Stock Management and Catalogue System

John Crawshaw
Computing BSc
(e.g., 2004/2005)

Summary

Software which gives a small business control over their stock can often be very over priced. This forces the business to formulate means of stock control which is not always entirely suitable and therefore inefficient. Should the business be able to afford the exiting solutions they are often over complex; leaving the majority of the functionality unused. A piece of software which allows this stock control at minimal price keeping the stock in a usable, accessible form would prove beneficial. Further work with the stock allows it to be opened up to customers in a real time dynamic catalogue.

Acknowledgements

I would like to thank the following people for their contribution to the project:

- My projects tutor Pat Hill for her support and advice.
- My project assessor Bill Whyte for taking the time to mark this report.
- My partner Natascia for her constant support, motivation and patience.
- Saturn Music for their support and input.
- All my family for their support.
- Everyone who I have failed to mention and should have done.
- Italy for the finest foods edible at any hour.

Table Of Contents

1. Introduction

1.1. Aim

The aim of the project is to replace the existing spreadsheet based stock listings with a system which allows for easy querying and updating by the user in a much more accessible form. This system should be able to be extended by further development that could make use of the catalogue in a variety of means.

1.2. Minimum Requirements

The minimum requirements are:

- Formulate the system requirements for the proposed system.
- Design the application using the system requirements.
- Create a working system in accordance to the design.
- Produce a storage system capable of holding all the data required by the system.
- Create an interface for the system influenced by HCI theory where possible.
- Evaluate the system with at least 2 users.

The possible extensions are:

- A searchable online product catalogue that will be updated real-time in accordance with the business owner's stock catalogue.
- A business website to surround the online product catalogue.
- Dynamic systems available to website users utilising the stock catalogue e.g. User notifications of items coming in stock, which will use user preferences.
- Manual for using the system and system installation.

1.3. Objectives

- Examine the current solution then explore the range of means available to provide a better solution.
- Examine the host's domain to evaluate what the system must do.
- Create a system able to hold all and manipulate the required information.

1.3.1 Deliverables

- Printed and bound: Final Report (including documentation in the appendix).
- Manual made available in Appendix E
- Most current implementation of software at http://www.saturnmusic.co.uk

2. Methodology and Tools

2.1. Introduction

Methodologies are used to help a developer create a piece of software which is able to fulfil the needs of the host and user. This is made possible by creating a “specified sequence of tasks” (Fiduk *et al*, 1990) for the designer to follow which aims to structure all aspects of the development. Tools are used to develop the software solution itself, which in this instance relates to the technologies that are used for storing and processing data.

2.2. Methodologies

2.2.1. SSADM

“SSADM has been used by the government in computing since its launch in 1981. It was commissioned by the CCTA (Central Computing and Telecommunications Agency) in a bid to standardise the many and varied IT projects being developed across government departments. The CCTA investigated a number of approaches before accepting a tender from Learmonth & Burchett Management Systems to develop a method" (Eva, 1994). SSADM is very well structured and thoroughly documented with a set ‘pathway’ for the developer to follow as the project develops.

2.2.2. RUP

The Rational Unified Process (RUP) is a software development methodology from Rational (Webopedia, 2005). There are four main phases for the developer to complete, each of which has its own iterations. The phases are:

- Inception
- Elaboration
- Construction
- Transition

“RUP is similar in concept to Extreme Programming in that only what is useful and required it produced and the development plan is updated throughout the process” (Webopedia, 2005).

2.2.3. Discussion

As the decision has been taken to use prototyping in section 2.5.3 the methodology that will be used should allow for this. The author has no previous experience with either of the methodologies the one that is best able to incorporate with prototyping and the most comprehensive set of steps from the beginning of the development should be used.

Due to the timescale and size of the project the decision was made to use the RUP as a guiding development process but not to be followed strictly. RUP was chosen because it did meet the previously stated criteria but as the author did not want to be too restricted to the development process it would be better used as a guide.

2.3. Programming

One of the most important decisions to be made in this project is whether or not to make the stock system accessible through the Internet. This decision directly affects the choice of programming language and the technologies that will have to be implemented. Restricting the software to one pc makes the implementation simpler and possibly more secure, but the use of the system declines substantially. Instead it would be better to make the system available online, in this way the system can be utilised more efficiently and lends itself to being more greatly re-useable or extendable. This section looks at some of the technologies that could be used when implementing an online approach for the system.

2.3.1. PHP

"The goal of the language is to allow Web developers to write dynamically generated pages quickly" (Meloni, 2002). PHP: Hypertext Pre-processor is available on all major platforms and for the major web servers. As this technology is a back end language for the server there is no need for it to be installed on the client machine, which allows the client to access the system from any machine with an internet browser. This is because when a PHP script is called by the server the server outputs plain HTML derived from the script, which is an added bonus for security purposes due to external users not being able to see the underlying code of the system. Lerdorf and Tatroe (2002) explain that PHP pages are HTML pages with PHP commands embedded in them, contrasting many of the other major dynamic web-page solutions, which are scripts that generate the HTML from scratch. This more efficient and heavily database integrated language would be an ideal solution for a web based application intended for this project.

2.3.2. Perl

Perl was "originally designed for text processing" (Patwardhan, 2002) which available for most major operating systems distributions and system architectures. Its main uses are for non-GUI applications but through the use of modules its functionality can be easily extended for ease of creation of CGI applications on the World Wide Web. The fact that the original intended use was aimed more toward text processing the speed of Perl CGI scripts suffers as this further functionality is extended. Like PHP, Perl is a scripting language and does not need o be compiled before being run on the server system but Perl does create the HTML files from scratch instead of only creating parts of the file like PHP. A helpful feature of the Perl programming language is one of the popular DBI modules. This module allows the Perl language to easily connect to databases and execute necessary queries.

2.3.3. Discussion

The two programming languages that were assessed by the author are both highly suited to use in an Internet based application. This functionality lends itself to the possible extensions of the project and also aspects of the minimum requirements making these languages ideal.

	PHP	Perl
Speed	Fast	Fast
Database Libraries	Yes	Yes
Knowledge/Experience	Yes	None

Figure 2.1: Programming Language Comparison.

To decide on the programming language to use for the backend of the system the author created Figure 2.1 to compare the different approaches. As experience was the distinguishing factor, it is clear that the language that should be used is PHP. Having more experience in a language not only allows the system to be better developed but also be more efficiently created saving time.

2.4. Storage Technology

2.4.1. Text Files

One of the most basic ways of storing data in any system is in a plain text file. The data contained in the file can easily be loaded into an array and accessed when the program requires. Problems become more apparent when the system begins to grow and the volume of data that is required increases dramatically.

Scalability is a large issue using this approach when a large number of files are required. For this project a file could contain the details of one product or of a single user. When the system reaches the point of storing the entire stock for a business and multiple users these issues arise:

- Some file systems are limited to the number of files they can access at any one time.
- Files that are spread about the disk drive will take longer to access.
- Keeping track of all the files needed for the system to work.

2.4.2. XML

“Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML” (XML.org, 2005). This mark-up language allows the developer to create specific data structures for the application. When the data in the files are formatted in such a way the information is much easier to extract especially with languages that have developed functionality in this area (e.g. Java). The problem with XML is that the system would still be file based and inherits all the problems outlined above even though the data would be better organised within the files.

2.4.3. Databases

MySQL is an open source, free command line database that is available for all major distributions of operating systems (mysql.com, 2004). This allows the stock catalogue data storage to be installed on any type of server in any location giving more freedom to the host and the system designer. As the database is command line driven the MySQL database is well suited to the scripting approach of interaction to create the properly dynamic website as all communications can be performed directly between the two technologies. The only drawback to this piece of software is that the MySQL database does not perform as well, when comparing to other major database solutions, dealing with large volumes of traffic. As the intended host is a small business the designer can assume that the access rates would not be huge and the bonus of the software being free would outweigh the drawbacks. Other data storage systems are considered in more detail in the final report.

PostgreSQL is another open source based database system but unlike MySQL is an object-relational database management system (ORDBMS). Being based on POSTGRES the database system is much more feature rich boasting abilities such as complex queries e.g. sub-selection (postgresql.org, 2005). This extended functionality leads to a downside in that the speed of the system is reduced when compared to MySQL which was originally designed to be a lightweight database, despite the fact that “each are converging in the others direction” (Gilfillan, 2003). The general agreement between PostgreSQL and MySQL fans is that the better system to use is the one that the programmer is more familiar with and the complexity of the requirements.

2.4.4. Discussion

The disadvantages from the use of text files as a means of storing data easily outweigh those of using a well managed and planned database. Therefore the author has decided to rule out this approach of data storage.

	MySQL	PostgreSQL
Features	Enough features to complete the project prototype.	Complex system features available.
Speed	More lightweight system.	More fully featured system can slow down simple queries.
Past Experience	Multiple previous projects completed using the database system.	None.

Figure 2.2: Database Decision Making.

Figure 2.2 shows that the most suitable database system for the project would be MySQL in this case. Past experience is the main factor in the decision to ensure that the database system can be setup faster and correctly. Other issues such as features and speed also lend themselves toward the small-scale project.

2.5. Implementing

2.5.1. Waterfall Model

An alternative to the prototyping methodology is the Waterfall Model. In comparison to others this methodology is much more restricted and could be seen as a fixed path to take when designing. The method is often found in the Systems Development Life Cycle (SLDC) and each step is seen as self contained and irreversible (Racoon, 1997) therefore when utilising the system the designer should be cautious about moving to the next step. Stages of the methodology consist of:

- Analysis
- Design
- Implementation
- Testing

An issue that can be seen with this methodology and this project type is that the model “fails to account for change” (Racoon, 1997) and the user’s expectations of the final system are likely to develop simultaneously along side the project. Using this methodology would create problems if major design changes had to be made or alterations in the core components of the design.

2.5.2. Prototyping

The prototyping methodology is the most likely candidate for a project of this nature. In any project the effects of errors in the early development stages can have large repercussions later on in the process, especially if the error cannot be easily undone, giving a great importance to early error recognition. The prototyping methodology aims to give the ability for early error detection by giving constant feedback to the system designer. This feedback is achieved by allowing users to interact with the system at every development milestone, allowing them to check new features as they are created, and then informing the creator of any issues that arise. Prototyping allows “for developers to discover critical properties of their product before making final decisions” (Purtilo *et al*, 1991) and therefore fewer errors at the time of project completion.

