Object Oriented Software Engineering – Excercise 1
A pharmacy Management System

Liran Zvibel, 040015059 Liel Man, 029517968

December 6th, 1999

Contents

Please note that the IEEE Std. 830 [2] specify several different ways to present the third chapter (Specific Requirements), and we chose the option discussed in section 5.3.7.7 (Functional Hierarchy) and then shown in appendix A.7 because it suits better the functional analysis we had to perform for this exercise.

The files (this LaTeX file and the figures) of this exercise can be found at the directory `~liranz/OOSE/ex1`.

Chapter 1

Introduction

This section contains an overview of the complete document.

1.1 Purpose of SRS

This is the Software Requirement Specification for the *pharmacy management* system. The purpose of this document is to convey information about the system's requirements, both functional and non-functional, to the reader. This document provides (a) a description of the environment in which the system is expected to operate, (b) a definition of the system's capabilities, (c) a specification of the system's functional and non-functional requirements, in cluding the (d) user-interface requirements and (e) a list of tentative future extensions and enhancements to the system.

This document is intended to several groups of readers. First, it is anticipated that the SRS will primarily be used by the application designers. Designers will use the information recorded here as the basis for creating the system's design. This SRS will also serve to establish a basis for the SRS of the pharmacy network management to be produced in the future to fill the missing part of this system. Third, the system maintainers will review the document to clarify their understanding of what the application does. Fourth, test planners will use information from this document to derive test plans and test cases. Finally, the project manager will use this document during project planning and monitoring.

1.2 Scope

The purpose of this software development project is to create a new system called: *Pharmacy Management*. The client for this project, The pharmacy network "Libriut", wishes to enhance its current outdated paper based pharmacy management system. The *Pharmacy Management* system should allow the complete management of all the pharmacy activities, excluding the cash registry operation, and to allow better cooperation of different pharmacies of the network. The application will provide the following capabilities:

- The system shall handle customer purchased.
- The system shall manage the inventory information.
- The shall manage all the interaction with the suppliers, including payments and special orders.
- The system should be able to prepare monthly reports.

The system will provide the following benefits to the client:

- All the pharmacies share their information, so every pharmacy has less management work to perform. Setting up a new pharmacy will also be easier for this reason.
- Since the computerized system will perform all the work, the pharmacist will have more time to help the customers.

- There will be a more accurate way to handle orders from the clients (as opposed to the pharmacist writing the name of the patient and the name of the medicine on a piece of paper).

Since the pharmacy network currently already have a relationship with a cash register provider, and all the branches already have an electronic cash register our system will not replace the cash register system (it will, however have to communicate with it).

1.3 Definitions, Acronyms and Abbreviations

The following is a list of definitions that should clarify the vocabulary to be used in this document:

- **Product –** Some goods that can be bought from a regular pharmacy (either medicine, first aid kits, bandages
- **Branch** – Every pharmacy store of the network will be called a branch.
- **Customer –** A person that buys at one of the branches.
- **Regular Customer** – A customer that regularly buys at the branch. Only workers of the network and chronic customers may become regulars. Every customer has one *home branch* with an account in that branch.
- **Home base –** The location of the main database servers for the network. This location might be in one of the bigger branches, and it might not be located in a branch at all.
- **WAN** – Wide Area Network. A (computer's) network that connects two geographically dispersed sites.
- **LAN** – Local Area Network. A local area network is a small network of interconnected workstations and associated devices that share the resources of a single processor or server within a small geographic area
- **Frame Relay** – Frame relay is a telecommunication service designed for cost-efficient data transmission for intermittent traffic between local area networks (LANs) and between end-points in a wide area network (WAN).
- **Barcode –** A barcode s the small image of lines (bars) and spaces that is affixed to retail store items, identification cards, and postal mail to identify a particular product number, person, or location.
- **Barcode reader** – An instrument that can optically read barcodes off products/cards and transfer the serial number associated with the product to the computer.The reader uses a laser beam that is sensitive to the reflections from the line and space thickness and variation. The reader translates the reflected light into digital data that is transferred to a computer for immediate action or storage.
- **Digitizer –** A touch-sensitive screen.
- **Database –** A database is a collection of data that is organized so that its contents can easily be accessed, managed, and updated.
- **Distributed Database –** A distributed database is one that can be dispersed or replicated among different points in a network.
- **RAID –** Redundant Array of Inexpensive Disks. A way to store logically integrated information on several small disks in a way that improves the performance of the whole system and provides some protection against disks failures.

1.4 References

Other documents or material of which the reader should be aware:
Please note the bibliography section at the end of this document.

- The SRS document for the home-base system to be coupled-developed with the system proposed here. The home base system holds the management of the distributed DB and reports.
- The protocol for the health care system.

1.5 Overview

A brief description of the content of each chapter is given below:

Chapter 1 : Introduction – Provides an overview of the project. Summarizes the major capabilities of the project.

Chapter 2 : General Description – The definition of requirements. Presents the environment in which the application is expected to operate, provide an overview of the system requirements, describe assumptions about possible users of the application, possible constrains on the project, and the underlying assumptions that on which the requirements analysis is based.

Chapter 3: Specific Requirements – The specification of requirements. Contains a complete description of the application's requirements, both functional and non-functional.

Chapter 4 : Possible Product Evolution – Provides a list of features or capabilities that are expected to change or be added in the future.

Chapter 2

General Description

This chapter will help the reader to become familiar with the application to be built. Both the functional and non-functional requirements are summarized in this chapter. Chapter 3 serves to present the actual requirements.

