved product records from the database and adds a check box for each. Each check box input type has two useful parameters – name and value. The name may be arbitrary and the value is either on (checked/enabled) or off. In order to delete the exact product from the database the name parameter is actually the product id in the table. Upon submission the deleteProduct( ) function will perform an SQL query erasing the correct product from the database. The following snapshot represents the delete product page interface.

Name:	Quantity:	Bulk Price:	Retail Price:	Barcode:	Last Modified:	To Delete:
Aspirin	10 x Tabletes	6	10	1234999	2012-08-23 17:52:01	<input type="checkbox"/>
Aspirin	10 x Tabletes	9	13	4321	2012-08-21 17:43:53	<input type="checkbox"/>

Figure 15: Delete Product Page

Upon submission the PHP script will iterate over the global \$\_POST array looking for the ticked check boxes and specifically the product ids which will be passed to the deleteProduct( ) function. It is important to note that the script is sorting through the array's names rather than their values. That is because a ticked check box value is simply the string “on”. This is quite useless for the SQL query. However the names are such as – name, brand, retail\_price, 12, 13 etc.

Using the is\_numeric() functions the script can easily retrieve only the product ids and nothing else.

```
if(isset($_POST['submit'])) {
    foreach ($_POST as $attributeName => $attributeValue) {
        if(is_numeric($attributeName)) {
            $errorMessage = $db->deleteProduct($attributeName);
            if($errorMessage == "") {
                echo "The product was successfully deleted.";
            } else {
                echo "There was an error. Please try again.";
            }
        }
    }
}
```

The following code snippet demonstrates how the remove product SQL query was constructed taking into account user selected products. It is important to note that there is no loop for multiple product ids. This is handled by the previously shown code snippet which uses the foreach loop to call the function each time there is a numeric match found in the \$\_POST array.

```
function deleteProduct($attributeName) {
    $query = sprintf("delete from product where id='%s'", $attributeName);
```

### 5.3.3 Administrator Database Control Problems

The following section describes some of the problems experienced during the development of the module in question. The five problems discussed are MySQL INSERT **permissions**, JavaScript **page loading** constructor, checking for **blank input fields**, **file uploads to the server** using HTML form and PHP super global arrays, and result **pagination**.

The administrator's database control page interface has to perform four simple operation corresponding to the MySQL syntax of SELECT, INSERT, UPDATE, and DELETE. These operations are only possible if the MySQL authorised user (username and password) has been granted the appropriate **permissions** to the database table. The initial setup of the Medical CMS was using a very restricted rights to the “medical” database in order to prevent security threats. Unfortunately problems with table access grants are difficult to establish using the ubiquitous PHP `mysql_error()` function. Fortunately executing INSERT and UPDATE commands using the MySQL command line tool produced the following error.

```
ERROR 1044 (42000): Access denied for user 'medical'@'localhost' to database 'medical'
```

To determine which database and table grants the MySQL user was missing the following command was executed.

```
mysql> show grants;
Grants for medical@localhost
GRANT USAGE ON *.* TO 'medical'@'localhost'
GRANT SELECT ON `medical`.* TO 'medical'@'localhost'
```

The problem was that user's permissions were set only to use and select records from the database. To fix this problem logged on as root user the following command had to be executed.

```
grant insert, update, delete, alter on `medical`.* to 'medical'@'localhost';
```

Browser or client side scripts are usually executed only when called from HTML elements such as button or input type. In order to execute a script without requiring obtrusive user interaction the JavaScript file can be called into action upon web **page loading**. There are several possibilities using native DOM (Document Object Module) functions and AJAX (Asynchronous JavaScript and XML). The easiest solution to the problem was using the following construct which calls the initialising function. It can acquire client browser data and call other functions as well.

```
window.onload = ini;
function ini() {
    cookies = navigator.cookieEnabled;
    filterResults();
}
```

To promote well structured Medical CMS it was necessary to prevent **blank input fields** from being submitted. Rather than constructing function for each HTML element input text field a different approach was used. As a result there is only one function that checks the currently blank input text field. Then using the DOM model it gets the value (e.g. Aspirin) and id of the HTML element. Thus the user can be easily notified which element was left blank. To activate the CheckIfEmpty routine the native JavaScript function `onblur( )` was used. It will execute the specified user routine when the user is not working the selected field any more.

**Uploading files** using PHP was not a straightforward procedure and required access to the server's terminal. The maximum file size is 1 Mega Byte for each product. This would prevent the server from running out of hard disk space. PHP uses the `MAX_FILE_SIZE` parameter in bytes to determine the restriction. To prevent misunderstanding the IEC approved units for binary were used. This means that 1 Kilo Byte equals 1024 Bytes. For 1 Mega Byte 1024 multiplied by 1024 equals 1048576 Bytes it total.

Common pitfall with uploading files to Linux servers is folder and file access permissions according to [4]. For example the directory named “Documents” part of the Medical CMS originally had 644 permission or read and write for the owner and only read for group and other users. Since the PHP script belongs only to the `www` (i.e. Apache Web Server) group, it is in fact considered as “other” user. In order to allow remote users to upload images and documents to the server, the “Documents” folder's permissions had to be altered to read and write for both group and other. The following code snippet shows how that was achieved using the “`chmod`” tool.

```
georgi@server:~$ chmod 777 Documents
```

The technique used to debug this problem was PHP's `print_r( )` function which can parse the array containing uploaded files – `$_FILES[ ]`. This function is similar to HTML's `<pre>` tag. It applies special formatting so that is easy to read and analyse.

From the output it was easy to determine that there were no errors during the upload from the client to the server and the file was stored in the temporary folder. The failure occurred during the stage of moving the temporary file to the destination prescribed by the script (i.e. `$_HOME/Documents`).

It was difficult to test the possibility of hundreds of products in the database. Nonetheless this is most probable after a month or more real system usage. In that case the problem of paginating the results will be significant. Paginating means that rather than displaying one thousand products on one page, the output is segmented or paged into equal size segments.

Traditionally this is achieved by placing numeric ranges at the very bottom of the web page. Thus results from the query will be displayed from zero to nine, 10 to 19, 20 to 29 etc. In order to do that an SQL command is used which will restrict the number of results fetched to the PHP array. The command requires two parameters – offset position and range (e.g. 10, 20, 30).

However there is another technique used in such cases which was implemented in this project. The range limitation remains in place. However the user can see more records from the database upon clicking on a button which changes the range. For example the initial number of records shown on the web page is 25. Upon pressing the button the page will refresh and display 50, further request will produce 75 records and so on.

The real benefit of this method compared with the traditional is that the user will eventually see all possible records. Whereas in traditionally paginated pages they will not remember on which paging range the desired product was shown. Thus they would have to keep going back 10 records in order to find what they are looking for. Another benefit is that the user can easily take control of the built-in Internet Explorer or other web browser's search functionality. This would be impossible if only 10 or 15 records are shown at a time.

The drawback of the implemented method is without a doubt high memory consumption. However it is worth to note that such a problem is not as relevant with modern personal computers. Irrelevant of the high number of records shown, the web browser will not require more than 200MB of volatile memory. This would seem insignificant when most PCs nowadays are equipped with upwards of 4GB of RAM.

### **5.3.4 Administrator User Database Control**

The administrator should be able to control the main asset of the content management system – the products. Also it is important for him / her to manage user accounts and their details and privileges.

The interface is similar to the product control interface. In other words authenticated users with administrator privileges (i.e. level 3 in the database) can view, add, update, and remove user information from the database. The novelty in the interface is the search and sort mechanism when viewing users, choice of privileges and annotation of the user profile during creation, replacement of the user profile image id, modification of the user password, and removal of user profiles with a single button.

The following diagram illustrates the functionality of the administrator's user view control script.

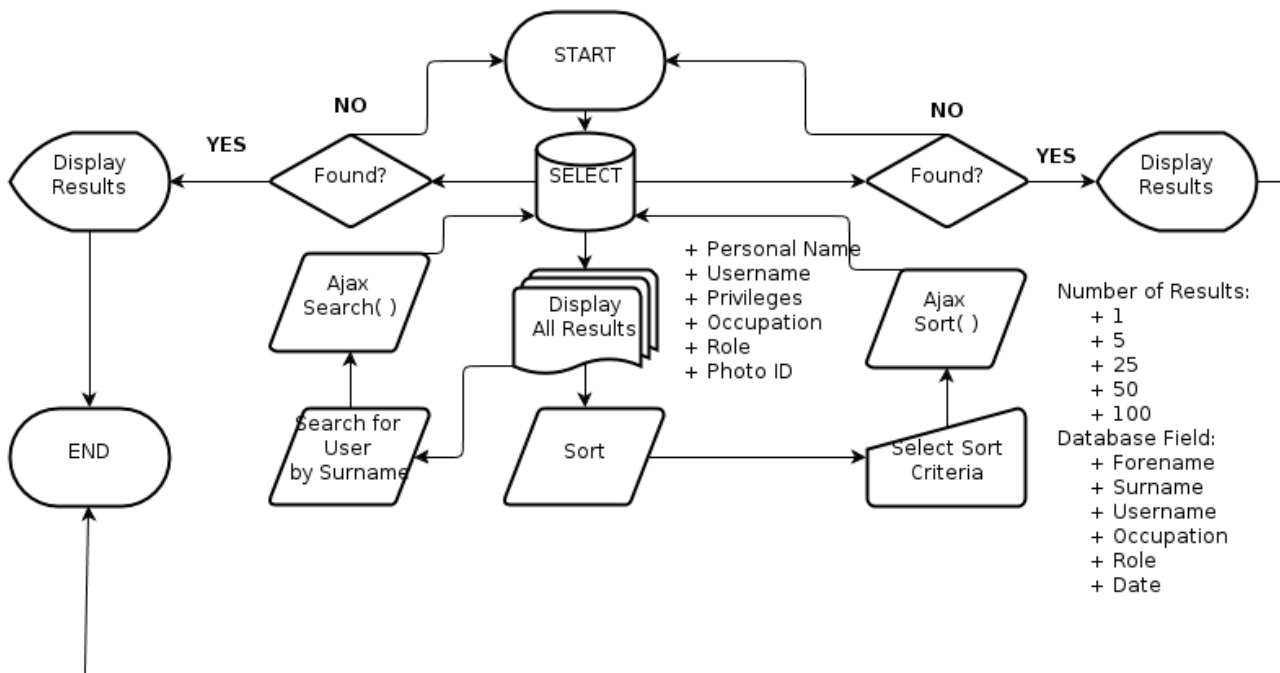


Figure 16: Admin View User Flowchart

The new feature implemented in the script is the usage of AJAX requests and responses to send the search and sort arguments and receive the result. The major benefit of using this JavaScript based framework is that the web page does not need to be refreshed every time a search query is sent for processing. Another important feature of this W3C standard technically known as XMLHttpRequest Level 2 is the ability to load plain text files, load XML formatted documents, and scripted files – most notably PHP and Microsoft's ASP. The practical application of which is that when searching, the user will not be inconveniently forwarded to another web page with search results. For this to be possible a JavaScript function creates XMLHttpRequest object which can handle interaction with external script files (e.g. PHP). It is important to note that by default it is impossible for JavaScript to load files due to security concerns, it is heavily restricted to writing and reading browser cookies. Another new feature of the script is the combined result for the user's names. Rather than wasting three table data fields (i.e. forename, surname, and gender) the user data is retrieved from the database and formatted in the following manner – Mr Georgi Butev.

