Integrated Inventory Management & Online Customer Order Processing System
For
Adonai African & Caribbean Shop
By

Livingston C. Turker
Computing & Management
Session 2005 / 2006

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material, which is obtained from another source may be considered as plagiarism.

(Signature of Student).....

Summary

The aim of the project was to design and implement an Inventory Management and Online Customer Order Processing Systems for Adonai African & Caribbean Shop. The shop specialises in selling African and Caribbean products, from groceries items to cosmetics. The shop also offers various services, which includes renting out African films and Money transfers. Adonai Shop currently use basic Ms Applications to manage inventory records and all business related processes. Due to the constantly changing business environment, which has led to an increase in the number of credit customers and product range available, Adonai therefore recorgnised the need for a system that would enable them in the processing of customer orders and managing the inventory records and also assist in the provision of other business services such as the renting of the DVD films and thereby improving the efficiency of business processes.

The project required the developer to adhere to software project management techniques in order to successfully obtain the company's requirements, before designing, implementing, testing and evaluating the system.

Acknowledgements

I would firstly like to thank my project supervisor, Dr. Kristina Vuskovic for her assistance during the project and my project assessor Martyn Clark for his valuable feedback during the progress meeting.

Secondly, I would like to thank Edison Bundy and Helen Robinson for the time and effort they have put into proof reading my reports.

Finally, I would like to acknowledge Lennox Cummings and Monica Z. Wang for their kind words when times were hard.

Thank you.

Table of Contents

1.0. Introduction

This chapter details a general overview of Adonai African and Caribbean shop business strategy. It outlines the aims and objectives for the project, specifies the minimum requirement, possible extensions discussed and it identifies the deliverables. In addition, the chapter gives a brief overview of the project significance to the developer's degree program and a brief information on the layout of this report.

1.1 Overview of Adonai Shop

Adonai shop specialises in selling African and Caribbean groceries and household products and also offers a number of services, which includes money transfer abroad and the renting of DVDs and VCRs of African films. The shop opened in 2001 in Leeds. The vast majority of Adonai products are export goods, although some of the products are purchased from other wholesalers in the UK mainland such as London and Manchester. The shop is currently managed by the staff, the manager and the till cashier. With the implementation of new e-commerce technologies, the manager hopes to expand the business and employ more staff.

1.2. Problem Definition

Adonai shop is a small business striving to compete in a segmented market share. Sales revenues have gone up since the shop opened in Leeds.

There are a number of regular customers in and around cities near Leeds that purchase items from the store on credit and their details are written on paper. Customers can place orders via the phone and the information is recorded on paper and then the store delivers to the customer's home. Sometimes paperwork goes missing and the customers don't receive their orders and this causes loss of business and trust from customers as they are often frustrated with the quality of service.

The manager uses spreadsheets and physical stock check to manage inventory. This method has been deemed inefficient, as it is not often apparent when stock levels are running low.

Due to the nature of the customers been served by Adonai shop, customers are currently not able to purchase items from home or browse the internet to see what is available, before driving to the stop only to find that the item they required has been sold out or has to be ordered from a supplier abroad.

In addition to selling food and house-hold items, the store also rent out DVDs and VCRs movies and several of these DVDs and VCRs go missing without knowing who borrowed them last. Inventory is currently managed by spreadsheets and other basic software applications, which is not very efficient as most items go unaccounted for.

1.3. Project Aim

The aim of this project is to develop an integrated inventory management and online customer order processing information technology solutions to manage inventory and daily services and transactions at Adonai African & Caribbean shop.

1.4. The Objectives of the Project

The overall objectives are to develop a thorough understanding of Adonai's business process and analyse the current business strategy whilst developing methods for improvements by using appropriate information systems methodology to design and implement the new system.

1.5. Minimum Requirements

Develop a prototype web-based ordering system.
The web based ordering system should enable customers to browse stock availability.
Develop a prototype inventory management database.
The database should be able to store and retrieve stock information.
The database should be able to store and retrieve customer details.
The database should be able to store and retrieve DVDs details.

1.6. Possible Extensions

The database shall be able to list stocks that are running low.
The web based ordering system shall allow for the creation of customers account.
The web based ordering system shall be able to take and process customer orders.
The web based ordering system shall be able to generate receipts with order reference.
The Inventory Management System shall integrate the functionality to loan DVDs to customers.
The inventory management system shall implement login password and username for security.
A User Manual.
The Inventory Management System shall implement functionality to list debtors and their details.

1.7. Deliverables

A database system that stores customer, items, DVDs, and order details.
A dynamic website that allows customers to browse stocks and place orders
The full project report.

1.8. Project Relevance to Degree Program

The project is relevant to my Degree Program as it will enable me to put together and utilise a number of skills I have acquired throughout my degree course and also to develop new skills and insight into the development of e-business solution to support a small business.
The project covers skills acquired from a wide range of the school of computing taught modules, which includes, Information Systems (IN11) and Software Project Management (SE22), which enables the application of appropriate methodologies, usability and time management consideration. Internet Systems Technologies (SY22), Advanced Databases (DB31and DB21) will help in the development of the database design and the web-based solution respectively.

1.9. Project Schedule

The initial Project Schedule can be found in Appendix (B) and may be subject to change during the course of the project due to a number of uncertainties that might be encountered.

1.10. Progress Report

I have met the milestones numbers 1-6 as indicated in the Project Schedule, which can be found in appendix (B).

I have critically examined and understood the problem situation, and as a result, I have performed a preliminary research into what technologies, methodologies and tools that will be required to provide solutions to the problem.

I am currently in the process of conducting various feasibility studies and analysis of the proposed solution.

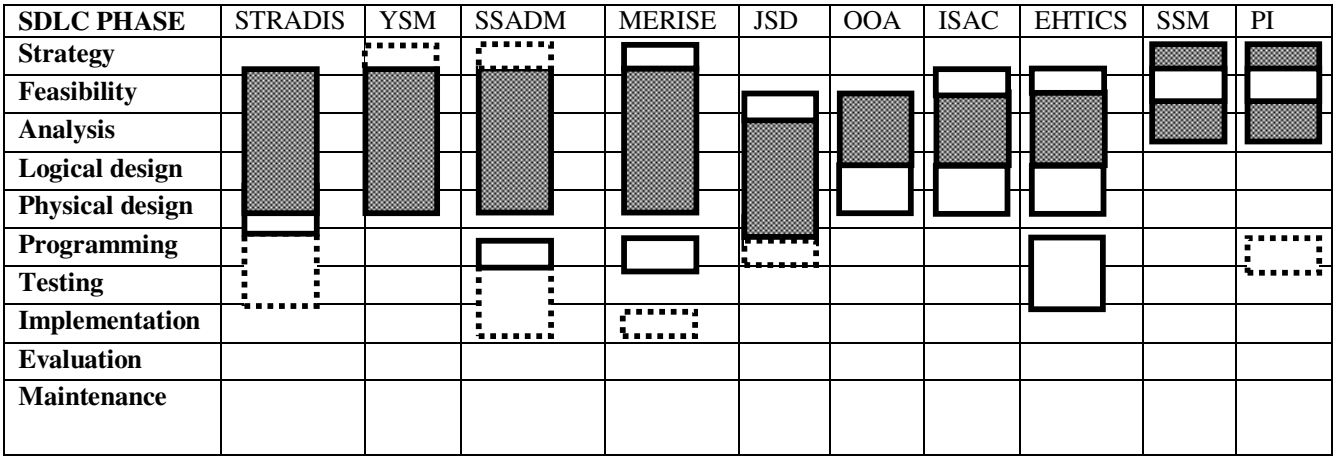
2.0. Background Research

This chapter detailed a selection of different methodologies, technologies and indicates their level of importance in a given information system development. It also gives detailed justification of the chosen methodology that the project will adhere to. This chapter also looks at usability and security concerns and their importance to the overall proposed system.

2.1. Methodology/Process Models and Technology Selection

An Information Systems Methodology is defined as a collection of procedures, techniques, tools and documentation aids which will help the system developers in their efforts to implement a new Information System (Avison and Fitzgerald [1]).
There are a number of methodologies to be chosen from when developing an information system. Selecting which one to be used depends entirely on the nature of the project being considered. If the conventional Software Development Life Cycle (SDLC) is to be used, therefore it is important that the chosen methodology covers all or the vast majority of the individual phases in the SDLC. Figure 1. depict various methodologies and their scope in terms of what aspects or phases of the SDLC they covered.

METHODOLOGIES



LEGEND

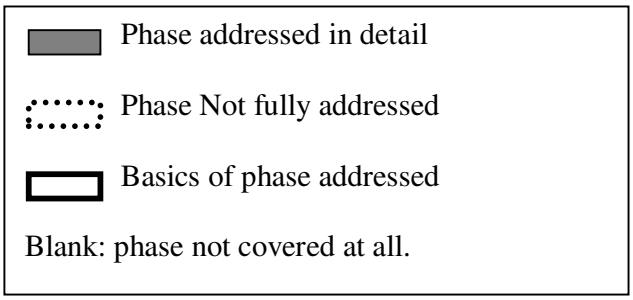


Figure 1.0

Source: Avison and Fitzgerald [1]

As shown in fig 1.0 above, the shaded areas indicate that the methodology covers the stage in some details, which may include the provision of specific techniques and tools of support. An unshaded area means that the methodology addresses that area,

but in less detail and depth and the broken lines indicate areas that are only briefly mentioned in the methodology.

2.2 Structured System Analysis and Design Methodology (SSADM)

SSADM is a methodology developed for and by the British Government for use in the Public Sector. This methodology is largely developed to cater for the implementation of database system, although it can be applied to other non-database systems (Avison and Fitzgerald [1]). As the vast majority of this project involves a database inventory management system development, it would be appropriate to consider or use this methodology along side others, in order to achieve the requirements of the client and the project.

2.3 System Development Life Cycle

According to Chrisanthi Avgeron and Tony Cornford [2], the System Development Life Cycle (SDLC) is one of the more traditional approaches to software development, as the model has been used extensively for more than 30 years. (SDLC) is a model for developing a system based on traditional problem solving with sequential steps and options for revisiting steps when problems appear.

However, the System development Life Cycle does has its drawbacks. It has been criticised as an inappropriate or over-rigid guide to information system development due to its linear pattern of tasks, which may result in inflexibility in software project management.(TUDOR & TUDOR [3]). Hughes and Cottrell [4] also explains that if a particular phase needs to be refined, the traditional SDLC methodology is insufficient. Although, Avgeron and Cornford [2] argued that the original linear form of SDLC has been modified in various ways to overcome its most dysfunctional aspects, such as the ability to effectively manage the long and complex development process. Also the life cycle provides a simple structure, which is easily understood and by following the life cycle it becomes easier to plan and manage that tasks of systems development.

The SDLC Methodology can be useful when the production of deliverables are an end result, as it involves extensive requirements analysis is needed whilst using SDLC, the methodology slows down the project, as producing a solution could take longer than expected. Since the client (Adonai shop) has a clear idea of what the solution should perform, which is to manage inventory and enable customer to process orders online, therefore the requirements have been well defined and understood.

2.3.1 The Waterfall Model

The Waterfall Model takes the main stages within systems development and represents them diagrammatically as a series of sequential steps with the flow of time and information from left to right Tudor & Tudor [3]. The major benefit of this model is that it allows project completion time to be forecast with more confidence than with some more iterative approaches allowing projects to be controlled effectively Hughes and Cottrell [4]. This feature would be useful in this project development as a later stage within the life cycle phases may reveal the need for some extra work at an earlier stage.

2.3.2 The ‘V’ Model

The ‘V’ Model is also a diagrammatic representation of the life cycle, however, its additional strengths over the Waterfall Model are that the products, which result from each stage, are passed to the following stages and the products, against which various levels of testing of the system can be performed Tudor & Tudor [3]. Due to the nature

of extensive testing involved, this model would not be appropriate for this project due to the limited amount of time allocated.

2.3.3 The Spiral Model

The Spiral Model according to Hughes and Cottrell [4], can be portrayed as a loop or a spiral where the system to be implemented is considered in more detail in each sweep. Each sweep terminates with an evaluation before the next iteration is embarked upon. A greater level of detail is considered at each stage of the project and a greater degree of confidence about the probability of success for the project should be justified, otherwise a decision to abandon the project would be reached. Avison and Fitzgerald [1] criticised this model as only the most important requirements are defined and implemented first. Following feedback from the client more sophisticated functionality is implemented. The major problem with this approach is that if the client has unintentionally missed an important requirement late into the development process, such changes could prove very costly. This can occur easily if the client, developer or even both parties misunderstand the requirements specification.
More-over, this methodology is said to be more useful with large-scale projects. Therefore it would not be appropriate for this project.

2.4 Methodology Choice

I have decided to use the SDLC methodology while incorporating an iterative approach borrowed from the Waterfall model in the implementation and testing phases. As I have earlier mentioned the client (Adonis shop) and the developer has a clear idea of what is required of the system, therefore revisiting analysis and design phases of the project extensively could result in bottleneck and valuable project time could be consumed unnecessarily. To ensure that the implementation provides a functional solution for the client, iteration will be carried out at 'Testing and Implementation' phases respectively. See diagram below:

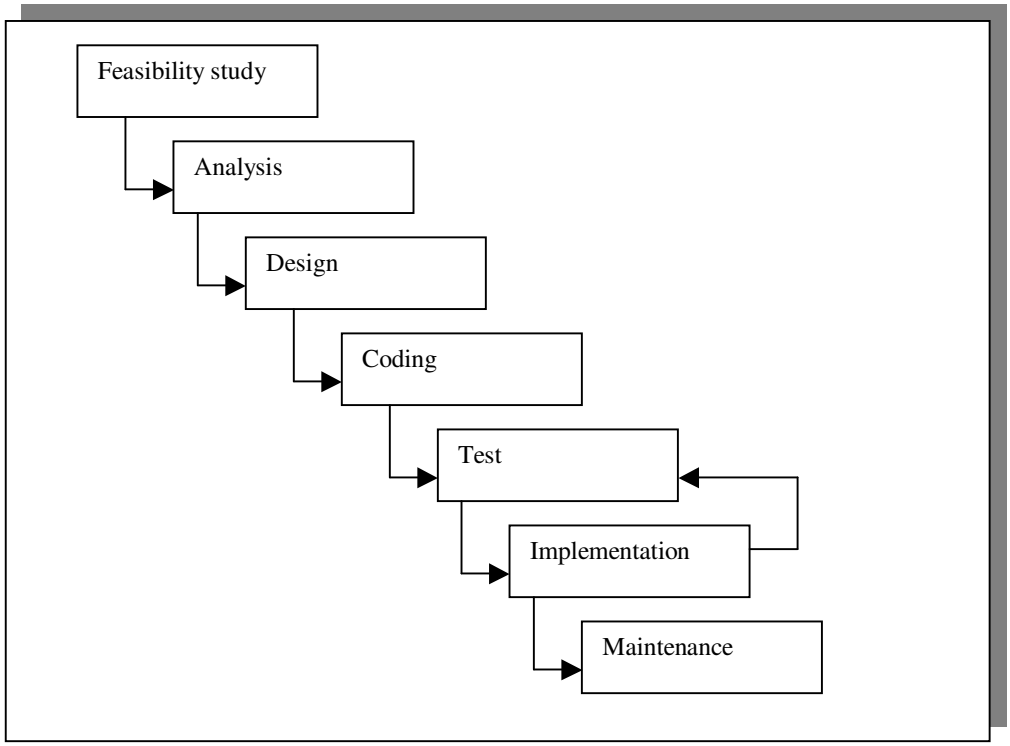


Fig 2.0: system development life cycle and the water fall model

2.5 Selecting appropriate Database System

According to C.J. DATE [5], a Database System is basically a computerised record keeping system; i.e., it is a computerised system whose overall purpose is to store information and to allow users to retrieve and update that information on demand. The information in question can be anything that is of significance to the individual or organisation concerned- anything in other words that is needed to assist in the general process of running the business of the individual or organization.

Various database applications exist, such as MS SQL Server, MySQL, MS Access and PostgreSQL, but it is important to carefully select the appropriate application to use in any given situation as some lack sophisticated functionality to achieve a particular task, such as web-site connectivity.

2.5.1 MS SQL Server

SQL is an example of a Transform-Oriented Language designed to use relations to transform inputs into required outputs. As a language, SQL has two major components: A Data Definition Language (DDL) and a Data Manipulation Language (DML) for retrieving and updating data (Connolyn &Begg [7]).

MS SQL Server is a product of Microsoft Corporation. It is very efficient in optimising features such as Trigger Managements. Microsoft claimed that it is the next-generation Data Management and Analysis Software, that delivers increased scalability, availability, and security to enterprise data and analytical applications while making them easier to create, deploy, and manage (Microsoft.com [8]). Inevitably, all these features come at a price and also another down side to MS SQL is that it is not platform independence. Therefore it would not be the appropriate choice for this project and the client.

2.5.2 PostgreSQL

Looking at fig 2.1 below, the obvious choice between PostgreSQL and MySQL would be the former. PostgreSQL seems to overcome all the shortfalls of MySQL. For example, PostgreSQL supports features to create for all trigger management, as indicated in the diagram. Although there is a downside to using PostgreSQL as it is very slow within a web-based development environment. Considering the fact that customer would like to browse their details and stock information online; the use of a slowly responding database would not be appropriate in this circumstances. More over, in any given e-commerce website, response time is considered to be of high importance as the impatient customers or potential customers gets frustrated very quickly. Therefore as this project involves web applications, and response time is one of the requirements, PostgreSQL would not be appropriate

2.5.3 MySQL

MySQL is an open source Relational Database Management system (RDBMS) that uses Structured Query Language (SQL), the most popular language for adding, accessing, and processing data in a database. MySQL is noted mainly for its speed, reliability, and flexibility. Most agree, however, that it works best when managing content and not executing transactions (shop-script.com [9]).

The rapid increase in connectivity between MySQL and PHP (Hypertext PreProcessor), will have to be taken into consideration, when also determining Server-Side Scripting. Some disadvantages of such an implementation include the fact that MySQL is not able to implement triggers and be efficiently supportive, where the database can have many front-ends (shop-script.com [9]).

The benefits of using MySQL outweigh the cost and disadvantages. From the diagram below, fig 2.1, the only major shortfall of MySQL is that it doesn't implement full use of triggers. Apart from this, MySQL is robust and reliable cost effective way for small business –like Adonai shop – to adopt. Scalability means as the client's customer base increases in the near future, the database system will still be fully functional to cope with an ever changing business environment. Platform independent would mean that the client wouldn't be tied up with various software licensing contracts from vendors as they can switch at any time.
Therefore it would be appropriate to use MySQL for the sake of the client's requirements and this project.

Comparison between MySQL and PostgreSQL

		MySQL	PostgreSQL
General			
Database Connections		Multiple	Multiple
Concurrent Access to Multiple Databases		√	√
Multi-version Concurrency Control		√	√
Unicode Support		√	√
Replication Support		√	√
License		GPL	BSD
Specifications			
SQL 99		**X**	√
ODBC		√	√
Relational Database Features			
Sequences/Auto-increment Column		√	√
User Defined Functions		√	√
Update-capable Views		√	√
Referential Integrity		√	√
Triggers	Statement / Row Level	√	√
	Before / After	√	√
	Nesting	**X**	√
	Compound	√	√
Domains		**X**	√
BLOB		√	√
CLOB		√	√
Name Length Limit		64	64
Delimited Identifiers		√	√
Stored Procedures		√	√
Procedural Languages			
PL/SQL (or equivalent)		√	√
Java		**X**	*
Python		**X**	√
PHP		√	√

Key to Symbols	
√	Feature supported
X	Feature not supported
*	External or unofficial support

Fig. 2.1. source: http://www.devx.com

2.6 Selecting a Server Side Scripting Language

Wikipedia.com online encyclopedia [10] defined a Server-side scripting as a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. It is usually used to provide interactive web-sites that interface to databases or other data stores. This is different from client-side scripting where scripts are run by the viewing web browser, usually in Java-script. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

2.6.1 Active Server Pages

Active Server Pages (ASP) is Microsoft's server-side scripting technology for dynamically-generated web pages that is marketed as an add-on to Internet Information Services (IIS).