2.1 Product Perspective

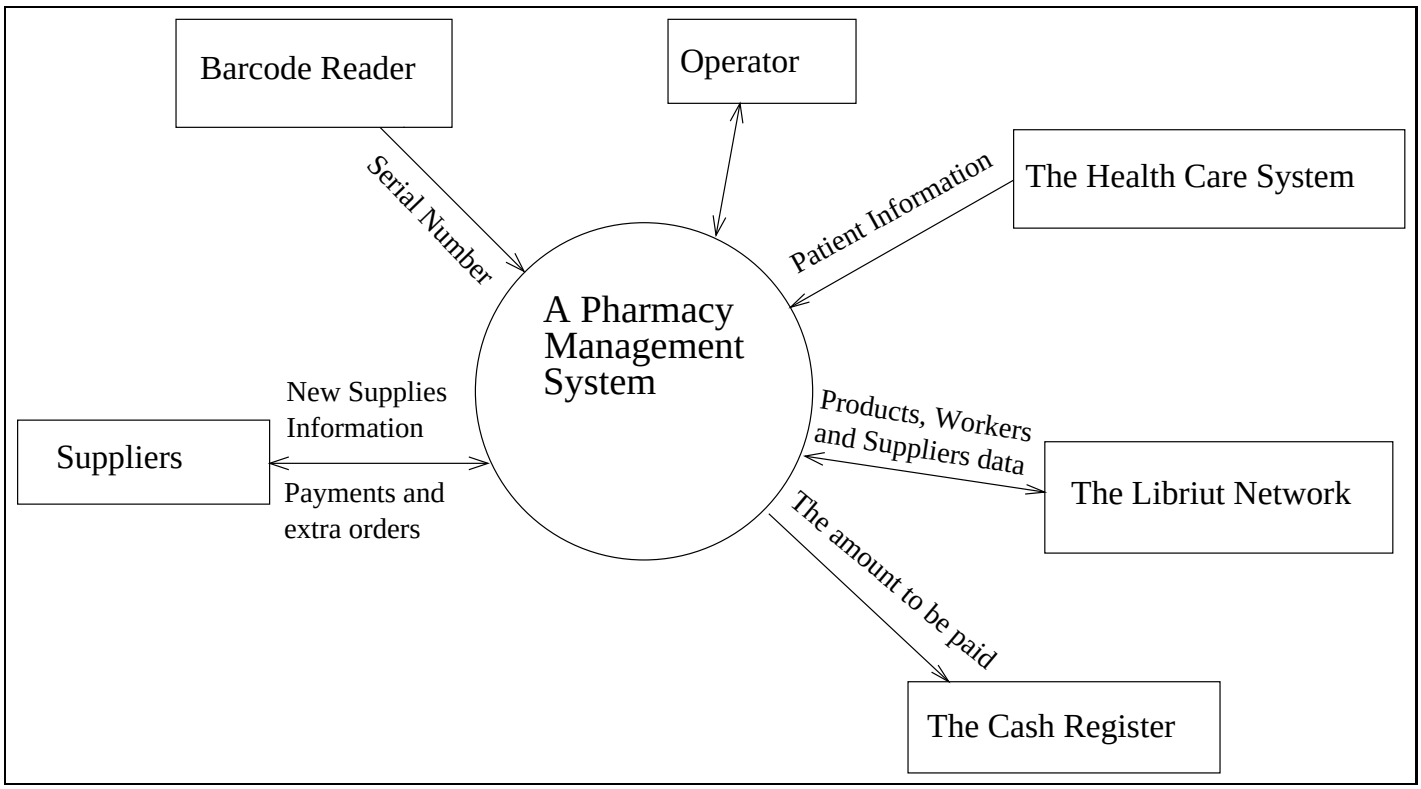

Figure 2.1: The Context Diagram

The Context Diagram for the system for the *Pharmacy Management System* is shown in Figure 2.1. It models the important ways of acquiring data to and from the system. Each entity is described below:

The Libriut Network This is the headquarters of the network. The database with the workers, suppliers and approved products are located here. In other parts of this document, it is also known as the "home-base".

The health care system This is the centralized location of all the patients of the country. It contains all the relevant information about all the patients in the country. It is used when a customer wishes to become a regular-customer, and some of his/her detailed have to be checked (i.e. whether s/he is a chronic patient or not, etc...)

The Barcode Reader An interface that optically read codes and transfers them to the computer. It is used when the customer has an optical I.D card, or when retrieving products information.

Operator Is either the pharmacist when s/he sells products to customers, part of the accounting personnel preparing reports or paying to suppliers, or part of the inventory personnel when updating the products information.

Suppliers The suppliers supply the products to the pharmacy system. The interaction with the system is either when the system calls them to notify them that some product is missing, or when they provide the monthly report.

Customers Do not interact with the system actively, but give and retrieve information through the pharmacist.

The Cash Register When the pharmacist finished inputting all the products the customer wishes to buy, the amount the customer has to pay is transferred to the cash register.

As seen from the context diagram, the pharmacy management system is not self-contained, in the meaning that it uses some other systems to perform its operation. Those interfaces will now be described:

2.1.1 System Interfaces

The *pharmacy management system* uses two other major systems to perform its work. The first is the *Pharmacy Network Management System*, which operates at the home-base of the pharmacy network, and its function is to synchronize between the different pharmacy branches, and store all the common tables and thus create a huge distributed database for the whole network to use. The second interface is the *Health Care Management System* which stores all the information about the patients. The pharmacy uses that system to retrieve information about the customers.

2.1.2 Hardware Interfaces

The pharmacy system will consist of one main server, and several terminals, each of those should have a different configuration. It is left to the designers to decide whether the terminal should be "*smart*", that is run the client programs, or "*dumb*", that is, the terminals are only there to present and gather information to/from the user.

The server should support at least up to 20 terminals.

The Server Configuration

The server will store all the information of the system (contain the database), connect to the home-base and call the suppliers when some extra-products are needed. Thus it should support the following extra interfaces:

- **A WAN communication card** – To communicate with the home-base. Please note that it is up to the designers to decide whether the system will be **Frame Relay** based, **ATM** based, **xDSL** based or any other appropriate interface. The interface should support IP based networks.
- **A Fast-Ethernet Card –** To communicate with the clients. Note that even though there are other LAN interfaces, a Fast-Ethernet is the most wildly used one currently, and it is not left for the designers to decide.
- **A RAID –** To support all the databases. A RAID is needed since regular disks, although satisfying the storage needs, don't satisfy the stability needs. The chosen RAID should have easy means of storage upgrading and disk replacing.
- **A Voice Enabled Modem** To perform the ordering of extra products when needed.

The Terminal Configuration

The clients will gather the information from the user, and present it to him/her in a graphical way. The terminal will have to support the following interfaces:

- **A Fast-Ethernet Card –** The the LAN connection.
- **A Digitizer –** A touch-aware graphical screen to help the user navigate through the menus, and choose between different choices.
- **A keyboard –** To type in information about products, customers, suppliers, etc....
- **A Barcode Reader** – To easily input serial-numbers or customers I.Ds.
- **A cash Register Connection –** To easily (and accurately) transfer the amount to be paid to the cash register.

Please note that since a digitizer is a required part of the system, there is no need for a mouse.

2.1.3 Software Interfaces

This section specify the required software packages to use with the system. That is, all the software that is not developed as a pert of the pharmacy management system.

Operating System

The operating system to be chosen should support all the hardware interfaces mentioned above (no device-level design and implementation will be performed for this project). The system should be stable and require minimum administarive care during operation.

Database System

Any database system that supports distributed databases will be suitable for this project. The designers should take into consideration the efficiency of the database system (take into consideration the TCP-C and TCP-D benchmarks, for example).

It should be taken into consideration that all the previous projects of this software company used Oracle as the database platform, and that is, any choice other then Oracle should be adequately explained.

Modem controlling software

The designers should either choose a library that deals with with the modem (and fax) or an external program that can be used (since it is located on the server, and the users should not work on the server, it does not matter that an external program is used.)

The most important consideration is development time. The designers should choose the option with the minimum (approximated) needed development time.

Voice Synthesis Software

To be used for ordering extra quantities of some products. Same requirements as of the previous section.

2.1.4 Communication Interfaces

There are three different communication interface levels:

- **The Physical Level**
  - **A WAN Communication Interface –** To connect the different branches of the network.

- **A LAN Communication Interface** – To connect the terminals to the server. Fast-Ethernet will be used.