The following code snippet demonstrates how an external script is called using JavaScript / AJAX and how the results are displayed to the user. It was constructed based on recommendations and source code from [6]. The searchAllUsers( ) function is called using the on key up event (i.e. the user has finished typing). The “search” argument is in fact the search keyword. Afterwards the XMLHttpRequest object is created and two functions are executed. The open( ) calls the external script file passing the keyword, in this case the user's forename using the GET method. The send( ) function is necessary to finalise the procedure so that results from the script can be received.

This is in fact achieved with the xmlhttp.responseText. The response from the script is appended to the HTML element identified by the keyword search. The response is generated based on the SQL script and formatted using PHP's echo function and HTML <table> tag.

```
$query = sprintf("select * from user where forname='%s'", $name);
```

It is important to note that readyState equals 4 means that the transfer is done. Also status equals 200 means OK or successful HTTP request. For example the popular status equals 404 means the web page was not found according to [6].

```
function searchAllUsers(search) {
    var xmlhttp;
    xmlhttp=new XMLHttpRequest();
    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {
            document.getElementById("search").innerHTML= "";
            document.getElementById("search").innerHTML=xmlhttp.responseText;
        }
    }
    xmlhttp.open("GET","ajaxSearchUsers.php?forename="+search,true);
    xmlhttp.send();
}
```

To sort the retrieved user results similar JavaScript / AJAX construct is used. The code snippet below demonstrates how this is achieved. The difference between the two is that there is an actual form button being pressed to sort the results. Also the JavaScript has to retrieve the set value for number of results to show and type of sorting from the drop-down HTML elements. In addition the GET method is used to send two arguments to the PHP script – type and limit. As a result the SQL query is slightly different as shown below.

```
$query = sprintf("select * from user order by %s limit 0, %s", $type, $limit);
```

```
For example: mysql> SELECT * FROM user order by date limit 0, 50;
```

The response from the script is displayed in similar manner using PHP's echo to generate user table.

```
function sort() {
...
    var type = document.getElementById("type").value;
    var limit = document.getElementById("limit").value;
...
    xmlhttp.open("GET", "ajaxSearchUsersSort.php?type="+type+"&limit="+limit,true);
...
}
```

The add new user interface was improved from add new product by automating some of the input tasks for the administrator. Thus he or she will spend less time adding new users to the database which can be very beneficial if the initial number is one hundred. These include automatic generation of **username**, **password**, and **pin number**. An attempt to guess the gender of the user based on the name was coded however no solution was found due to script failure discussed later.

The **username** is generated on focus event which means that the user has clicked inside the text field or tabbed into it.

The following code snippet is executed which takes both the forename and surname, concatenates them in lower case, so that the surname precedes the forename which is only the first character of the original. For example – Georgi Butev will become butevg.

```
function generateUsername() {  
...  
    var username = surname + forename.charAt(0);  
    document.getElementById("username").value = username;}  
}
```

The **password** is generated based on the user's surname, the length of the surname, and current day of the week (e.g. Monday) with the first character in upper case. This is medium strong password because it is alpha-numerical, contains upper case letters, and it is more than six characters long. For example Friday5Butev where 5 is the length of the user's surname. It is important to note that unlike PHP's function date( ), JavaScript returns only numerical presentation of the day of the week. The solution to this issue was to use a switch statement based on the date( ) function's output and output hard coded days of the week.

The **pin number** is generated based on the JavaScript's Math library. The random( ) function can generate pseudo-random numbers between zero and one. To create a pin number that is four digits long the result can be multiplied by 10000 and in combination with floor (i.e. real number not floating point) to get useful output. The code snippet below demonstrates how this was achieved.

```
function generatePin() {  
    var pin = Math.floor((Math.random()*10000) + 1);  
    document.getElementById("pin").value = pin;  
}
```

Other than the previously mentioned improvements the SQL script remains similar to add new product script except for the password field. Since this is admin only interface the password is not protected from prying eyes by the asterisk. That is because the administrator should be experienced ICT person with knowledge of IT security. Also this is useful in order to prevent password mismatch. In addition to few new database columns such as notes (for annotations such as warnings), repeat\_password column is used to store the previously used password by the user. Thus if he or she was to forget their current / latest password they can try with the outdated one.

Some of the **problems encountered** were due to JavaScript and formatting AJAX responses. Some of the JavaScript files did not respond to any request even though the files were copied and barely modified from the original – Registration.php. It was at that point that the author found out about JavaScript debuggers. Microsoft introduced web development tools with Internet Explorer 9, Google Chrome since inception, and Mozilla Firefox is proud of Firebug. Thanks to the debugger the few mismatched names were fixed and the script was responsive again.



One of the major differences between the registration script and the admin add user script is the ease of automation. Several JavaScript functions were added to generate suggestions based on the forename and surname of the user. One of them was the `guessGender()` function. The solution was to access the data gathered by the authoritative and trusted source – the official website of the USA social security statistics. From the following web page [5] users can download a list of 1000 popular names in the United States. The text file contains the gender associated with the name and number of occurrences. It is important to note that this would work only for people from countries like the US, UK, Australia, Canada etc. My name would not be recognised even though it is very similar to George. That is because the script was using regular expression patterns and returned true only on exact match. In addition some languages like Bulgarian and Czech use the suffix "a" to denote a female name (e.g. Cvetelina, Jitka). This could be implemented by future developers if the system is used in Eastern Europe. The JavaScript files did grind to a halt once more. This time the debugger could not determine the cause, restarting and clearing the browser's cache did not fix the issue. The most probable cause for this problem is the fact that JavaScript cannot load files from the hard drive and assigning 1000 names to a single variable brakes the Internet Explorer's JavaScript engine.

Another encountered problem was that after sorting, the database results were displayed in many separate tables. This was not functionally incorrect but visually disturbing. The culprit was the while loop which retrieves and displays the database records. Due to the usage of AJAX, the HTML `<table>` tag has to be reset each time a JavaScript function is called which causes unexpected output. This was achieved by referencing a `<div>` tag by its id which encompasses the table filled with user data. Thus it was easier to hide the division with the following JavaScript.

```
document.getElementById("search").innerHTML= "";
```

The solution was to use the ubiquitous PHP echo command to print out the table and the HTML tags before the loop. The benefit of using AJAX interfacing between an HTML page and script file is that the PHP script can use various formatting which will not be obstructed by JavaScript. In other words the output is not assigned to a single variable, it has direct access to the HTML page.

The **update** user page is also similar to update product page. The more important challenges were file upload, display of user privileges, and password update. The script had to make sure that when updating the user's details the existing image identification link is not replaced. There were two ways to accomplish this. First the use of `$_FILES['avatar']['error']` array which can be checked for errors – file was too big (1 and 2), file was partially uploaded (3), no file was uploaded (4), server access error (6). Second the use of `$_FILES['avatar']['size']` array which can be checked whether it is zero. Furthermore to make sure that files are unique PHP's `mt_rand()` function was used.

According to PHP's official manual it is faster and more unpredictable based on Mersenne Twister's algorithm compared with `rand()` which is much slower.

The user privileges are stored as `TINYINT` in the database. The solution was to use switch statement based on the retrieved digit and hard coded string values. Also the drop-down form had to be generated before displaying the results from the loop.

Finally the password should be changed with user update. That is because SHA2 is only a hashing algorithm and the password cannot be decrypted and displayed to the administrator. It can only return true or false based SHA2 matching two strings – the original password and user supplied.

The remove user page implements two significant differences for the administrator compared to remove product page. First the administrator can delete only one account at a time with the form button supplied with every record displayed to the user. This was implemented to prevent accidental removal of users that should not be deleted from the system. Also after some testing it proved much more convenient to use single button click compared to check boxes. Second was the use of the title HTML attribute. When the administrator hovers with the mouse pointer over the name of a user a floating tool tip appears displaying the date record from the database. Also hovering over the user's avatar a floating tool tip appears displaying notes about the user (e.g. warnings).

### **5.4.1 User Database Control**

The user database control available to registered and authenticated users to the content management system is similar to the administrator's. The user can see server statistics, view, search, add, update, and remove products from the database with the exception of managing CMS users. The reason is to ensure a secure system. Only users elevated to administrator's privileges can manage user details in which case they will use the administrator interface. This section of the document describes only the improvements made to the scripts that perform the previously mentioned database control tasks. The most significant improvement / change to the scripts is the use of JavaScript and AJAX.

The benefit of using AJAX for this project is that scripts can be loaded using generic JavaScript functions in the HTML web page. The result of the script is displayed to the user directly without the need of refreshing the web page or redirecting to success / fail web page. This feels more like a desktop application and it is a popular trend when developing web applications rather than simply dynamic web sites.

It is important to note that the use of JavaScript frameworks was considered due to the increased quantity of JavaScript source code for CSS, DOM, and AJAX.

Popular frameworks such as YUI (Yahoo!' User Interface library), Dojo (IBM's JavaScript library), jQuery (most popular), and MooTools (animation and graphic effects). The benefit of using any of these frameworks is the fact that less source code is written and a lot of available extremely useful and common to web development functions (e.g. AJAX, loading images, animation, generating pie charts, rich text editor). For example selecting HTML elements or using DOM properties is significantly easier.

The drawback is the fact that these frameworks alter the way JavaScript is written. Without a doubt the purpose is rapid software development. However the target audience is clearly the enterprise (e.g. jQuery, Dojo) and professional software developers working in teams of dozens and applying the Model – View – Controller methodology.

The reason NOT to use JavaScript frameworks is actually to learn JavaScript scripting without unnecessary additives and code abstractions. For example all AJAX interaction was achieved without frameworks and so were basic animation effects discussed later.