Programming ASP websites is made easier by various built-in objects. Each object corresponds to a group of frequently-used functionality useful for creating dynamic web pages such built-in objects are: Application, ASPError, Request, Response, Server and Session. Session, for example, is a cookie-based session object that maintains variables from page to page Wikipedia.com [10]. However, it does have its drawbacks, Microsoft co-orporation has purposely developed Active Server Pages for its (IIS) internet information server and therefor this made it platform dependent, i.e MS Windows Operating systems only.

ASP would not be suitable for this project due to its limitation for sole use with IIS and the cost of purchasing it.

2.6.2 Java Server Pages (JSP)

Java Server Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.

In Java 2 Platform, Enterprise Edition (J2EE) v1.4, JSP technology has simplified the page and extension development models with the introduction of a simple expression language, tag files, and a simpler tag extension API, among other features. This makes it easier to build pages based on JSP technology (java.sun.com [12]). In comparison with PHP, JSP, there are similarities in both scripting languages. However, JSP offers more advanced features and capabilities, such as the Standard Tag Library (JSTL). The main benefit of this library is that it has the ability for a simple formatting of data within the JSP page without altering the database in the server. Although PHP does offer a more flexible choice and on this occasion, Java server pages will not be chosen at the moment, however it may be considered during the course of the project, should it be necessary to do so.

2.6.3 Hypertext Pre-Processor (PHP)

The PHP (Hypertext Pre-processor) is a scripting language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications Wikipedia.com [10].

PHP can be used on all major operating systems, including Linux and many Unix. PHP has also support for most of the web servers today. This includes Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, and many others. For the majority of the servers PHP has a module, for the others supporting the CGI standard, PHP can work as a CGI processor.
So with PHP, there is the freedom of choosing an operating system and a web server. Furthermore, there is a choice of using procedural programming or object oriented programming, or a mixture of them. Although not every standard OOP feature is implemented in PHP 4, many code libraries and large applications (including the PEAR library) are written, using only Object Oriented Programming (OOP) code. PHP 5 fixes the OOP related weaknesses of PHP 4, and introduces a complete object model.
With PHP there is no limitation to output HTML. PHP's abilities, includes outputting images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. You can also output easily any text, such as XHTML and any other XML file. PHP can auto-generate these files, and save them in the file system, instead of printing it out, forming a server-side cache for a dynamic content (Php.net [11]).
One of the strongest and most significant features in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. And most importantly for the purpose of this project and the requirement of the client, PHP supports MySQL, Therefore, it would be appropriate to use PHP as the scripting language as the choice of database as earlier mentioned is a MySQL.

2.7 Usability Issues

Usability is defined as the degree to which people (users) can perform a set of required tasks. It is the product of several, sometimes conflicting, design goal (Brink, Gergle & Wood [13]).
Functionally correct; the primary criterion for usability is that the Customer ordering system and Inventory management system correctly performs the functions that the users need, such as purchase items online. If the system does not allow customers to do this, it would be rendered useless.
Efficient to use; efficiency can be measured by the time or actions required to perform a task, for example transaction time with regards to customer details retrieval from the database via the web interface should be reasonable. This is one of the reasons it has been decided that MySQL would be the choice of database to be used as procedures that are faster tend to be more efficient.
Easy to learn: ease of learning determines how quickly new users can learn to accurately perform a task procedure, for example the steps taken for a particular customer to open a new customer ordering account online should be kept to an absolute minimum as research evidence suggests that the fewer steps a procedure contains, the easier it is to learn.
Error tolerant: Error tolerance is determined by how well errors are prevented, how easily they are detected and identified when they occur, and how easily they are corrected once they are identified (Brink, Gergle & Wood [13]). The stock control and customer order processing system would be automated and minimise the number of manual entries when customers are placing their orders in order to reduce input errors, extensive input validations will be performed within the browser at run-time without even accessing the database in the server.

2.8 Security Issues

2.8.1 Secure Sockets Layer (SSL)

Secure Sockets Layer is a system for encrypting data sent over the Internet, including e-commerce transactions and passwords. With SSL, client and server computers exchange Public Keys (PK), allowing them to encode and decode their communication.

Due to the nature of this project, security will depend entirely on the web server hosting company. However, it is important that the appropriate hosting company, that has a server, which supports SSL or equivalent security application is used –(More below).

2.9 Hosting/Web Server Selection

Following research into various web hosting companies and package deals available, it has been decided that StreamlineNet is the preferred choice. The reason for this is that StreamlineNet web server offers unlimited gigabyte web space and bandwidth, its server supports PHP and MySQL databes, which are the choices of scripting language and database for this project. The cost is reasonably low and having spoken to the client, he has agreed to proceed with StreamlineNet. Finally and most importantly, StreamlineNet Web Server supports Secure Sockets Layer for secure connectivity and communications as specified above.

2.10 Project Amendments
Following the return of the Mid-Project Report, further research was conducted. The Gant Chart showing the project schedule was slightly readjusted in order to be in line with the actual progress of work. The revised Gant Chart is depicted in appendix (C).

3.0 Analysis
This Chapter outlines the issues that affect Adonai's Business Process, as well as the analysis of the current system. The implementation of an Information Systems/Information Technology Strategy will be discussed, the resulting information from the IS/IT strategy will enable the identification of the requirements for the proposed system, which will help the design stage incorporate these requirements successfully.

Analysing sample documents, observations at the shop and the interview with the user helped in understanding how and what is recorded in the shop, in order to model the proposed system to closely match the current business process and help implement the IS/IT strategy. The Chapter concludes with an investigation into existing similar systems to see whether they could solve the problems and/or meets the user requirements as well as conform to the IS/IT strategy of the business.

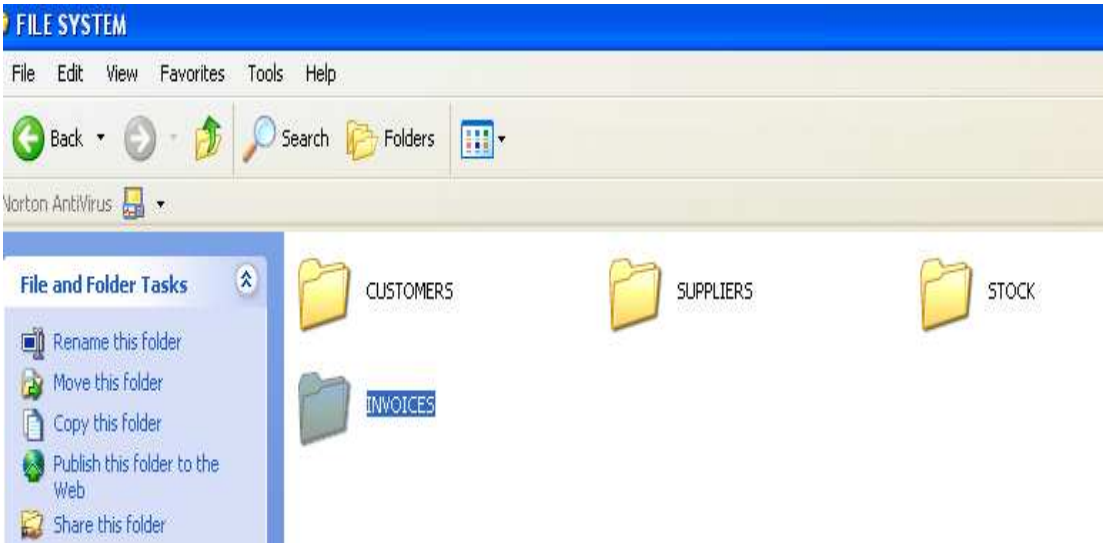
3.1 Analysis of the Current Business Process
According to Maciaszek [13] in order to determine the requirement, the developer must understand how the Business processes are currently performed. Appendix D depicts the current ordering processes using a UML activity diagram. The diagram shows the users involvement at each stage.

3.2 Current Customer Ordering Process
Customers can place their orders over the phone or visit the shop in person to purchase their goods/items. When orders are placed over the phone, the shop assistant takes the order details on A4 sheet of paper with the customer address. The address is then compared with the ones stored on the Microsoft spreadsheet to see if the customer is already known or registered. If he/she is a new customer, the spreadsheet is updated and the customer order is picked and then dispatched to the customer. If for example any item is out of stock, the manager will then initiate a purchase order from suppliers.

3.3 Current Purchase Ordering Process
The store manager purchases stock from various suppliers. Some purchase orders are delivered to the shop by the suppliers, and the store manager also travels to suppliers to purchase stocks. The invoices are paid immediately.

3.5 The Existing System
The current system is mainly a paper-based system with the support of Microsoft Excel Spreadsheets. The spreadsheets hold information about customers, suppliers, addresses and products inventory information. The system uses Excel macros and report calculating tools to perform statistical operations on the data held on the spreadsheet, to help generate stock reports and summaries of debtors.

The major drawback of this system is that understanding the process of reporting using Excel Spreadsheets is highly complex.
The statistical operations executed on the spreadsheet are such that a user must have a great deal of knowledge as to how the system works. The difficulty of understanding the method by which these queries run on the data indicates a system that not only has evolved beyond its original specification, but is one that is no longer efficient.
Chapter 1 outlined most of the drawbacks of the system within the problem definition section.



3.6 Shortfalls of the Current System

Following the analysis of the current system, several limitations will need to be addressed in the new system. Firstly, it is not often apparent when stock levels are running low. Due to the nature of the system, data duplication is inevitable as several spreadsheets hold individual customer/suppliers addresses, which was already recorded elsewhere. The system needs to reduce the amount of paper work to improve speed of order processing and efficiency of inventory management and eliminate data inconsistency. Usability issue as I have earlier discussed in Chapter 2 has not been considered during the design of the current system - as it is very unusable to novice without good excel knowledge.

3.7 Analysis of an off-the-shelf Systems

There are a number of similar off-the-shelf System available on the market, some of those systems offer a variety of functionalities which caters for most of the problems specified above. Following some research, two off-the-shelf packages have been identified and the systems will be fully examined to determine whether the solutions they offer could be adopted.

3.7.1 OrderWise

OrderWise is a stock control and order processing software, its developers claimed that it could “expand your current stock control and order processing capabilities”.

The OrderWise systems key features that could meet the user requirements are summarised below:

• **Key Features** • Product audit trail • Batch traceability • Serial no traceability • Expiry date traceability • Single item traceability (reels of cable; bars of steel; vats of liquid; etc.) • Multiple product purchase and sale unit quantities	• Product grouping • Product usage forecasting • Seasonal trend forecasting • Fast & flexible product searching • Stock turnover control - FIFO; LIFO; Expiry Date; Manual • Multiple stock locations • Flexible customer pricing - Discount Structures; Price Lists; Special Prices; Special Offers; Qty Breaks; Global Discounts; Manual Discounts	• User configurable fields • User configurable grids • Compatible with most leading accounts packages • Can be used as a standalone system

Table 3.7.1- *source www.orderwise .co.uk*

Strengths
OrderWise can be used as a standalone system and it enables the user of the system total control over inventory information. This would be a key feature of any system Adonai would want to implement, as stock is the main asset of the company.

OrderWise also offer the ability to control stock turnover and flexible customer pricing. Implementing these tools in a system would be welcomed by customers.

Weakness
OrderWise is a system that deals with stock control and all issues related to good inventory management, however it does not fully address the issue of online customer order processing.
The system also does not integrate any web-based interface to enable the store manager to operate in different locations.
Most importantly, the system does not implement any function, which may help Adonai rent out any of its DVDs and VCR films to customer.
The system is also too expensive costing between, £1000 - £9000 and this is out of the store manager's budget.

Conclusion
OrderWise offers most of the solutions required to cater for the inventory management problem at Adonai but it does not deal with all other requirements and also the cost of the system is another concern for the manager as this is a small business. Therefore, OrderWise would not be a suitable system.

3.7.2 Stockit!

Stockit! is another inventory management system, according to the developers, "this dynamic product will bring tangible benefits to any small to medium enterprise needing to improve stock handling efficiency". In addition to inventory management, Stockit! system offers other functionality such as report builder.

The key features of Stockit! System is as follows:
High-performance, industry-standard relational database
Secure multi-user access
Easy creation of multi-level parts lists
Minimum stock levels and order thresholds
Automatic order creation
Product serial numbers
Stock valuation
Report Builder for custom reports

Strength

In addition to complete inventory management, the Stockit! system enables the user to manage inventory at remote locations, which is additional benefit for the manager.

The system also offers some automation of order processing and report building interface for custom reports of inventories. Another advantage is that it offers some sort of security features.

Weakness

The system places a great emphasis on the issue of inventory management and does not address the overall issue relating to IS/IT strategy, such as continuing to build customer relationship and attracting new customers while delivering cost effective products, which is required by Adonai in order to achieve competitive advantage.

It also does not offer full functionalities as indicated in the user requirement. The order processing is mainly internal and does not offer any online customer ordering processes. And finally, the functionality that allow for the renting out DVDs and VCR to customers has not been implemented.

Conclusion

The solutions offered by Stockit! system deals with the issue of inventory management but it does not fully cater for the overall user requirement and online customer ordering systems. The order processing functionality of the system is mainly internal and does not allow the customers to place orders online.
The cost of multiple user licensing of the system is also too expensive for the business. Therefore, Stockit! system would not be a suitable solution for Adonai shop.

3.8 The Business Analysis and IT Strategy

According to Yeates and Wakefield [31], the development of an IT strategy is a vital ingredient in the success of any business that uses computers.

Following several informal meetings with the store manager, the manager has told the developer that they currently do not have any Business Strategy. According to Ward [14], for any business to achieve and sustain a competitive advantage, the business must define and adopt a business strategy together with its IS/IT strategy, which must be implement in line with the business strategy.

A business strategy can be defined by the business drivers an organisation uses to operate Ward [14], Adonai's business drivers are summarised below:

- "To continue to satisfy existing customers"
- "To continue to attract new customers and find new markets"
- "To be able to expand and open more stores in and around Leeds area"

Competitive advantage will be achieved by aligning the IS strategy with the business strategy, Ward [14] state that, The essence of business strategy lies in creating future competitive advantage faster than competitors do. Adonai will create future competitive advantage through the implementation of a new IT system.

3.8.1. IT SWOT (Strength Weakness Opportunity and Threats)

SWOT analysis has been applied for many years as a structure and tool for strategy development. It identifies the features of a proposal or idea against the Strength Weakness Opportunity and Threats of a business.

SWOT Analysis of Adonai Current Information System Applications

Application	SWOT Conclusion	Action
BT/AOL Internet broadband	Adequate as it provides access to the internet with email client	Keep
Microsoft Excel spreadsheets	Inadequate, as its drawback outweigh the benefits it brings to the business with little functionalities.	Discard
Email	Adequate as it is useful for communicating with suppliers and customers	Keep

Table 3.8.1

The Strength, Weakness, Opportunity and Threat (SWOT) analysis above clearly shows that the current application used at Adonai shop does not meet the requirements of the business. Microsoft Excel spreadsheet does not offer the full functionality that is required to meet the business objectives. Therefore the Excel spreadsheet should be discarded. The AOL Internet broadband is adequate for the business as it provides useful benefits to the business, suppliers and its customers. The use of e-mail clearly advances the business and

allows customers/suppliers to communicate with Adonai shop in a more convenient manner. According to Davis & Benemati [15], e-mail is one of the most powerful co-operative tools, however commercial use of e-mail is still relatively immature and despite it's widespread use, few businesses understand its nature, constraints, benefits and issues. For these reasons, it was concluded that both the e-mail and the AOL Internet broadband should be kept and improved upon.

3.9 User Requirements

Requirements Gathering Techniques

The developer believed that it would be important to understand the order processes and systems that already existed in Adonai shop because they would have to be closely integrated with the Web ordering system. Three requirements gathering techniques proved to be helpful in understanding the current systems and processes – document analysis, interviews, and observation.

First, the developer collected existing reports such as order forms, screen shots, see section 3.5. That shed light on the current system. The developer was able to gather a good amount of information about the brick-and-mortar order processes and systems in this way. When questions arose, the developer conducted a short informal interview with the store employee.

Next, the developer interviewed the store manager-Mr Japhet Uhu, for the order and inventory system to get a better understanding of how those systems worked.

Finally, the developer spent some time visiting the shop and observing exactly how the order and inventory management processes worked in the brick-and-mortar facilities.

To ensure correct analysis of the problem, user requirements were then identified from the result of previous sections by using the requirement gathering techniques mentioned earlier, which includes the analysis of user interviews, business documents, as well as observations of the current business processes. The identified user requirements for the new system have been divided into functional and non-functional requirements. According to Lawrence Chung [16], functional requirements are functions that the system must be capable of performing. These are software requirements that describe the behavior of the system. Non-functional requirements are the requirements that describe not what the system will do but how the system will do it. Non-functional requirements can be grouped into; the system performance requirement, the system quality attributes and the systems external interface requirement. The requirements are as follows.

3.9.1 Functional Requirements

1	The database shall be able to store and retrieve stock information such as, an item id, quantity in stock, price, minimum level and description.
2	The database shall be able to store and retrieve customer details such as the name, address, town, postcode, and their telephone number and id.
3	The web based inventory system shall allow for the creation of customer

	accounts.
4	The web based ordering system shall be able to generate order confirmation and receipts.
7	The web based ordering system shall allow customers to order items.
5	The system shall store an order_id, item quantity, item code, description, unit price and total price.
6	The system shall provide efficient search facilities for locating stock, orders and customers.
7	The system shall automatically calculate the price that the customer must pay for an order.
8	The system shall allow the user to add, edit and delete customers, product and orders to and from the system.
9	The system shall be able to produce the following reports: • Customer Details • Product Details • Stock Level Details • Overdue Dvds • Lists of Debtors • Dvd Details • Dvd Borrowings • Employees Details
10	The system shall allow for the issuing of dvd to customers.
11	The system shall record dvd details include, id, title and description.
12	The system shall record employee name, address, tel no, post code.
13	The ordering system shall allow user to add items to the shopping cart.
14	The ordering system shall allow user to remove items from the shopping cart.

Constraints

1	Storage of address entries to a maximum of two, excluding the City and Postcode fields.
2	The system should not allow an order to be confirmed if the order line/shopping basket is empty.
3	Numeric integrity constraints will be enforced on fields, which must contain numeric data.
4	The system should not allow an order to be confirmed if the order the delivery details formed is not filled in.

3.9.2 Non-Functional Requirements

Performance
All user inputs must be validated within the client's browser to reduce error and increase throughput in processing of orders.
The execution of user demands as rapidly and efficiently as possible.
The system will rollback transactions such as update appointment incase of hardware or software failure.
To be able to handle the anticipated volume of data, including throughput and storage.

Maintenance
Allowing shop manager to undertake simple routine housekeeping tasks, such as database backing-up.
The system shall be scalable – it shall be able to grow with the business, allowing future updates and modifications to the system.
To have security features such as passwords where necessary for access to the system (Data Protection Act) and backup procedures.

Usability
To be able to operate on any platform Such as Microsoft Windows and Mac. Operating systems.
To be able to operate within any major web browser, such as Internet Explorer, Netscape, Mozzilla and others.
Use of uniform colours and text layout throughout the system, maintaining consistency within the Interface.
Allow a user to cancel an action and navigate forward or backward within the application.
System should have familiar and easy to use interface that can be learned in a short period of time by the user.
System will help the client carry out the tasks and achieve their goals effectively and efficiently by using drop down lists where appropriate.
System shall keep response times to user interaction to a minimum.