- **The line level** – The IP protocol. The terminals will use this protocol to transfer the information to the server.

- **The Database Transaction and distributed protocol** – The protocol used by the database to synchronize the database-server and database-clients.

2.2 Product Functions

The purpose of the *Pharmacy Management System* is to allow the pharmacist to easily sell the products. Thus the system should perform most of the time consuming operation itself, in order to save the pharmacist time. It should be noted that even though some pharmacy shops do have an in-house accounting and inventory departments, most of the pharmacies do not, and the pharmacist should perform some administrative tasks as well.

A summary of the system's functionality is given below:

- The system should handle all customer purchases, and log them to a database (keeping track of the inventory).

- The system should also handle all the pharmacy purchases and orders from the suppliers, including paying the suppliers.

- The system should also manage pre-ordering of some products (if a product ran out, and a customer would like to order it). Further more, the system should also manage the list of awaiting customers.

- The system should support information collaboration with other pharmacies of the same network, i.e. all of the pharmacies will use the same medicine and suppliers databases which will be based in a central location.

- Regular Customers may choose not to pay for the current bill, and add it to their account.

- The system shall produce monthly reports, with extensive information about the status of the inventory, payments to the suppliers, purchase made by customers and the unbalanced regular customers' accounts.

2.3 User Characteristics

The users of the system are not experienced computer users. The current pharmacy management system is paper based. The system should be very easy and intuitive to use. Furthermore, the system should include extensive help screens, help pop-up messages and other means to help the users use the system.

The "Libriut" network will train all the workers before the system becomes functional, and then the system will be at the pharmacies for at least a month before it is operational for the staff to experience with it.

2.4 General Constrains

2.4.1 Interfaces to other systems

If either the Pharmacy Network Management System or The Health Care System fails, the pharmacy management system should still work. It may not provide all the information, and allow to perform all the operations, but the following operations should still be (even partially functional):

- Customer Purchases.

- Inventory Updates.

- Regular Customers adding current bills to account.

2.4.2 Security

The system should support several different levels of security. A pharmacist may only sell the products, the people from the accounting department may only pay the suppliers and the people from the storing-room may only update new status on products.

It should be noted, though, that in small pharmacies, one person might have several responsibilities, and s/he should be able to perform those using his/her login name.

2.5 Assumptions and Dependencies

The requirements analysis for this application was undertaken based on the following assumptions. Should any of these assumption later change or prove to be unreasonable, it will likely cause changes to occur in the application's requirement.

- For now, a regular-customer will be able to pay his/her account only after performing a purchase.
- For now, each branch will pay the suppliers separately.
- For now, a regular customer of one branch cannot add to his/her account purchases from another branch.

Chapter 3

Specific Requirements

This section contains all the details which are relevant for the design phase to follow. Specific requirements should be such that one may objectively determine whether they are fulfilled or not.

3.1 External interface requirements

This section provides a detailed description of all inputs into and all outputs from the software system.

3.1.1 User Interfaces

The system's user interface will be based on graphical digitizers, and the user interface design should take it into consideration. Since there are several levels of security every screen will show only the options that are available to that kind of user. The navigation between different parts of the system should be through touchable numbered menus, if the user wishes s/he can touch the option (which should be wide enough to be easily touched), or type the number on the keyboard.

Each screen should also have a help screen associated with it. The help screen will appear if the user clicked the help button, which will always be at the same location on the screen. Another way to get into the help screen for the current screen is to type F1 on he keyboard.

Whenever a data item can be inserted using the barcode reader an indicator should pop-up on the screen to notify the operator. The system will automatically use the information gathered from the barcode reader. In cases when the user both clicked information on the keyboard, and inserted a serial number from the barcode reader, the value of the barcode reader will be used, and the user should be notified (the notification should be "passive", that is, the user don't have to acknowledge it).

Please note, that since this is the first computerized system maid for pharmacies, and there is not enough information about efficient user interfaces for pharmacies (that is colors that look good in the lighting conditions, size of buttons, representation of the menus, etc...), the user interface might (and possibly will!) be changed after the first version, thus the design of the system should allow an easy modification of the UI.

3.1.2 System Interfaces

This system has two system interfaces. One is the health-care system, that provides information about the patients. Such information is needed in order to decide whether a customer can become a regular-customer. The second system is the pharmacy management system that stores all the information that can be shred between different branches of the pharmacy network.

3.1.3 Hardware Interfaces

There are two types of hardware interfaces in the system. One is for the server of the branch, and the other describes the end machine (terminal).

The server configuration

The server shall store all the information of the branch (contain the database for this current branch), all the database transactions will be performed on that machine (even if the designed choose to use "smart" terminals), since the database server will be running on that machine. This machine will also maintain the connection to the home-base and the health-care network. The following extra-interfaces shall be supported by the server machine:

- **A WAN communication card** – In order to communicate with the home-base and the health-care system. This document does not specify the low-level protocol to be used (even though the chosen protocol should not be worth in latency and bandwidth consideration then the following candidates: Frame Relay, ARM, xDSL).

  The designers should take into consideration that if the card WAN communication card will be installed in the server, the server will also have to be the router of the LAN, and it might decrease its performance. The designers might consider adding a router to the system. Even though it is a new element it is considered as part of the server.

- **A Fast-Ethernet Card** – To communicate within the LAN the the branch. Since a Fast-Ethernet cards are so wildly used these days, all the modern operating systems support them.

- **A RAID** – To support all the databases. The chosen RAID interface should be supported by the operating system used, and be used as any regular disk.

  The chosen RAID should support hot disks addition and replacement, snapshots, and direct connection to a backup device. The designers should also choose a backup device that works with the chosen RAID. Backups of the database will be applied manually by the operators of the system (unless backups are done manually, people often forget to replace the backup tape.)

- **A Voice Enabled Modem** – When there is a need to perform a special order of some products, the system will automatically call the supplier with the modem. The modem should be voice enabled since the transaction will be performed automatically using a voice synthesis system.

The Terminal Machine

- **A Fast-Ethernet Card** – To be used for the internal communication between the terminal and the server (the branch'es LAN).

- **A Digitizer** – The graphical user interface will be based on a touch-aware screen. The system should support it.

- **A keyboard** – To type in information.

- **A Barcode Reader** – To easily input numerical information to the system. The main considerations when choosing the barcode reader should be its accuracy and ease of programming.

- **A Cash Register Connection** – This interface is used to transfer the amount of the current purchase to the cash register. The pharmacy already have an electronic cash registers in all the branches, and this system should interface with the model used by the "Libriut" network.

3.1.4 Software Interfaces

This section describes the interface to all the software systems that are used in the *Pharmacy Management System* but not an integral part of it (that is, developed independly of the system).

3.1.5 Operating system

The operating systems (the server and the terminals might have different operating systems) chosen should support all the hardware and software requirements proposed in the rest of this document. For stability reasons, the operating systems chosed should be at least one-year old, and used in other distributed projects. Please note that it is advised to choose the same operating system for the server as the operating system chosen for the "Pharmacy Network Management System".