The product web page allows the user to perform the four database operations without being redirected to other web pages. Initially the web page is loaded with empty HTML <div> tags which later are used to display the view, add, update, and remove product HTML forms. When the user selects any of the icons, a JavaScript function makes an AJAX call to load the HTML form from different script (e.g. ajaxViewProduct.php).

The function will also hide the rest of the divisions (e.g. add, update, remove) to make sure the user sees only the view form. This logic is applied to the other links too – the active division is shown to the user and the inactive hidden from sight.

Afterwards the user can perform operations involving the database itself such as SELECT for searching against matching user supplied keyword, INSERT for adding new entries, UPDATE for altering existing entries, and DELETE for removing existing entries in the database. With the exception of INSERT, the user first has to search for a product before performing a subroutine. This exception is due to the fact that the user only uses the HTML form for the INSERT subroutine.

The following figure #17 illustrates Product.php and its operation dependencies.

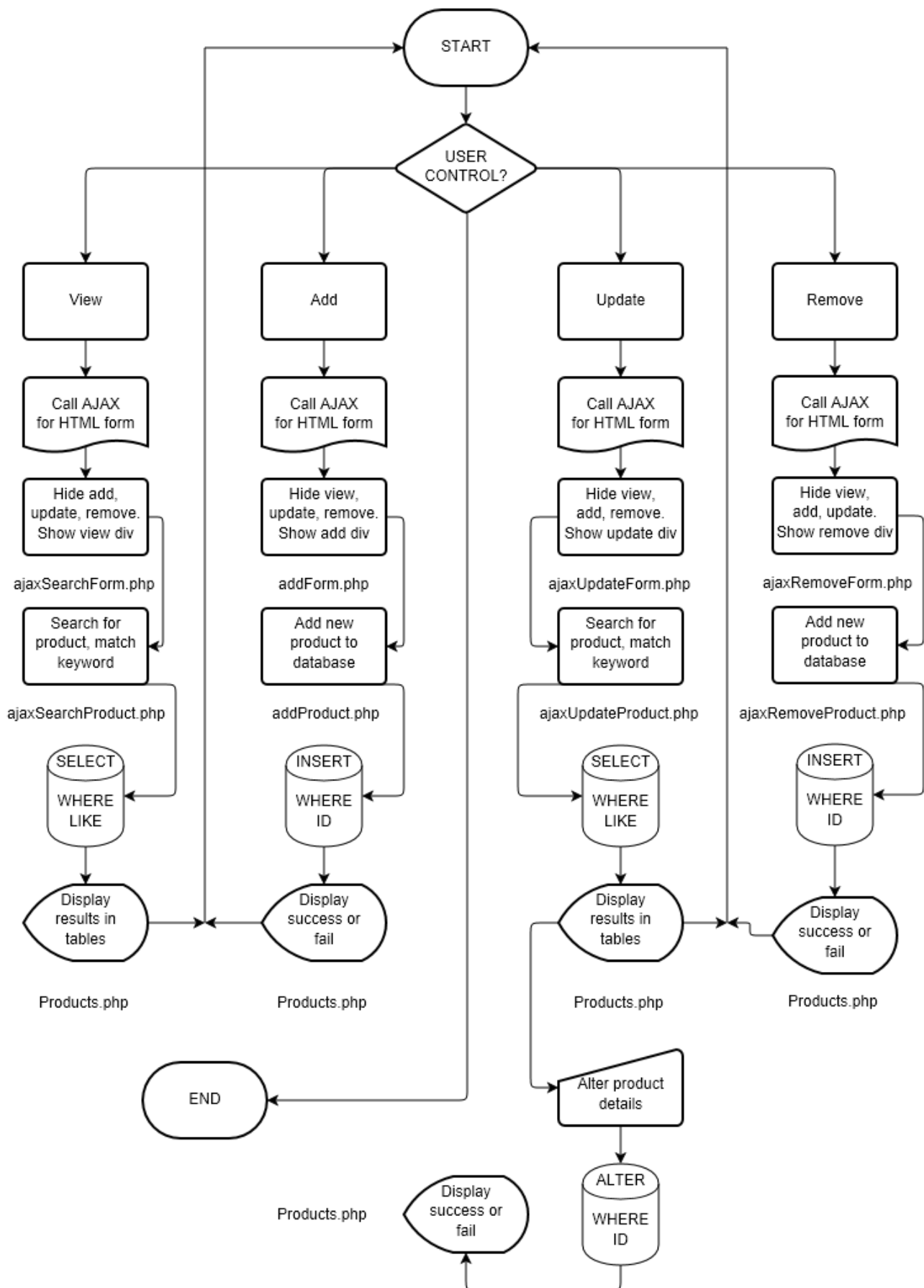


Figure 17: User Database Control

## 5.4.2 User View Product

This section describes the function of the view product script. The JavaScript view ( ) function is called whenever the user performs on click event – clicking with the mouse once on an anchor tag. Then a form will be displayed for searching products (from ajaxViewProducts.php) in the database based on the following criteria – name, Latin, brand, manufacturer, category, barcode, man\_price, retail\_price, and prescription. The results can be ordered alphabetically, by date, and by retail price. There is no limit in the number of retrieved results.

The following code snippet demonstrates how the HTML form is displayed to the user. The AJAX construct was based on recommendation from the W3C organisation [6]. First the XmlHttpRequest object is created. Then a function is executed only if ready state equals 4 (i.e. request has finished successfully) and HTTP status code equals 200 (i.e. OK). The function clears out the destination division and then overloads it with the text reply from the script. The open( ) and send( ) functions are also necessary to complete the XML Request. It is important to note that the GET method is used which passes an argument notifying the script that only the form is required – no SQL query. The GET method is more convenient because no data is visible to the user (due to AJAX) and using the POST method is more convoluted even though there is no real benefit from it. That is because POST requires additional data to be sent such as HTTP headers. The “true” parameter instruction means asynchronous data transfer which is used in conjunction with onreadystatechange. Alternatively false could be used for synchronous data transfer which defeats the purpose of AJAX.

```
var xmlhttp;
var submit = "no";
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if(xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        document.getElementById("viewProducts").innerHTML = "";
        document.getElementById("viewProducts").innerHTML = xmlhttp.responseText;
    }
}
xmlhttp.open("GET", "ajaxViewProducts.php?submit="+submit,true);
xmlhttp.send();
```

While the above script is executing, the following code snippet hides inactive divisions from the user with fade-out animation. Usually this would require JavaScript framework such as jQuery (250KB in total). However my solution was to use the CSS element property opacity to achieve the fade-out effect. Using the native JavaScript function setTimeout( ) which is similar to sleep( ) in Java and wait( ) in C++ delays the execution for number of specified milliseconds. After 500 milliseconds the remaining division's (text and images) opacity will be reduced to 0.5 units. Then after 500 milliseconds the remaining visible divisions will be hidden.

```

var interval = 500;
var opacity = 0.5;
var hide = 1000;

setTimeout(function(){document.getElementById("addProducts").style.opacity = opacity;}, interval);
setTimeout(function(){document.getElementById("updateProducts").style.opacity = opacity;}, interval);
setTimeout(function(){document.getElementById("removeProducts").style.opacity = opacity;}, interval);
setTimeout(function(){document.getElementById("addProducts").innerHTML = "";}, hide);
setTimeout(function(){document.getElementById("updateProducts").innerHTML = "";}, hide);
setTimeout(function(){document.getElementById("removeProducts").innerHTML = "";}, hide);

```

The fade-in and fade-out effect supplied by jQuery can be fully imitated using this method – a while loop which increases / decreases the division's opacity by 0.1 units every 200 milliseconds.

The actual searching routine is called using the on key up event – the user has finished typing the keyword. The AJAX function is slightly improved because it also checks for ready state 1 (function has been called), ready state 3 (loading response text), and HTTP status codes 400 (failed request), and 404 (page was not found). Apart from notifying for request errors, ready state 3 is used to notify the user that content is being loaded. The request is sent with three arguments – keyword, type, and order. The result from the following SQL is displayed to the user in HTML table format.

```
$query = sprintf("select * from product where %s='%s' order by %s", $type, $keyword, $order);
```

The result from the following SQL is used to search for products from the database. Since it is impossible to match long text with simple SELECT statement, the LIKE statement is used. The MySQL LIKE statement is very convenient in this case because it is similar to grep and sed tools part of Unix operating systems. Rather than making a perfect match, the user can find a product in the database by matching some words in the prescription column. This is possible due to LIKE's support for regular expressions such as start with char, end with char, containing characters.

```
$query = sprintf("select * from product where %s like '%s%' order by %s", $type, $keyword, $order);
```

Minor format issues are taken care of when displaying the result to the user. For example delivery time (i.e. day, week, month) should be in plural form if delivery number is bigger than one.

```

if($row['delivery_num'] > 1) {
    $delivery = $row['delivery_num'] . " " . $row['delivery_days'] . "s";
} else {
    $delivery = $row['delivery_num'] . $row['delivery_days'];
}

```

Also when displaying product covers, the path should be corrected by going back two directories where all the images are stored.

```
echo "<td><img src=\"../../\" . $row['cover'] . "\"" width=\"100\" height=\"100\"></td>";
```

### 5.4.3 User Update Product

This section describes the function of the update product script. The procedure is similar to user view product with several important differences. First there is a four tire structure which corresponds to four different AJAX calls to a single PHP file script.

if(\$getForm == "search") { ... }	Only display HTML form.
else if (\$getForm == "update") { ... }	Use SQL to retrieve results and provide the user with ability to add, update, and remove product.
else if (isset(\$_POST['submit'])) { ... }	Perform SQL query.
else { ... }	Raise unhandled exception.

Second to properly display drop-down select options the following if else control structure is used for all options part of a single select. It makes sure that the retrieved record in the database is marked as selected while at the same time displaying all other possible options which should not be marked as checked.

```
if ($row['intake'] == "Peroral") {  
    echo "<option selected=\"selected\" value=\"Peroral\">Peroral</option>";  
} else {  
    echo "<option value=\"Peroral\">Peroral</option>";  
}
```

Third the success / fail of an SQL operation and notification was modified. The executionSuccess() method part of the ExecutionSuccess class is called to determine how many MySQL rows were affected by the query. Usually MySQL will return 0 on fail and more than 1 on success. The class is represented in the following class diagram.