2.5.3. Discussion

The rigidity of the waterfall model creates a design process that is too restrictive to fully explore all the user requirements without being prone to major problems further along the development progress. In the prototyping methodology the system can be created with the influence of the user at every stage, allowing large parts of the system to be altered as time progresses. As the author believes that the system should be designed using this constant user feedback and the flexibility of changing certain functionality the Waterfall model will be employed in this situation.

3. Analysis and Design

3.1. Introduction

The aim of this section is twofold. Firstly information must be gathered to find exactly what the system must do through the use of requirements gathering. This information must then be extended and further refined using UML (unified modelling language) techniques such as Use Cases Realisation and Concept Level Class Diagrams. The second part of this section looks at how the system works. The technical aspect of how the system performs tasks is left for section 4 (Implementation), in this section however UML notation can be seen describing the functionality set out in the analysis stages.

3.2. Business Modelling

3.2.1. Business

Saturn Music is a small family run business based in the town centre of Wakefield. The shop stocks most, if not all, popular musical instruments and the accessories for them. Also stocked are items required for live performances, from smoke machines and lighting to synthesisers and amplifiers. This huge variety leads to very large stock lists with a wide variety of categories.

3.2.2. Stakeholders

There are two major stakeholders involved with Saturn Music. The first of which acts mostly as the general manager and sole staffed member of the shop. His job varies hugely from day to day ensuring tasks are completed for the business. These tasks include:

- Ordering new stock to maintain stock levels.
- Updating stock sheets from the business based machine.
- Providing sales assistance to the shoppers then any following technical support that may be required.
- Advertising of the business.

The second stakeholder is more of a reserve member in terms of the workload of the business but still has as great a stake in the stock maintaining system. Her job is to ensure that the stock lists are kept up to date and accurate for taxing and insurance purposes which is carried out on a home machine.

3.2.3. Problems and Alternatives

Multiple locations of the stock system create problems as copies of the stock have to be carried around meaning that it is not always obvious which is the most current and that the method for transferring data is not the most reliable. The stock sheets themselves are cluttered and do not have much organisation, some sheets having hundreds of items in one list.

There are solutions such as Actinic Catalogue (http://www.actinic.co.uk/) but problems arise when Saturn Music specific data needs to be stored. The end result in a trial with this software lead to over complication and an unnecessary expense, as most of the functionality of the software would not be used.

3.3. Requirements

The aim of this section is to find, from the users, what tasks the system must be able to perform and in some cases the importance of these tasks. The intention of this section is not to describe how tasks will be performed, instead only concentrate on what tasks the system will be capable of completing. As prototyping will be used in the development stages it is unlikely that all of the user requirements will be realised at this stage, the beauty of this approach is that more functionality is added as the software solution 'matures' and the user submits extra input. This is another benefit to the prototyping approach as it would be close to impossible to document all possible requirements prior to the development stage.

3.3.1. Gathering

Some methods of user requirement gathering include:

- Questionnaires
- Interviews
- Focus Groups

To ensure that the best requirements gathering tool is used each one must be judged using a set of criteria specific to this project. For every different application where requirements are used there are always a set of variables which determine which would be the most efficient to apply. To find the must appropriate a set of criteria will be created:

- Time (1) – This criterion aims specifically at how long the requirements gathering would take using this approach including both the questions to ask a user and the time taken to turn replies into requirements.

- Organisation (2) – The second criterion looks at how difficult the requirements gathering would be to perform. This could be arranging a meeting or finding a place to distribute questions.
- Feedback (3) – The last criterion attempts to measure the value of the feedback that is obtained using the chosen method. This assesses both the quality of the answers gained and the likelihood that answers will actually be obtained.

The criteria will be judged using a numerical valuation in the range of 1-3, 1 being the lowest and 3 the highest.

	Questionnaires	Interviews	Focus Groups
Criteria 1	2	3	1
Criteria 2	2	2	1
Criteria 3	1	2	3
Totals	**5**	**7**	**5**

Figure 3.1: Evaluation of Requirements gathering.

The results in Figure 3.1 show that Interviewing is the best approach for this project. The decisions for each of the values are described below:

- Criteria 1 – Although interviews can take a little time to organise as soon as the interview begins no time is wasted with the requirements gathering and so scores the highest. Questionnaires can take a little time to develop and prepare for the distribution but also there is a large delay from waiting for the results, which then need to be transferred into usable requirements. Focus groups take a lot of time to schedule a suitable time for all the people concerned; the session itself can then take a large period of time to complete.
- Criteria 2 – Both questionnaires and interviews require about the same amount of organisation. For each a set of questions has to be developed which is then put to the target audience. Interviews require a meeting to be arranged where as questionnaires require permission for distribution and afterward collections. The focus groups are more difficult to organise, as there are lots of people involved who all need to have free time at the same time, hence the lower score.
- Criteria 3 – Focus groups offer the best feedback for the user requirements. This is because the person overseeing the group can direct the discussion of the group and then record the decisions made. As there are multiple members in a group, areas of disagreement can be addressed consequentially producing better results.

Questionnaires can produce varied results depending on how the candidates interpret the questions and the return rate of the questionnaires themselves. Interviews have a good feedback rating as the person performing the interview can ask further questions and direct the level of detail that is obtained 'on the fly'. This method is much more personal and can continue until the interviewer is content with the results obtained.

An informal interview was arranged with the stakeholders to discuss requirements, the aim was to find a basic outline, which could then be refined as the prototyping took place. Figure 3.2 shows these initial requirements:

Num	Requirement	Staff
1	Well-structured item database - easy to navigate and search.	Kev, Ann.
2	Ability to add remove and change items.	Kev.
3	Ability to add remove and change categories.	Kev.
4	Create key summary information, such as stock value currently held.	Ann.
5	Detailed information of items to be stored.	Kev, Ann.
6	Make the stock information available to customers.	Ann.
7	Adapt the site to registered users (mail shots/personalisation).	Ann.
8	Administrative ability on the users (remove/change password etc)	Ann, Kev.

Figure 3.2: Initial Requirements.

3.3.2. Use Case Diagrams

Bennet *et al.* (2001) state that use cases “provide a good way of getting an overall picture of what is happening in the existing system or is planned to happen in the new system”. These use case diagrams can be realised through the initial requirements made with the stakeholders in addition to more feedback from further interviews. As an added bonus the diagrams are very simple with little notation so it is easy to communicate the developers understanding and intentions with the stakeholders to ascertain that they are in agreement.

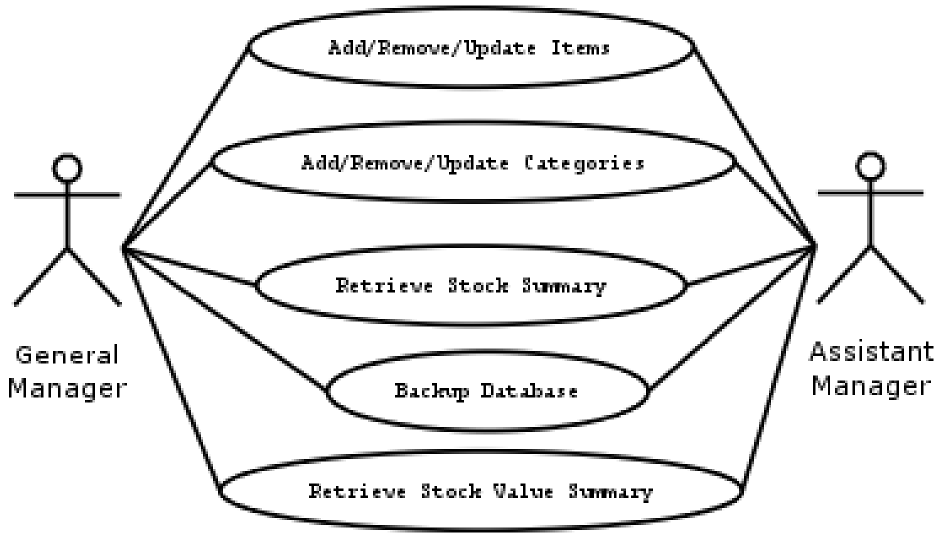


Figure 3.3: Stock Management Use Case

Figure 3.3 shows how the two stakeholders will be able to interact with the new system. This use case involves the general maintenance for the stock levels including the backing up of the database and retrieving stock summaries.

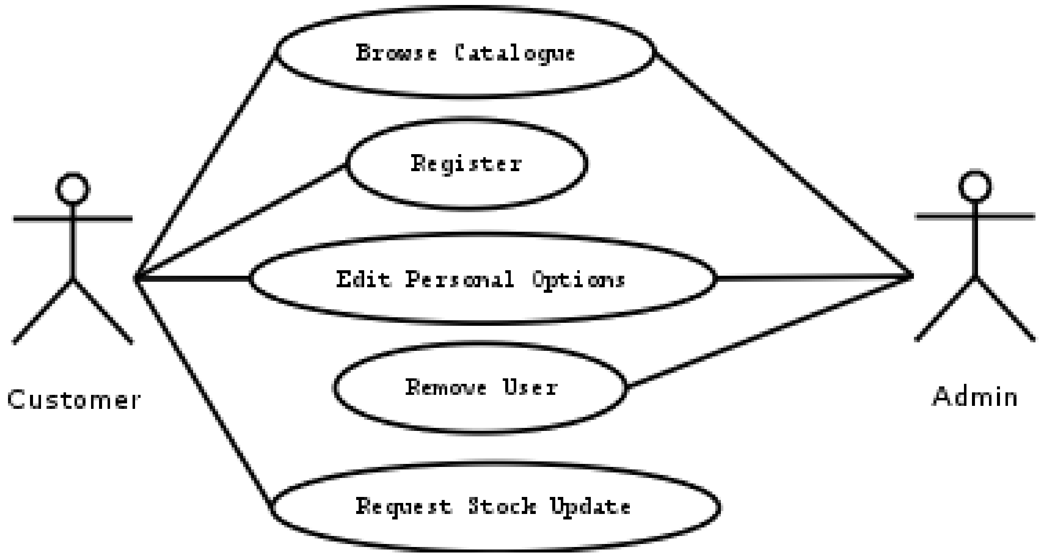


Figure 3.4: Customer stock interaction use case.

Figure 3.4 shows how external customers will be able to interact with the stock catalogue and the surrounding system. There are items that the customer is unable to interact with but the admin is able to (and vice versa) meaning the system must have some means of authentication for certain tasks. This means of authentication will also help for the identification of users in requirements such as mail shots.