All the device drivers for the hardware and other software interfaces should be either provided with the operating system (or by the vendor of the operating system), or be developed by the vendors of the interfaces (In this case, the vendor should explicitly state that the device driver was written for the operating system type and version used in this project.)

Database System

The database chosen should support distributed databases. The database should also score well in some OLTP (On Line Transaction Processing) benchmarks such as TPC-C and TPC-D, and allow for dynamic reallocation of the tables (it is of no use that our RAID can add additional space on-the-fly, but the database system will not be able to use it.)

Up until now, all the projects performed by this company used the Oracle database server, which supports all the needed features. Should the designers of the designers of the system decide to choose another database, they will provide an additional document describing the reasons of their decision.

Please note that the database used in this project has to be the same one used in the *Pharmacy Network Management System*.

Modem controlling software

The designers should choose a library that deals with the modem. The modem will be used to perform the special orders in the middle of the month through a modem connection or a phone call using a voice synthesis system. The modem will also be used to send the monthly fax to the supplier with the regular order for that month.

Since the users do not directly interact with the server, the modem controlling software may either be a library, or a software that can be controlled from another program.

Voice Synthesis Software

This software will be used when performing special orders from the suppliers. The system should be able to produce an understandable voice. The program/library used should allow the tuning of the frequency of the voice. Anyway the interface to the voice synthesis system should be performed in such a way that another engine could be easily added to the system.

3.1.6 Communication Interfaces

The communication interfaces supported by the system will be the WAN communication protocol, the LAN communication protocol (Fast-Ethernet), and the IP protocol for the higher level communication. The system shall use the communication protocol provided by the database vendors to interact with the database, and an internal protocol shall be designed to communicate between the terminals and the server. It should be noted, that some systems, such as the X Windowing system already support that kind of operation, and this protocol should be considered.

Another communication protocol shall be developed in conjunction with the *Pharmacy Network Management System* to connect the pharmacy-branches to the home-base.

3.2 Functional Requirements

3.2.1 Information Flows

In this section all the Data Flow Diagrams are presented in a DFS order. Data types are explained in the uppermost DFD where they introduced to the system.

Data Flow Diagram 0

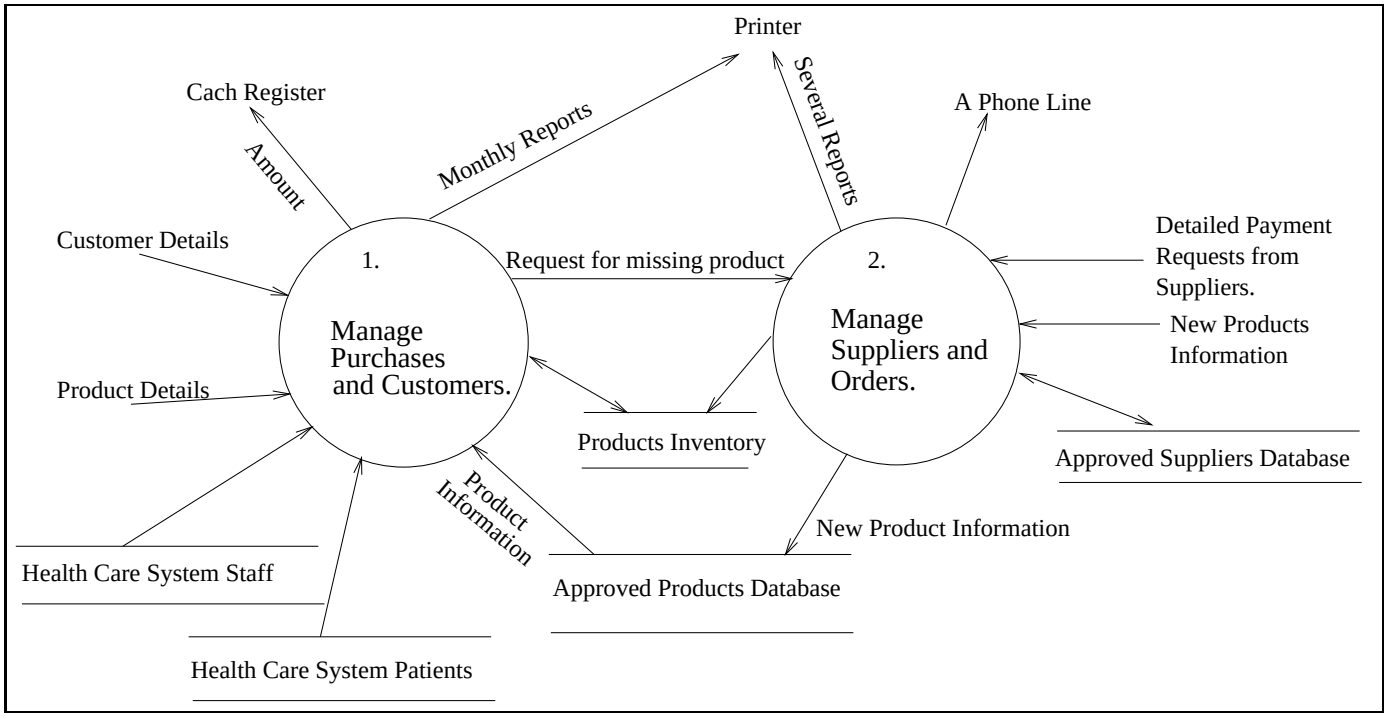

Figure 3.1: DFD 0 – The Pharmacy Management System

1. **Data Entities**

- **Health Care System Staff** – An external database (this database is part of the health care system, and contains all the staff of the health care network, and the "Libriut" pharmacy network which is part of the health care network.
- **Health Care System Patients** – An external Database (located at the health care network home-base) that contain information about all the patients of the network.
- **Approved Products Database** – An external database (located at the "Libriut" pharmacy network home-base), that contains information about all the approved products (that is, products that can be sold at the "Libriut" pharmacies.
- **Approved Suppliers Database** – An external database (located at the "Libriut" pharmacy network home-base) that contain all the suppliers that can supply products to the Libriut pharmacies.
- **Products Inventory** – An internal database, that contain information about the status of all the products within the current Libriut branch.
- **New Product Information** – When a new product is realized into the system. It is enough that one branch enter the data, and the rest of the branches will be able to use it.