4.0 Design

According to Allan Dennis [17], the purpose of the analysis phase is to Figure out what the business needs. The purpose of the design phase is to decide how to build it. This Chapter will deal with the logical, physical design and architecture design of the proposed system.

The design phase has been split into three different parts: The System's Process Modelling, which involves the creation of Data Flow Diagrams (DFD) showing the overall business process and the data flows to and from external entities and data stores. The second part is the design of the Data Modelling, which involves the creation of Entity Relationship Diagrams for the database, and the third part deals with the design of the overall System Architecture, which involves the hardware and network connectivity.

Normalisation, Functional Dependencies and Database Constraints will be dealt with in this Chapter also. The final component of this Chapter will detail the design of the user interfaces.

4.1 Process Modeling

4.1.1 The Context Data Flow Diagram (DFD) of the Proposed System

The context DFD shows the overall business process as just one process (i.e., the Integrated Inventory Management system/online customer order processing system itself).

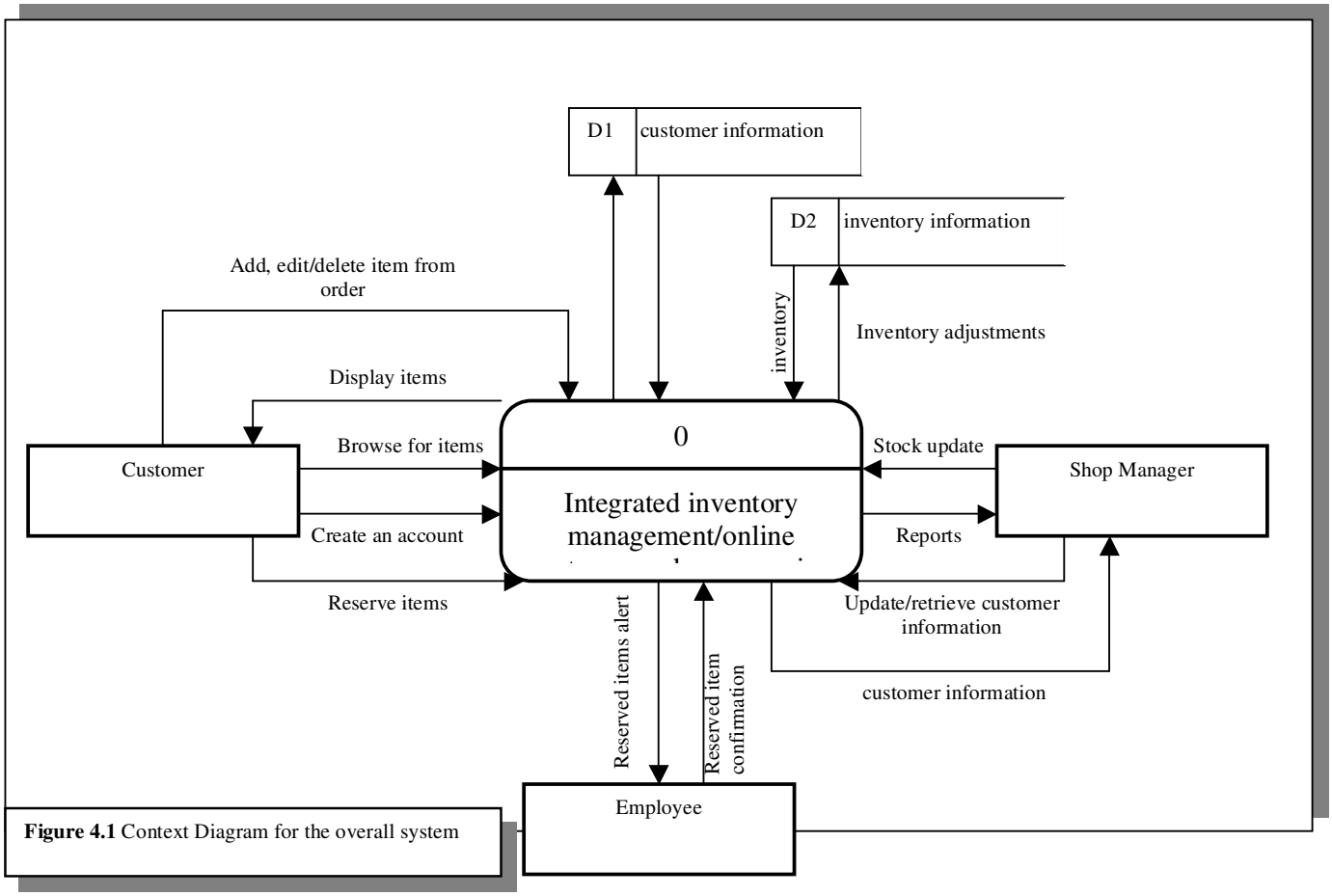


Figure 4.1 Context Diagram for the overall system

Above is a very high level DFD of the entire system showing the input and output from the system and interactions between the internal and external entity. The next step was to create a more detailed lower level DFD for each process within the proposed system and this can be found in Appendix L.

4.2 Data Modeling

Data model describes the data that flows through the business processes in an organization. It is a formal way of representing the data that are used and created by business system, it illustrates people, places or things about which information is captured and how they are related to each other [17]. The Data Modeling Technique that is used is Entity Relationship Diagram (ERD).

4.2.1 Entity Relationship Diagram

An entity relationship diagram is a picture that shows the information that is created, stored and used by a business system. Below is the entity relationship diagram created inline with the user requirements listed in previous Chapter.

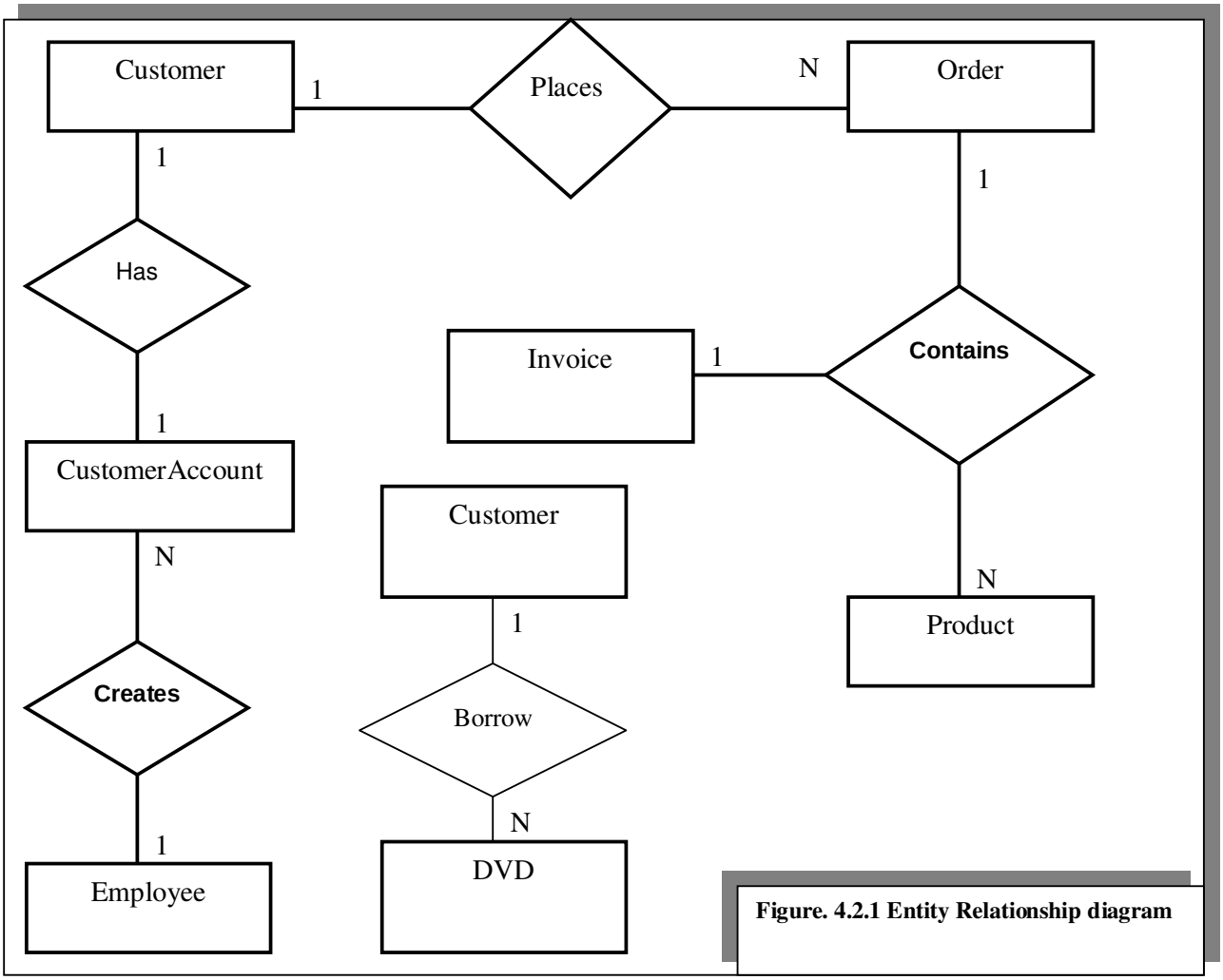


Figure. 4.2.1 Entity Relationship diagram

Notice above is a relationship of degree three (Contains) see Figure 4.2.1, with three participating entities, namely Order, Invoice and Product. Thomas & Begg [18] refer to relationship of degree three as Ternary. The purpose of this is to represent the situation where an invoice contains a customer order and a customer order contains product.

The relationship between the Customer and the DVD (Borrow), is a weak entity, which has its own attributes see Figure 4.2.2. According to date & darwen [23], A weak entity is an entity that cannot be uniquely identified by its own attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key. The foreign key is typically a primary key of an entity it is related to. Therefore, this would make use of the customer id and the DVD_ID.

A Customer can place one or more Order, which results in a one-to-many relationship between both relations. A customer has the following attributes:
Customer id, first name, surname, address1, address2, Telephone number, town and postcode.

One Customer Order contains one or more Product
One Invoice contains one Order.
One Employee creates one or more CustomerAccount
One Customer has one CustomerAccount
One Customer borrows one or more DVD

4.2.2 The Database Schema

Elmasri and Navathe [19], state that to ensure a sound database design, database schema must adhere to entity and referential integrity. These ensure that the primary key is never null and no unmatched foreign key values exist respectively. Entity Integrity is achieved by setting particular fields to NOT NULL., and Referential Integrity by recording the foreign key value for the primary key in the referring table. Using low level detail from the Analysis project phase, the schemas can be designed now to include, entities, attribute details and relationships between them.
The following are lists of entities and their associated attributes identified from the initial Entity Relationship Diagram depicture in Figure 4.2.1.

ENTITIES	ATTRIBUTES
Customer	**Customer_id**, first_Name, Surname, Address1, Address2, city, PostCode, Tel_No.
Store_Orders	**Order_id,** *Product_id,* Quantity, Customer_id, Date, TotalAmount, order_name, order_address, order_tel, order_postcode, status,
Store_Item	**item_id,** description, Quantity_in_Stock, Price, Min_stock_level
Invoice	**Invoice_No,** *Order_id***,** Customer_id, Date.
CustomerAccount	**Account_No**, *Customer_id*, Credit_Limit, Outstanding_balance Order_id.
Employee	**Emp_id,** first_name, surname, address1, address2, city, tel_No
DVD	**Dvd_id,** title, description
Borrow	***Customer_id, dvd_id*****,** date_borrowed, due_date, fine, returned

LEGEND:
Primary keys are underlined and emboldened
Foreign Keys are italicised

Figure.4.2.2 The Initial Database Schema

The entities and attributes listed above may be subject to some changes and adjustment where necessary in order to fall in line with the business rules, database constraints, functional dependencies and normalization process

4.3 Constraints

Detailed constraints of which the inventory management and online customer order processing system will be tested for are as follows:

4.3.1 Business Constraints

- Customers cannot check out their orders if the delivery details form is not completed.
- Customer cannot place any order if the amount of the order exceeds his/her credit limit.
- A customer cannot purchase an item if the required quantity is greater than quantity_in_stock.

These constraints will be executed on the client side within the browser.

4.3.2 Functional Dependencies

Thomas & Carolyn [18], describes functional dependencies (FD) as the relationship between attributes in a relation. For example, if A and B are attributes of relation R, B is

functionally dependent on A (denoted A → B), if each value of A is associated with exactly one value of B. The Customer table is such that functional dependencies exist.
Notice the above schema (Figure 4.2.2) *CustomerID* → *first_Name* as CustomerID is unique and determines the firstname of a customer. However we can deduce that CustomerID is not functionally dependant on first_Name because the first_Name does not uniquely identify CustomerID, as two customers may have the same firstname. Consequently we can say the firstname is truly functionally dependent on CustomerID. Understanding functional dependencies is important as it will be useful in the normalisation process later.

4.3.3 Primary Key Constraints

Primary Key constraints are used to ensure that all rows in a table are uniquely defined and are vitally important especially when using insert, update and delete commands. Exploiting the benefit of primary key constraints will be achieved through using the " auto increment" and " not null" features of MySQL, and will be employed throughout the tables. An example of this would be the Customer table and the attribute CustomerID. The system must at all times ensure that this constraint be correct so as to ensure data consistency.

4.3.4 Domain Constraints

According to Elmasri & Navathe [19], Domain constraints specify that within each tuple, the value of each attribute A must be an atomic value from the domain dom(A). i.e. these types of constraints specify what data type the attribute will accept and the format with which this data type will be accepted. For example, the Customer table specifies that the CustomerID of a Customer must be an (AutoIncrement), hence the domain constraint on this attribute is that it must be of type that increments automatically as new customers are added to the system.

4.4 Normalisation

According to Thomas & Carolyn [18], Normalisation is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise. Elmasri & Navathe [19], further defined normalisation of data as a process of analyzing the given relation schemas based on their functional dependencies and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anormalies.

The purpose and benefits of normalization can be deduced from the definitions above and for the purpose of this report, the data contained in the schema depicted in (Figure 4.2.2) has been normalized as the full normalization process is beyond the scope of this project and therefore will not be covered in depth.

4.4.1 Boyce Codd Normal Form

Adhering to a particular normal form ensures a level of consistency and data integrity will be achieved. Four different normal forms are commonly used to normalise system schemas, with Boyce Codd Normal Form (BCNF) adhering to the strongest constraints, C.J. Date [5]. It can be concluded that all tables in the schema in Figure 4.2.2 adhere to

third normal form. This states that any attributes dependant on each other must be separated, to be dependant upon the super key for that class.

4.5 System Views Design

4.5.1 User Interface Design

A standardise template has been created (see Figure 4.5.1) to ensure consistency in the layout of the web site, The header content will change according to which part of the system the user has navigated to. When a user has logged into the system the header will display options to chose from and work with a table. When a user is not logged in, the shop name will only be displayed with limited functionality buttons. Shackle & Richardson [20] state that the user should not be able to access parts of the system that are not appropriate to them. Therefore, the Menu buttons on the side will display the number of buttons available to a user depending on their access privilege. The contents will be displayed in the middle of the page, a Help button will also be available on each screen and finally a search box will be located near the menu buttons on the side. And other marketing information such as discount, offers, and adverts will be present across the top of the page in the (INFO BAR). PHP, HTML, JavaScript are used to create the interface.

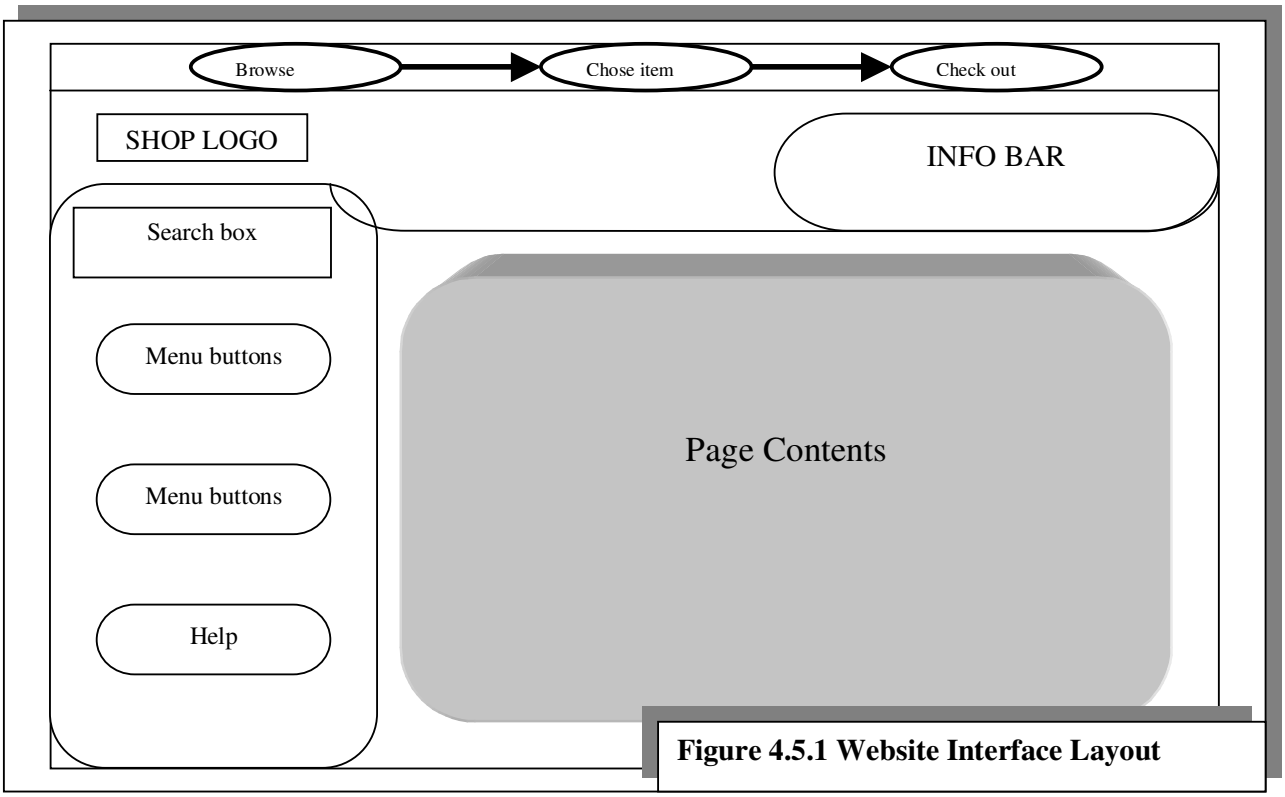


Figure 4.5.1 Website Interface Layout

4.5.2 Usability Design

The basic usability framework used in the design of the system is such that it promotes ease of use. In three summarised easy steps, a typical customer can complete a basic task by simply; browsing for items, then choose item and then checkout. The chain of event will be mapped across the top off every page indicating what stage the customer is in within the page in the form of a simple diagram as shown below in Figure 4.5.2.

Consistency according to Neilsen [21], Users should not have to wonder whether different words, situations, or actions mean the same thing by using consistency in system colours to represent the different features available. Such design enables users to gain system familiarity. The consistency in the inventory management/ online customer order processing system is in accordance with the general principles of usability defined by Nielsen [21] and will aid the employee and customers as all users quickly becoming accustomed to the system is paramount.

A number of colours will be tested to identify the ones that best match the menu bar and headings. Furthermore, Nielsen [21] advised the use of contrasting colours where necessary. All errors will be flagged in red colour to stand out against a white background. Using contrasting colours will enable users to quickly read data off screen. Create, Retrieve, Update, Delete and View actions are dark blue, help and information displays are white in a black background. Cascading Style Sheets (CCS) support will be incorporated into every web page. According to W3Schools [22], specifying format attributes in one CSS file makes all changes global, i.e. to all pages using the style sheet. This also allows for interface consistency throughout the system and easy modification if required.