3.3.3. Concept Class Diagram

Bennet *et al.* (2001) state that class diagrams “show the building blocks of any object orientated system”. When in an analysis stage the class diagrams remain as concepts that are later used in both the design process and then the implementation stages of the development. In this case the concept level class diagrams describe the system for one particular instance i.e. for a manager using the system and for a customer using the system.

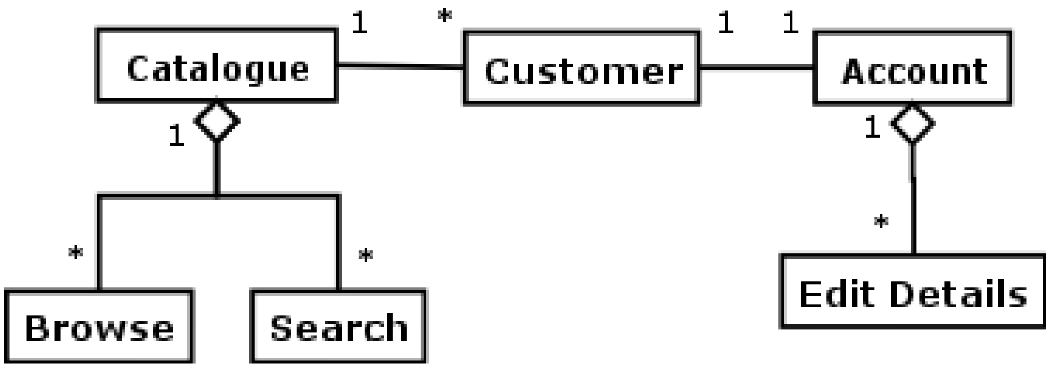


Figure 3.5: Customer based concept class diagram.

Figures 3.5 and 3.6 represent the system in use by both the administrator (who in this case would be the stakeholders) and the customer (who would be browsing a website). The numbers and asterisks in the diagrams represent the cardinality of the classes (e.g. many admin can oversee one catalogue) and the diamond shaped arrow represents one class that is part of another (e.g. Searching is part of the catalogue class).

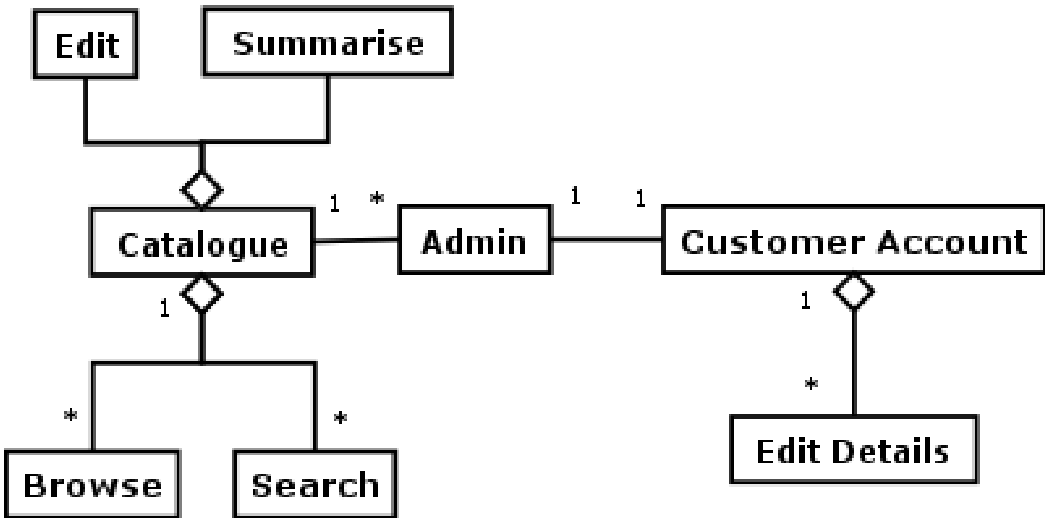


Figure 3.6: Admin based concept class diagram.

3.3.4. Requirements Listing

As requirements are developed they can be segregated into two groups. The functional requirements are tasks that the system must be able to be performed by the user. Non-functional requirements are effectively constraints placed on the solution by either the system itself or the users. Figures 3.7 and 3.8 show the functional and non-functional requirements of the system, these requirements have been realised through the analysis in the use cases and concept level diagrams along with further interviewing of the stakeholders and refinement of the original requirements.

Num	Requirement	Importance
1	Compatible with any server operating system and most, if not all, main stream internet browsers.	Essential
2	Ensure all users are able to perform tasks they expect to be able to.	Essential
3	Enable stakeholders to more efficiently complete tasks when compared to previous system.	Essential
4	Ensure that authentication systems are in place so users can only gain access to the correct areas.	Essential
5	Enforce record concurrency and referential integrity in the database.	Essential
6	Ensure that the system can cope with a large amount of stock (scalability issues).	Essential

Figure 3.7: Non-Functional Requirements.

Num	Requirement	Importance
1	Searchable catalogue.	Essential
2	Browse able catalogue.	Essential
3	Customers - Ability to register with the website.	Essential
4	Customers - Ability to update personal data.	Essential
5	Customers - Front page tailored to their preferences.	Essential
6	Customers - Request an email update for when items are restocked	Desirable
7	Admin - Add new items.	Essential
8	Admin - Remove items.	Essential
9	Admin - Update items (stock/information).	Essential
10	Admin - Add new categories.	Essential
11	Admin - Remove categories.	Essential
12	Admin - Rename categories.	Essential
13	Admin - View all users.	Desirable
14	Admin - Remove users.	Essential
15	Admin - Reset user's passwords.	Essential
16	Admin - Mail outs to all registered users or dependant on set preferences.	Desirable
17	Admin - Request stock summaries.	Essential
18	Business information website to surround stock catalogue system.	Desirable

Figure 3.8: Functional Requirements.

3.4. Business Function

3.4.1. Collaboration

Collaboration diagrams attempt to show the communication between the objects that make the system. When all the message paths are highlighted the overall picture shows how all the objects work as a whole to provide the solution. Bennet *et al.* (2002) explains that “this ‘working together’ to produce some useful result is what is meant by collaboration”. Figure 3.9 attempts to show this ‘working together’ by extending the concept class diagram with extra classes realised from the further requirements. The arrows depict the flow of data between two classes e.g. an admin requests a mail-shot from the email class, which itself retrieves the email address information from the accounts class.

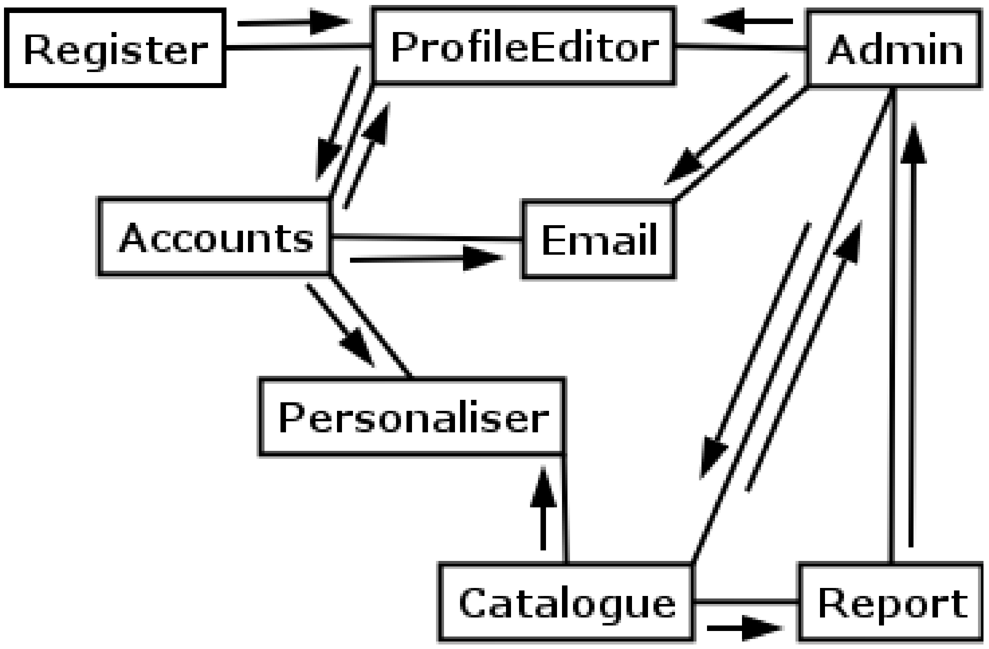


Figure 3.9: Collaboration Diagram.

3.4.2. Sequence

The sequence diagram is somewhat of an extension to a collaboration diagram. The transferral of messages between classes are still shown but with the difference that the ordering of class interaction to complete a task is shown. Bennet et al. (2002) explain that “Sequence diagrams are used to show the same interaction as in a collaboration diagram, but they emphasise the order of the message over time”. To better understand the sequence diagram one could imagine it to be plotted as a graph, where the x axis would represent objects and the y axis representing time.

Figure 3.10 illustrates this ordering of messages over time for a user utilising the system. First the user registers with the system and then begins to browse the catalogue.