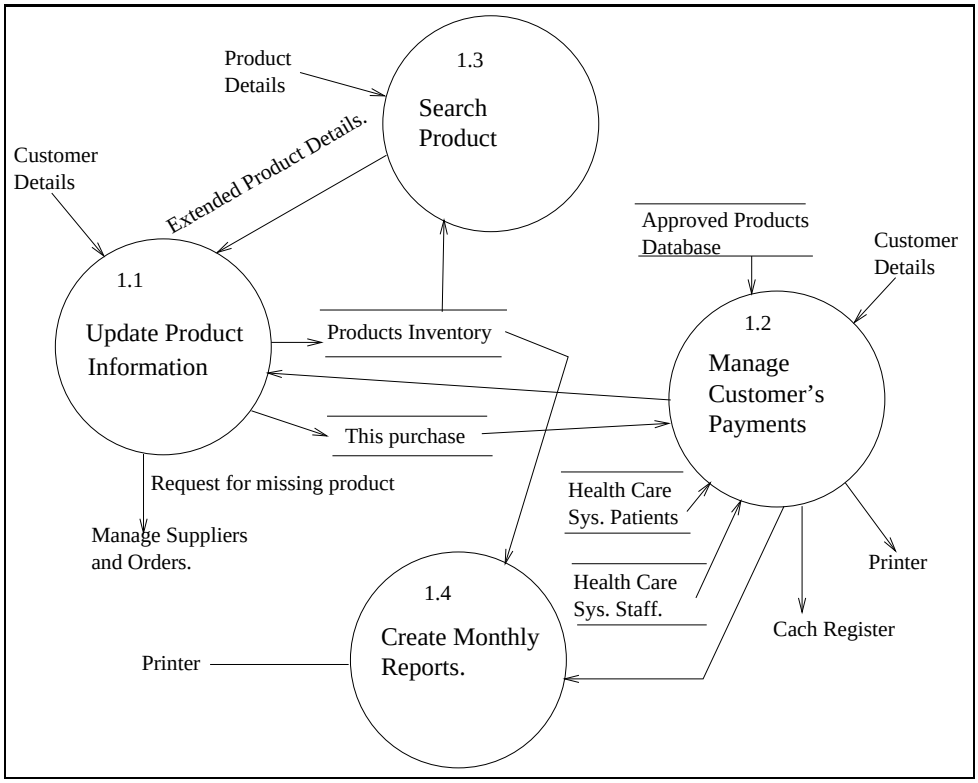

Figure 3.2: DFD 1 – Manage Purchases and Customers

- **Detailed Payment Request from suppliers –** At the end of every month the suppliers send the branch a detailed payment request, containing all the products that the supplier supplies, and the sum of all the product costs.
- **Printer –** The printer can print either regular reports, or special printable cheques as payment for the suppliers.
- **A phone Line –** Is used when ordering extra products, and for the normal orders.
- **Cash Register –** The system transfers the amounts to be paid to the cash register.
- **Customer Details –** The details of the customer that perform the purchase.
- **Product Details –** The details of the product that is currently added to the total sum.

2. **Pertinent Processes**

   (a) **Manage Purchases and Customers –** Perform all the needed computation included in a customer purchase, and in addition/removing/updating regular-customer information.

   (b) **Manage Suppliers and Orders –** Perform all the needed computations included in paying the suppliers, ordering products and updating the inventory.

3. **Topology** – The product inventory database is used by both processes (to update the inventory information when a purchase was made, or a shipment arrived).

   When a product is not present in the inventory, and the customer would like it, a special order is performed, but since the second process handles the orders, the information is moved there.

Data Flow Diagram 1 – Manage Purchases and Customers

1. **Data Entities**

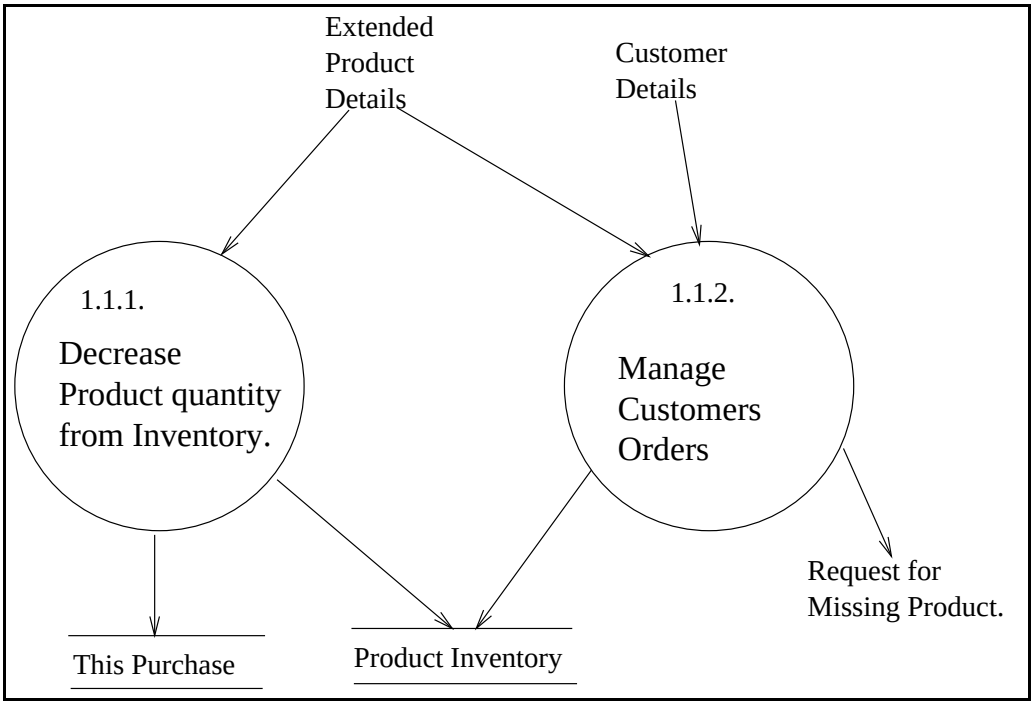

Figure 3.3: DFD 2 – Update Product Information

- **This Purchase –** Contains information about this purchase.
- **Extended Product Details** – Contain both the quantity that the customer buys, and the quantity that s/he wishes to order.

2. **Pertinent Processes**

   (a) **Update Product Information** – Updates the status of the product in the inventory, and updates the current purchase. If the product is not in the inventory orders it.

   (b) **Manage Customer's Payments –** Produce the payment information, updates the purchases database, lets regular customers move the current purchase into their account or lets them pay all their debts. It also manage the regular-customers database.

Data Flow Diagram 2 – update Product Information

1. **Data Entities**
   None new.

2. **Pertinent Processes**

   (a) **Decrease Product Quantity from Inventory –** Decreases the quantity of the current product from the inventory database, and add the quantity and the product information to the information about this purchase.

   (b) **Manage Customer Orders –** Adds the customer and the quantity to the waiting list of the current product. Since this process don't have information about the suppliers, it also notify the *Managing Suppliers and Orders* process that an order should be maid.