All buttons within all forms will be of the same size, and all forms containing text boxes will enforce tab indexation and dropdown option lists where necessary. Required fields are also marked with a (*) to minimise users receiving validation error messages. The login/logout button will be at the top for easy access.

The user will be warned if they attempt to delete information about an entity that will affect corresponding tuples in connected tables. The user will then be prompted to decide whether they wish to continue even if corresponding tables will be affected.

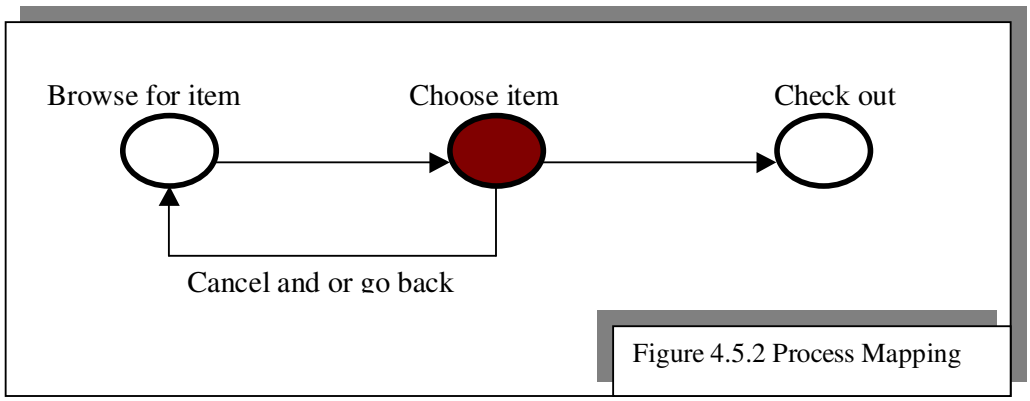


Figure 4.5.2 Process Mapping

4.6 System Architecture

The architecture of the system is depicted below using a logical and physical design. The logical design shown in Figure 4.6.1 is a data flow diagram that describes the flow of information requests and responses for the system. The physical design shown in Figure 4.6.2, describes the hardware and software needed to realize the logical design.

4.6.1 Logical design for the website

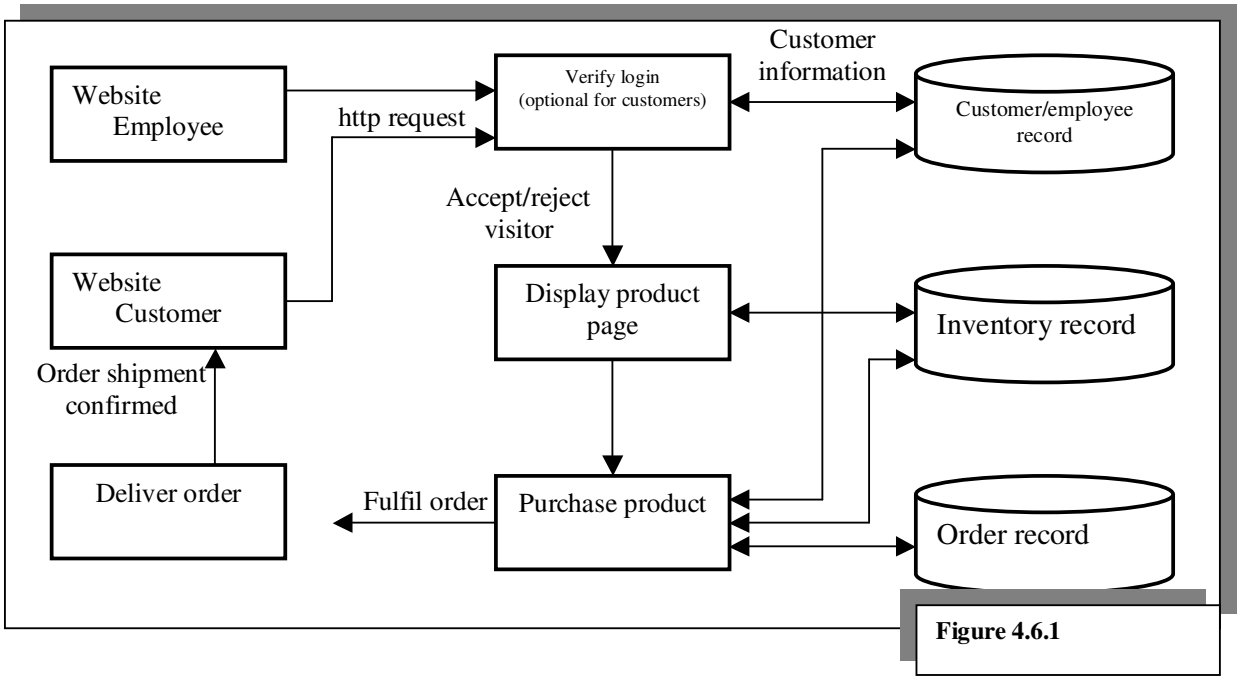


Figure 4.6.1

The idea behind the logical design is that the system will be accessible to both employee and customers via web interface, using http request, an employee can log onto the system and will be granted access to update, modify, delete and/or amend any record within the system.

A customer can access the website without loging in however they will be restricted to certain functions and will not be able to modify any inventory record. The reason for this technique is that research evidence suggests that passwords and logins can be problematic for customers and could also deter potential customer thereby resulting in loss of sales.

4.6.2 Physical design for the website

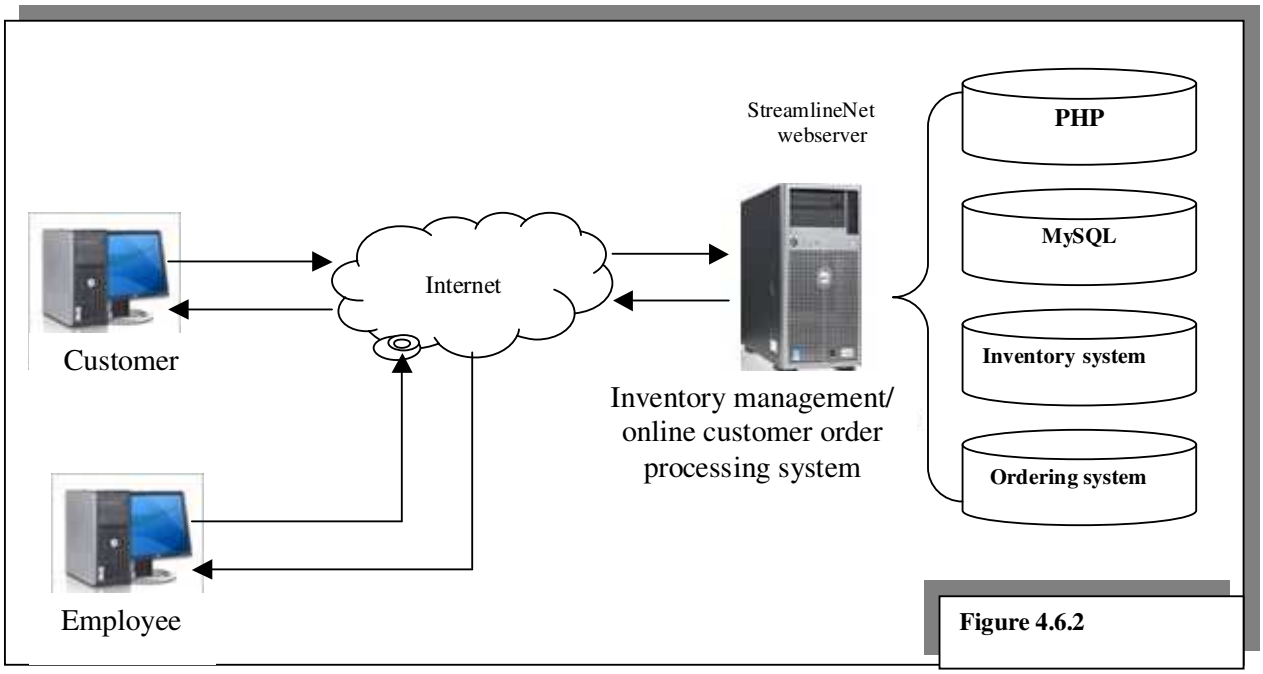


Figure 4.6.2

5.0 Implementation

This Chapter details the tasks, which was performed in order to implement the Design as described in Chapter 4. The installation of relevant software required for the system development was explained. This Chapter documents the sequences of operations performed in order to implement a system that would meet the requirements. The Chapter will also discuss database, server, applications and security implementation that occurred during this phase of the project.

5.1 Software Installation

The software necessary for creating the required system was available through a CD-ROM that came with a textbook called PHP, MYSQL & APACHE [24]. The CD-ROM includes MySQL version 4.0.21, Apache Web Server version 2.0.52 and PHP version 5.0.2. These software programs were installed on the developer's computer for development purposes. Installing Apache and PHP on the developer' s computer were achieved through the use self-extracting versions of the Apache and the installation of the PHP was achieved by referring to configuration documentation, which can be found in PHP&MYSQL [25].

5.2 Database Implementation

The initial phase of the implementation was creating the database and its tables. As discussed in Chapter 2, MySQL was the chosen database technology. This tool allowed for the rapid creation of schema and all tables, constructed via a MySQL graphical user interface (GUI) application called Mysqlyog. The Mysql statements that were used to create each table can be found in Appendix E.

5.2.1 The Database Tables

There were twelve tables created that relates to the system's functionality. When creating tables in MySQL, each table is assigned a name and the table fields are specified. Each field specified contains its own name, a primary key, foreign keys if appropriate, data type and field length.

Primary Key Constraints

Some of the tables were assigned with a primary key that was set to auto_increment. The primary key would ensure record integrity by uniquely identifying each record entry within the system. The auto_increment flag automatically assigns integers in sequence to new records, aiding integrity maintenance as suggested by Atkinson [26]. The system user need not worry about maintaining primary keys for the store_orders, new customer, customer_accounts, employees and store_orders_item tables. The item categories, store_items and DVD tables does not use auto_increment, because these tables contain items that already have a unique identification numbers.

The '*borrow*' table contains a composite key referencing the primary keys on DVD and *Customer* tables respectively. This will allow for the system to tie the customer and DVD tables in other to recognise the borrower of any listed DVD and retrieve their details. The composite keys are two or more columns designated together as the primary key.

Data Types

The vast majority of the data type used within the system was the VARCHAR, which allows inputs of alphanumeric data items. These were used for post code and address fields.

All the tables that required date/time, the DATE data type were used to maintain integrity. The DATETIME data type was used where a time was needed to be stored, such as the storage of new orders in the *store_orders_items* and *store_orders* table. All Prices fields used the FLOAT data type, with a format of (6, 2). This imposes all prices to have a maximum of two numbers after the decimal place.

Other Constraints

The fields in the system that should not be left empty in other to maintain data integrity, were flagged NOT NULL- to ensure that the database required data to be present for those particular fields. Furthermore, some fields were initialised with a default flag, being set if no data was persisted for that field. For example, when an order is confirmed, corresponding entries for the appropriate order-lines are made in the *store_orders_items table*, with a default value of 'PENDING', which indicates this is a new order. Referential integrity constraints between tables- as defined in the Design section in Chapter 4 of this report, is upheld by referencing primary and foreign keys within each relevant table in the form: FOREIGN KEY (*field_name*)REFERENCES *parent_table*(*field_name*). If a particular action violates this constraint type, such as deleting a record in the parent table, MySQL will generate an error.

Below is an example of MySQL syntax extracted from Appendix E, to demonstrate constraints and some data types discussed:

create table Customer_Account (
Account_No int not null primary key auto_increment,
Customer_id int not null,
Credit_Limit float not null,
Outstanding_balance float not null,
Order_id bigint (20) not null,
FOREIGN KEY (customer_id) REFERENCES customer (customer_id),
FOREIGN KEY (order_id) REFERENCES store_orders (order_id)
);

5.3 Implementing the Ordering System

The following section will document the procedures, techniques and software use in creating the ordering system. The PHP files were implemented using a programme called PHP Designer and a MS note-pad.

5.3.1 The Ordering System Tables

In total, three tables are required namely: *store_categories, store_items*, and *store_item_size*. Although the table named '*store_items*' has already been implemented in previous section during the implementation of the inventory management database.

Table Name	Attributes
store_categories	Cat_id, cat_title, cat_desc.
store_items	Item_id, cat_id, item_title, item_price, item_desc, item_image.
store_item_size	Item_id, item_size.

Table 5.3.1 ordering system tables

The store_category table has two attributes besides the cat_id attribute: cat_title and cat_desc, for the title and description. The cat_id is the primary key and cat_title is a unique field because there's no reason to have two identical categories.

In the store_items table, the cat_id attribute relates the item to a particular category in the store_categories table. This attribute is not unique because we will want more than one item in each category. The item_image attribute will hold a filename-in this case its assumed to be local to the server, which will be used to build an HTML *<IMG>* tag when it is time to display the item information.

The store_item_size table is optional. However it has been implemented for the purpose of system scalability as specified in the user requirement in Chapter 3. This will enable the client to store complex items information in the future if for example they do decide to extend their product range to include items with varied sizes such as clothing.

5.3.2 Displaying Categories of Items

Using PHP scrip, a file named '*seestore2.php*' has been implemented to list categories and items. The system has been implemented such that all categories and items will not

be listed at once as soon as the page is displayed, instead the user is given the option to immediately pick a category, see its items and then pick another category. In other words, the file *seestore2.php* will serve two purposes: it will show the categories and then show the items in that category. Sample implementation code is shown below:

```
//Show categories first
$get_cats = "select cat_id, cat_title, cat_desc from store_categories order by
cat_title";
$get_cats_res = mysql_query($get_cats) or die(mysql_error());

// Then Show items
$get_items = "select item_id, item_title, item_price from store_items where cat_id =
$cat_id order by item_title";
 $get_items_res = mysql_query($get_items) or die(mysql_error());
```

Figure 5.3.2 Displaying Categories of Items

5.3.3 Displaying Items Information

The file '*showitem.php*' is used to display all the items information including the item image. MySQL query has been implemented and issued to retrieve the category and item information. The query is a table join. Instead of selecting the item information from one table and then issuing a second query to find the name of the category, the query simply joins the table and the category id to find the category name. Sample implementation code below:

```
$get_item = "select c.cat_id, c.cat_title, si.item_id, si.item_title, si.item_price,
si.item_desc, si.item_image from store_items as si left join store_categories as c on
c.cat_id = si.cat_id where si.item_id = $_GET[item_id]";
$get_item_res = mysql_query($get_item) or die (mysql_error());
```

Figure 5.3.3 Displaying Items Information

5.3.4 Implementing the Shopping Cart (basket)

In order to successfully implement the shopping cart, two new tables has to be created in addition to *store_order* table that was created for the inventory systems earlier. The new table will be called *store_shoppertrack*, it will be used to hold items as users add them to their shopping cart.

Appendix E depicts the SQL statements that were used to create the new tables.

However, for the sake of clarity in the following explanation, I will include the attributes of the store_shoppertrack table in this report as follows:

Table Name	Attributes
store_shoppertrack	Shop_id, session_id, sel_item_id, sel_item_qty, sel_item_size, date_added.

Notice from above that the only key in this table is the *shop_id* attribute. The session_id cannot be unique; otherwise, the users could order only one item from the ordering system, which is not a good business practice.
The value stored in the *session_id* attribute identifies the user; it matches the value of the PHP session ID assigned to them. The sel_* attributes hold the selections by the user: the selected item, quantity of the item, and the selected size if applicable. Finally, there's a *date_added* attribute. Many times, users place items in their cart and never go through the checkout process. This practice leaves straggling items in the tracking table, which need to be cleared periodically. For example, the system administrator may want to delete all cart items more than a week old- this is where the *date_added* attribute is helpful.

5.3.5 Adding Items to Cart (basket)

A file *addtocart.php* has been implemented to tackle this requirement. This file will simply write information to the *store_shoppertrack* table, and will automatically redirect the user to view of the shopping cart. To start with, the file initialises a PHP session using the function '*session_start()*'. This is important because the system needs to recorgnise and capture the user's session ID to write to the store_shoppertrack table. Sample code shown below:

```
<?PHP
session_start();
$addtocart = "insert into store_shoppertrack values ('', '$PHPSESSID',
'$_POST[sel_item_id]', '$_POST[sel_item_qty]', '$_POST[sel_item_size]', now())";
  mysql_query($addtocart);
```

Figure 5.3.5 Adding Items to Cart

5.3.6 Viewing the Shopping Cart (basket)

A file *showcart.php* continues the user's session using the session ID with the
records in the *store_shoppertrack* table. This has been implemented with a joined SQL query, in which the user's saved items are retrieved. The *shop_id, sel_item_qty*, *sel_item_size* attributes are extracted from store_shoppertrack, and the *item_title* and *item_price* attributes are retrieved from the *store_items* table, based on the matching information from *store_shoppertrack.*
If there are no results, the user has no items in the store_shoppertrack table and a message is written to the user advising accordingly. Sample code shown bellow:

```
<?PHP
session_start();
//check for cart items based on user session id
$get_cart = "select st.shop_id, si.item_title, si.item_price, st.sel_item_qty, st.sel_item_size
from store_shoppertrack as st  left join store_items as si on si.item_id = st.sel_item_id where
session_id = '$PHPSESSID'";
$get_cart_res = mysql_query($get_cart) or die(mysql_error());
```

Figure 5.3.6 Viewing the Shopping Cart

5.3.7 Removing Items from the Shopping Cart (basket)

The file *removefromcart.php* is used to issue a query and redirect the user to another script *showcart.php*. It enables users to remove unwanted items from their cart. This is based on the users session ID with the records in the store_shoppertrack table. Sample code shown below:

```
<?PHP
session_start();

if ($_GET[id] != "") {
   $delete_item = "delete from store_shoppertrack where shop_id = $_GET[id] and
session_id = '$PHPSESSID'";
   mysql_query($delete_item) or die(mysql_error());

   //redirect to showcart page
   header("Location: showcart.PHP");
   exit;
```

Figure 5.3.7 Viewing the Shopping Cart

5.3.8 Checking out Orders

The checking out process is very straight forward as there will not be any credit card processing involved due to the requirement of the client. A form *checkout.html* and *checkout.php* has been created to tackle the checking out process. The .html file uses java-scripts and HTML to take and validate users input before calling the .php file to process the form and send the orders to the inventory management system by automatically populating relevant tables within the system.

5.4 User Interface Implementation using Iterative Approach

To conform to SDLC and the Waterfall model methodology defined in Chapter 2, a standard interface prototype was created thus allowing the developer to further improve the design during numerous iterations whilst adhering to the usability framework as identified in Chapter 4. According to Neilsen [21], this process is advantageous as it " allows designers to take advantage of any insights into user needs that emerge from the tests", these insights can then be used to improve the interface from iteration to iteration.