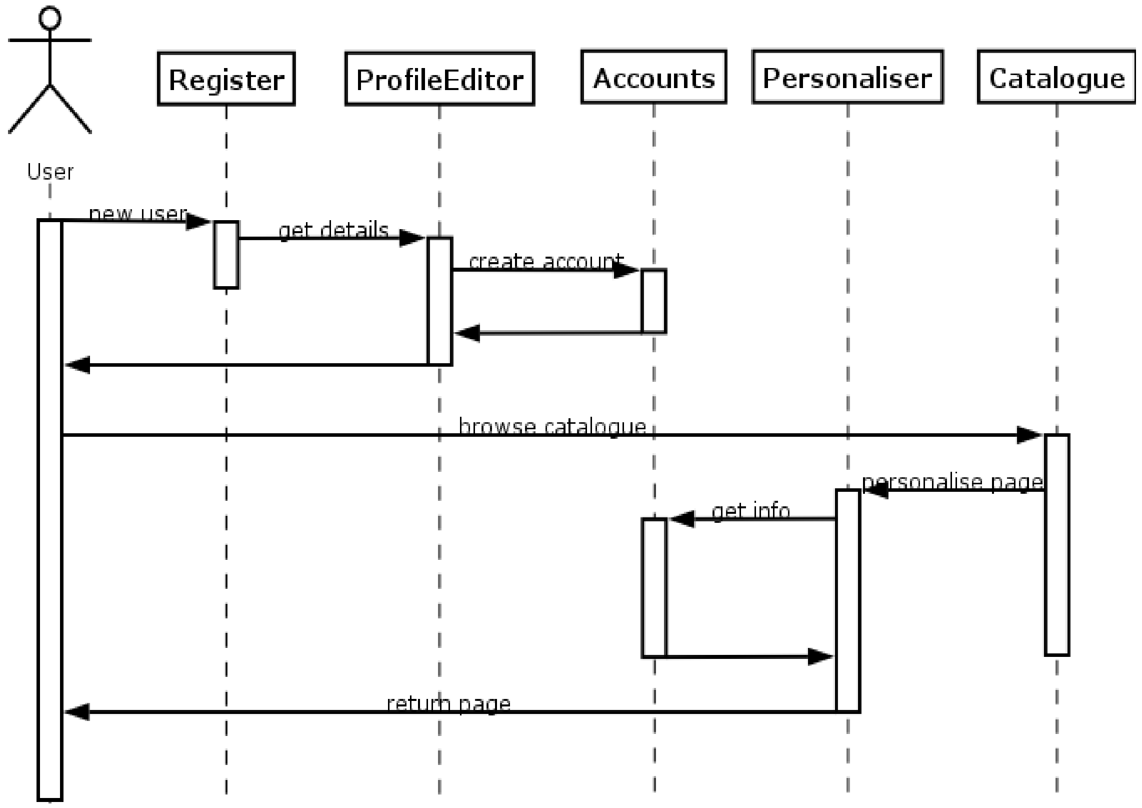


Figure 3.10: Sequence diagram for users.

Figure 3.11 shows how an administrator interacts with the system. As an authenticated administrator has different access privileges to that of a normal user some of the classes which are available to them differ.

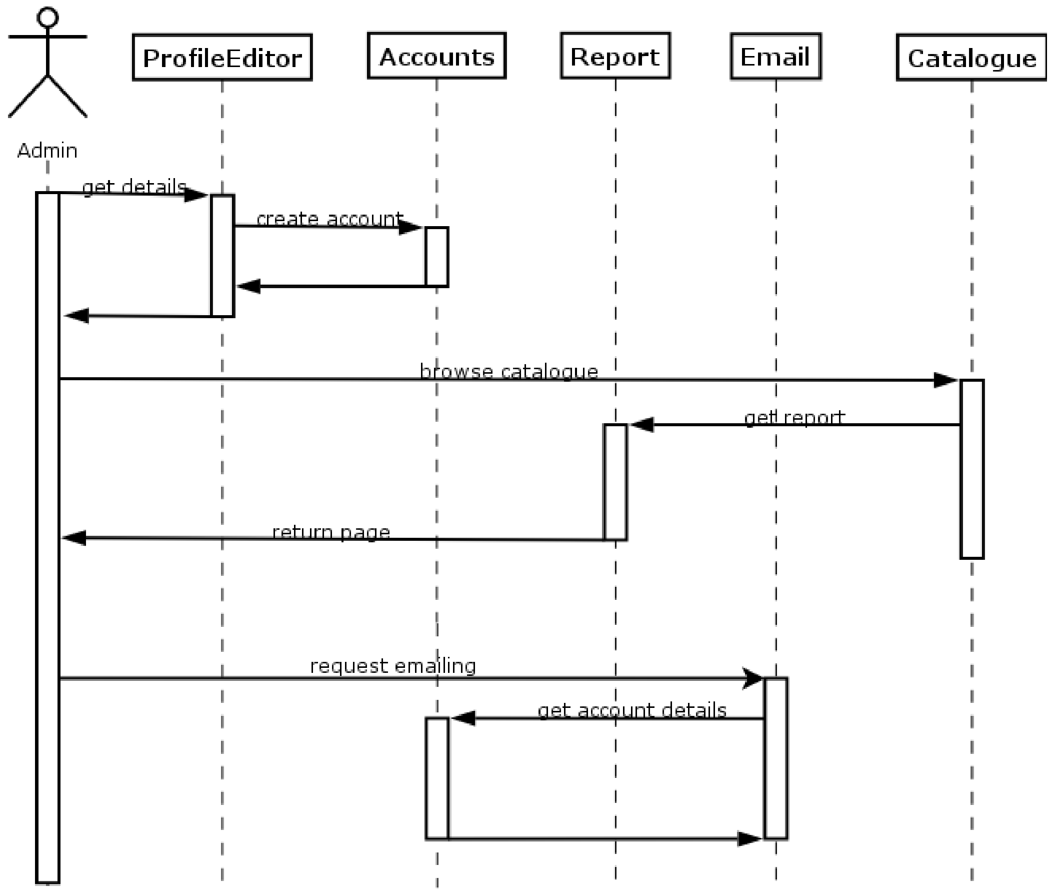


Figure 3.11: Admin based sequence diagram.

3.4.3. Activity

“Activity diagrams are a means of describing workflows and can be used in a variety of ways. At the design level, they can be used to describe in details the flow within an operation” (Bennet *et al*, 2002). Figure 3.12 attempts to show the workflow for the scenario described in figure 3.10. Diamond shapes in the flow represent a choice that is made and the round edged boxes represent the state of the user while black dots represent the start and end of the workflow. An important note on figure 3.12 is that the user can choose to register and update their profile before they browse, a further extension of this image would be to show that the user would not have to do this should they choose not to.

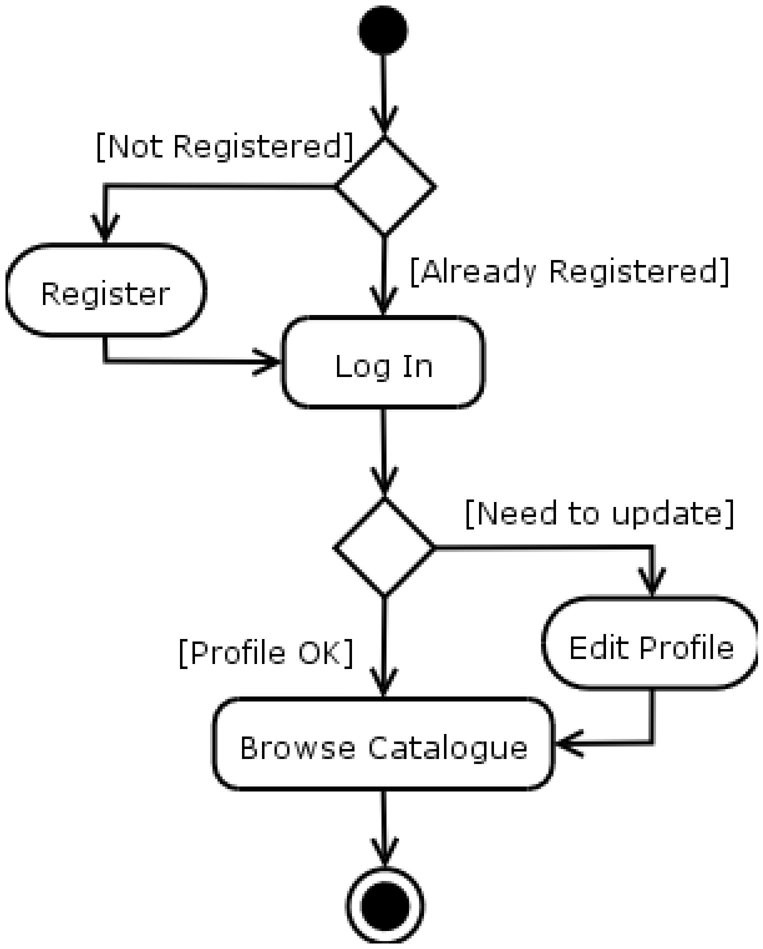


Figure 3.12: Activity Diagram.

3.5. Data Management

The current stock handling solution of the business uses flat, unordered spreadsheets to store all the items that the business can sell. To make all the data fully accessible and maintain integrity this spreadsheet has to be restructured into a formal database. The database management systems allow for strict rules which includes regulations that maintain the concurrency of items and how entities relate to each other, ensuring that the relationships are not wrongly represented or broken. The ER Modelling allows the structure of the data to be mapped as a whole and describes these relations.

3.5.1. ER Modelling

To model how the data currently is stored and then how this is developed in a true normal form Entity Relationship modelling will be used. First entities and their relations can be described, secondly adjustments can be made to ensure normalisation then lastly transcribe this information into the construction of the database. Here is an image of the current solution represented in an ER Model:

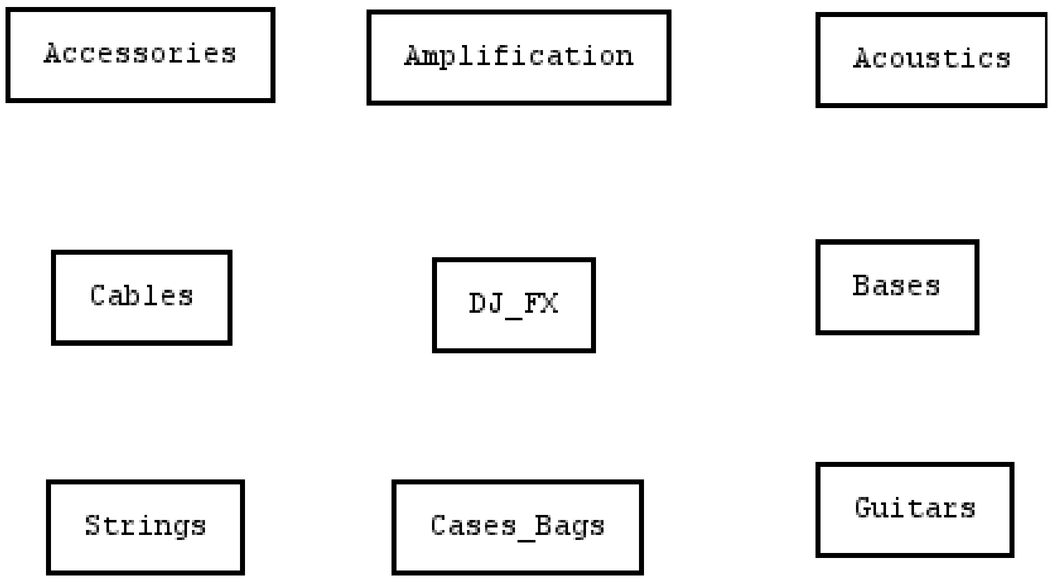


Figure 3.13: Original ER Model

Figure 3.13 shows nine objects that are the entities of the existing system. In this instance these represent the categories of the items. If the attributes were to be shown in this image they would branch from the entities and describe qualities of them items, e.g. Name, Brand, Price etc. but to reduce cluttering these have been omitted. Relationships are shown as lines between the entities which are further explained by the numbering that accompanies them.