ExecutionSuccess	
+ success: String	+ errorNumber
+ errorMessage	
+ __constructor( ): void	+ executionSuccess( ): Boolean
+ executionSuccessNoDebug( ): String	+ getErrorNumber( ): Integer
+ getErrorMessage( ): String	

Forth the result from the SQL query is displayed in a different division placed beneath the HTML search form for more appropriate display format.

#### 5.4.4 User Database Control Problems

Some of the problems experienced during the development of this module were – base path location, working with AJAX requests, fade-out effect algorithm, opacity bug, search engine product reference, alternatives to success / fail MySQL operation.

During the development of the administrator's interface all PHP file scripts were in the same directory or were placed in directories directly referenced from the base path. However this was not the case with the user's interface because of the build structure – Home/GeorgiButev/Products.php. This meant that all necessary files such as CSS, images, icons, JavaScript libraries could not be referenced. There were three possibilities to solve this problem. First was to use two functions placed in the Settings.php file – setRootPath and getRootPath. Using PHP's `exec( )` function it is possible to gain access to the terminal and execute the “pwd” (print working directory) command. Since Settings.php is placed in the base path it will print - /home/georgi/project which has a soft link to /var/www/project. Second was to use some of PHP's built-in functions such as `realpath( )`, `basename( )`, `dirname( )`, and `pathinfo( )` to extract the base path. Unfortunately their usage is targeted towards extracting data from a URI such as a domain name. Third was to use the “cd” command to change directory to the appropriate path. Common to all Unix systems is the single and double dot locations in each directory. The single dot is used to refer to the directory itself and double dot to the directory above in the root tree. Thus by using “../” the script can traverse back to Documents (for images), media (for CSS and icons), and others.

Another problem was constructing AJAX requests and appending the output. The original W3C manual or rather source code documentation was somewhat confusing. The problem was that the response text from the PHP script was not displayed. The solution came from [7] which holds practical examples of interaction between JavaScript, AJAX, and PHP. Furthermore the book contains references to AJAX ready states and HTTP status codes which are extremely useful not only for debugging but for user convenience also.

The fade-out algorithm was achieved without the need for JavaScript frameworks but the implementation was not straightforward. The main problem was decreasing the opacity for each division from 1 to 0 by 0.1. That meant that there were  $((10 + 1) \times 4) = 44 \times 4$  for each control = 176 lines of JavaScript code which was unacceptable. Initially the solution was a while loop which runs for 2.5 seconds, decrements the opacity, and increments the time interval. Unfortunately this solution was not functional and did not decrease the opacity slowly but immediately.



The workaround solution was to simply decrease the opacity to 0.5 units thus achieving the desired effect with minimal amount of lines of code.

There was another software bug associated with the opacity. When view products is selected and then add products, the view division should fade-out and disappear, while at the same time add division appear fully. This was problematic because even though the first division disappeared, the second division remained hidden. The solution to this problem was to reassure the visibility of the division with instructions of opacity that equal 1.

```
document.getElementById("viewProducts").style.opacity = 1;
```

Another problem was the automatic referencing of medical products based on their retail or Latin name. To achieve this the script was supposed to perform a Google search and retrieve the first result which is inserted into the database. Unfortunately Google does not support such operations through their API. The company would only index a site and provide for free a search widget (e.g. php.net) for it. On the other hand Microsoft's Bing did provide an API that does what was necessary. Unfortunately their service requires lengthy registration and impose traffic restrictions. An alternative to these two is DuckDuckGo which does provide free search services. Their API is capable of receiving simple GET method requests (e.g. [www.duckduckgo.com/?q=lsbu](http://www.duckduckgo.com/?q=lsbu)) and responding with easy to work with XML or JSON format. Unfortunately after testing the API it did not return results based on convoluted medical drugs' Latin names. The final solution was to simply check the response code from [www.drugs.com](http://www.drugs.com) by appending the name of the drug as such:

```
http://www.drugs.com/Levothyroxine.html
```

It is important to note that this is a potential software bug. If the Latin name of a product is mistaken or it is missing from their database, the Medical CMS will hyper link medical staff to non-existing web page. Surprisingly PHP 5.3.10 does not support try and catch exception block for `fopen( )` and `file_get_contents( )` which are responsible for opening and reading URLs. Thus it is impossible to open a 404 web page without breaking the script. It is also important to note that the simplest solution was to use JavaScript's DOM links array. It contains all links on a web site. For example the following will print out the first search result from Google:

```
document.link[9] => http://www.drugs.com/Levothyroxine.html
```

Another minor problem was the fact that after each SQL query, several lines of code are necessary to determine its success or failure. To produce more coherent source code, the previously mentioned `ExecutionSuccess` class was created which can be applied to the dozen SQL queries.

## 6. CMS Improvements for the User and Administrator

The following improvements and their necessity is explained in this section – messages between users, PDF reports, and accounting graphs. The first improvement of the system is the private messaging system. It allows registered users to send and receive text messages similar to electronic mail. The benefit is that there is no need for commercial email service – users can communicate between each other and prevent unwanted spam from external sources. The second improvement of the system is generating PDF file reports which are saved to the server's hard disk and offer the user to download the document to his or her personal computer. The third improvement of the system is generating colourful data graphs based on database records which can be printed as part of a medical report. The last two improvements heavily rely on third-party open-source software libraries which provide utilities unavailable from a standard PHP 5 setup.

### 6.1 Messages

The Messages interface offers the user the following functions – view, search, send, and remove messages from that database. The interface uses JavaScript and AJAX to switch between all four so that the user will not be redirected and loose focus. AJAX is used to first load the HTML form and then upon user triggered event, execute database operation. The structure of the database table is simple – id, sender, receiver, subject, date, file, reviewed, and message. The algorithm is also straightforward – the SQL query will select all entries in the database matching the currently logged user. Thus only relevant messages will be retrieved from the database and displayed in HTML tables. The two more interesting fields are file and reviewed. The file field contains a link to the local server hard disk where message attachments are stored. This way all medical users can exchange medical images and documents not related to the product database table. The reviewed field is highly useful because it visually instructs the user whether he / she has read the textual message. If that is not the case the user can click on a button and trigger an SQL update operation, which will change the reviewed field from false to true. Another fine touch to the interface is the possibility for the user to sort / filter the returned results. Rather than using HTML radio buttons or check boxes, the user can click on the HTML table header (e.g. sender, date) which will call the same SQL SELECT statement but with different ORDER BY argument which will immediately be displayed back to the user in the desired format. The figure below demonstrates the look and feel of this particular interface.

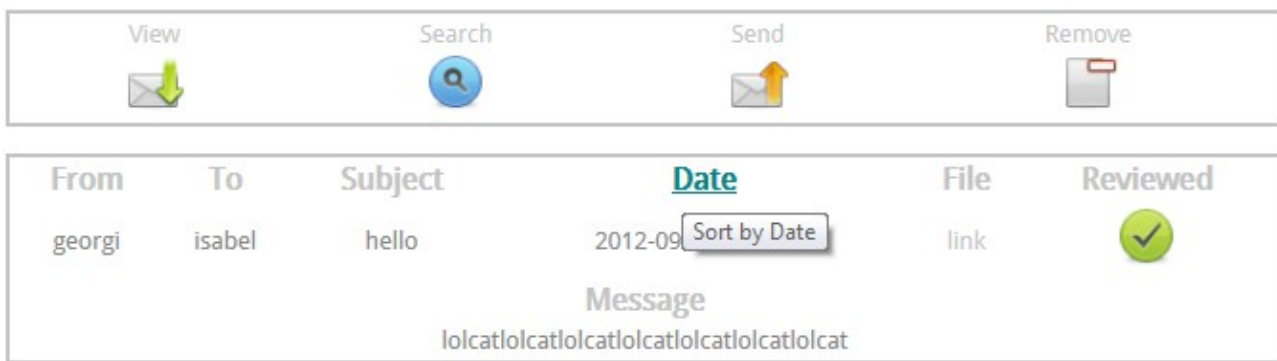


Figure 18: View Messages

The search function is very useful because a user may have dozens of messages to sort through to find a past message of interest. The search form simply takes a single keyword and PHP performs not one but multiple SQL statements matching the supplied keyword with all rows and fields (e.g. sender, receiver, message). The script will even regular expression on the message's contents if no result was found yet. Valid results or matches were checked using PHP's `mysql_affected_rows()` function which returns 0 only when no table rows were affected – no records. The regular expression is constructed using MySQL's LIKE syntax (i.e. `%keyword%`) which will match messages which contain the given keyword anywhere within the text. To visually notify the user, the matched field is highlighted in bright colour and larger font size. The figure below demonstrates the look and feel of this particular interface.

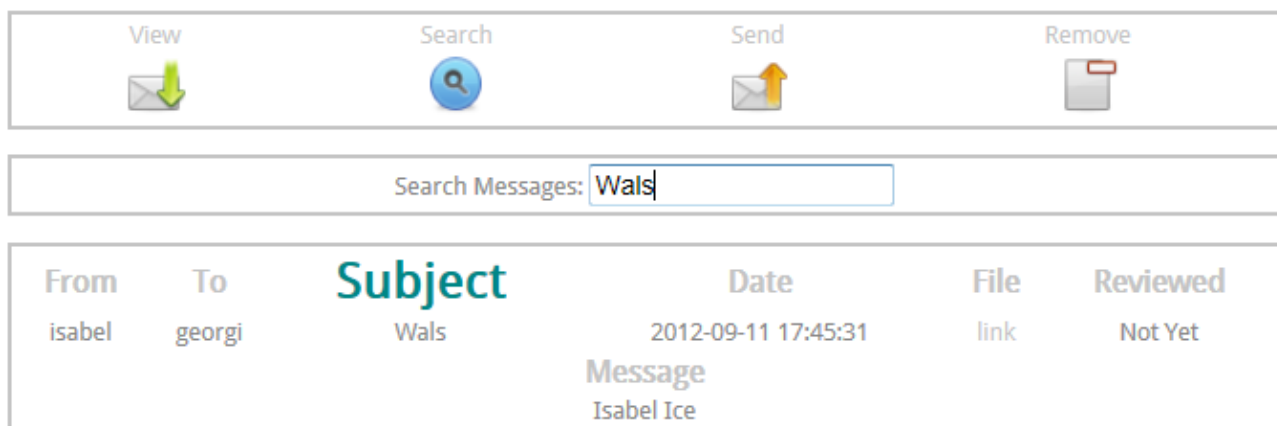


Figure 19: Search Messages

The send function displays an HTML form to the user with which he / she can compose a private message to another existing user in the database. If the user does not exist in the database the message will be rejected and the sender notified. Similarly to the add product script, users can upload files to the server which will be referenced as part of the private message. This script moves the temporary file on the server to the Medical CMS attachment directory. This directory can be easily examined by the administrator because the file name contains both the sender and receiver.