Data Flow Diagram 2 – Manage Customer's Payments

1. **Data Entities**

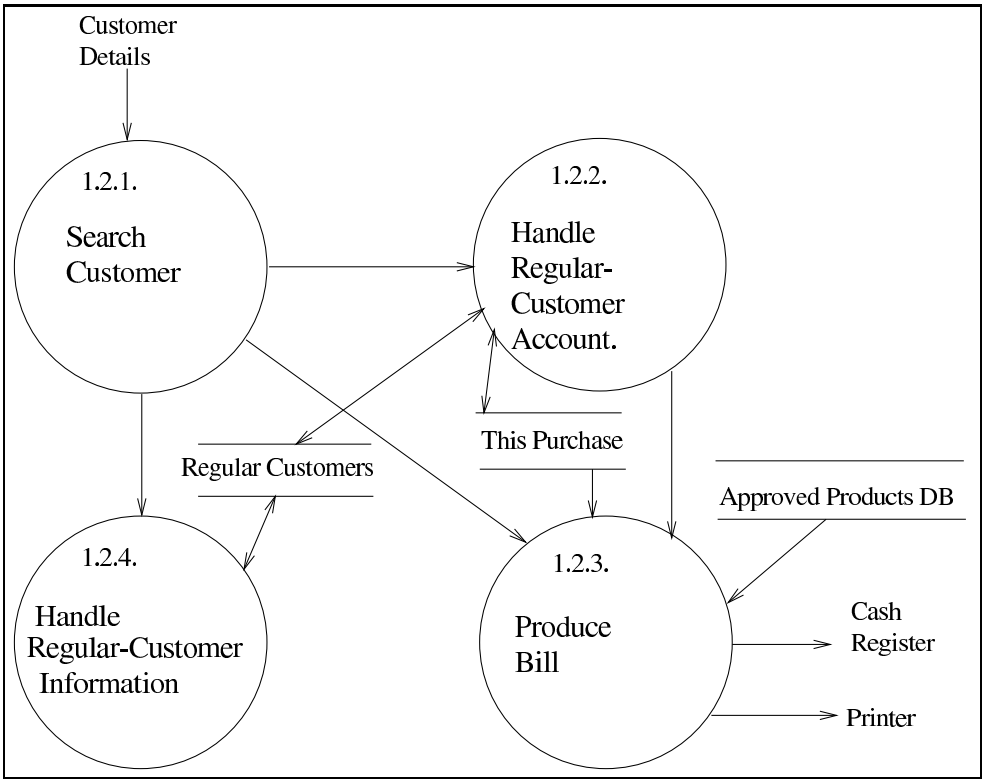

Figure 3.4: DFD 2 – Manage Customer's Payments

- **Regular Customers –** The internal database of the regular customers.

2. **Pertinent Processes**

   (a) **Search Customer –** This process takes any form of customer detail, and outputs the serial number of the customers, or returns that the customer does not appear in the database.

   (b) **Handle Regular-Customer Account –** checks whether this customer has other unpaid purchases, and add them to this purchase, or add this purchase to the customer's account.

   (c) **Produce Bill –** Outputs the amount to the cash registry, and prints the information of this purchase.

   (d) **Handle Regular-Customer Information –** Adds/ Removes and updates regular customers.

Data Flow Diagram 3 – Search Customer

1. **Data Entities**
   None New

2. **Pertinent Processes**

   (a) **Search By Name –** Searches the customer by name. It returns the same details as serial number s/he it is a regular-customer, by ID if just a patient, and no information if the patient was not found.

   (b) **Search By serial –** Verifies that the serial number exists. Outputs as by Name.

   (c) **Search By I.D. –** Like by name, but just by I.D.