An example of this numbering is:

- 1:1 (one to one) - Is where only one attribute of one entity is stored in another entity.
- 1:* (one to many) - Is where one entity may have many attributes in another entity, e.g. A guitar category may contain lots of guitars.
- *:* (many to many) – Is where two entities attributes overlap considerably.

The entities are represented in a database as tables in which the attributes describe the ‘columns’; this includes the data types and any rules that apply. The relationships that are described above are practically applied differently depending on which is used. For a one to one relationship primary keys are associated with the relevant attributes in the other tables. The one to many and many to many relationships are more complex however; these relationships require a third table where the corresponding items from each entity can be matched. When querying is then performed a join can be made between the two matching tables.

Looking at this current solution it is easy to see how certain entities will contain a long list of items, which for a true DBMS is easy to search through and manage but in a spreadsheet can cause major problems for the operator especially with no strict item naming and categorising. A system that ensures all the information that is required is actually obtained would ensure the database stays correct and complete. The next step is to adjust the ER model into a normalised form that can be used for the new system.

3.5.2. Normalisation

Before the normalisation can begin the other tables that will contain information for the system will be added. Below in Figure 3.14 these changes can be seen:

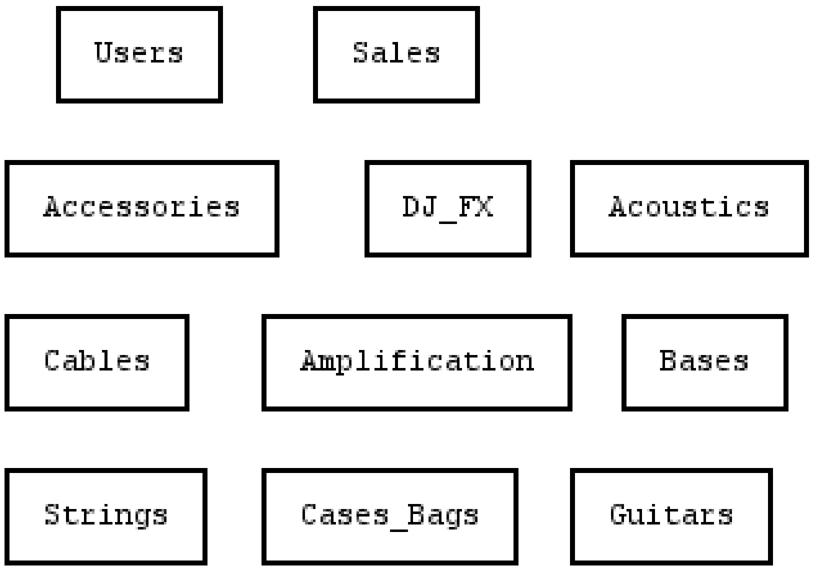


Figure 3.14: ER Model with system entities.

According to mysql.com (2005) “Normalization is the process of removing redundant data from your tables in order to improve storage efficiency, data integrity and scalability”. This key process is critical to designing and creating a well functioning database that will stay useable even as the system grows and loads placed upon it are increased. Ensuring that the database does actually conform to the normal forms requires following simple rules for each different normal form. For the purpose of this project reaching the third normal form level of normalisation will be adequate as the fourth level is use for production databases and fifth levels of normalisation are mainly used in academic databases. One drawback of normalisation is as the level is raised so is the complexity of the database as more tables and relations are required to satisfy the rules and constraints.

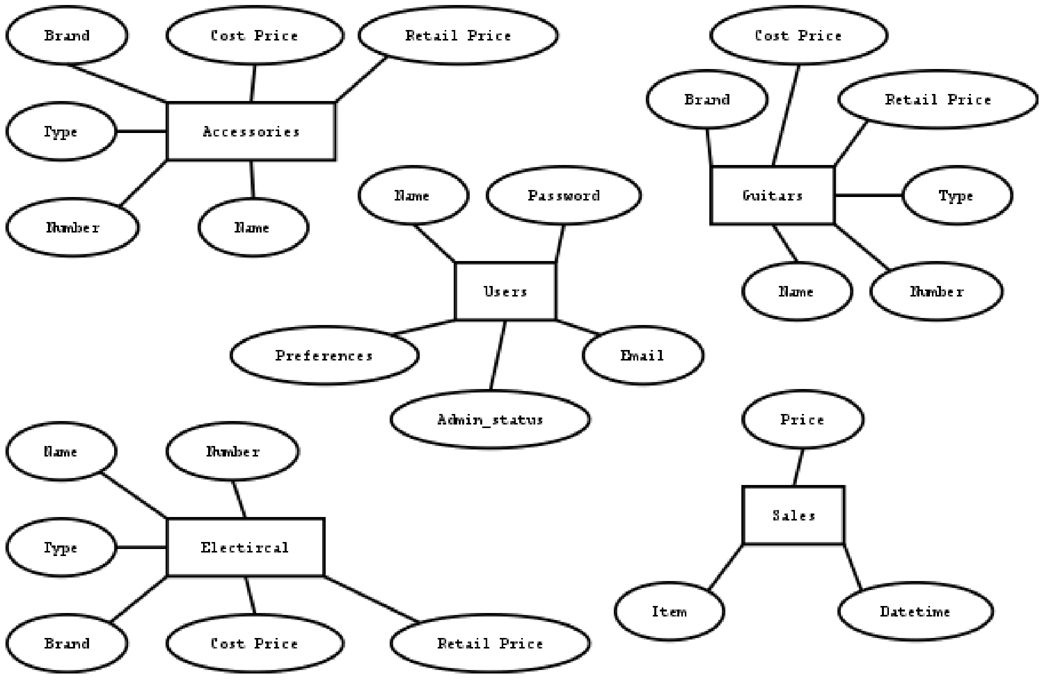


Figure 3.15: Normalisation Stage.

Figure 3.15 shows how first normal form has been achieved. To reach the first normal form (or 1NF) the requirement is:

- Horizontal rows cannot contain redundant data; every value must be atomic, ensuring that no sets of values are used.

The next stage is to progress with the normalisation. Requirements for the second normal form are:

- Meet requirements for the first normal form.
- No redundancy in the vertical columns; ensuring that no values are unnecessarily repeated.

Third normal form then requires:

- Meet requirements of the previous normal forms.
- Removal of data that is not fully dependent on the primary key.

Figure 3.16 shows how these changes have been implemented with the new tables to hold the system data (such as 'users' and 'sales').

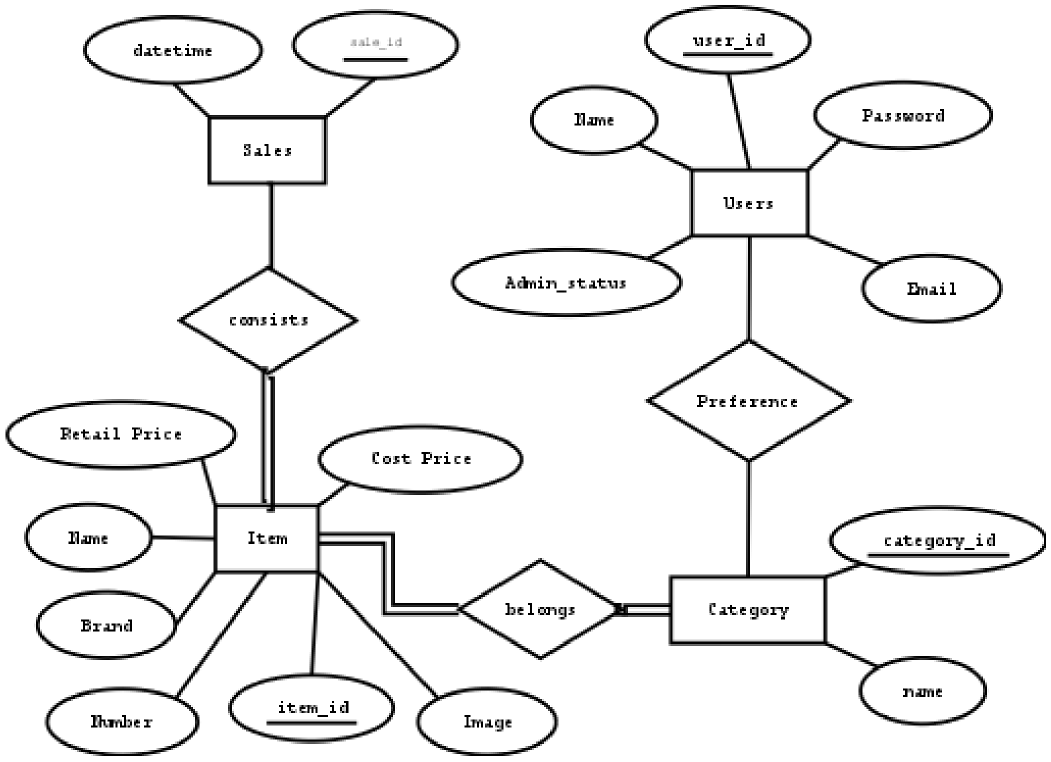


Figure 3.16: Normalised database.

The changes that have been made to accomplish the database normalisation can be seen in Figure 3.16. Primary keys have been assigned so that every entry and the database can be uniquely identified, helping to achieve third normal form. This is attained using the automatic incrementing key assigning data type provided by the database. The separate item tables have been consolidated into one to which a category table is related. This ensures the database conforms to the second normal form and gives two major benefits: 1) Categories can easily be added and edited. 2) The preference relation between category and users allows users to choose their preferred category. The sales entity is an optional table that can be used to track transactions by making single receipts constituting of single or multiple items. The SQL commands to create this final scheme can be seen in Appendix B.

3.5.3. Integrity

Two important issues must be addressed while designing a database to ensure its integrity these are the entity integrity and referential integrity constraints. Without these two constraints the database could return unexpected results and threaten both the reliability of the database and the data itself.

Entity integrity aims to ensure that all the tuples in a table will be able to be distinguished from one another. Elmasri and Navathe (2000) express that “The entity integrity constraint states that no primary key value can be null”. The primary key is used to identify tuples in a relation; if no key is set there is a high risk of not being able to distinguish between two tuples adversely effecting queries placed on that relation.