The delete function displays all relevant messages to the user in HTML tables format. On top of each is placed a delete button which performs an SQL statement when clicked. The SQL statement simply erases a row from the table which matches the product's id. To prevent misunderstanding the delete button is disabled (after successful operation) so that the user has a visual notification. The more interesting aspect of this function is how the product's id is retrieved. That is because the delete button cannot call the JavaScript function for removal by sending an id contained in one of the dozens of tables. The workaround was to assign each delete button's title property with the product's id. That way the correct id will be passed referencing the button element itself as follows.

```
echo "<input type=\"button\"
value=\"Delete\"
title=\"\" . $row['id'] . \"\"
onclick=\"removeMessage(this.title)\" />";
```

The figure below demonstrates the look and feel of this particular interface.

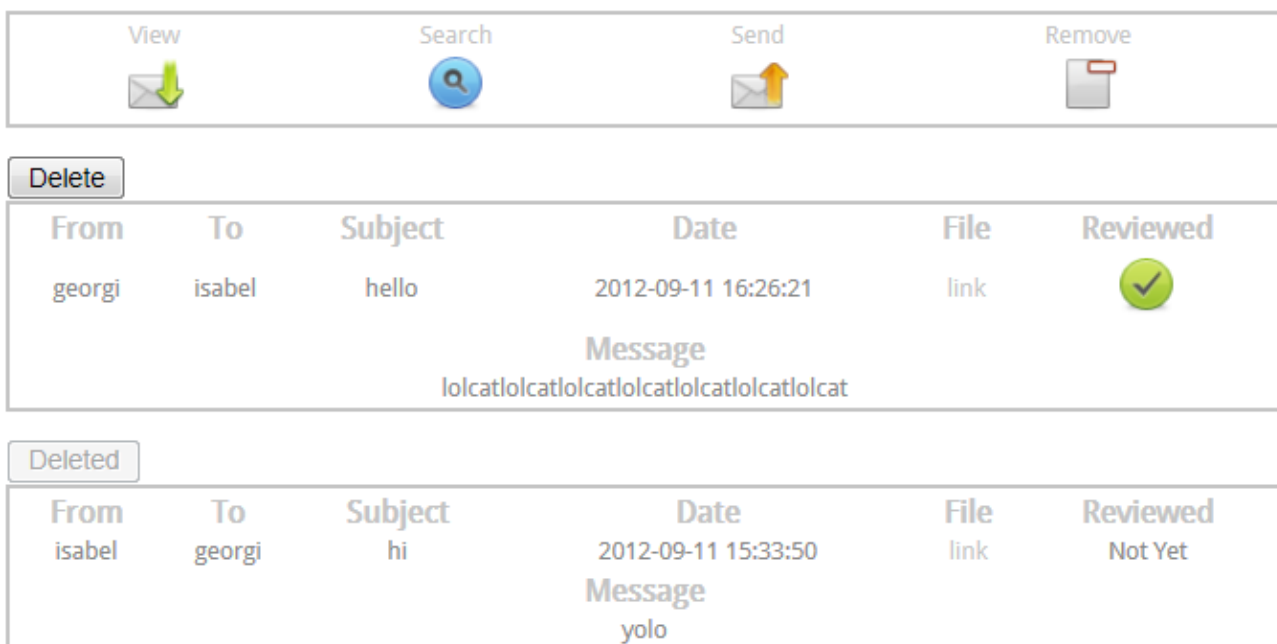


Figure 20: Remove Message

## 6.2 PDF Report

Authenticated users can generate PDF reports directly from the Reports interface. This is useful for medical users which may be required to produce some of this statistics by hand or requesting this from the administrator. The first option is quite tedious and also may not be possible because it may require direct access to MySQL's command line. The second option is also not optimal even though the administrator can construct complex SQL statements.

That is because the administrator will not be able to deal with many requests each with different requirements. Thus the user is presented with four options to generate documents containing up to date data from the database.

- MySQL Statistics
- Messages
- Medical Products
- Users

Upon request a PDF document will be generated by a PHP script containing data from the database and the user's personal comment, formatted into tables similar to Microsoft Excel spreadsheets. The report can be downloaded to the user's PC and stored on the server.

Since there is no native PHP class to create portable document format files, third-party software was necessary. The official PHP documentation [8] recommends the usage of PDFLib which provides full range of PDF creation functionality. However this library is not free of charge – commercial license is needed. The documentation also mentions a free alternative which is also developed by PDFLib GmbH conveniently named PDFLib Lite. Unfortunately there are two main drawbacks:

- The lite version can only create documents with text – no images, tables, web links, page numbers, and other essentials.
- Both versions are considered as PHP Extension Community Library which means they have to be installed and manually configured corresponding to php.ini and apache configuration.

Hopefully there was another alternative to PDFLib which is in fact free, open-source, and easy to install and configure. Rather than struggling with terminal commands it is simply a PHP class which can be extended. Thus all functions inside the FPDF class can be overwritten to produce the desired output (e.g. header, footer, page numbering, page break). Even though its features are not as good as those of a modern word processor, the generated reports are well formatted and clearly display the necessary information. It is worth mentioning that it was possible to implement full-featured PDF creation tool. The PHP script can create a text document using data fetched from the database formatted in LaTeX. Then using Ghostscript terminal commands, two documents will be generated – Postscript and PDF. The Medical CMS GeneratePDF class algorithm is straightforward. First the header and footer are initialised (e.g. title, author, default font). Second the table headers are created based on the type of report. Third the table is populated with data from the database based on the type of report.

Finally the report is stored with a unique file name – report-georgi-01-09-2012-16-59-59-pm.pdf. One of problems encountered during the implementation of the FPDF library was no output:

```
FPDF error: Some data has already been output, can't send PDF file
```

After reviewing a dozen of solutions and workarounds found in online forums, the problem remained persistent. My solution was to open the original FPDF class and modify `_checkoutput()` function. After commenting out the problematic if else control structure which did not apply to the project the problem vanished and all outputs were successfully formatted afterwards.

## 6.3 Accounting

The accounting web interface provides registered users the ability to see visual graphs based on otherwise inaccessible to them data from the database. These illustrated graphics can be easily included as part of a medical or book-keeping report by screen capturing the image on the screen. There are two types of graphs displayed on the web page – pie and column chart. In addition to the visual display the user can also refer to a table containing the combined data on which the graphs were based. It is important to note that there is no native PHP library which can provide such imaging on the user's side – the browser. This is due to the fact that PHP is a server scripting language unlike JavaScript which is scripting language for internet browsers such as Internet Explorer and Mozilla Firefox. That is why the displayed pie chart and column chart had to be generated with the aid of an external tools.

The **pie chart graph** contains information for the following – total users, total products, total messages, and total visitors to the Medical CMS. The algorithm for calculating these values is straightforward – select all rows from each of the four tables and use the value returned from the PHP `mysql_num_rows()` function. The four tables from the database are as follows – user, product, message, and log. The last table contains entry logs for all users that sign into the web site. The structure is as follows – id, ip (10.0.0.3), language (en-UK), browser (MSIE 9.0), os (Windows NT 6.1), osbits (64), username (georgi), and date.

In difference with other database constructs, rather than calling four different database class functions, one function contains four SQL statements and the results are returned as an associative array, which then is incorporated into an HTML table. The last step is important because the JavaScript graphical tool has to load data from single or multiple HTML elements.

The figures below demonstrates the graphical output – pie chart.

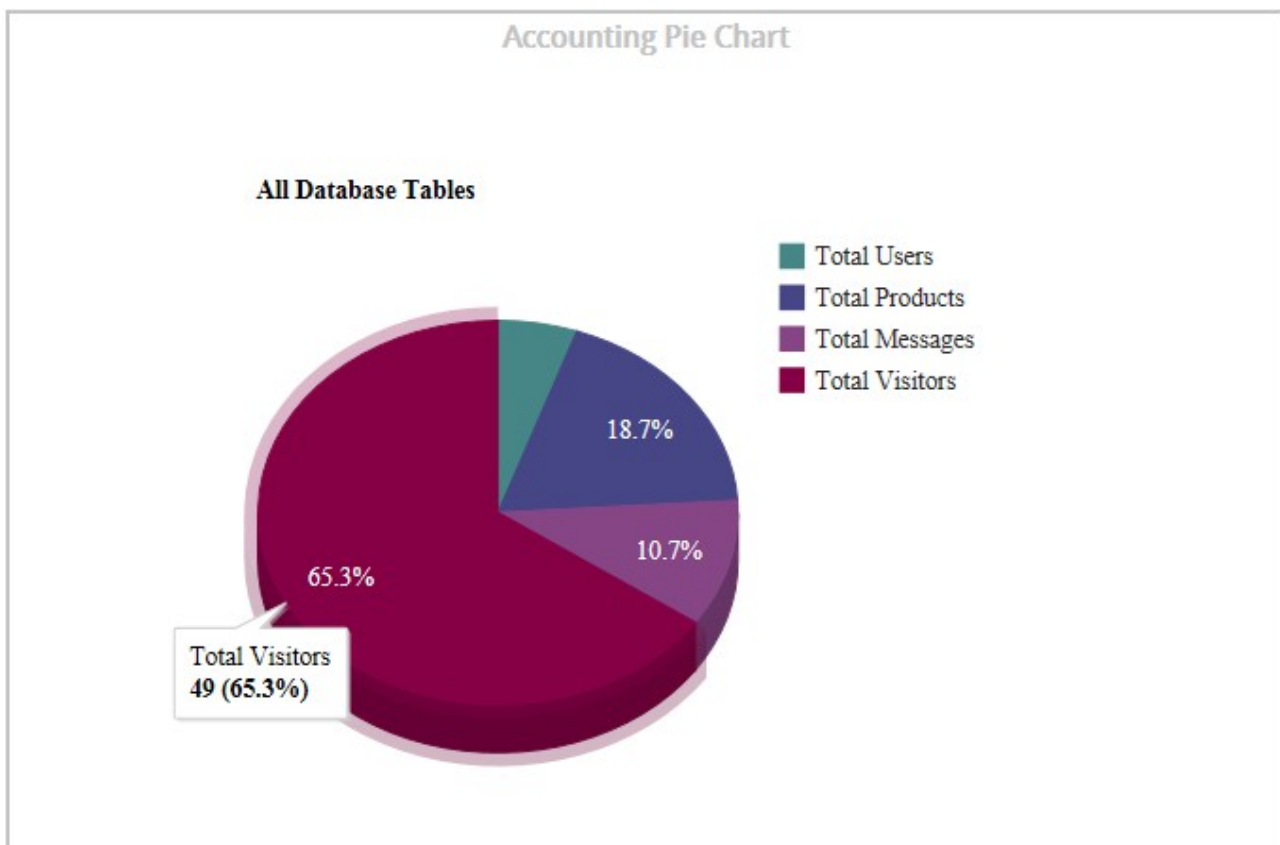


Figure 21: Accounting Pie Chart

The figure below demonstrates the second graphical output based on manufacturer and retail price calculations.