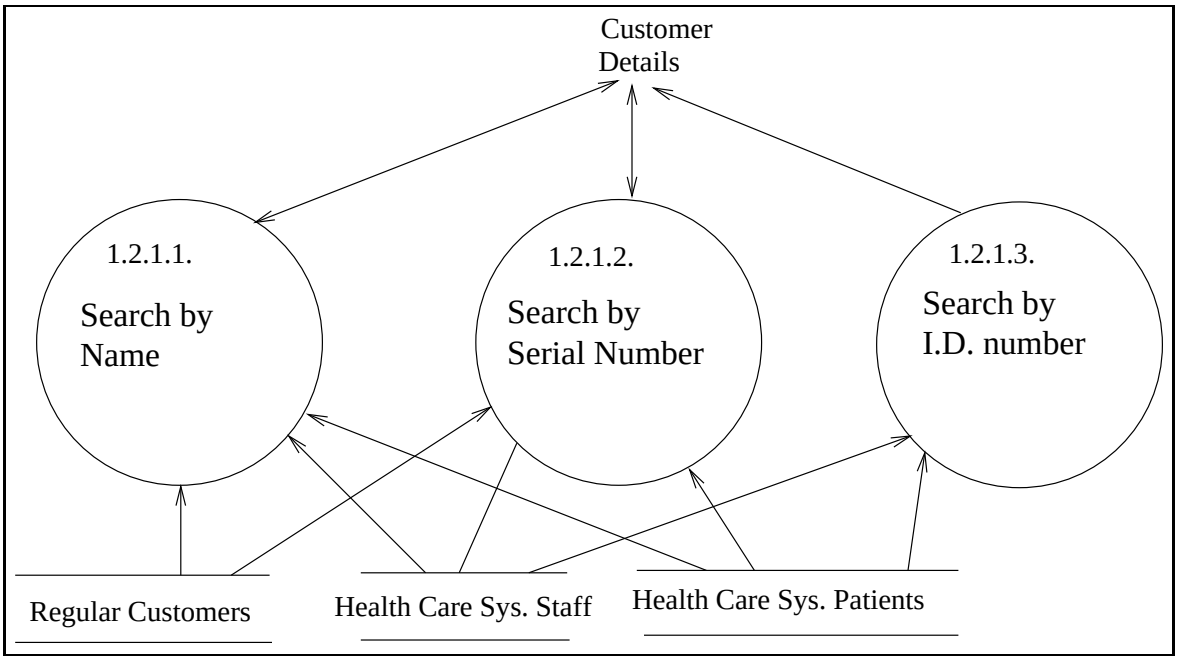

Figure 3.5: DFD 3 – Search Customer

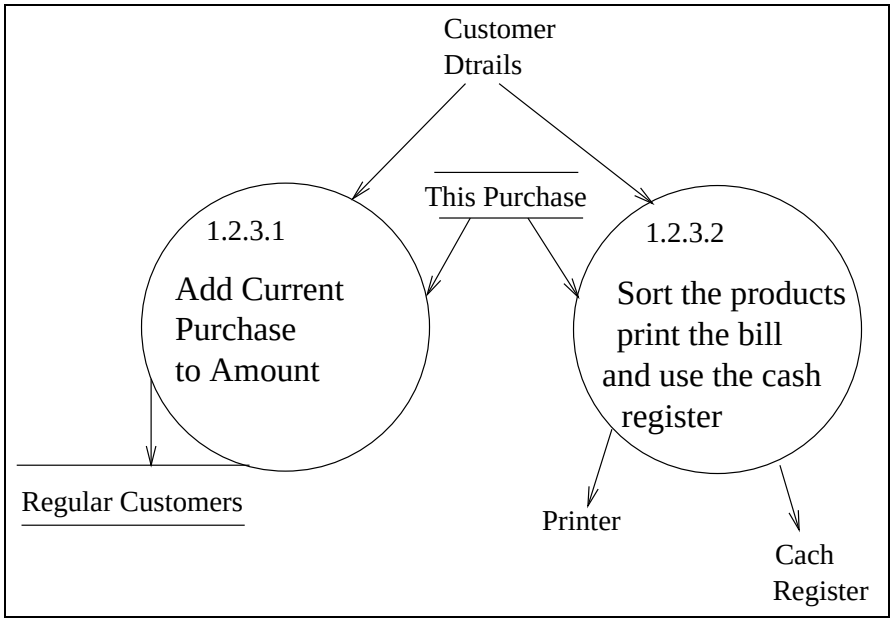

Figure 3.6: DFD 3 – Produce Bill

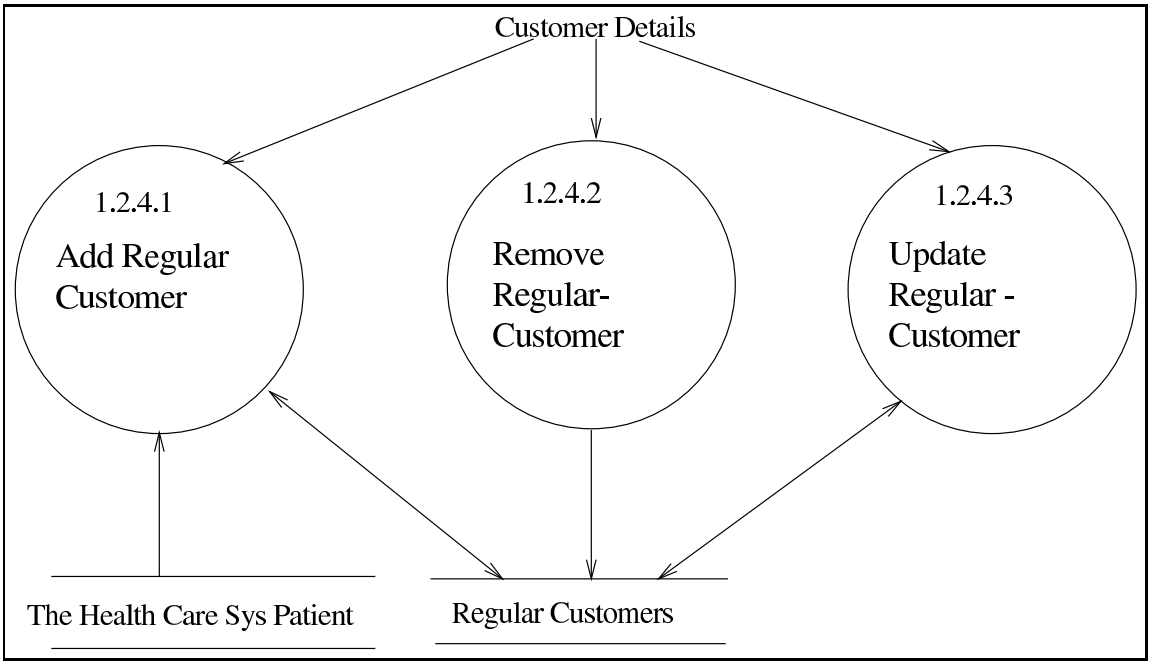

Figure 3.7: DFD 3 – Handle Regular-Customer Information

Data Flow Diagram 3 – Produce Bill

1. **Data Entities**
   None New
2. **Pertinent Processes**
   (a) **Add current purchase to amount** – adds the current amount to the account of the customer.
   (b) **Sort the products, print the bill and use the cash register** – Does what the name says.

Data Flow Diagram 3 – Handle Regular-Customer Information

1. **Data Entities**
   None New
2. **Pertinent Processes**
   (a) **Add Regular Customer**
   (b) **Remove Regular Customer**
   (c) **Update Regular Customer**

Data Flow Diagram 2 – Search Product

1. **Data Entities**
   None New
2. **Pertinent Processes**
   (a) **Search By barcode**
   (b) **Search by Name**
   (c) **Show location on shelf**
   (d) **Check whether a recipe is needed**