Referential integrity is aimed toward foreign keys and multiple relations (unlike entity integrity). Elmasri and Navathe (2000) show that “the referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples of the two relations”. This constraint aims mainly to stop a foreign key in one relation pointing to a non existing entity in another; an example would be having an item in a category which no longer existed. Both this constraint and the entity constraints are mainly enforced by the DBMS but careful consideration will still need to be taken when designing the queries that will be used.

4. Implementation

4.1 Introduction

In this chapter we look at how the features of the system were implemented to ensure that the user requirements were satisfied. These are the more technical aspects of how functions were created and their purpose. Due to length full code listings are not shown but instead snippets are used to highlight points. Even though the author opted for using the prototyping method of development most code that had been implemented was not changed in the next iteration, instead just extra functionality added according to the requirements.

4.2. Database

Recovering relevant data from the database into the system is a key part to making the system work. Any part of the system which generates dynamic content is likely to need access to the main database. The best way to do this is to create a central access point of the database which is then included into all the files that need database access.

This code defines variables needed to connect to the database then creates a variable the holds the actual database connection. One major advantage of this method is that should any of the above information change it only has to be edited in one place.

Above is an example of how a script imports the database file then executes a query. Once imported PHP makes querying very easy, all that needs to be done is to create a string containing the query, execute the query with the mysql_query function then create a new variable containing the matching results.

4.3. Queries

Some queries in the system are more complex as they have to retrieve information from multiple tables. As the database has been designed to be in third normal form and respects the entity and referential constraints, the task can be easily performed using inner joins.

This query shows how a one to many relationship is created in SQL. Three tables are joined together where there are matching keys, allowing a search to be conducted on both the items and categories tables. This makes it possible to find an item in one specific category, which is used in both the catalogue and stock management scripts.

4.4. Functionality

The implemented functionalities which will be discussed in this section are those which were specified in the possible extensions. The first of these is the search function available to the catalogue. When this is used from the main page the text entered into the search box is used to look through all the item names and item brands, which then returns the results accordingly. When a user is browsing the catalogue and is looking in a specific category searches which are made within the category look within the relevant items only, allowing the user to quickly find the item they are looking for. This functionality is passed onto the admin section of the website allowing searches to be carried out in the same way, again allowing the stock control to be more efficient for the stakeholders.

Another of the possible extensions was the business based website to surround the cataloguing system. To ensure that the system can be used as a business medium, relevant information about the business has to be provided, which is located in the contact section. To add to the business feel the site has a professional appearance (consistency, colour scheme etc) which is complemented by the dynamic stock catalogue and functionality.

One of the possible extensions which were derived from the user requirements was the ability to give feedback to the registered users. Mail-outs are a quick and easy way to communicate with potential customers, whether this function is used to deliver special offers or newsletters the task is easily completed by simply filling a text box and clicking ‘Send’. Although not fully developed, another idea brought by the user requirements was for the user to be able to request updates for items which is currently out of stock simply by clicking an icon while they are logged in. When the stock is updated the system then creates a standard email which informs the user that the item is back in stock, effectively performing a task that the business owner would have to do themselves in order to make a sale to that person.

4.5. Security

In this day and age it is a sad fact that any machine connected to the internet is potentially at the threat of an attack. One of the best ways of dealing with this problem is try to prevent this happening in the first place and also minimise the damage if any threat becomes an attack. This section shows the attempts made to keep the system secure and also explains the 'worst case scenario'.

As this application is dynamic there is likely to be, at some point, input from the users so one of the golden rules of programming should be addressed "*Never trust your users*". As PHP is executed by the web server (usually running with high priorities) all information that is passed to the server should be inspected. The first means of doing this is to create code which will check for 'injection' attempts (i.e. an attacker adding their own code and tricking the server into executing it). This is done using the code below:

This approach uses two possible methods for checking input data: 1) Magic Quotes which automatically removes characters that can affect the system in a way which the developer would not want or 2) The language and database specific real escape string which performs slightly better due to the specificity. Which one is used depends on if Magic Quotes are enabled in the php.ini file (allowing the administrator to control this security feature).

The other means of securing the web application is to ensure that insecure global variables are not used at all costs. These variables can be easily changed using the address bar in a browser by adding code onto the end of page requests (as this information is also processed by the server). One of the most common mistakes made is using an authorising global variable to check if users are logged in or not; this is particularly devastating should an attacker decide to set the administrator as 'logged in'. A better method for authorising users is to utilise the session ability of PHP. Sessions can be used in conjunction with cookies, but as some web browsers do not accept cookies and they can also be altered by potential attackers a system which controls the session variables itself would be preferred.

This code is taken from the logging in program. Should the user's username and password match then this code is run. A session name is assigned, cookies are turned off and the session is started on the server. Details about the user are taken from the database to ensure no security breaches and then the user is forwarded to the successful login page with their new 'session ID'. The setback with the approach is that every time a user progresses to another page the session ID must be passed on for them, meaning that every link on the website has to be accounted for.

Should an attack attempt be successful the access to the database machine is still restricted to the account which holds the system database. This account is likely to not have administrator privileges on the server so the damage done to the database server could be restricted to the one account (and easily rectified). The backup function in the system requirements would then make it possible to easily recover the database to its previous state in a matter of minutes (after the password has been changed) providing that a backup is performed regularly.

4.6. GUI

The GUI is an important part of the implementation as it is the only part that is viewable to the customers and is critical for efficient use when administrating the stock control. The system should ensure that the catalogue is usable by people with varying levels of computer experience, ensuring it is easy to navigate and easy to read.

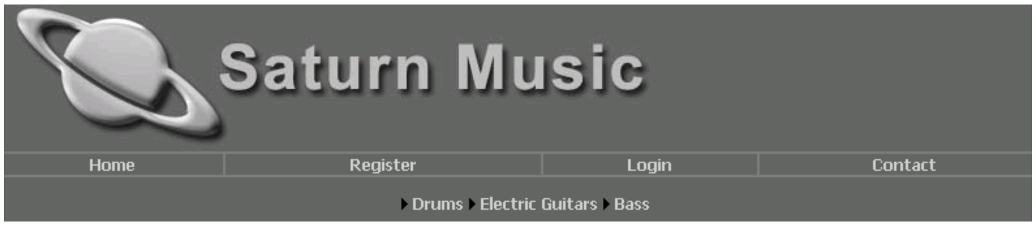


Figure 4.1: Navigation.

Figure 4.1 shows the main navigation of the website. This page is at the top of all of the system pages allowing users to find their way no matter where they are. These four main menu items also update when a user logs in as seen in figure 4.2. Underneath the main locations of the site there are the current existing categories of the stock. When new categories are created they appear dynamically at this location, this flat hierarchy can be easier for the users to navigate as all the currently available groups are visible at the same time. Tree like navigations can be more complex to both navigate and effectively implement.

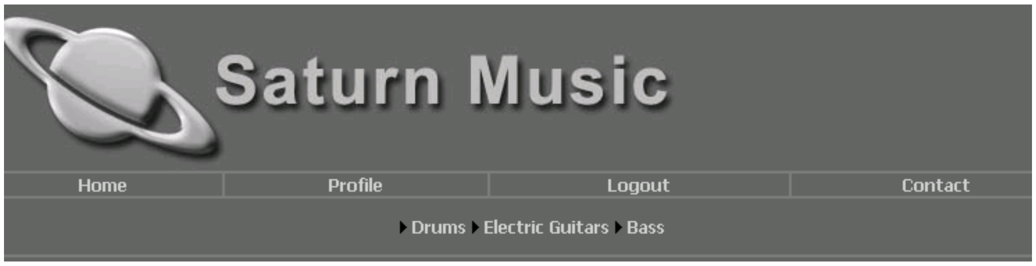


Figure 4.2: Logged in menu.

When creating the GUI's for the system the author aimed to ensure consistency at all times. This was achieved by both creating the previously mentioned menu system and also having all items being displayed in the same format. This format is also re-used to list users, items and categories for the admin in the admin section as well as search results.

The colours used in the system were kept simple and close to greyscale to ensure that text is always easy to read. Important messages, like login error messages, are displayed in red to draw the user attention to them. Sections of the system are segregated using contrasting background colours, which also contrast with the text colours to ensure that it stands out.

The main page of the system is tailored, using HCI theory, to the current user if one is logged in. This utilises the information given by the users in their profile to display the top resulting items for their favourite category. This is relatively simple to implement but gives added value to the user browsing their site as they feel that the business is more interested in their needs and shows them the items that they like.

5. Evaluation

The last stage of project aims to assess how well the system has successfully completed its aims. Looking at how the system can fulfil the user requirements and how the project requirements have been completed shows if these aims have been achieved. Testing of the system gives a good indication of how the system is performing and gives back information, which can be used in to perform this final evaluation.

5.1. Software testing

Software testing is used the check if the is working as the developer would expect it to work. Major flaws in functionality can be found pre-release when using software testing properly. To efficiently test the software and ensure that all code is thoroughly checked a structured test plan can be used. As the developer used prototyping some testing has already occurred during programming, as each function is created the code was then checked allowing for the developer to correct any obvious mistakes in the next iteration.

5.1.1. Test Plans

The approach the author has decided to use to assess if the software is working or not is to try to perform each of the requirements set out in section 3.3.4. This approach was chosen because all the created code would be used as all the functionality is used and any flaws would be shown. The table below shows the results of the tests that were made, both the test that was carried out and the outcome each one returned.

The second test carried out on the system was to have the users interact with the system. To create an accurate measure where users can be compared fairly scenarios were created. Scenarios are a good test of seeing how the system is able to interact with people and also how people accept the system to work with. Issues such as the GUI and ‘feel’ of the system can easily be highlighted when carrying out these tests. Results of these scenario tests are discussed in section 5.3 alongside the feedback from users.

Num	Test	Outcome	PASS?
1	Searchable catalogue.	Catalogue could be searched.	PASS
2	Browse-able catalogue.	Catalogue could be browsed.	PASS
3	Customers - Ability to register with the website.	Registered as expected (system showed errors).	PASS
4	Customers - Ability to update personal data.	Able to change email, password and personal preferences.	PASS
5	Customers - Front page tailored to their preferences.	Display appears different for different users.	PASS
6	Customers - Request an email update for when items are restocked	Email not sent.	FAIL
7	Admin - Add new items.	Item successfully added.	PASS
8	Admin - Remove items.	Item successfully removed.	PASS
9	Admin - Update items (stock / information).	Stock levels, name and category successfully changed.	PASS
10	Admin - Add new categories.	Category successfully added.	PASS
11	Admin - Remove categories.	Category successfully removed.	PASS
12	Admin - Rename categories.	Category successfully renamed.	PASS
13	Admin - View all users.	Able to view all current registered users.	PASS
14	Admin - Remove users.	Successfully removed user.	PASS
15	Admin - Reset user’s passwords.	No option to reset password.	FAIL
16	Admin - Mail outs to all registered users or dependant on set preferences.	Created email aimed at guitar enthusiasts, sent successfully.	PASS
17	Admin - Request stock summaries.	All summary information displayed.	PASS
18	Business information website to surround stock catalogue system.	Business contact details and address visible on website.	PASS

5.1.2. Success Measure

After the tests were complete the successfulness of them can be seen by looking at the pass and fail rates. The failing tests must be addressed and investigated to see why these tests failed and if there is a large detriment to the system.