The Standard Interface Prototype

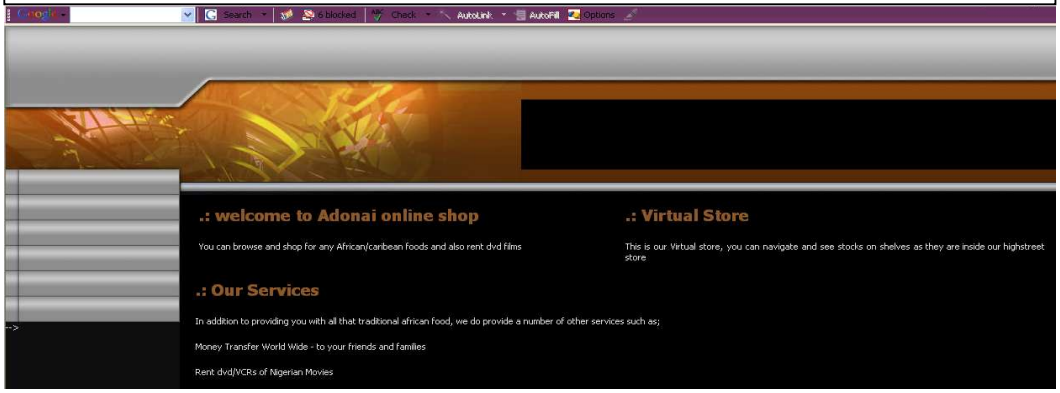


Figure 5.1 The first iteration of the interface

The initial iteration of the interface design dealt with the important principles identified by Neilsen [21], such as " recognition rather than recall" and " aesthetic and minimalist design" Recognition is fulfilled by the system menu buttons, which are of the same format and color. With this basis continued throughout the whole system i.e. the Inventory Management System and the Online Ordering System the users ability to memorise functions will be reduced, which in turn will improve efficiency. The aesthetic and minimalist design will not only present information to the user that is relevant it will ensure that the user is not overloaded with information Neilsen [21].

Interface of the Inventory Management System

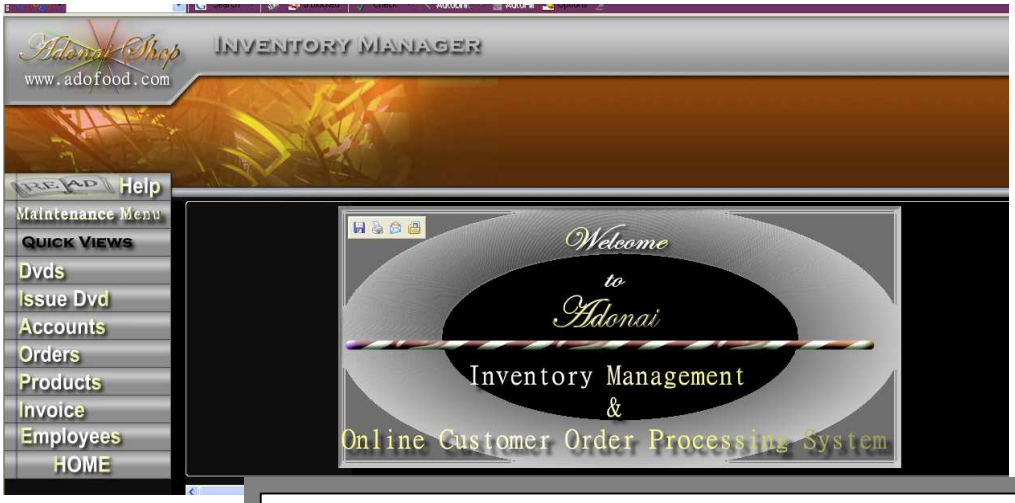


Figure 5.2.1 The final iteration of the interface

Interface of the Online Ordering System

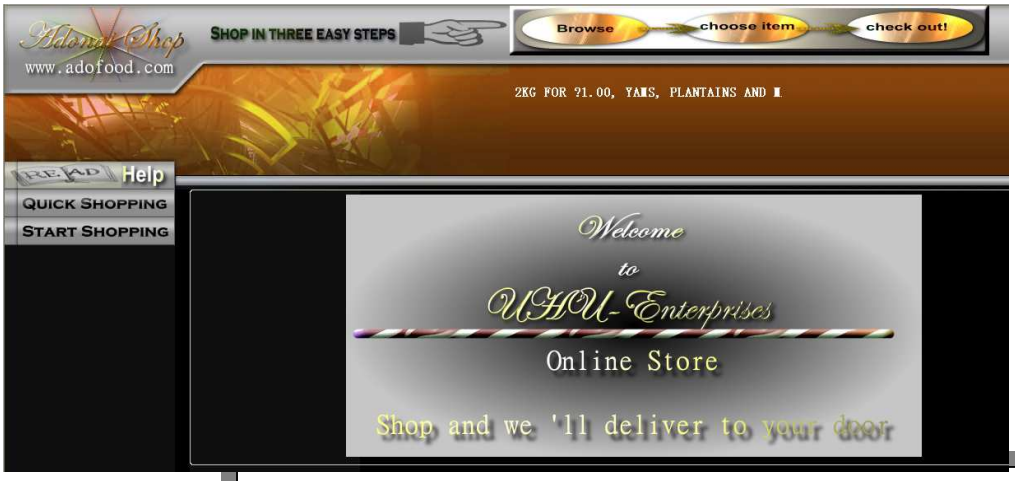


Figure 5.2.2 The final iteration of the interface

The final user interface has been built using HTML, Cascading Style Sheets (CSS), PHP and Java scripts. Together, these techniques have built functionality, validations and usability into the application.

Following several iterations, the final interface was finalised as depicted on Figure 5.2.1 and Figure 5.2.2. These interfaces included further principles outlined by Nielsen [21]. Help was a key issue identified that would aid a user considerably, although the preference for any system is that help should not be necessary, a little help to guide the user can be hugely beneficial. The menu bar offers a help button with different help topics available to the user. Also the menu bar used words, which are relevant to the business as oppose to system oriented descriptions such as the 'start shopping' button on the customer interface. Nielsen [21] also indicated that " Flexibility and efficiency of use" in a system is of very importance and the use of accelerators should be used to enable users to work more effectively. The "Quick Views" buttons on the Inventory Manager menu (see Figure 5.2.1) was implemented to enable common user tasks to be performed faster than the original method. This keeps the amount of processing time to a minimum.

5.5 System File Structure

5.5.1 The Inventory Management System Files Structure

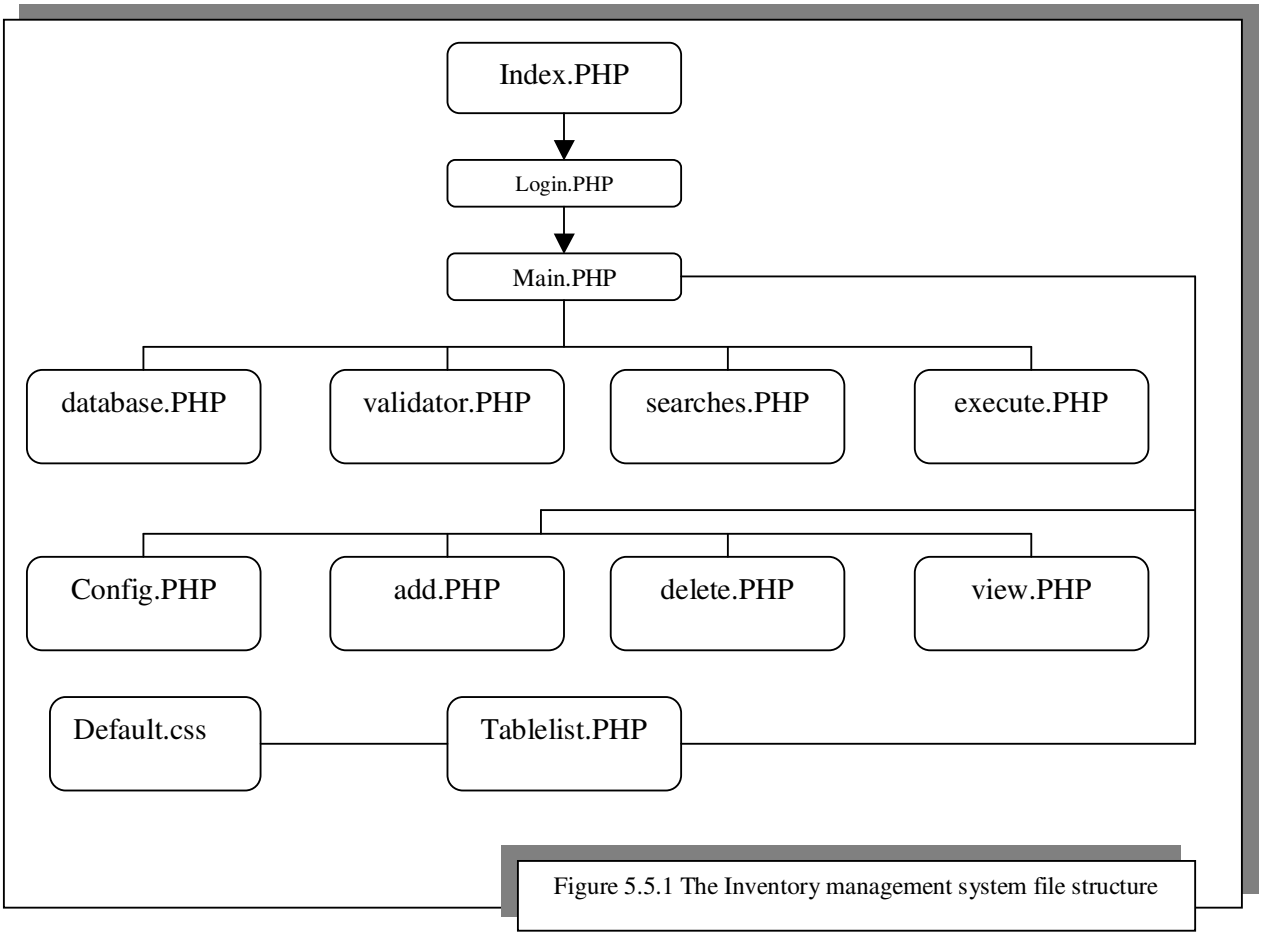


Figure 5.5.1 The Inventory management system file structure

The file named '*index.php'* is used to present the login instructions to the user, which after a successful login, the file '*main.php*' is executed with lists of all the available tables within the database such as borrow, customer_accounts, DVD etc. the file '*main.php*' incoporates all other '.php' files using the PHP function '*require("config.php");*'. The file *config.php* defines all the tables and the security checks required for login onto the system and the file *database.php* is used to defined and establish the connection to the database. *Tablelist.php* file is used to fetch all the available tables from the database and then passes them to '*main.php*' for display. Difining these configuration and functional files as a separate software module, allows a consistent mimalist design to be achieved throughout the system and hence helps the user more easily adapt to different functions. For example, *the add.hpp, delete.php, edit.php and view.php* files where defined once but called to perform their function where required on different tables rather than defining them separately on individual tables. I.e. *add.php* file will enable the user to add new customer to the customer table and can also be used to add new items, accounts, DVD and all other tables in the database.

5.5.2 The Online Ordering System File Structure

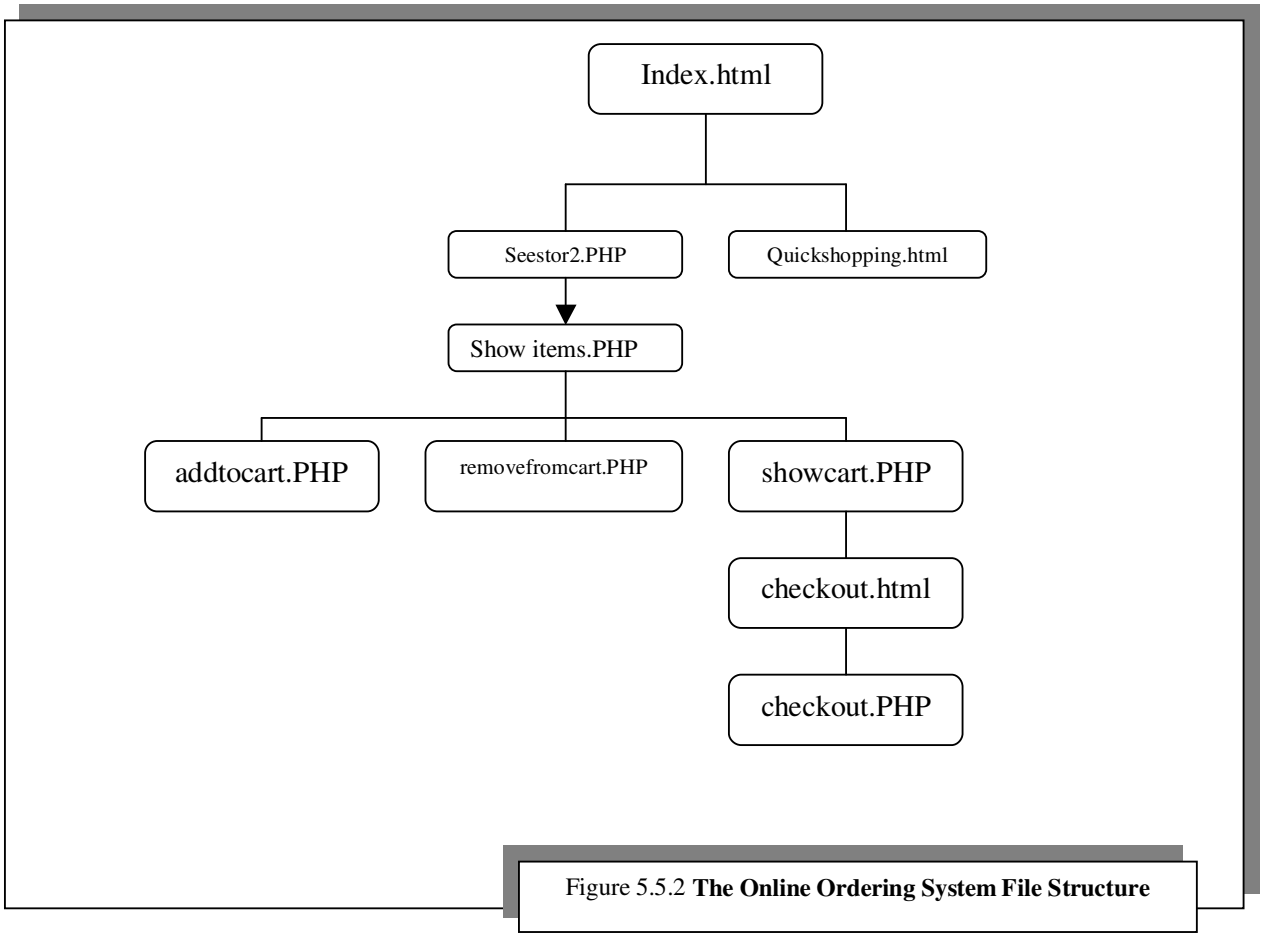


Figure 5.5.2 **The Online Ordering System File Structure**

The file '*quick shopping.html'* is used to execute the processing of customer shopping list. This is a HTML form, which when submit will send an email to the back office inventory manager system for staff to pick the items.

The file '*seestore2.php*' is used to list the store item categories, the idea here is that a customer do not have to see all the different items available at once. The listed categories must be click or selected by a customer to allow them to display available items under that category.

The file '*showitem.php*' is executed when a customer clicks on any given item category, this file looks in the database and fetch the required item from the selected item category. The '*addtocart.php*' file is executed when the addtocart button is clicked, this then picks the selected item and build a shopping cart for the customer. The file '*removefromcart.php*' is used to delete specified item from the shopping cart. *Checkout.html* file contains the form which is used by the customer to submit their delivery details once the information is entered onto the form, the file '*checkout.php*' is then executed to populate the database with the new order and also retrieve and output the new *order id* onto the screen for the customer record.
Detailed implementation of these files will be discussed later.

Cascading Style Sheets (CSS) was embedded onto PHP and HTML files, replicated throughout the system in the form *"<link rel="stylesheet" type="text/css" href="<?PHP print($CSS); ?>" />*". The *'.css'* file defines the format of the interface such as the font sizes and predefined colours for the pages. The html files contains data such as the HTML for the navigational menus as well as the header bar displays. This reuse of code allowed a consistent "minimalist design" to be achieved throughout the system and hence assist the user by adhering to one of the defined usability issues in Chapter 2. In a number of circumstances the interface was improved by validate all user inputs within the client's browser rather than the database and the use of drop down lists such as the selection of item sizes, this idea will help mitigate the issue of *referencial integrity* within the database.

The completed interface design was embedded onto a html *"<frameset>"* this technique allows the interface to remain constant such that when a user navigates to a different page within the system, the interface will remain unchanged, only the content area will change.

5.6 Security and Privacy

5.6.1 Passwords

Password protection was a system requirement specified by the non-functional user requirements in Chapter 4. Password protection is only required within the system maintenance interface to prevent unauthorised user accessing customer accounts and inventory information. Therefore the username and password for the system has been defined in the *'config.php'* file. When a user logs onto the system, through the file *login.php*, this file then activates *config.php*, which checks the user's supplied password against the predefined one, if there's a match, access is granted onto the Inventory

Manager system otherwise the user is advised login failure. The *config.php* file is hidden in the server and cannot be accessed without access to the servers control panel, which is administered by the hosting company called Streamlinet.
If a user's login is successful, then a session is started, signifying that a valid user is working on the system.

5.6.2 Session Enabling

After a session has been initialised, a session variable is created for the user, this variable is currently set to true and will expire when the system is idle for a PHP predefined period of inactivity. When this occurs, the user is redirected to an error page and is advised to login again.

5.6.3 MySQL Injection Vulnerability

According to Vikram Vaswani [25], SQL injection is a technique for exploiting web applications that use client-supplied data in SQL queries without stripping potentially harmful characters. The threat can be dealt with in a number of ways such as the use of a PHP function called '*stripslashes()*' to strip off all the slashes in all inputted strings or data.
For example the query below uses 'stripslashes' to remove any potential harmful characters such as backslashes from a string value item category description *"$cat_desc":*

"$cat_desc =* ***stripslashes****($cats[cat_desc]);"

5.7 System Functionality

5.7.1 Viewing, Editing and Deleting Records

Individual records are viewed by clicking on the view button built in every item line, when a user clicks on this, the item details is displayed in full. To edit a record, when the '*edit*' button is clicked, the a query is issued and the required data is retrieved from the database and returned to pre-fill a form (see Figure 5.6.1 below), the data in the form can then be altered and saved. A record set can be deleted by clicking on the '*delete*' button.

Figure 5.7.0 View, Edit and Delete Menu

Figure 5.7.1 Editing Record

5.7.2 Adding Records

All data are inputted onto the system via text fields as shown below in (Figure 5.6.2). When the user clicks the submit button each and every field is checked against the constraints that the variable must meet. If for any reason the input does not meet the constraint defined for that particular variable the input will not succeed and the user will be advised accordingly through an error message.