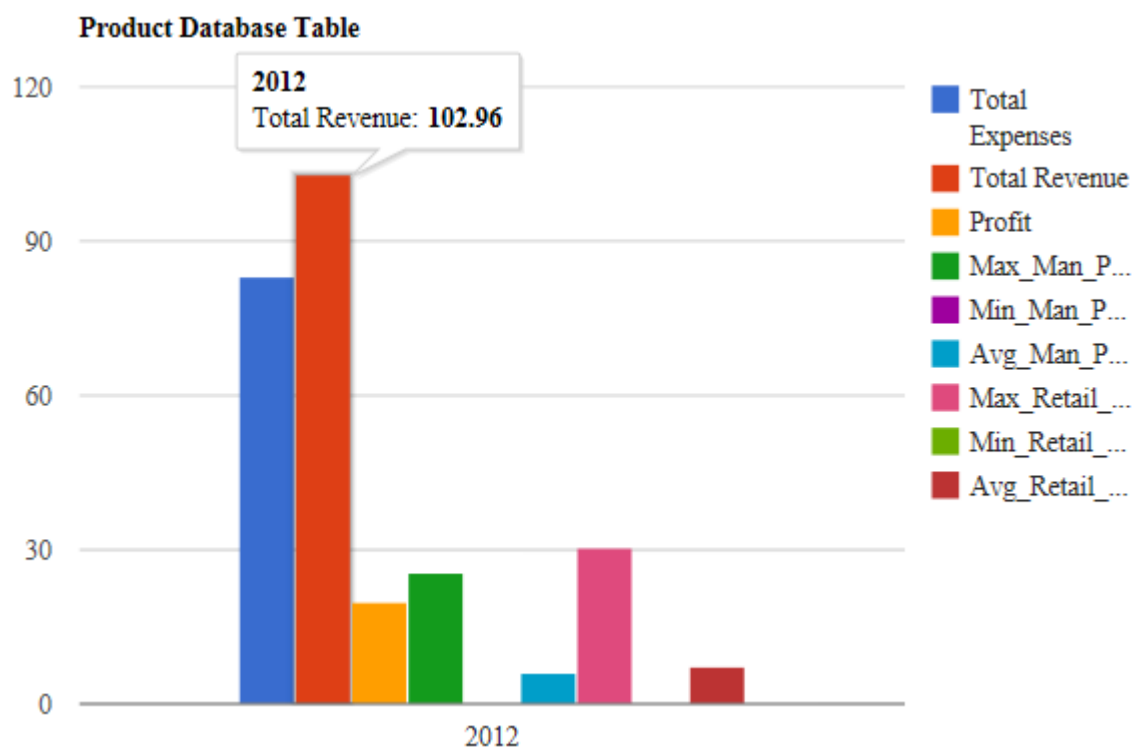


Figure 22: Accounting Column Chart

The **column chart graph** contains data for number of years in the database. Since this project was started in 2012 and the date field in the database is MySQL TIMESTAMP, only 2012 is displayed. Rather than performing calculations with PHP using simple arithmetic operators, MySQL group and aggregation functions were used as recommended by [10]. The following explains which was used for what purpose:

- Total expenses – SUM( ) of all products' manufacturer prices.
- Total revenue – SUM( ) of all products' retail prices.
- Maximum manufacturer price – MAX( ) of all products' manufacturer prices.
- Minimum manufacturer price – MIN( ) of all products' manufacturer prices.
- Average manufacturer price – AVG( ) of products' manufacturer prices.
- Maximum, minimum, and average for based on all products' retail prices.

The following code snippet demonstrates how total revenue is calculated. After query execution only one row is retrieved from the database since the result is contained in a single row resource. To prevent unnecessary precise digits (e.g. 3.14159), the result is rounded to the nearest number with 2 digits after the decimal point.

```
$query = "select sum(man_price) from product";  
$result = mysql_query($query);  
$row = mysql_fetch_row($result);  
$total_expenses = round($row[0], 2);
```

To produce graphics on a web page based on non-static data from a data source (e.g. database) the following may be used – GD, ImageMagick, JavaScript and CSS, JavaScript and HTML 5 Canvas API, Dojo, jQuery etc. The Graphics Library and its alternative ImageMagick are the most popular choices according to [11]. Unfortunately they are not always pre-installed on the server and are mostly used for their advanced features. The final decision was to use Google's charting tools for three reasons. First similar to Google online fonts tool, it is accessible without the need to download and install – the software is simply referenced in the script source address. Second the tools were specifically developed for charting data with pie, column, line, bar, area charts. Third the other technologies required to be downloaded (several MBs in size) and configured after which only a single feature would be utilised out of the rich Dojo or jQuery library.

The algorithm for creating a Google chart is as follows. First the external library resource is loaded by JavaScript. Second a JavaScript function is called which contains the data (either from AJAX call or in this case from HTML table elements).



Third the data is sent to the Google server as an associative array (chart credits and actual values). Forth the chart type and options (title, colours, font, size, 3D) are selected again using array construct. Finally the `chart.draw()` function is called which displays the graphical output in the designated HTML division.

One of the particularly interesting problems during the implementation of the Google graph was all grey charts. There was no debug information from JavaScript and Google. The problem was that SQL queries return string types even if the result is integer or in this case decimal float. The solution was to use PHP's `parseInt()` and `parseFloat()` functions.

## 7. Results and Discussion

This section of document discusses the results from the author's work on the project. Since the Medical CMS is not a scientific experiment, the discussed results are based on objective project evaluation. This was achieved by performing software tests, functionality tests, and usability feedback from ordinary (non technically savvy) users testing the CMS. The test procedure is also explained and results are presented in tabulated format. Also since this application could be deployed by an organisation, computer network security aspects are discussed – threats and measures against SQL injection, Cross Site Scripting, PHP 5.3.10, MySQL 5.5.24, Apache 2.2.22, and Ubuntu Server 12.04 LTS security flaws.

### 7.1 Software Tests

The following table contains software test description, result, and brief comment.

Test Description	Result	Comment
Index → Are all products displayed correctly?	Pass	PHP prints warning that the MySQL result was empty when the database table is empty. Also only ASCII encoded strings are displayed correctly, Cyrillic and Czech letters are obfuscated.
Login → Is it possible to brake the pin cod?.	No	The quick login feature accepts only four digits and correctly counts trial and error timing.
Login → Is it possible to proceed any further without password?	No	The script signs in users with both username and password. If a user types <code>ProcessLogin.php</code> or <code>FinaliseLogin.php</code> the script will not allow access to the system – login button remains disabled.
Register → Is it possible to brake the captcha code.	Maybe	It is possible to type either of the two words (image) and continue. Is it also possible to use the audio fall back (accessibility option) to pass the test.

Register → Is it possible to proceed any further?	No	The script always checks the user input for empty text fields. If a user types ProcessRegistration.php or FinaliseRegistration.php the script will not allow access to the system – block button.
Is it possible to access any restricted page without credential?	Maybe	The admin and user script always check whether the SESSION array contains information about logged user. However the session is not automatically destroyed. If the user did not log out, someone else can easily gain access to sensible pages.
Admin → Correct products display?	Pass	Some images may not be displayed. This is due to broken uploads, incorrect address, large image size (above the limit). Rows are displayed with empty entries, maybe an issue.
Admin → Is it possible to brake the add product script?	Yes	The script did not check whether the name is all numbers or the price is text. Also it is possible to upload non-image file for the product cover(brake the link) because the file type is not checked.
Admin → Is it possible to brake the update product script?	Yes	It was possible to update a product and not upload image cover. This did brake the link and the product image was not displayed. Also similar problem with alpha-numerical characters input.
Admin → Is it possible to brake the remove item script?	Maybe	Since it was possible to remove multiple products from the database, the user can unintentionally remove an item. There should be “are you sure about action” dialogue window or page.
Admin → Is it possible to brake view users script?	Maybe	The AJAX search call crashed the browser (unresponsive) only when the script had to search 100 plus users. Also the search looks only for exact matches – “georg” will not match “Georgi”.
Admin → Is it possible to brake add user script?	No	All submitted text field are checked. Also most fields are populated with automatically generated data (including the password). Also there are drop-down lists which prevent issues.
Admin → Is it possible to brake the update script?	Maybe	By default all users from the database are displayed. This could cause memory buffer overflow and cause browser unresponsiveness. The results should be paginated.
Admin → It is possible to mistake user for removal?	No	Rather than multi-tick selection of check boxes, this script allows only for single row erase of data. Because missing product is not such a big problem as a user not being able to use the CMS.
User → Manage products problems?	Maybe	The functionality is enhanced by several AJAX functions and animation. The problem is that sometimes the tabs get stuck and display wrong content, no content, and invisible content.
User → View messages problems?	Yes	The user message file attachments were inaccessible because of the directory hierarchy. The folder was placed in the base working path. If the attached file is deleted the link will remain in the database and point to non-existing file on the server.
User → Mark message as reviewed problems?	Maybe	When a message is marked as reviewed, it remains visible on the page. The user cannot mark it again or undo. It would be better if the division display property is set as non-visible.
User → Corrupt search, update or delete message?	No	The search feature performs regular expression match on all message table fields. All update text input and file upload are checked for errors and incorrect input. Delete message functionality has similar problems to mark as reviewed option.

## 7.2 CMS Interface Functionality and Usability

This section of the report discusses the functionality and usability of the content management system as seen and experienced from non technically inclined users. Their feedback is extremely valuable because medical personnel will have altogether different opinion on how the application works compared with the strictly technical view of the developer.

The test required the participation of five different people at different ages and occupations that are not similar in nature and not related to ICT. The collected feedback was analysed and the results shown below are in question and answer format. The most interesting responses were taken into consideration for this report.