Test 6 indicated a failure when tested. This test is aimed in part to the human-computer interactivity functions of the website. Should an item be out of stock a user would be able to register their interest in the item with the system. When the business owners then restock the item an email would be sent out to the user notifying them of this. The test failed in this instance because the email was not sent when new stock was entered. This failure was due to a non-finished function of the system. At the final iteration of the website the system was able to email users but not without prompting from administrator. To complete this task changes would have had to be made to the structure of the database and further programming required (possibly in another language for the system to remain platform independent) to regularly checking items with users interests and the stock levels of these items.

Test 15 also resulted in failure when tested. This test was part of the administration area of the system. Should a user loose or forget a password it is usually the job of the administrator to fix this. Other systems use a combination of secret questions and personal information to reset the password automatically. The reason that this feature did not work in this system is that the implementation was removed. Further thought into the ethics of this method of password resetting showed that it might not be the best possible way. An attacker could easily impersonate another user causing the password to be incorrectly reset, then revealing personal information stored by the system to the attacker. After this reconsideration the author decided that this feature would be better left disabled until further development.

With only two fails when testing the system against the user requirements the system could be considered a success. The two failures that occurred either needed more thought to ensure the best method was used or lots of extra time and database restructuring.

5.2 Implementation

The overall success of the system and how it was implemented should be further investigated to decide if choices that were made were the best made for the system. In this section some choices of how the catalogue were implemented and the methodology used for the system development shall be explored.

The catalogue system used a flat structure to represent the categories. An alternative to this would be a hierarchical approach to organise the items. The decision in this case was to stay with the flat representation due to the nature of the business. A music business has several high level distinguishable items (e.g. Drums / guitars) but below that level do not vary enough to constitute a separate sub category. For other businesses, such as a computer parts and accessories company, a hierarchical structure would be ideal to organise the stock more effectively. This business dependence lends itself to further development (section 5.4).

Another function incorporated into the catalogue system was the personalisation for the registered users. Seven benefits to the use of personalisation in a system can be seen below:

- Efficiency
- Effectiveness
- User Satisfaction
- Trust
- Accessibility
- Improved revenue
- Customer loyalty

Efficiency and effectiveness relate to the speed and performance of a user performing tasks with the system. It can be argued that in this system these attributes are not improved because of the requirement to login before these benefits take effect, which itself slows the user down when using the system. Degrees of user satisfaction, trust and accessibility have been improved by adding personalisation to the system. The fact that the users can register with the system, then personalise how they view the site gives an added sense of value and makes the users happier to use the system. The way that the system is designed users are able to choose any of the available categories as their favourite giving accessibility to all on the site. Improved revenue and customer loyalty may not be directly influenced by the personalisation implemented in this system. Two reasons for this is that the stock notifications were not fully completed and as the website is not e-commerce enabled there is not an accurate measure of how other features effect the customer's transactions. Some drawbacks to the use of personalisation in the system are kept to a minimum where possible. Drawbacks such as speed and complexity are not affected because of the size of the application. Had the database been much larger and other methods of personalisation were used (e.g. tracing) problems with speed may have occurred. One possible area for a drawback to become apparent is frustration of the user. Should a user, who has preferences set to guitars, log in and want to search for drum kits the system is not tailored to their needs and to fix this would require them to change their preferences.

The methodology used to develop the system was the best choice with the approach taken. The author found the prototyping to be much more relaxed and flexible especially when compared to that of the restrictive waterfall model. Prototyping allowed the development to happen gradually with constant feedback from the stakeholders.

5.3. User Satisfaction

The feedback from the users of the system and the stakeholders has played a large part throughout the project. For evaluation purposes scenarios were set for administrators which were then timed to compare with the original system (where possible). As there was nothing to compare the catalogue users tasks to, because no system previously existed, feedback was examined instead. The proposed scenarios were as follows:

- Scenario 1 – Add/Remove/Edit Item.
- Scenario 2 – Add/Remove/Rename Category.
- Scenario 3 – Search for an item.
- Scenario 4 – Get stock summary.

Results of these scenario tests can be seen in figure 5.1:

Scenario	New System (average time)	Old System (average time)
1	2m 46 second	3m 2 seconds
2	1m 12 seconds	5m 32 seconds
3	4 seconds	8 seconds
4	3 seconds	1m 8 seconds

Figure 5.1: Scenario times.

The difference between the systems varies from scenario to scenario but overall a speed improvement can be seen when using the new system. When talking to the volunteers this was due to the system being much more specific when compared to the spreadsheet application.

Scenario 1 showed little difference in the times taken to complete the set of tasks. Even though the new system has a specific set form for creating a new item and editing an existing item it is still very simple to add a new items in a spread sheet application. In both the spread sheet application and the new system removing existing items is a simple two click operation.

Scenario 2 highlighted how the new system out performed the existing system. When creating a new category (represented as another sheet) in the existing spreadsheet application it not only has to be named but also linked to the summary generating pages with

a number of extra calculations. In the new system this only needs to be done when the summaries are generated and then is done automatically by the system.
The third scenario shows again how the new system has no real advantage over the existing system. The search function is always visible in the footer of the website so is very easy to find and utilise. In the spreadsheet a simple shortcut (CTRL+F) or selection from the tool menu allows access to a more advanced search tool.

The last scenario showed the ease of acquiring a summary through the use of the new system. For the users to complete this task now only requires two clicks of the mouse. With this existing system this almost always required the addition or alteration of spreadsheet formulas, with they new system this is automatically done for all existing categories ensuring all items are accounted for.

Some feedback gained from these scenarios was:

- '*The item creation is easy to use and find*'.
- '*Handling categories is now much easier and faster*'.
- '*Summaries give the needed information quickly*'.

To gain feedback from general catalogue users an email was sent out to the author's peers asking a few simple questions. These questions included asking the users to register with the system and use multiple features. Below are the comments which highlighted concern for some of these features:

- '*After registration I am not logged in…*'

This behaviour of the system was a conscious decision made in the development stages. Making the user login manually after the registration stages ensures that their login identification and passwords were noted.

- '*The [product interest] registering button returns an error message*'.
- '*Item listings may look better in a visible table*'.

This functionality flaw was discussed in section 5.1.2. The visual feedback could be used as a further development when refining the GUI. A simple colour alternating table could create an effect which better distinguishes between items.

5.4. Further Development

Further development of the system would give greater functionality in both the catalogue area and the administration of the stock system, but too much further development would undermine the project aims. When originally looking at the existing solutions for a problem of this nature the author found them to be complex and having unnecessary functionality in some areas, at a higher cost. Over further development might see this system overcomplicate in the same way.

The developments that are left available without undermining the aims remain somewhat limited. The obvious first possibility is to finish the user requirements that were out of scope for a project in these time limitations. This would see further feedback to the user from the system and a more elaborate database system. Also possibly storing slightly more information from the user so that password retrieval could be automated (even using the same emailing system).

Another development that could be implemented is to create more options in terms of how the catalogue is displayed. A development which allowed the administrator to choose between flat or hierarchical ordering would allow the system to be available to more businesses. For this to be fully effective template work would have to be added so business logos and custom text could be easily added and changed as the admin liked.

A development that pushes the boundary of the project aim would be to allow the system some e-commerce functionality. The ability to purchase online would add a new angle for the business to gain revenue from and therefore expand. This development would require a lot of research and programming to ensure the system is created as securely and efficient as possible, definitely something that is out of this projects scope.

5.5. Conclusion

Overall the author considers the system to be a success for the stakeholders. All of the requirements were satisfied along with the objectives of the project. The personal reflection of the author can be seen in Appendix A, which discusses the personal feelings about the project especially in aspects such as time management. The project success was largely due to the prototyping approach used in the development process, allowing the users of the system to have a constant feedback and actually use the system before it was finished. The information given by the users proved to be invaluable for the most part and somewhat stressful in other occasions but overall allowing the system to be developed how the stakeholders envisaged it themselves.

R. References

S. Bennet, J. Skelton and K. Lunn, (2001). **Schaum's Outline of UML**. McGraw-Hill.

R. Elmasri and S. Navathe, (2000). **Fundamentals of Database Systems, Third edition**. Addison Wesly Longman.

M. Eva, (1994). **SSADM Version 4: A User's Guide (McGraw-Hill International Series in Software Engineering)**. McGraw-Hill.

K. Fiduk, S. Kleinfeldt, M. Kosarchyn, E. Perez (1990) *Design methodology management-a CAD framework initiative perspective:* **ACM/IEEE Design Automation Conference**, 27: pp 278-83.

I. Gilfillan, (2003). *PostgreSQL vs MySQL: Which is better?*, URL: **http://www.databasejournal.com/features/mysql/article.php/3288951** [20th April 2005]

R. Lerdorf and K. Tatroe, (2002). **Programming PHP**. O'Reilly.

J. Meloni, (2002). **PHP fast & easy web development**. Premier Press.

MySQL (2005) *MySQL Reference Manual*, URL: **http://www.mysql.com/** [20th April 2005]

N. Patwardhan, E. Siever and S. Spainhour, (2002). **Perl In A Nutshell**. O'Reilly.

PostgreSQL, (2005). *Documentation*, URL: **http://www.postgresql.org/** [20th April 2005]

J. Purtilo, A. Larson, J. Clark, (Oct, 1991). *A methodology for prototyping-in-the-large*: **International Conference on Software Engineering**, pp 2-12.