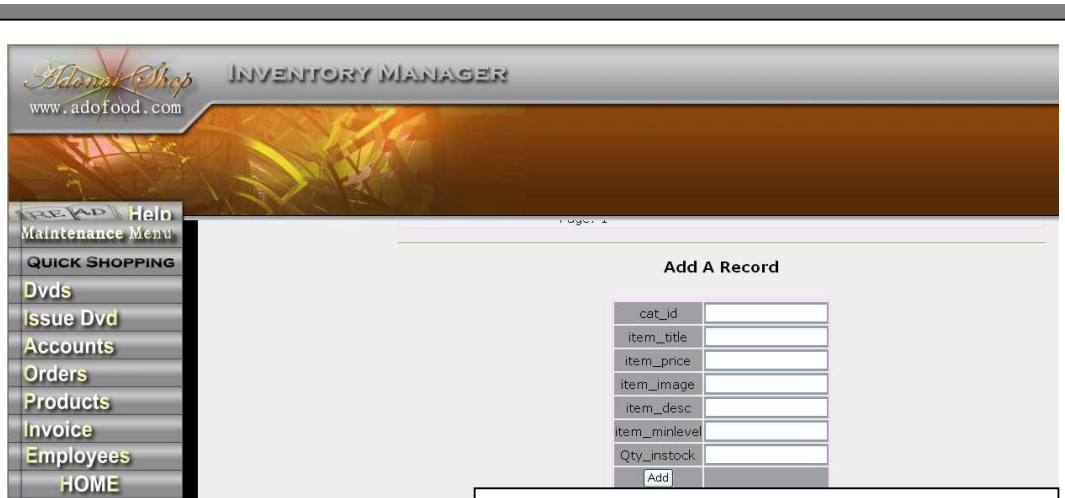
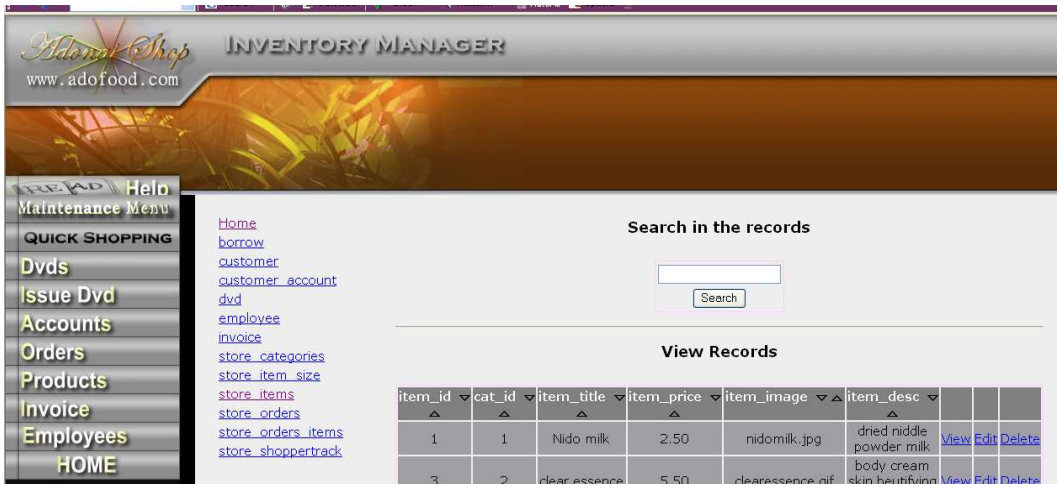


Figure 5.7.2 Adding a record

5.7.3 Searching for Records

The search facility available on the system uses a "wild-card" to perform a query on all available fields within a given table. The wild card matches any inputs such as strings, characters and numbers.

"$QUERY .= "`$Field` like '%$_POST[QUERY]%'";"



5.7.4 Generating Reports

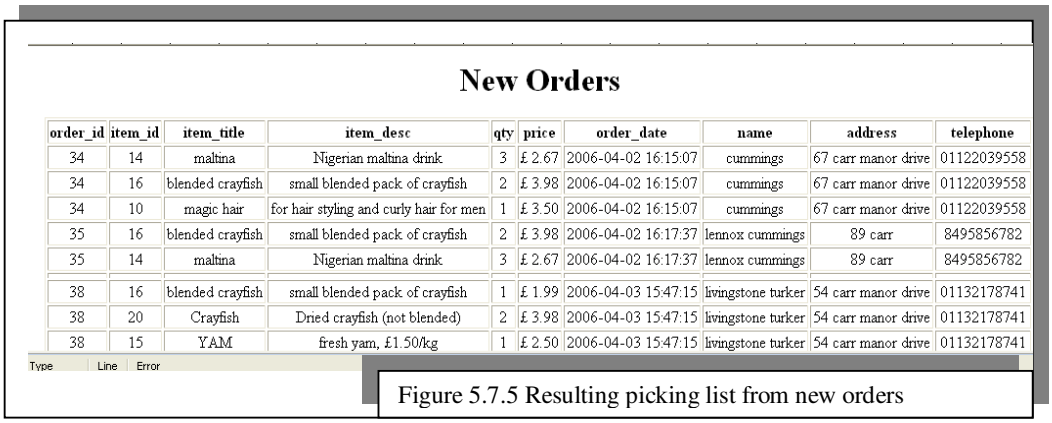
Functional requirements number 12, defined in the requirements specification outlined in Chapter 3 were features identifying reporting needs. The majority of the reporting needs defined in the specification, were such that they must not only be viewable on screen but also printable. It has been decided that most of the reports will be based on the user's screen, however if the user wishes to print a hard copy of the report, this can be achieved by clicking on the file print option from the browser.

Retrieving Customer Orders

To view and generate a picking list for new orders, the button on the menu of the Inventory Management System has been assigned to a file called *'view_neworders.php'* , which is activated once the button is clicked. The script together with an integrated MySQL query (see Figure 5.7.4) is used to query the following tables and retrieve new and pending orders only - (when a customer places an order, the order is automatically marked 'pending' and populates relevant tables): store_items, store_orders and strore_orders_items tables. The resulting data is then displayed on the screen (see Figure 5.7.5) or the user can print a hard copy of this report by simply clicking on the file – print icon on the users browser.

```
$get_order = "select soi.order_id, soi.sel_item_id, si.item_title, si.item_desc,
soi.sel_item_qty, soi.sel_item_price, so.order_date, so.order_name,
so.order_address, so.order_tel from store_orders_items as soi inner join
store_orders as so on soi.order_id=so.order_id inner join store_items as si on
soi.sel_item_id=si.item_id where so.status='pending' order by order_id";
$get_order_res = mysql_query($get_order) or die(mysql_error());
```

Figure 5.7.4 Mysql qery to retrieve new orders



New Orders

order_id	item_id	item_title	item_desc	qty	price	order_date	name	address	telephone
34	14	maltina	Nigerian maltina drink	3	£ 2.67	2006-04-02 16:15:07	cummings	67 carr manor drive	01122039558
34	16	blended crayfish	small blended pack of crayfish	2	£ 3.98	2006-04-02 16:15:07	cummings	67 carr manor drive	01122039558
34	10	magic hair	for hair styling and curly hair for men	1	£ 3.50	2006-04-02 16:15:07	cummings	67 carr manor drive	01122039558
35	16	blended crayfish	small blended pack of crayfish	2	£ 3.98	2006-04-02 16:17:37	lennox cummings	89 carr	8495856782
35	14	maltina	Nigerian maltina drink	3	£ 2.67	2006-04-02 16:17:37	lennox cummings	89 carr	8495856782
38	16	blended crayfish	small blended pack of crayfish	1	£ 1.99	2006-04-03 15:47:15	livingstone turker	54 carr manor drive	01132178741
38	20	Crayfish	Dried crayfish (not blended)	2	£ 3.98	2006-04-03 15:47:15	livingstone turker	54 carr manor drive	01132178741
38	15	YAM	fresh yam, £1.50/kg	1	£ 2.50	2006-04-03 15:47:15	livingstone turker	54 carr manor drive	01132178741

Type Line Error

Figure 5.7.5 Resulting picking list from new orders

Checking Stock Level

The stock level details can be retrieved by using the '*lowitems.php'* file to lookup the store_items table and return all quantities less than or equal to the minimum stock level. Again the result will be displayed on screen or the user can print out a hard copy. The SQL statement for this is:

```
$get_lowitems = "select item_id, item_title, item_desc, item_price, qty_instock
from store_items where qty_instock<=item_minlevel order by qty_instock";
$get_lowitems_res = mysql_query($get_lowitems ) or die(mysql_error());
```

Retrieving Lists of Debtors

This is achieved by clicking on debtors button, which then looks up all customer accounts and retrieve payment outstanding over a specified period of time. This implementation was achieved by concatenating relevant tables within the database as shown below:

```
$get_debtors = "select a.customer_id, a.outstanding_balance, c.first_name, c.surname,
c.tel_no, email from customer_account as a inner join customer as c on
a.customer_id=c.customer_id where outstanding_balance>0";
```

Retrieving Products and Customer Details

This can be done by login onto the Inventory Manager system and selecting the appropriate option to view its details. The system uses the file '*advview.php'* to generate views for all the table. The file '*view.php'* is used for viewing individual customer or product records.

6.0 System Testing

The purpose of the testing procedures used throughout the system is to ensure that the system functions as intended by the developer. The processes of the testing phase partly occurred during the system's implementation. The idea is to allow, individual system software modules to be assessed as well as overall system functionality. This Chapter discusses the use of realistic sample data, testing methods undertaken and their results.

6.1 Sample Data Deployment

The sample data was obtained from data stored on the current manual system. Existing customers details, items details and the new pending orders were used to populate the system's database. In addition, each order has more than one order-lines associated to it. This allowed for the system's functionality to be examined and help simulate a real world environment by shopping online and entering appropriate data onto fields and selecting items from the database.

6.2 Software Module Testing

The purpose of Unit Testing is to ensure that each software module in the system functions properly as specified in the requirements. This testing strategy was applied to each module in the system. These test procedures ensured that valid input data was accepted and invalid data was rejected. Each unit within the application was tested to verify that all links and buttons navigated as expected. Usability testing has also been incorporated to examine whether error messages are clear and understood.

The database schema was also monitored to ensure insertions, updates and deletes were occurring and with the expected changes. The results for software module Testing can be found in Appendix F. All the tests have proved successful. For those tests, which failed initially, the causing factor was identified and remedial action was taken in order to pass the test concerned. The detail of remedial action taken has been documented in Appendix F.

6.3 Black Box Testing

The purpose of Black Box or functional testing is to assess the system's internal workings. However the overall aim of this test strategy is to examine results without knowing how the system arrived at that result. In effect, a tester only requires knowledge of the system specification rather than underlying architecture. This caters for such testing to be performed from an end user's perspective rather than a Designer's perspective. Furthermore, any ambiguities that may exist between the Black Box test results and the original specification are easily detectable, as an unexpected output would occur. The results of Black Box testing can be found in Appendix G. The tests performed here are based around a user's input and actions. The expected and actual system output for each test case has been documented.

6.4 White Box Testing

The purpose of White Box testing is to certify that the underlying system architecture functions correctly. This contrasts to Black Box testing, which examines system output from knowledge about the system's use of syntax. White Box testing has been undertaken for this system, in order to examine the changes of state for each of the database tables within the system. White Box testing has two immediate benefits.

Firstly, by examining the table states during update, create and delete processes for example, the developer is able to verify that the right tables are being queried or affected. Furthermore, the developer can query the record contents, to ensure the correct and relevant fields in one or more tables have been correctly queried or affected, as appropriate. The second benefit is more aimed at the client; as such testing can simulate

system stress that it may endure once deployed to see how it reacts. The results of the White Box testing can be found in Appendix H.

6.5 System Integration Testing

The purpose of performing Integration Testing is to ensure that the different parts that make up the full system function correctly, when combined to form a single working application. Integration Testing was performed at the end of each phase of the development to prove that the system still maintained functionality. The benefits become more apparent as the system increases in size and functionality. For example, at the completion of the Online Ordering System and the Inventory Management System were two separate systems (at that point in development). However, the addition '*checkout.php'* in the Ordering System relies on successful integration with the Inventory Management System by automatically populating some database tables within Inventory Management System. For such reasons, testing occurred at each phase's completion to verify that new additions have not affected existing functionality. Navigation from one part of the system to another was also examined, by testing the links within the system's sidebar menu (the Quick Views). The implementation of CSS helps to uphold a consistent layout format as mentioned in Chapter 5 for each page in the system. This was also examined. The result from the integration testing can be found in Appendix J.

6.6 User Acceptance Testing

User acceptance testing was conducted at the end of the implementation phase in order to determine that the system had met the requirements defined in Chapter 3, the shop manager and the developer has gone through a series of evaluation, which involves the developer demonstrating the software and the shop manager using it briefly. The results of the testing procedures can be found in Appendix K.

7.0 Evaluation

This Chapter deals with the evaluation of the entire project by starting with the assessment of how the minimum requirements were achieved.

The project will then be evaluated against the functional requirements defined by the user, this will verify whether the project meets the needs of the user by solving the problems as specified at the beginning. This will then follow by the evaluation of the system's usability. This is because poor usability was one of the major drawback of the previous system. In addition, methodology used in the project will be evaluated to determine whether project management was carried out effectively. And finally, the system will be compared against similar existing systems, to enable the evaluation of the quality of the system and also help conclude whether the development of this system was worthwhile.

7.1 Meeting the Minimum Requirements

Develop a prototype web-based ordering system.

The Design and Implementation Chapters of this report document how the web-based order system was designed and implemented. Section 5.3 Figure 5.2.2 shows the interface prototype of the ordering system. Detailed screen dumps of the ordering system can be

found in Appendix I. Following the development of the system's prototype, implementing other function to enhance the system was a clear-cut. Iterative development approach through the use of PHP scripting language and cascading style sheets were appropriate helps promote consistency of the system. However, as the development progresses, and further functions implemented, the system became more and more complex in structure.

The web based ordering system should enable customers to browse stock availability.
The prototype ordering system implements the functionality to allow for the browsing of stock and checking of stock availability. Figure 7.1 below depicts this action in a screen shot taken from the ordering system.

Figure 7.1 Browsing items online

Develop a prototype inventory management database.
The inventory management system has been developed and implemented beyond the prototype specifications. Figure 5.2.1 in section 5.3 of this report depict the screen dump of the inventory management system. A detailed documentation of the implementation and functionality of the inventory management system can be found in Chapter 5 - (implementation) - of this report.

The database should be able to store and retrieve stock information.
The database has been implemented such that it is able to stock as well as retrieve stock (item) information. Figure 5.6.1 in Chapter 5 shows the retrieval of stock information and Figure 5.6.0 shows the stored inventory data.

The database should be able to store and retrieve customer details
Figure 7.2 below shows the storage of customer details in the database. The retrieval of customer details has been implemented and can be achieved through the use the 'view' button to the right hand side of the table. Details of this process can be found in section 5.6.1 in Chapter 5 of this report.

Figure 7.2Customer records

The database should be able to store and retrieve DVDs details.

The database is also able to store and retrieve DVD details, from the diagram above, Figure 7.2. It is clearly shown on the menu items the lists of DVD records stored on the database also see Figure 7.3 below.

Figure 7.3 stored/retrieved DVDs

7.2 Exceeding the Minimum Requirements

Figure 7.4 below shows the list of enhancements that were implemented

No.	Functionality
1	The ability to be able to list stocks that are running low.
2	The functionality to allow for the creation of customers account on the inventory management system.
3	The ability for the web based ordering system to be able to take and process customer orders.
4	The functionality for the ordering system to be able to generate receipts with order reference.
5	The ability for the inventory management system to integrate the functionality to loan DVDs to customers.
6	The functionality to allow customers do quick shopping.
7	The ability to generate picking lists for customer orders.
8	The implementation of user login and online help for the Inventory Management System.
9	The provision of a user manual.
10	The provision of online help for the ordering system.
11	The functionality to list debtors and their details.
12	The functionality to list overdue DVDs that has not been returned.

Figure 7.4 System enhancements.

7.3 Future System Enhancements

7.3.1 Customer Online Accounts Creation.

The ability for the system to allow customers to create their own online accounts would further improve efficiency and delegate some of the back office information processing to the customer thereby freeing valuable amount of time for the staff to concentrate more on other tasks within the store.

7.3.2 Improved Administrator Features

At present, the store manager has to use the tool called MySQL/PHP Administrator platform –available via the hosting company (streamlinenet)-, to perform a backup or restore of the database's contents, a more user friendly interface could be built in to the system. A useful feature here would be allowing Administrators to perform a back to a specified file location at the click of a button.

7.3.3 Automatically Generate Purchase Orders

Another useful feature for the inventory management system's user would be to quickly generate a purchase order from the system by referencing all items running bellow minimum stock level. This can be achieved by storing suppliers details along with each product, the idea would be that the system will be able to obtain suppliers information relating to any item running low and then use that information to automatically produce purchase orders.

7.3.4 Customisable Interface and Page Layout

Implementing the functionality to allow users to be able to sort results by any column heading by clicking on that column would be beneficial and improve efficiency as data are currently displayed in a predefined sorting format. Furthermore, allowing users to customise page layout and colour schemes improves overall system usability for distinct user.

7.4 Evaluation of Chosen Methodology

As mentioned in Chapter 2, the chosen project methodology was a variation of the Software Development Life Cycle. This catered for the implementation and testing phases to be iterated using an iterative approach. This approach proved useful in these phases, as debugging the system during the process of implementation was probably less costly than finding errors at the end of development. However, the analysis and design phases were examined to ensure that all the requirements were being catered for. It may have been better to also iterate these phases to ensure that requirements were met while possibly saving more time for development.

The original and revised Gantt charts are illustrated in Appendixes B and C respectively. Initially, a time period was dedicated to the feasibility study phase, however, it was later decided by the developer that this would consume more project time and therefore it would be more appropriate to include the feasibility stage with the analysis phase of the project thereby saving some much needed development time.

7.5 Evaluation of Chosen Technologies

PHP scripting language was considered to be far easier during the implementation phase and the developer felt PHP script were a superior choice, particularly when connecting to MySQL.

The Apache/PHP MyAdmin interface catered for connectivity between the MySQL database and PHP views. MySQL database allowed for all the requirements to be fulfilled. However, MySQL does not currently support views. Therefore this downfall required the explicit creation of displaying particular fields each with a specified alias and resulted in rather complex queries.

To ensure consistency throughout the system, CSS implementation allowed all pages to adhere to the same textual and layout format. Changing the properties in a single CSS file meant changes were global to the entire system and therefore timesaving.

7.6 Evaluation of Design Stage

The preliminary database schema needed to be readjusted and further entities were identified and introduced were necessary as the development progresses. Most entities were further decomposed and normalised to an appropriate degree.
The usability design also had to be reworked to adhere to the defined usability issues identified in Chapter 2 and cater for varied users such as experts and novices.

7.7 Evaluation of Implementation

The process of implementation was carried out effectively and thoroughly which resulted in the delivery of an effective solution. In the initial implementation, installation and configuration of the required software applications (Apache, PHP and MySQL) on the developers computer was one of the problematic task. And also the creation of the shopping cart for the online ordering system was problematic. This required successful operation between, the MySQL database, PHP engines and Apache webserver. The implementation was adhered to a rapid and iterative development approach of the system as it was divided into sub-phases, which helped the developer concentrate on one phase at a time. This ensured that the developer could test the system at each stage and ensuring that all requirements were met before developing the next phase of the system.

7.8 Usability Evaluation

A number of potential users have been tasked to evaluate the online shopping system and also the staff and the manager at Adonai shop were asked to evaluate the inventory management system. Each user had the same set of tasks to perform under the new system.

The tasks undertaken by the nominated users were to create a new order, updating an existing customer/item data on the database, creating a new account for customer and searching for records.

Create Customer Account

User	**Time in old system (mins)**	**Time in new system (mins)**
Mr Japhet (shop manager)	4.07	3.10
Florence (shop assistant)	5.04	4.12

Tasks involves recording customer details, retrieving accounts number /customer ID.

Issue DVD to Customer

User	**Time in old system (mins)**	**Time in new system (mins)**
Mr Japhet (shop manager)	2.09	2.11
Florence (shop assistant)	2.05	3.08

Task involves recording borrowers/DVD details

Update/Editing Records

User	Time in old system (mins)	Time in new system (mins)
Mr Japhet (shop manager)	6.10	1.05
Florence (shop assistant)	8.06	2.04

Tass involves retrieving appropriate records editing it and save.

Searching Records

User	Time in old system (mins)	Time in new system (mins)
Mr Japhet (shop manager)	5.09	0.09
Florence (shop assistant)	6.02	0.09

Task involves searching for a specific record or data and displaying the results.

Online Shopping

User	Time in old system (mins)	Time in new system (mins)
Monica (customer)	N/A	4.10
Lennox (customer)	N/A	4.04
Livingstone (customer)	N/A	3.50

Task involves purchasing a maximum of two items and checking out.

Creating customer account produced slightly varying results compared with the current system whereas updating an existing record showed faster results compared to the current system and searching for records was significantly faster and straight forward process in the new system.