Q:	Are you satisfied with the home (index) page?
A:	Most were satisfied with the simple and intuitive design. Some suggested that mouse hover over a product should display additional information about it. Also rather than simply listing categories of products, they should lead to a page which list all products associated with it.
Q:	Are you satisfied with the products page?
A:	The amount of detail is satisfactory but it would be nice if expensive products could be easily distinguished from cheap ones, similar to tag cloud. The highest priced products could have larger font size whereas the lowest price products smaller font size.
Q:	Do you find the login procedure intuitive and functional?
A:	Everyone part of the questionnaire answered that they really like the three-stage process of logging into the CMS. Especially the feature which tells them whether the username or password input is incorrect rather than simply stating error. Also the visual notifiers (icons) were something that the majority enjoyed.
Q:	Do you think that quick login is a useful feature, would you use it?
A:	Three out of five answered that they would not use it and even though they want fast access to the CMS, they prefer username and password authentication. The rest answered that they cannot be bothered remembering passwords for all the websites they visit. They agreed that they would prefer if the authentication was only by PIN code, except for the admin.
Q:	Do you find the registration procedure easy to use and functional?
A:	Four out of five answered that they found the registration process intuitive and straightforward. They did not experience any issues with it. They appreciated the visual notifications, well designed forms, and the ability to go back one stage. However all of them said that the captcha code was unnecessary for a small organisation and problematic to guess.
Q:	Do you think admin home page should display server statistics?
A:	The majority answered that they really like the interface. However 4 out of 5 said that the information was too much and the database log was sufficient. According to them only visitor IP address, web browser version, and database uptime should be displayed.
Q:	Do you find the admin manage product and users usage of JavaScript adequate / beneficial?
A:	All but one responded positively, saying that what looks like tabs is useful for daily operation.

Q:	Do you find the current search mechanism useful or you would rather use a search button?
A:	Three out of five replied that they appreciate this feature because it would save them time when searching for products. Even more so when using the filter result options. The rest stated that they are used to a search button but they would not dismiss the feature entirely.
Q:	Do you find the add, update, and remove product pages suitable for daily operation.
A:	Two of five responded that they find the interface highly usable. Especially the fact that all products are displayed when updating and multi-selection is possible when deleting. Three of five said that they would like to see review page. This would be a page which appears right after add, update, remove operations and tells the user what they are about to commit. Thus the user can review and continue or go back and edit.
Q:	Should the delete page be multi-select (products) or single-select (users)?
A:	All of the participants in the questionnaire were undecided. They said that they like both and one is not better than the other. However they would like to see something similar to a recycle bin. If a product or user is deleted from the database, it may be retrieved from the trash later.
Q:	What are your thoughts on the search and filter results mechanism for users in the database?
A:	Most agreed that the mechanism would be adequate in daily use especially with result sorting.
Q:	Do you think the add user page should skip input contents check when the admin is using it.
A:	All of them agreed with the exception made for the administrator. They said that it would be annoying if he/she had to spend a lot of time entering data.
Q:	How well structured and usable are the update and remove pages?
A:	All of them agreed that the remove page was very usable. However the update form according to them was all over the place and could be well improved.
Q:	Do you have personal comments about the admin interface?
A:	Two of the five said that they would like to see common database operations automated and included in the interface with a single button. For example account management (GRANT), utility (DESCRIBE, SHOW), compound (BEGIN/END/, IF), transaction (COMMIT) etc.
Q:	Do you appreciate the AJAX and animation enabled user pages?
A:	The majority of participants were positive and like the uncluttered interfaces. They expressed their desire for auto completion when adding new products to the database and especially when dealing with the product cover file, product knowledge base file, and barcode. They also shared that animation should be queue when a product is deleted so that it disappears.
Q:	Do you think that the report page is useful? Would you use in your line of work?
A:	Most responded that the contents of the reports are valuable and well formatted. However they would most probably use only small part of the document on the job. They found the messages report useless to them as well as MySQL Statistics report. Ideally a report could be composed of sections and data they select with check boxes or similar.
Q:	What do you think of the accounting page?
A:	All but one of the participants found the pie chart somewhat useful. Their opinion of the column chart was a lot more favourable and said it will be useful in their line of work. The table information at the bottom of the page would have been sufficient, since they could use Microsoft Excel. They would like to see which user adds/updates/deletes products and his/her work load for Human Resources statistics.

Q:	Overall satisfaction with the interfaces, functionality, and usability?
A:	Pavlina → 4/5, Jiří → 3/5, Ivan → 4/5, Steven → 3/5, and Jitka → 3/5.
Q:	What are the major drawbacks of the content management system?
A:	Majority of the participants stated that the web design could be much improved. Even though they appreciate the simplicity and table structures, they would prefer something modern in style similar to Facebook and Twitter. Also the administrator should be able to change the look, feel, and functionality of the CMS directly from the interface rather than from the script.
Q:	What are the major benefits of the content management system?
A:	Majority of the participants stated that the web application is very well structured, with sensible but conservative design. All pages and forms operate as expected and there was no need to think how to accomplish a task. Also even though small and compact, the application seems to be ready for a smaller organisation after several small changes reflecting specific needs of an organisation.

### 7.3 Security Threats

In order to evaluate the security threats which the Medical CMS is facing, all possible server software security flaws have to be examined. These include the operating system (Ubuntu Server 12.04 LTS) and all installed applications part of the LAMP software suit – Apache web server, MySQL database server, and PHP scripting language. All these can be taken care of using a simple system update which should upgrade the software and thus remove security exploits. The following two commands are necessary to fetch a list of updated software packages (.deb for Ubuntu and Debian) and then download / configure / update the current version – security patch.

```
georgi@server:~$ sudo apt-get update
georgi@server:~$ sudo apt-get upgrade
```

It is more important to test the content management system itself for possible security bugs. The testing cases were constructed and executed using two popular security exploitation software – Acunetix and Metasploit. The first can be installed on Microsoft Windows operating system. Then the administrator should create a new scan project by selecting the target host and type of exploits. After significant amount of time Acunetix will produce report on the weak points of the system. On the other hand Metasploit is used by professional system hackers. First a case has to be compiled by selecting a target host then select or scan for particular vulnerability (e.g. remote memory violation). Afterwards using the interactive Ruby / Metasploit shell, exploit properties are configured (e.g. port) and the source code is executed. These exploits are written by software engineers for what is known as zero day exploits – vulnerabilities which are not known and for which no security patch exists. They are written in the Ruby scripting language for fast deployment.

The following table represents the vulnerability test result data and their brief analysis is presented.

Vulnerability Description	Impact Level	Comment
Blind SQL Injection → AdminUpdateProduct.php	High	“Your script should filter meta characters from user input” [12] on the following text fields – barcode, id, man_price, name, quantity.
Cross Site Scripting → AdminUpdateProduct.php ProcessLogin.php ProcessRegistration.php	High	“Your script should filter meta characters from user input” [12] on the following text fields – (barcode, name, quantity), (password, username), and all text input fields in ProcessRegistration.php script file.
Application Error Message → AdminRemoveUser.php	Medium	The script produces error message to notify the user upon unsuccessful operation. It is possible for an attacker to use this to disclosed internal system data.
Directory Listing → /project/library /project/media	Medium	The server did not restrict access to these directories. The first contains sensible database information and scripts whereas the second – images and CSS file.
Error Message on Page → /project/library/fpdf Register.php	Medium	“This page contains an error/warning message that may disclose sensitive information.” [12] Error output caused by PHP warnings being enabled on the server.
HTML CSRF Protection	Medium	CSRF (Cross Site Request Forgery) protection maybe needed where HTML form elements are used.
User credential are sent in clear text via HTML forms.	Medium	It is possible for an attacker to intercept the data sent from the user to the server when using unencrypted HTTP rather than HTTPS communication channel.
Apache mod_negotiation discloses directory listing.	Low	It is possible for an attacker to send corrupted HTTP header to the server to respond with 406 – Not Acceptable. This will also disclose sensible server data.
File Upload	Low	If the file type and size are not verified by the script, an attacker can upload malicious code on the server.
Brute Force Guess Password → FinaliseLogin.php	Low	Equipped with a file containing numerous passwords, an attacker can guess the password and gain access to the CMS. The number of trial-error should be restricted.
Enable HTTP Options on the Apache 2 web server.	Very Low	It is possible for an attacker to acquire valuable information about the server – should be disabled.
Unprotected Sensitive Files	Very Low	The settings.php file and test.php are placed in the base working path of the project.
Autocomplete attribute set to ON for password input fields.	Very Low	In regard to login and registration, the password text fields will show previously typed entries – passwords. This could be dangerous if the attacker has physical access to the victim's personal computer.
Session cookie HTTPOnly Flag not set	Very Low	If the flag was set then the server will only accept cookie requests by the server, not by the client – possibly an attacker.

It is important to note that Acunetix was scanning and performing rigorous security tests on the Medical CMS and server setup for well over 8 hours. The results were surprising because most vulnerabilities listed in the table above were prevented by the system and server but there was damage done to the system. The unprotected user input text fields allowed for SQL injection and XSS security exploits listed below, in spite of the usage of `mysql_real_escape_string( )` as recommended by PHP's online manual.

- Dozen of newly created directories in the project's base working path. Directories with cryptic looking names, containing random bits of information.
- Dozen of newly created users in the database which were clearly artificial in nature – integer values for names and empty (null) values.
- Over 20 000 newly created products in the database which were clearly artificial in nature. These were the most shocking not only because of the volume but because loading them in the internet browser made it crash completely. Also the names and attributes were those of the `/etc/passwd` file – the script was trying to extract login credentials from the server.
- The mess caused by the script which could have been done by an attacker was significant.
- There was an attempt to extract passwords from the user table which was unsuccessful because all passwords are hashed using the SHA2 algorithm.

However it is also very important to note that the damage did not cause the CMS to halt its operation. All pages and scripts remained functional, no files or directories were deleted, no database table structure was altered, and both Apache web server and MySQL database server remained responsive if not a little sluggish during the Acunetix attack. Furthermore the following command was used to perform backup of the data stored in the MySQL server database.

```
georgi@server~$ mysqldump medical --add-drop-database -dump_date --password="toor" > medical.sql
```

Performing security penetration test using the Metasploit framework was different as listed below.

1. Scan for server vulnerabilities using nmap.
2. Search the online Metasploit database for the targeted vulnerability.
3. Start the framework from the Linux shell, load the exploit, and follow the instructions found in the online database.
4. Record the results from the penetration testing.