L. B. S. Racoon, (1997). **Fifty years of progress in software engineering**. SIGSOFT Softw. Eng. Notes, pp. 88-104.

W3, (2005). *Extensible Markup Language*, URL: **http://www.w3.org/XML** [20th April 2005]

Webopedia, (2005). *RUP*, URL: **http://www.webopedia.com/TERM/R/RUP.html** [20th April 2005]

Appendix A – Personal Reflection

Undertaking such a project gave the rare opportunity to see how everything I have learnt in a degree can work together to provide a complete solution. Skills ranging from information system development to programming and database manipulation had to be used to create the system. Looking back at how I performed during the project, if I had another chance, some parts would have been approached completely differently. Advice that I would give to people just starting projects would be as follows:

Value Your Support

Any support that is heading your way should be taken graciously no matter whom it is from. There are times in a project where this support is invaluable. Family and friends are always a big help even if not directly with the task at hand!

Start Early

I imagine that everyone says this but I wish I had listened. The time taken to complete the actual design, implementation and then write the project up is surprisingly phenomenal. Start as early as possible. As seen in appendix C and D there was a large shift in my workload.

User Feedback

Ensure that your users get stuck in to your system. The main reason for this is that your are creating a system for them, so it is possible they will like it more if they get given the opportunity to have constant input into the designs. This at times can be frustrating as the users often change their mind or think of new things which could be implemented. The more that the user feels valued during the development process the more they will be helpful and responsive when the evaluation time comes.

Subject

Choose a subject which you are very comfortable with, both in terms of level of interest and capability. If you are lucky enough to be interested and capable in an area of computing stay in that area; a project of this length will be much more satisfying to carry out with an underlying interest and natural talent in the subject.

Appendix B – Database Creation

MySQL statement to create system database:

```
CREATE TABLE `belongs` (
  `category` int(11) NOT NULL default '0',
  `item` int(11) NOT NULL default '0',
  PRIMARY KEY  (`category`,`item`),
  KEY `item` (`item`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `category` (
  `cat_id` int(11) NOT NULL auto_increment,
  `name` varchar(30) NOT NULL default '',
  PRIMARY KEY  (`cat_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=9 ;

CREATE TABLE `consists` (
  `sale` int(11) NOT NULL default '0',
  `item` int(11) NOT NULL default '0',
  PRIMARY KEY  (`sale`,`item`),
  KEY `item` (`item`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `item` (
  `item_id` int(11) NOT NULL auto_increment,
  `brand` varchar(30) default NULL,
  `name` varchar(30) default NULL,
  `cost_price` int(11) default NULL,
  `retail_price` int(11) default NULL,
  `amount` int(11) default NULL,
  `image` varchar(20) default NULL,
  PRIMARY KEY  (`item_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=8 ;

CREATE TABLE `preference` (
  `category` int(11) NOT NULL default '0',
  `user` int(11) NOT NULL default '0',
  PRIMARY KEY  (`category`,`user`),
  KEY `user` (`user`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `sales` (
  `sale_id` int(11) NOT NULL auto_increment,
  `time` datetime default NULL,
  PRIMARY KEY  (`sale_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

CREATE TABLE `users` (
  `user_id` int(11) NOT NULL auto_increment,
  `name` varchar(30) NOT NULL default '',
  `password` varchar(41) NOT NULL default '',
  `email` varchar(40) NOT NULL default '',
  `admin` tinyint(1) NOT NULL default '0',
  `reg_date` datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=8 ;

ALTER TABLE `belongs`
```

```
  ADD CONSTRAINT `belongs_ibfk_1` FOREIGN KEY (`category`) REFERENCES
`category` (`cat_id`),
  ADD CONSTRAINT `belongs_ibfk_2` FOREIGN KEY (`item`) REFERENCES `item`
(`item_id`);

ALTER TABLE `consists`
  ADD CONSTRAINT `consists_ibfk_1` FOREIGN KEY (`sale`) REFERENCES `sales`
(`sale_id`),
  ADD CONSTRAINT `consists_ibfk_2` FOREIGN KEY (`item`) REFERENCES `item`
(`item_id`);

ALTER TABLE `preference`
  ADD CONSTRAINT `preference_ibfk_1` FOREIGN KEY (`category`) REFERENCES
`category` (`cat_id`),
  ADD CONSTRAINT `preference_ibfk_2` FOREIGN KEY (`user`) REFERENCES
`users` (`user_id`);
```

Appendix C – Project Schedule

	October	November	December	January	February	March	April
Preliminary Investigation	11th	26th					
Research	11th	🟦	3rd				
Mid-Project Report			9th-17th				
Host Interviews				1st-14th			
Produce Software				7th	1st		
Software Modification					1st	1st	
Write Final Report						1st	20th

Appendix D – Revised Project Schedule

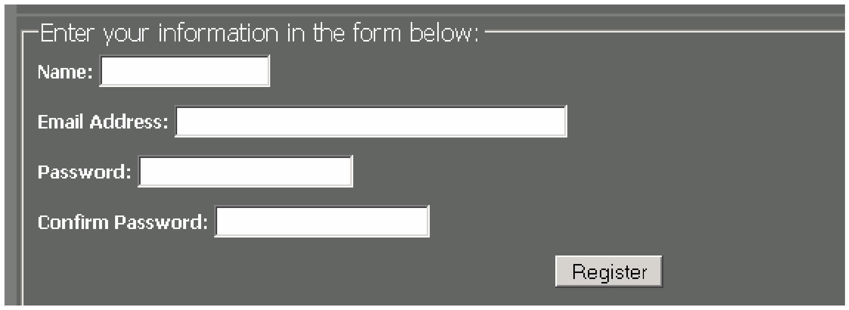
	October	November	December	January	February	March	April
Preliminary Investigation	11th	26th					
Research	11th	🟦	3rd				
Mid-Project Report			9th-17th				
Host Interviews				1st	🟩	🟩	15th
Produce Software				7th	🟥	5th	
Software Modification					1st	🟦	5th
Write Final Report						1st	26th

Appendix E – User manual

User guide

Register with the system:

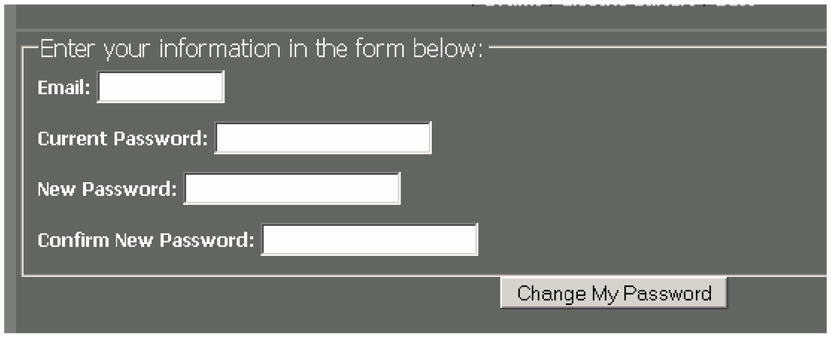
1) Click register in the main menu.
2) Complete the form seen below:



3) Click the register button.
4) If successful the user will be prompted with a success message.

Change password:

1) When logged in click the profile item in main menu.
2) Fill out the form below and click the button to continue:



3) Users will be prompted with a message if successful.

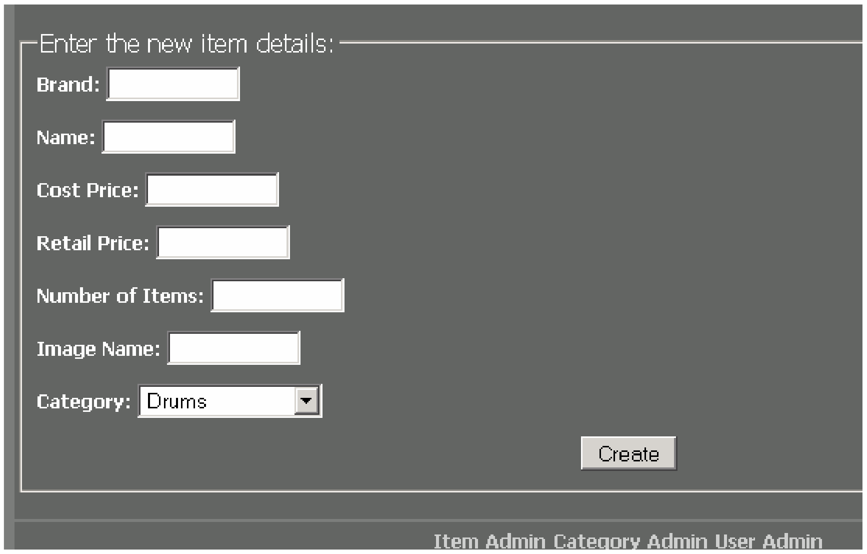
Change personalisation details:

1) When logged in click the profile item in the main menu.
2) Select from the list of existing categories the preferred ones.
3) Click the button to save the changes.

Admin Guide

Add New Item

1) When logged in as an administrator, select the ‘Item Admin’ from the footer menu.
2) Fill out the form seen below:



Enter the new item details:

Brand:

Name:

Cost Price:

Retail Price:

Number of Items:

Image Name:

Category: Drums

3) Click the button to confirm the item creation.

Remove Item

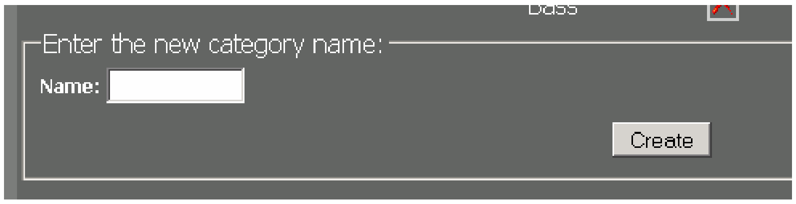
1) When logged in as an administrator, select the ‘Item Admin’ from the footer menu.
2) With the current available items listed select the ✗ icon next to the relevant one.

Edit Item

1) When logged in as an administrator, select the ‘Item Admin’ from the footer menu.
2) Click the ‘Edit’ text next to the relevant item then repeat the form process from ‘Add New Item’ above.

Add Category

1) When logged in as an administrator, select the ‘Category Admin’ from the footer menu.
2) Fill out the form seen below:



Enter the new category name:

Name:

Remove Category

1) When logged in as an administrator, select the ‘Category Admin’ from the footer menu.
2) With the current available categories listed select the ✕ icon next to the relevant one.

Edit Category

1) When logged in as an administrator, select the ‘Category Admin’ from the footer menu.
2) Click the ‘Edit’ text next to the relevant item then repeat the form process from ‘Add Category’ above.

Remove User

1) When logged in as an administrator, select the ‘User Admin’ from the footer menu.
2) With the current available users listed select the ✕ icon next to the relevant one.