It is anticipated that more use of the system will increase operational familiarity, as far as users are concerned. Therefore the time taken to process orders and update customer account and issue DVDs should decrease over time with more system usage.
Users provided verbal feedback of the system after the test procedures. All users were generally pleased with the system's look and feel and that features were highly specified and easy to master.
The shop manager liked the idea of being able to view and manage details away from the premises when required as access to the system is via http (Internet). Although when the system was demonstrated via a URL (www.adofood.com), the system was slightly slower, taking nearly 5 seconds extra to respond to some actions. However, the response time depends on the speed and type of connection of the user's Internet. Results shown here were demonstrated using ADSL broadband Internet connection.
The shop manager appreciated the ability for the system, to automatically generate order references, although, these references are set to automatically populate appropriate tables within the database system, which will reduce the amount of entries staff had to perform, thereby also reducing input errors and validations and also improve efficiency and accuracy of stored data. The ability to calculate orders total automatically means no more complicated amendments on excel spreadsheets.

Overall the client was impressed with the system's features and is eager to commission the system as soon as possible.

7.9 Comparisons to Alternative Systems

Chapter 3 of this report details the analysis phase of this project of which two existing products where identified, namely: OrderWise and StockIt! Systems. Figure 7.9 below shows a comparison between the two systems against the new Adonai Inventory Management and Online Customer Ordering System solution. The table clearly identifies that although the new system does not offer all the functionality that a combination of OrderWise and StockIt! Systems offer, it must be distinguished that Adonai system is tailored to meets the requirements of Adonai's business.

Functionality	Adonai Systems	OrderWise	StockIt!
Stock control	Yes	Yes	Yes
Issue DVDs	Yes	No	No
Create customer accounts	Yes	No	No
Web based ordering capabilities	Yes	No	No
Process orders (in house)	Yes	Yes	Yes
Login/Password protection	Yes	Yes	Yes
Remote access via Internet	Yes	Yes	Yes
Maintain customers and debtors information	Yes	No	No
Minimum stock levels and order thresholds	Yes	Yes	Yes
Provide sales statistics	No	Yes	Yes
Produce Marketing / promotional materials	No	Yes	Yes

Table 7.9 Comparison with existing systems

7.9.1 Conclusion

The development of the new system has helped improve the short-falls of the existing systems by implementing most of the OrderWise and StockIt! Systems functionalities together with the requirements of Adonai's business, which then makes a complete system fully customised to suit the business process. Therefore it would be concluded that the development of this system is a success and was worthwhile.

References

[1]. Avison, D.F., G., Information Systems Developments: Methodologies, Techniques &Tools, 2nd Edition, 1995: Mc Graw Hill.

[2]. Chrisanthi Avgerou and Tony Cornford Developing Information Systems: Concepts, Issues and Practice, 2nd edition. 1998: MacMillan.

[3] D.J. Tudor & I.J. Tudor Systems Analysis and Design: A comparison of structured Methods, 1995: NCC Blackwell.

[4]. Bob Hughes and Mike Cotterell, Software Project Management, 3rd edition. 2002: Mc Graw Hill.

[5] C.J. DATE, An Introduction to Database systems, 7th edition, 2000: Addison Wesley.

[6] http://www.devx.com Open Source Database Feature Comparison Matrix.

[7] Thomas Connelly, Carolyn Begg and Anne Strachan, Database Systems: a practical approach to design, implementation and management. 2nd edition, 1998: Addison Wesley.

[8] Microsoft.com (Cited 22nd November 2005): available here http://www.microsoft.com/sql/howtobuy/default.mspx

[9] http://www.shop-script.com/glossary.html#m (Cited 29th November 2005)

[10] Wikipedia.com Encyclopedia (cited 02nd December 2005) http://en.wikipedia.org/wiki/Server-side_scripting

[11] PHP.NET (cited 02nd December, 2005) http://uk.php.net/manual/en/intro-whatcando.php

[12] Java.sun.com (cited 02nd December, 2005) http://java.sun.com/products/jsp/

[13] Macaizek, L. Requirments Analysis and System Design. 2001: Addison Wesley.

[14] John Ward and Joe Peppard, Strategic Planning for Information Systems third edition, 2002: John Wyley & Sons, LTD.

[15] Davis & Benamati, E-commerce Basics (technology foundations and e-business applications), 2003: Addison Wesley.

[16] Requirements in Software Engineering. Available here John Mylopoulos , Lawrence Chung , Stephen Liao , Huaiqing Wang , Eric Yu, Exploring Alternatives During Requirements Analysis, IEEE Software, v.18 n.1, p.92-96, January 2001 .(cited 10th December, 2005).

[17] Allan Dennis, Babara Haley Wixom, System Analysis and Aesign, second edition 2003: Wyley.

[18] Thomas Connolly and Carolyn Begg, database systems, a practical approach to design, implementation and management. Third edition, 2002: Addison Wesley

[19] Elmasri and Navathe, Fundamentals of Database Systems, 4th edition 2003: Addison Wesley

[20] Shackle & Richardson, Human Factors for Informatics Usability: Cambridge University Press 1991.

[21] Jakob Neilsen's Ten usability heuristics. 2002 [cited 04th February 2006]: Available from: http://www.useit.com/papers/heuristic/heuristic_list.html

[22] CSS tutorial, http://www.w3schools.com/css/default.asp (cited 05th februry 2006).

[23] C.J. Date, Hugh Darwen, Nikos A. Lorentzos, Temporal Data and the relational model, 2003: Morgan Kaufmann.

[24] Julie C Meloni, PHP, Mysql and Apache, second edition, 2005: SAMS.

[25] Vikram Vaswani, how to do everything with PHP & MYSQL, 2005: McGrawHill.

[26] Atkinson, L., Core MySQL, The Serious Developer's Guide. 2002: Prentice Hall.

[27] Sommerville, I., Software Engineering, 6th Edition, 2001, Boston: Addison-Wesley.

[28] Jennifer Niederst, Webdesign in a Nutshell, A Desktop Quick Reference, Second Edition, 2001, O'Reilly & Associates, Inc.

[29] Paul Bocij, Dave Chaffey, Andrew Greasley & Simon Hickie, Business Information Systems, 3rd Edition, 2006: Prentice Hall.

[30] G. Winfield Treese, Lawrence C. Stewart, Designing Systems for Internet Commerce, Second Edition, 2002: Addison Wesley.

[31] Donald Yeates and Tony Wakefield, System Analysis and Design, Second Edition, 2004: Prentice Hall.

[32] Alan Dennis, Barbara Haley Wixom, Systems Analysis Design, Second Edition, 2003: John Wiley & sons.

Appendix A

Project reflection

At the beginning of the project I thought I had a clear idea of what I need to do in order to produce the software solution for the client. I totally underestimated the workload involved in the development. I designed a Gant chart and worked steadily as scheduled on the Gant chart, but when I had my mid project meeting with my project supervisor and assessor, I soon realised that the Gant chart I was working with is not as realistic as I had thought when I was advised by my assessor and supervisor that I was somewhat "8 weeks behind" on the entire project. I quickly had to readjust and accelerated the pace of the project. My advice to future students is not to underestimate or take for granted the amount of time required in software development and beware project development may take longer than originally anticipated as there may be some unforeseen obstacles that could be encountered such as, debugging codes and or something is not working on the code.
Here goes the old saying "earlier the better". I would advise all future students undertaking similar project to start at the earliest possible time, do not make the same mistake I made by underestimating the amount of time required as this project is very time consuming. After the mid project meeting with my supervisor and assessor, I had to work late night and on average 14hrs a day in order to complete the project.

Following the completion of all the necessary preparation, I began to implement a prototype of the system and then rapidly implementing all the required functionalities by applying knowledge and concepts acquired from the School of Computing modules including but not exclusive to Database Principles and Practice (COMP2400), Software project Management (COMP2660), Information Systems Strategy (IS31) and Internet Systems Technologies (SY23). Doing this project has not only assured these skills but has allowed me to develop new skills in both technical and non-technical areas as well as gain real world experience in solving business problems.

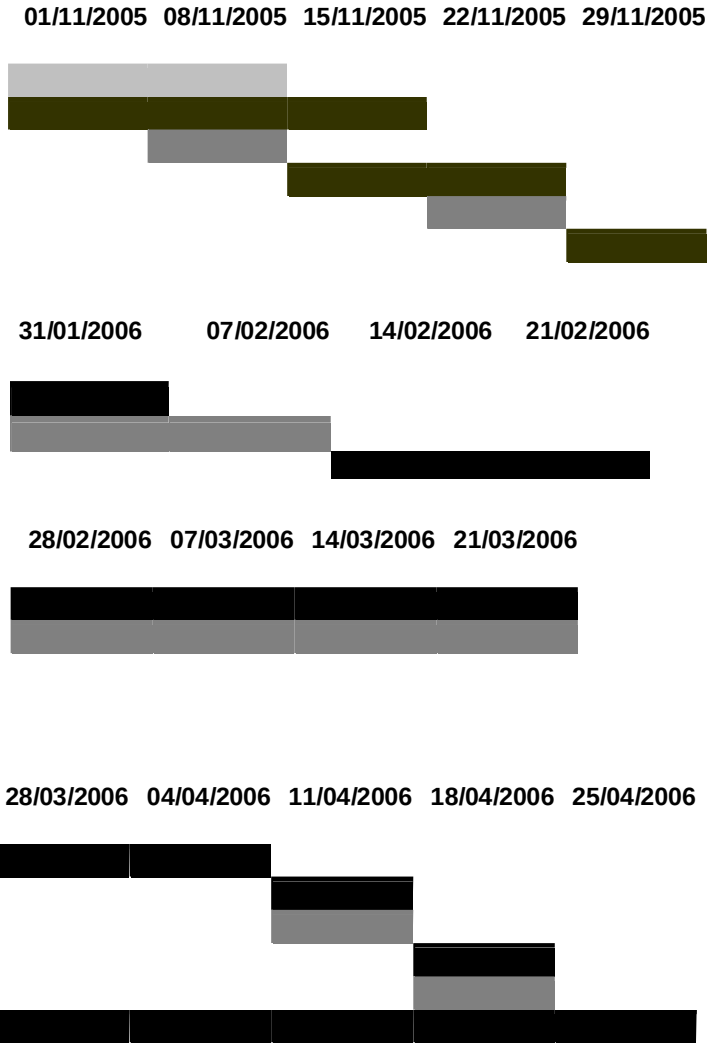
Most of the interviews I conducted during the gathering of the requirements were rather informal. However, there were some occasions where I had to present myself in a professional manner and communicate as a Computing Professional this enabled the client to have more faith in me and my proposed software solution. My advice to future students who may want to conduct interviews in the future is to be prepared and have a basic understanding of business environment, processes and practices and use less technical jargons as this would be irrelevant to business people and could probably bore them to the point where the interviewer will not be able to collect all the necessary information required.

If I have the opportunity to do the project all over again, I would be more disciplined in terms of time management and literature research as I placed a huge emphasis on some phases, which left me with little time to spend on others. My advise to future students would be to pay detailed attention to the preliminary background reading before making decisions with regards to their choice of technologies and methodologies to use. This is because, by the time I realised that there were problems with my chosen technologies, it

was too late to alter it, I had to deal with all the consequences that came with it and also this can be frustrating for the developer and time consuming.
Finally, my opinion is that this project is very challenging and if any future student could pull it off, it is also equally rewarding as I have learnt a great deal in software development, time management and application of various methodologies to solving real world business problems.
This project has enhanced my computing skills portfolio plus it looks good on the CV as well.

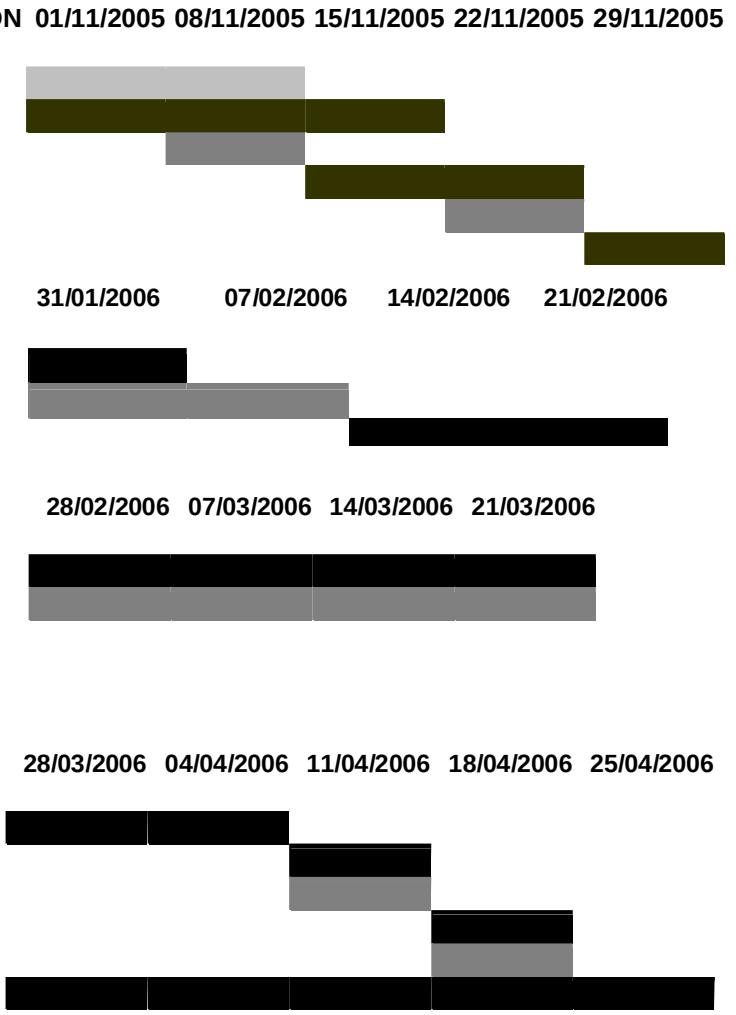
APPENDIX (B) PROJECT SCHEDULE (INITIAL GANT CHART)

No	TASK	START	FINISH	DURATION
1	Problem definition, aims and requirements	01/11/2005	14/11/2005	2 wks
2	Meetings with Adonai manager	01/11/2005	21/11/2005	3 wks
3	Interview with Shop assistants	08/11/2005	14/11/2005	1 wk
4	Methodologies & technologies research	15/11/2005	28/11/2005	2 wks
5	Finalise methodoly and technology	22/11/2005	28/11/2005	1 wk
6	Finalise mid-project report	29/11/2005	05/12/2005	1 wk
7	Conduct feasibility study	31/01/2006	06/02/2006	1 wk
8	System investigation and analysis	31/01/2006	13/02/2006	2 wks
9	System design	14/02/2006	27/02/2006	2 wks
10	Implementation: The inventory manager	28/02/2006	27/03/2006	4 wks
11	Implementation: The ordering web-site	28/02/2006	27/03/2006	4 wks
12	System testing & Integration	28/03/2006	11/04/2006	2 wks
13	Write up a User Manual	11/04/2006	17/04/2006	1 wk
14	Write up technical documentation	11/04/2006	17/04/2006	1 wk
15	Deploy system	18/04/2006	24/04/2006	1 wk
16	Evaluate system	18/04/2006	24/04/2006	1 wk
17	Write-up Final Report	28/03/2006	01/05/2006	5 wks



APPENDIX (C) PROJECT SCHEDULE (FINAL GANT CHART)

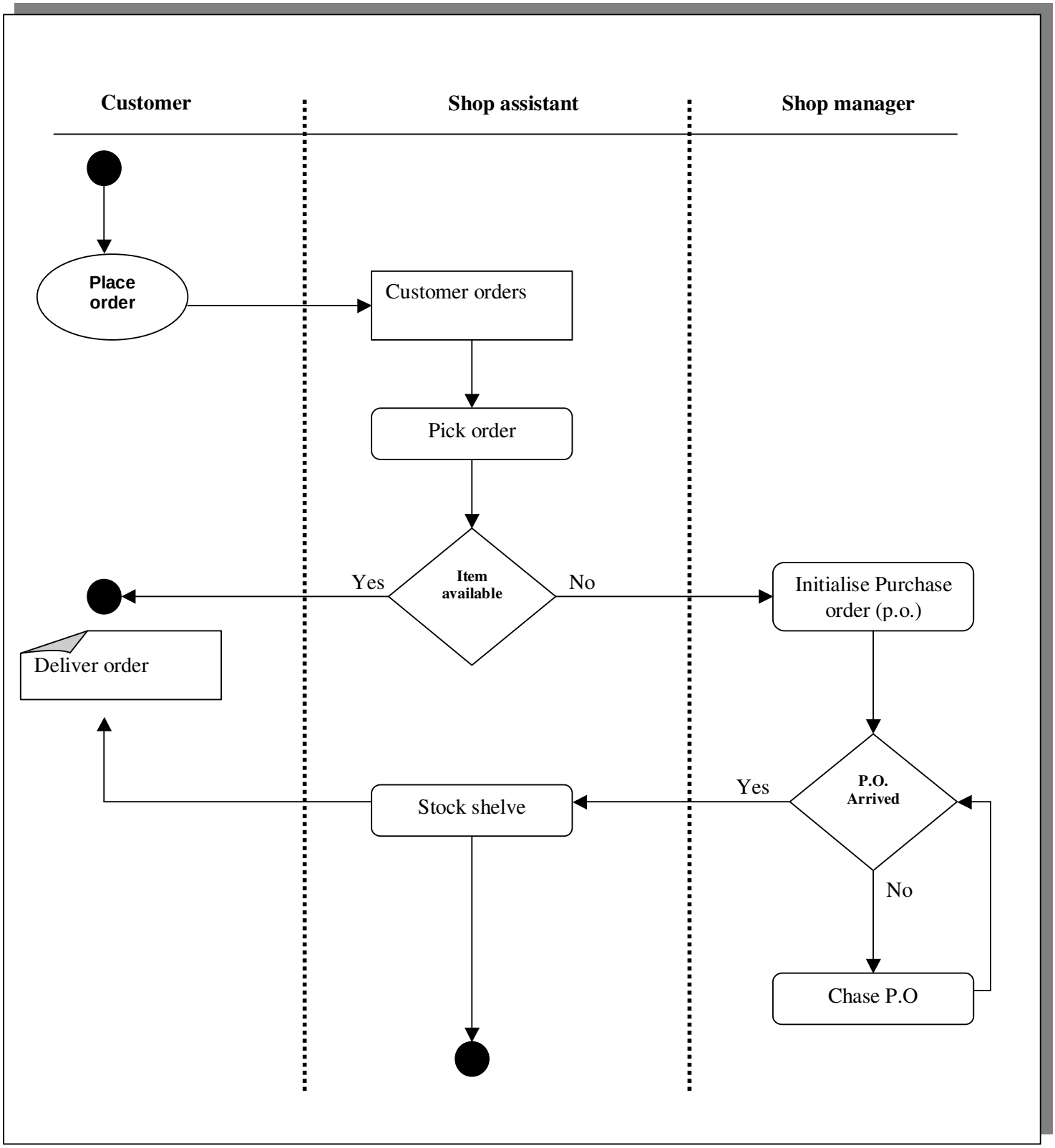
No	TASK	START	FINISH	DURATION
1	Problem definition, aims and requirements	01/11/2005	14/11/2005	2 wks
2	Meetings with Adonai manager	01/11/2005	21/11/2005	3 wks
3	Interview with Shop assistants	08/11/2005	14/11/2005	1 wk
4	Methodologies & technologies research	15/11/2005	28/11/2005	2 wks
5	Finalise methodoly and technology	22/11/2005	28/11/2005	1 wk
6	Finalise mid-project report	29/11/2005	05/12/2005	1 wk
* 7				
8	System investigation and analysis	31/01/2006	13/02/2006	2 wks
9	System design	14/02/2006	27/02/2006	2 wks
10	Implementation: The inventory manager	28/02/2006	27/03/2006	4 wks
11	Implementation: The ordering web-site	28/02/2006	27/03/2006	4 wks
12	System testing & Integration	28/03/2006	11/04/2006	2 wks
13	Write up a User Manual	11/04/2006	17/04/2006	1 wk
14	Write up technical documentation	11/04/2006	17/04/2006	1 wk
15	Deploy system	18/04/2006	24/04/2006	1 wk
16	Evaluate system	18/04/2006	24/04/2006	1 wk
17	Write-up Final Report	28/03/2006	01/05/2006	5 wks