The results from the vulnerability scan are detailed in the table below. In total there were 22 vulnerabilities out of which only 4 are critical and require to be dealt with in timely manner.

Brief Description	Detailed Description
Default SSH password.	The password was root spelled backwards which means that hackers can gain complete access to the system using secure shell at port 22.
Apache Tomcat server example scripts information leakage.	Even though these are not part of the Medical CMS they are vulnerable to cross site scripting. These could easily be removed from the server.
CIFS NULL session is permitted.	Since the Samba server is turned on, an attacker can retrieve valuable information about the server. Also make anonymous RPC calls to the server. The server can be reconfigured.
IP source routing is enabled.	The attacker can exploit this to bypass the server's firewall. Usually this feature is disabled by default but configuration of the following file may be necessary – /etc/sysctl.conf.
SMB signing disabled	The authenticity of packets is not checked. The attacker can perform man in the middle attack.

The following chart was generated from MetaSploit data to visually demonstrates the security risks.

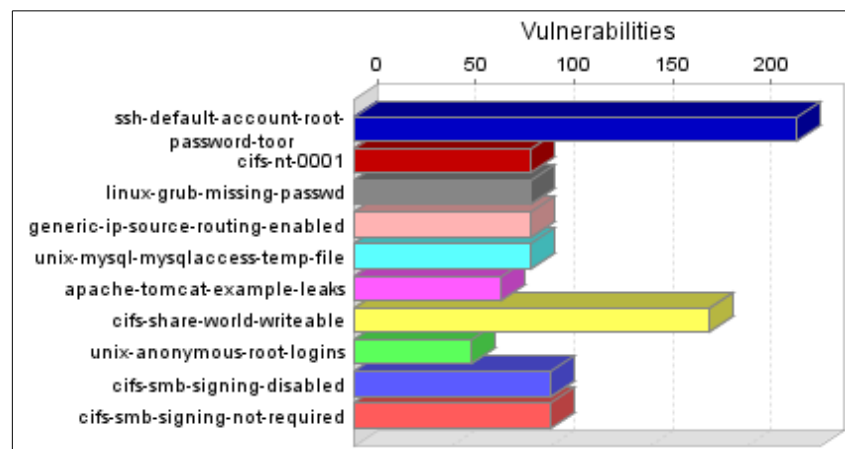


Figure 23: MetaSploit Report

Overall the server is very secure excluding the default configuration of applications such as Tomcat and SSH. Also the majority of security flaws were due to the Samba server/client software. It was enabled by the author to easily transfer and edit script files between the host operating system Windows 7 and guest OS Ubuntu Server 12.04. It is important to note that Samba is a software used on all UNIX operating systems to interact with Windows OS via the SMB / CIFS network protocol.



## 8. Conclusion and Recommendations

The aim of this project was to create medical content management system suitable for small organisations managing medical products. In order to develop this software application several goals had to be achieved. They were met by using the following methodology – feasibility study, significant and thorough research of the topic, time planning and management with deadlines, advanced understanding of the used programming languages, draft planning for the design of the application, testing of each module for software bugs, review of lectures from London South Bank University, and consultation with the supervisor and senior lecturers.

The second aim of this project was to create free and open-source CMS using similar tools. To achieve this the following were used – Ubuntu Server (Linux operating system), PHP (scripting language), MySQL (database server), JavaScript (browser scripting language), FPDF (PDF documents creation), Sublime Text (advanced text editor), and LibreOffice (advanced word processor). These are all alternatives to closed-source and expensive software tools from Microsoft such as – IIS, ASP.NET, MS SQL Server, VBScript, PDFLib, MS Visual Studio, and MS Office.

The success of the implementation of the technical research and technical approach was discussed on the basis of testing the functionality and usefulness of the CMS. Also the project was evaluated by independent participants in a test to determine the usability of the system. Their feedback was mostly positive and favourable with few exceptions – design and site features. The author considers this project to be a success due to the following:

- The CMS is operations at all levels (guest, user, and administrator).
- The administrator can manage products and users in the database with ease.
- The user can manage the products in the database using single interface (AJAX) as well as generate product PDF reports and statistics graphs.
- Both groups can send, receive, and manage private messages.
- The guest user can view site information and list products and users.
- The web application offers secure registration and login procedures.
- The complete content management system can be easily deployed because it is part of virtual server environment which contains a network operating system, web server, database server and plethora of tools for software development such as PHP, Perl, Python, and GCC.

Unfortunately not all of the planned features of the program were implemented. Some of the discussed components required additional hardware devices (i.e. barcode scanner), some required capital investment (i.e. purchasing medical products online), and others required too much working time (i.e. purchasing medical products by email request), and others were out of the project's scope.

The following list represents some of the recommendations that developers or students at London South Bank University from the MSc Computer Systems and Networks academic programme can work on individually or as part of their MSc dissertation.

1. Incorporate feature to purchase medical products from online stores using PayPal, bank transfer, VISA/Mastercard/American Express card payment, and email request.
2. All purchase data should be stored securely in the database. It can be extracted and presented to the administrator or accounting department part of the organisation.
3. Implement barcode scanning feature directly from the web interface during the add/update product process. This should be tested with actual hardware with Java preferably. The reason is that the JVM can run on almost all modern operating systems. The application should also support bulk scanning for the user's convenience.
4. Redesign of the main CSS theme for the content management system. It is recommended that the new design should be modern (e.g. Twitter, Facebook, and YouTube) and feedback from medical users should be gathered to match their expectations of great looking software since this is very subjective matter.
5. Implement support for UTF-8 encoding and decoding. The CMS provides minimal support for Unicode characters, mainly variations of the Latin alphabet such as umlaut and circumflex. It is recommended to include support for simplified and traditional Chinese as well as Arabic and Hebrew script.
6. It is recommended to implement recycle bin database table. Analysis of the user feedback determined that review of all database operations is necessary before they are committed, especially when removing product from the database. This could either be achieved using MySQL transaction features or by moving data to special database table. Thus when required from the user this data will be “recycled” and the changes to the database reverted.
7. It is recommended to expand the current visitor logging features. The CMS could acquire more information about each visitor and present it to the administrator in a well formatted data structure suitable for analysis and export to PDF report documents.

## 9. List of References

1. L. Ahn *et al.* *CAPTCHA: Using Hard AI Problems For Security*. CAPTCHA, 2008. [Online] Available from: [www.captcha.net/captcha\\_crypt.pdf](http://www.captcha.net/captcha_crypt.pdf) [Accessed 4 July 2012].
2. J. Katzmeyer. *Iphone PIN codes: 10 Most Common*. CIOinsight, 2011. [Online] Available from: <http://www.cioinsight.com/c/a/Mobile-and-Wireless/iPhone-PIN-Codes-10-Most-Common-402679/> [Accessed 15 July 2012].
3. M. Mar. *Understanding User-Agent Strings*. MSDN, 2012. [Online] Available from: [http://msdn.microsoft.com/en-us/library/ms537503\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms537503(v=VS.85).aspx) [Accessed 20 July 2012].
4. PHP. *File Upload Errors*. PHP Online Manual, 2012. [Online] Available from: <http://www.php.net/manual/en/features.file-upload.errors.php> [Accessed 2 August 2012].
5. U.S. Social Security Administration. *Beyond the Top 1000 Names*, 2011. [Online] Available from: <http://www.socialsecurity.gov/OACT/babynames/limits.html> [Accessed 15 August 2012].
6. A. Kesteren. *XMLHttpRequest Level 2*. W3C, 2012. [Online] Available from: <http://www.w3.org/TR/XMLHttpRequest/> [Accessed 5 September].
7. K. Hadlock. *Ajax for Web Application Developers*. Indianapolis: Sams Publishing, 2006.
8. PHP. *PDFLib Introduction*. PHP Online Manual, 2012. [Online] Available from: <http://www.php.net/manual/en/intro.pdf.php> [Accessed 10 September].
9. FPDF. *FPDF 1.7 Reference Manual*, 2012. [Online] Available from: <http://www.fpdf.org/> [Accessed 10 September].
10. Oracle. *GROUP BY (Aggregate) Functions*. MySQL 5.5 Manual, 2011. [Online] Available from: <http://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html> [Accessed 15 September].
11. L. Welling, L. Thompson. *PHP and MySQL Web Development*, 4<sup>th</sup> edition. New Jersey: Pearson Education, 2009.
12. P. Engebretson. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Waltham: Syngress Publishing, 2011.

## 10. Bibliography

1. L. Ullman. *PHP and MySQL for Dynamic Web Sites: Visual QuickPro Guide*, 2<sup>nd</sup> edition. Berkeley: Peachpit Press, 2003.
2. A. Hudson and P. Hudson. *Ubuntu Unleashed*, 2<sup>nd</sup> edition. Indianapolis: Sams Publishing, 2007.
3. R. Nixon. *Learning PHP, MySQL, and JavaScript: A Step-By-Step Guide to Creating Dynamic Websites*. Sebastopol: O'Reilly Media, 2009.

## 11. Project Planning Diagram

The following Gantt chart represents the breakdown of activities that were planned at the beginning of this project in order to successfully develop and finalise the application and report on time.

May		
Week 1 →	Week 2 →	Mile Stone
Approval of Project	Final consultation with supervisor before end of second semester.	5%

June		
Week 1 & 2 →	Week 3 & 4 →	Mile Stone
Literature review and topic research.	Report the findings and software aim, objectives, and technical background in the first 20 pages.	20 %

July				
Week 1 →	Week 2 →	Week 3 →	Week 4 →	Mile Stone
Install, configure, and learn how to use Ubuntu Server.	Install, configure, and learn about LAMP. Start PHP learning course.	Start MySQL learning course.	Start JavaScript and Dojo learning course.	30%

August				
Week 1 →	Week 2 →	Week 3 →	Week 4 →	Mile Stone
Admin – manage users script.	Admin – manage products script.	Login, register, and authentication script.	Debug Admin. Write 20 pages report. User – HTML mock-up.	60 %

September				
Week 1 →	Week 2 →	Week 3 →	Week 4 →	Mile Stone
User – manage products script.	User – manage messages, reports, and accounting.	Debug User. Write 20 pages report.	Perform software and visual test. Gather participants for questionnaire – feedback.	90 %

October				
Week 1 →	Week 2 →	Week 3 →	Week 4 →	Mile Stone
Write 10 pages for conclusion, references etc.	Complete review of source code.	Complete review of report.	Print, bind, and attach DVD. Submit the envelope.	100 %

## 12. Appendix A

The attached DVD-R medium contains all deliverables listed in the beginning of this report.