Note: (*) Number 7, feasibility study eliminated from the final project report

APPENDIX D

A UML Activity diagram showing the current method of processing customer orders



APPENDIX E

----------> SQL Files <----------

```
create table customer (
Customer_id int not null primary key auto_increment,
first_Name varchar (30) not null,
Surname varchar (20)not null,
Address1 varchar (50) not null,
Address2 varchar (40)not null,
city text,
PostCode varchar (10) not null,
Tel_No varchar (25),
email varchar(40)
);


create table Invoice (
Invoice_No int not null primary key auto_increment,
Order_id bigint (20) not null,
Customer_id int not null,
Total_amount float not null,
Invoice_Date date not null,
FOREIGN KEY (order_id) REFERENCES store_orders (order_id),
FOREIGN KEY (customer_id) REFERENCES customer (customer_id)
);

create table Customer_Account (
Account_No int not null primary key auto_increment,
Customer_id int not null,
Credit_Limit float not null,
Outstanding_balance float not null,
Order_id bigint (20) not null,
FOREIGN KEY (customer_id) REFERENCES customer (customer_id),
FOREIGN KEY (order_id) REFERENCES store_orders (order_id)
);

create table DVD (
Dvd_id varchar (10) not null primary key,
title text not null,
description text
);

create table Borrow (
Customer_id int not null,
dvd_id varchar (10)not null,
date_borrowed Date not null,
```

```
due_date Date not null,
fine float,
returned_yes_no text not null,
FOREIGN KEY (customer_id) REFERENCES customer (customer_id),
FOREIGN KEY (dvd_id) REFERENCES dvd (dvd_id)
);

create table employee (
Emp_id int not null primary key auto_increment,
first_name text not null,
surname text not null,
address1 varchar (40) not null,
address2 varchar (40) not null,
city text,
post_code varchar (10)not null,
tel_No varchar (25)
);
```

----------**Store Front Tables**----------

```
CREATE TABLE store_categories (
cat_id int not null primary key auto_increment,
cat_title varchar (50) unique,
cat_desc text
);

create table store_items (
item_id int not null primary key auto_increment,
cat_id int not null,
item_title varchar (75),
item_price float (8,2),
item_image varchar (50),
item_desc text not null,
item_minlevel int not null,
qty_instock int not null,

FOREIGN KEY (cat_id) REFERENCES store_categories(cat_id)
);

create table store_item_size (
item_id int not null,
item_size varchar (25),
FOREIGN KEY (item_id) REFERENCES store_items (item_id)
);
```

---------->Shopping Basket Tables <----------

```
create table store_shoppertrack(
shop_id int not null primary key auto_increment,
session_id varchar (32),
sel_item_id int,
sel_item_qty smallint,
sel_item_size varchar (32),
date_added datetime
);

create table store_orders(
order_id int not null primary key auto_increment,
order_date datetime,
order_name varchar (100),
order_address varchar (255),
order_city varchar (50),
order_postcode varchar (10),
order_tel varchar (25),
order_email varchar(100),
item_total float (6,2),
status enum('processed', 'pending')
);

create table store_orders_items(
order_item_id int not null primary key auto_increment,
order_id int,
sel_item_id int,
sel_item_qty smallint,
sel_item_size varchar(25),
sel_item_price float(6,2),
FOREIGN KEY (order_id) REFERENCES store_orders (order_id)
);
```

APPENDIX F

UNIT TESTING

1). Inventory Management System

MENU BUTTONS AND QUICK VIEWS

Test	Expected Result	Actual Result	Comments
Click dvd button from sidebar menu.	List of all the dvd id, title and description in dvd table (non-editable data).	As expected	Test successful
Click dvd_history button from sidebar menu	List of all the dvd transactions to date including CUSTOMER_ID, DVD_ID, DATE-BORROWED, DUE_DATE, FINE, RETURNED(Y/N) in borrow table (non-editable data).	As expected	Test successful
Click account button from sidebar menu	List of all customer accounts details including ACCOUNT_NO, CUSTOMER_ID, CREDIT LIMIT OUTSTANDING BALANCE, ORDER ID from customer_accounts table (non-editable data).	As expected	Test successful
Click on New_orders button from sidebar menu	Generate picking lists for all new and pending orders by listing the following data order_id item_id item_title item_desc qty price order_date name address telephone (non-editable data).	As expected	Test successful. Although the initial test failed and corrective actions where taken to rectify the query.
Click on product button from the sidebar menu	List all the inventory records including ITEM_ID CAT_ID TITLE PRICE DESCRIPTION MIN_STOCK_LEVEL QUANTITY IN STOCK in the store_items table (non-editable data).	As expected	Test successful. Although the initial test failed and corrective actions where taken to rectify the query.
Click on the Reports button from the sidebar menu	Display the menu page for report creation with different report buttons available	As expected	Test successful
Click on the employee button on the sidebar menu	List of all employees details including EMPLOYEE_ID FIRST NAME SURNAME ADDRESS_1 ADDRESS_2 CITY POST-CODE TEL-No in the employee table (non-	As expected	Test successful

	editable data).		
Click on the home button from the sidebar menu	Display the welcome interface	As expected	Test successful
Click on help button from the sidebar menu	Display help page (non-editable data).	As expected	Test successful

SYSTEM LOGIN

Test	Expected Result	Actual Result	Comments
Click maintenance menu button on the sidebar menu	Display login page with username and password box provided	As expected	Test successful
Enter invalid or no username and click login button	Display error messages advising Login fail	As expected	Test successful
Enter invalid or no password and click login button	Display error messages advising Login fail	As expected	Test successful
Enter a combination of invalid username and password	Display error messages advising Login fail	As expected	Test successful
Enter a valid username and password and click login	Display login ok and grant access to the system	As expected	Test successful

BORROWED DVDS TABLE
VIEWING, SEARCHING, ADDING, EDITING AND DELETING RECORDS

Test	Expected Result	Actual Result	Comments
Click on borrow link on the maintenance menu	Display the records from borrowing table, display the search box and the add record form.	As expected	Test successful
Click on the link 'view' on the right hand side of the table	Display the corresponding record on a single row with all the details	As expected	Test successful
To search for record, enter id, or a wildcard caracter	Display result that matches the input in the box	As expected	Test successful

in the search box			
Adding new dvd for loan, enter a valid Customer_id. dvd_id, date_borrowed, due_date, returned_yes_no	Insert a new record onto the borrow table and display successful insert message	As expected	Test successful
enter invalid date_borrowed, due_date,	Display error message and advise invalid input	As expected	Test successful
Click on the Edit link to edit record	Retrieve the corresponding dvd on loan details and displayed in a editable form for editing	As expected	Test successful
Alter records on the editable form and click 'edit' button	Update record and display message advising successful editing	As expected	Test successful
Viewing a recored, click on the view link on the right hand side of the table	Delete record and display message on screen advising delete successful	As expected	Test successful

Note: All the order tables used the same php file, which was implemented globally for adding new records, editing records, searching records and viewing records, and therefore the same testing procedures that was applied to the table 'Borrow' was applied to all other tables, which includes customer, dvd, customer_accounts, employee, store_items, store orders and store_order items.

APPENDIX G

BLACK BOX TESTING

1). Inventory Management System

Test	Expected Result	Actual Result	Comments
Open a web browser and enter the URL	Welcome page shown with company logo with buttons on the sidebar menu	As expected	Test successful
Login via the maintenance menu button	Display the main menu listing all the available table on the system	As expected	Test successful
Attempt to work with tables after session timer times out.	Advise user to re-log onto system	As expected	Test successful
Attempt to navigate to a page without logging onto system	Redirect user to login menu	As expected	Test successful
View Reports	Display the reports creation page with all links and buttons	As expected	Test successful
Create a New customer and customer_account	New accounts and customer creation procedures executed and New customer and accounts information shown in all Individual appropriate views	As expected	Test successful
Edit customer or customer accounts	Edit procedures executed and edit evident for those fields shown in view all customers or customer accounts	As expected	Test successful
Delete Customer or customer accounts	Deletion procedures executed and delete evident for those fields shown in view all customer or customer accounts	As expected	Test successful
Issue dvd to customer	All borrowing processes procedures executed. Borrowing evident for those fields shown in view all	As expected	Test successful

	borrow table		
Process new orders	All order processing procedures executed, new orders evident for those fields shown in view all customer accounts, store_orders table and store_orders_items table	As expected	Test successful (initial test reveals a number of errors which were rectified accordingly)
Add new dvds	All insert procedures executed and new dvd evident in view all dvd	As expected	Test successful
Delete dvd	All delete procedures executed and deletion evident in view all dvd	As expected	Test successful
Add new employee	All insert procedures executed and new employee evident in view all employee	As expected	Test successful
Delete employee	All delete procedures executed and deletion evident in view all employee	As expected	Test successful
Add new items	All insert procedures executed and new item evident in view all store_items	As expected	Test successful
Delete item	All delete procedures executed and deletion evident in view all store_items	As expected	Test successful

2). Online Ordering System

Test	**Expected Result**	**Actual Result**	**Comments**
Open a web browser and enter the URL	Welcome page shown with company logo and appropriate buttons on the sidebar menu (non-editable data)	As expected	Test successful
Use the quick shopping button to open the form and submit shopping lists	Quick shopping form opens, record shopping lists and emailed to store (non-editable data)	As expected	Test successful
Start shopping	Opens the store categories page and list item categories (non-editable data)	As expected	Test successful
View items on mouse click on item category	Show lists of items under category with prices (non-editable data)	As expected	Test successful

Move to View details of selected item by clicking on item listed under category	Display item's image, description and price (non-editable data)	As expected	Test successful
Add item to shopping cart	Item added to shopping cart and shopping cart displayed and option given to continue shopping or check out. (non-editable data)	As expected	Test successful
Move to remove item from shopping cart	Item successfully deleted from shopping cart and evident shown in shopping cart. (non-editable data)	As expected	Test successful
Checking out: Invalid or no details entered on the check out form	Do not allow the user to proceed to check until details are entered on required fields of the form (non-editable data)	As expected	Test successful
Checking out: valid details entered on required fields of the check out form	Proceed to check out, generate order receipt with order reference number, total amount and a message for the customer (non-editable data)	As expected	Test successful (initial test failed and corrective actions where taken to rectify the fault with order reference and total price display)

APPENDIX H

WHITE BOX TESTING

1). Inventory Management System

Test	Expected Result	Actual Result	Comment
Log onto system	Execute config.php file to validate username and password and grant access if valid user.	As expected	Test successful
View help	Return help page	As expected	Test successful
View dvd	Query of dvd table and return results	As expected	Test successful
View dvd history	Query of borrow table and return results	As expected	Test successful
View accounts	Query of customer_accounts table and return results	As expected	Test successful
View orders	Query of store_orders, store_orders_items and store_items table and return results	As expected	Test successful
View product	Query of store_items table and return results	As expected	Test successful
View reports	Return report page menu	As expected	Test successful
View employees	Query of employee table and return results	As expected	Test successful
Create customer account	Insert into customer_account table	As expected	Test successful
Edit customer account	Update specified customer account fields for a given id value	As expected	Test successful
Delete customer account	Update field of customer_account table to -1	As expected	Test successful
Issue dvd for loan	Insert into borrow table	As expected	Test successful
Edit borrowings	Update specified borrow fields for a given id value	As expected	Test successful
Delete borrowing record	Update field of borrow table to –1	As expected	Test successful
Add new customer	Insert into customer table	As expected	Test successful
Edit customer	Update specified customer fields for a	As	Test

	given id value.	expected	successful
Delete customer	Update field of customer table to –1	As expected	Test successful
Add new items	Insert into item table	As expected	Test successful
Edit items	Update specified store_items fields for a given id value.	As expected	Test successful
Delete item	Update field of store_item table to –1	As expected	Test successful
Process orders	Auto populate store_orders table and store orders items,	As expected	Test successful

2). Online Ordering System

Test	**Expected result**	**Actual result**	**Comment**
Enter quick shopping	Display quick shopping page	As expected	Test successful
Submit quick shopping form	Send details to store for processing	As expected	Test successful
Start shopping	Display item category page, allow customer to browse for products, add items to cart, remove items from cart, calculate total amount for that order.	As expected	Test successful
Checking out	Display checkout form that must be filled in and on submit, generate order receipt.	As expected	Test successful

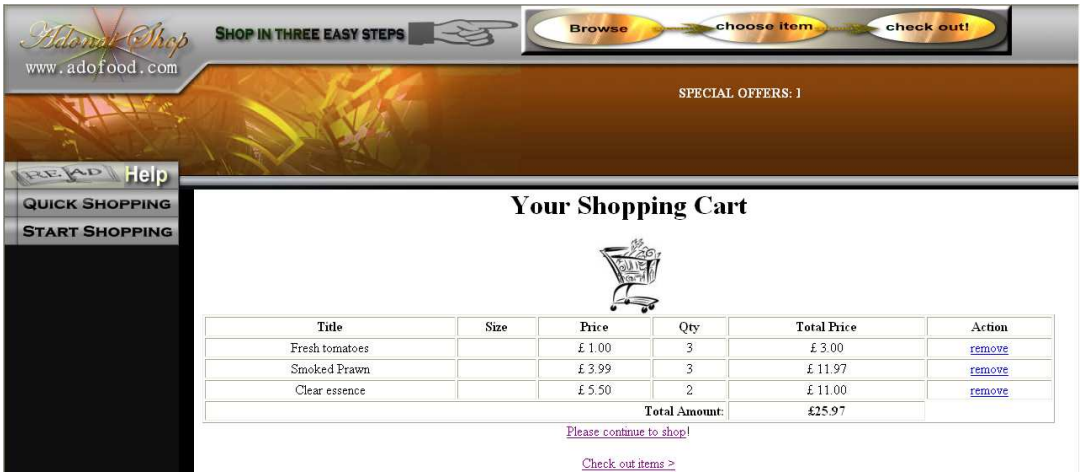
Appendix I

System screen dumps

Figure 1- Interface of the online ordering system

Figure 2- Viewing item under groceries categories

Figure 3- Viewing the shopping cart



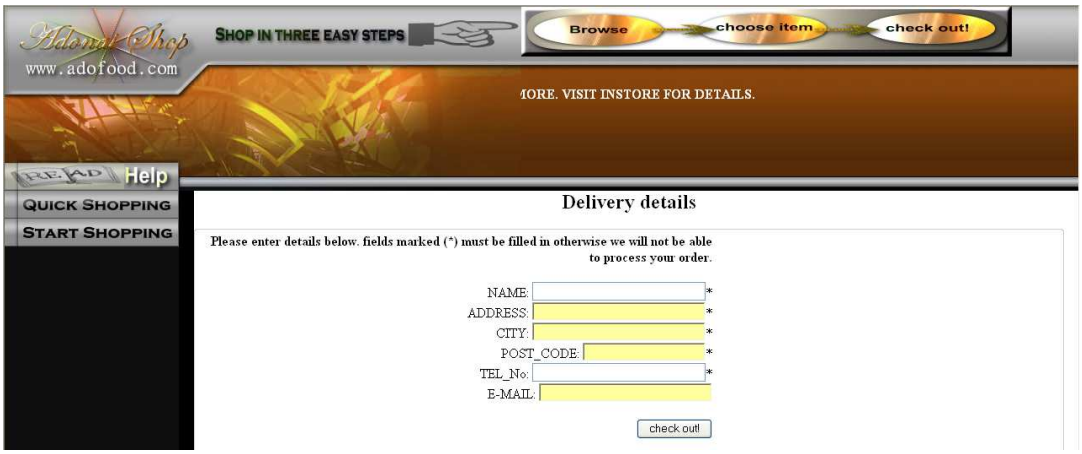


Figure 4- Checking out form

Appendix J

System Integration Testing

Integrating Inventory Management System with the Online Ordering system

Test	Expected result	Comment
File viewitem.php on ordering system to interact with inventory management system	Items retrieved from database and displayed on ordering system	Test successful
File checkout.php on ordering system to interact with inventory management system	Auto populate store_orders, store_orders_items tables on the inventory management system	Test successful

Global system integration using '.CSS' file

Test	Expected result	Comment
Default.css file to format all page layout within the inventory management system	Uniform colours and fonts displayed across all pages	Test successful

Integrating appropriate '.php and .html' files for the systems

Test	Expected result	Comment
File seestore2.php successfully linked to view item.php and showcart.php	Display items via link and show cart via link	Test successful
Showitem.php file successfully linked to addtocart.php	Provide option to add to add item to cart and showcart	Test successful
Showcart.php file successfully linked to checkout.html	Open checkout form	Test successful
Checkout.html successfully linked to checkout.php	Successfully generate order receipts and clear shopping cart on completion of shopping	Test successful

Integrating all buttons and links

Test	Expected result	Comment
Maintenance menu button successfully linked to login.php file	Display login page	Test successful
Quick view button successfully linked to appropriate pages	Display pages corresponding to appropriate button	Test successful

View, Edit and Delete links successfully linked to individual file	View performs view.php functions, Edit performs edit.php and Delete performs delete.php functions accordingly	Test successful

APPENDIX K.

USER ACCEPTANCE TESTING

Functional requirements	Result
The database shall be able to store and retrieve stock information such as, an item id, quantity in stock, price, minimum level and description	Test successful
The database shall be able to store and retrieve customer details such as The name, address, town, postcode, and their telephone number and id.	Test successful
The web based inventory system shall allow for the creation of customer accounts.	Test successful
The web based ordering system shall be able to generate order confirmation and receipts	Test successful
The web based ordering system shall allow customers to order items.	Test successful
The system shall store an order_id, item quantity, item code, description, unit price and total price	Test successful
The system shall provide efficient search facilities for locating stock, orders and customers	Test successful
The system shall automatically calculate the price that the customer must pay for an order	Test successful
The system shall allow the user to add, edit and delete customers, product and orders to and from the system.	Test successful
The system shall be able to produce reports	Test successful
The system shall allow for the issuing of dvd to customers	Test successful
The system shall record dvd details include, id, title and description	Test successful
The system shall record employee name, address, tel no, post code.	Test successful

Constraints

Storage of address entries to a maximum of two, excluding the Town, County and Postcode fields.	Test successful
The system should not allow an order to be confirmed if the order line/shopping basket is empty.	Test successful
Numeric integrity constraints will be enforced on fields, which must contain numeric data.	Test successful

Performance	Result
All user inputs must be validated within the client's browser to reduce error and increase throughput in processing of orders.	Test successful
The execution of user demands as rapidly and efficiently as possible.	Test successful
To be able to handle the anticipated volume of data, including throughput and storage.	Test successful

Maintenance

Allowing shop manager to undertake simple routine housekeeping tasks, such as database backing-up	Test successful
The system shall be scalable - it shall be able to grow with the business, allowing future updates and modifications to the system	Test successful
To have security features such as passwords where necessary for access to the system (Data Protection Act) and backup procedures.	Test successful

Usability	**Result**
To be able to operate on any platform Such as Microsoft Windows and Mac. Operating systems.	Test successful
To be able to operate within any major web browser, such as Internet Explorer, Netscape, Mozzilla and others.	Test successful
Use of uniform colours and text layout throughout the system, maintaining consistency within the Interface.	Test successful
Allow a user to cancel an action and navigate forward or backward within the application.	Test successful
System should have familiar and easy to use interface that can be learned in a short period of time by the user.	Test successful
System will help the client carry out the tasks and achieve their goals effectively and efficiently by using drop down lists where appropriate.	Test successful
System shall keep response times to user interaction to a minimum.	Test successful

APPENDIX L

LEVEL 1 DFD SHOWING THE INVENTORY MANAGEMENT/ONLINE CUSTOMER ORDER PROCESSING SYSTEM