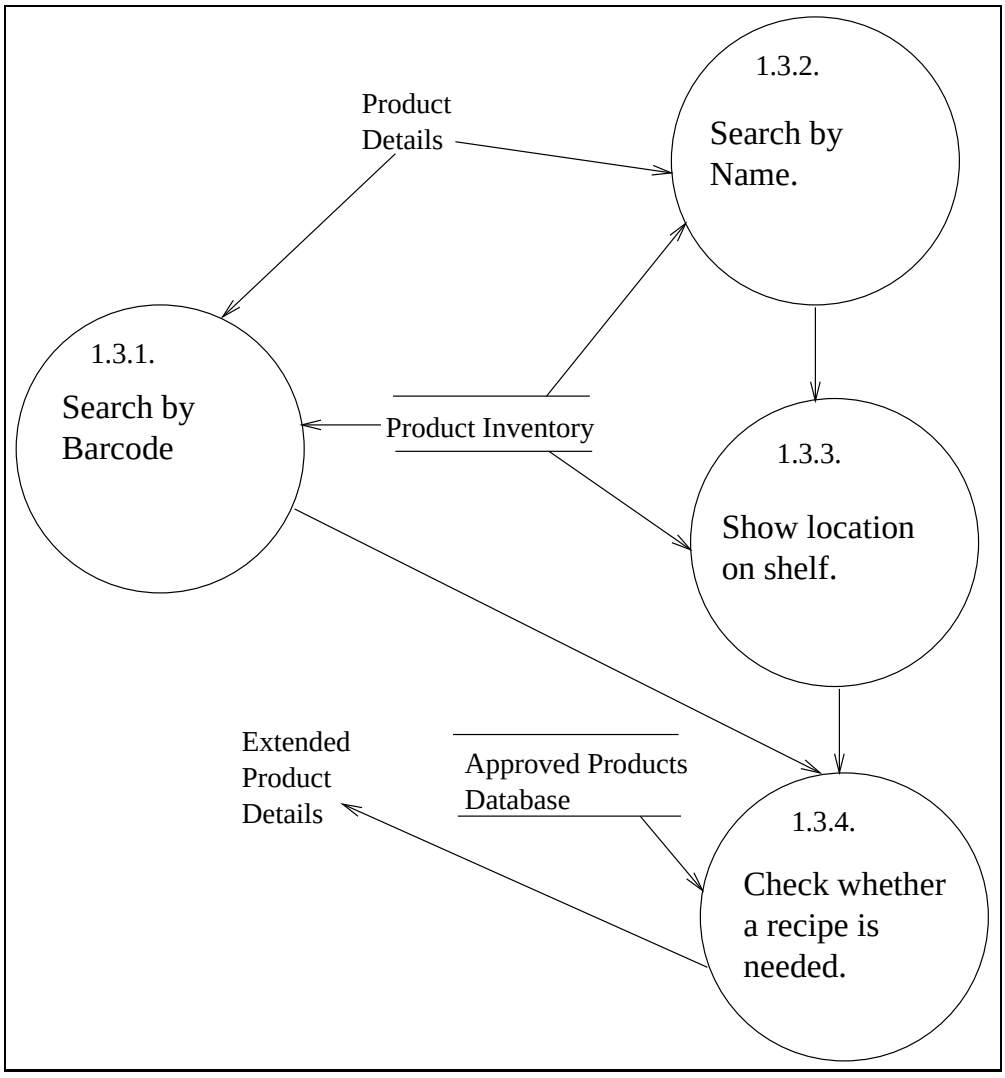

Figure 3.8: DFD 2 – Search Product

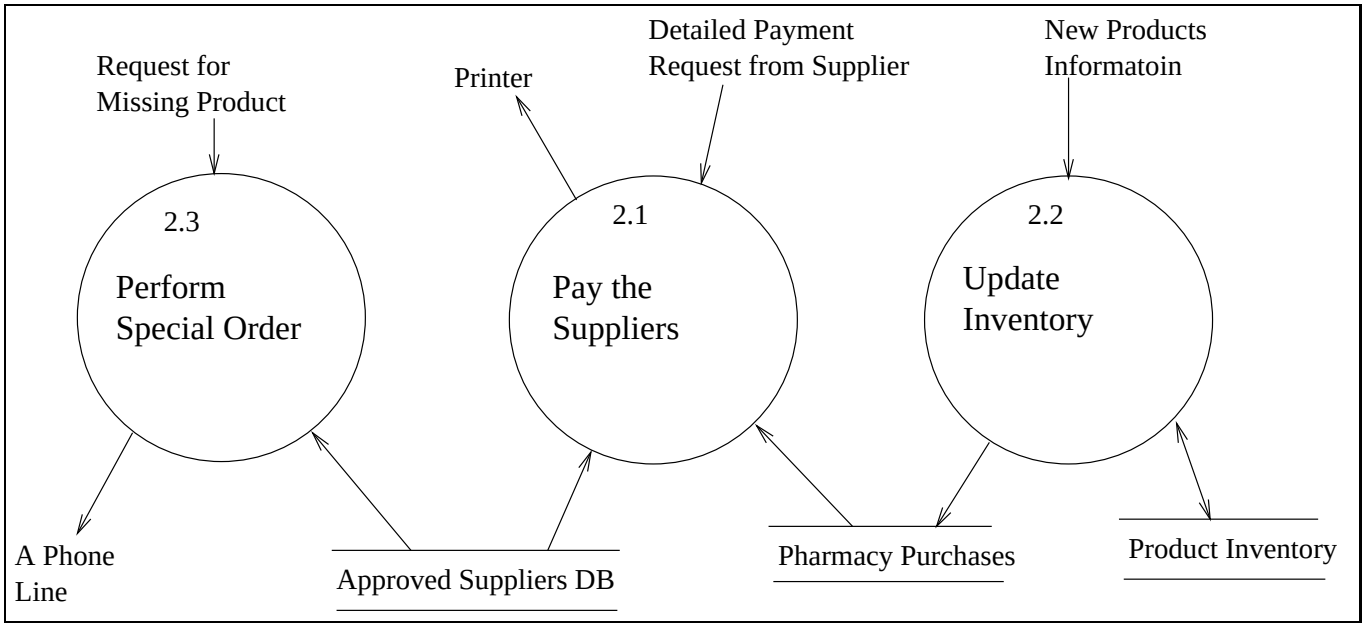

Figure 3.9: DFD 1 – Manage Suppliers and Orders

Data Flow Diagram 1 – Manage Suppliers and Orders

1. **Data Entities**

   - **Pharmacy Purchases** – A database that includes information about the new products delivered to the pharmacy.

2. **Pertinent Processes**

   (a) **Pay the Suppliers** – Check the payment request against the Pharmacy Purchases database, and then print the cheque.

   (b) **Update Inventory** – When new product arrives, we should update the inventory.

   (c) **Perform special order** – Since the inventory is updated in Figure 3.2.1, all we have to do is call the supplier and notify them that we want some more products.

Data Flow Diagram 2 – Pay the Suppliers

1. **Data Entities**
   None New

2. **Pertinent Processes**

   (a) **Get Next Supplier** – The payment process should go over all the suppliers, so we just start from the beginning and go over all the suppliers.

   (b) **Compare The request and the purchases DB** – Checks whether the detailed payment request is indeed correct.

   (c) **Print Cheque** – After all the details are gathered, we should print the cheque.

   (d) **Check whether a recipe is needed**

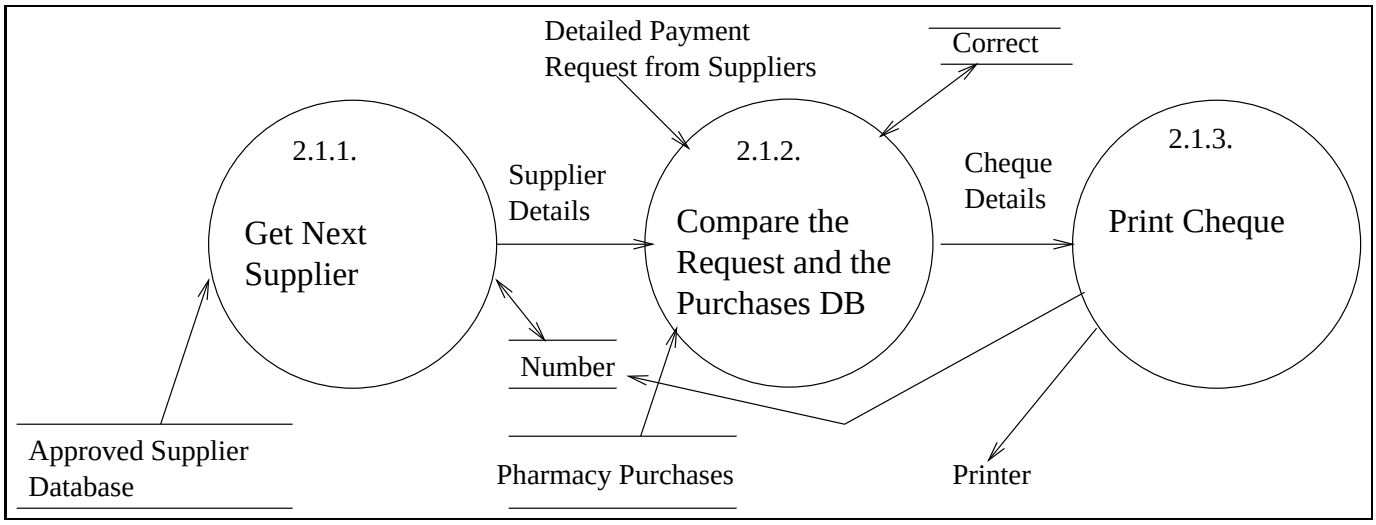

Figure 3.10: DFD 2 – Pay the Suppliers

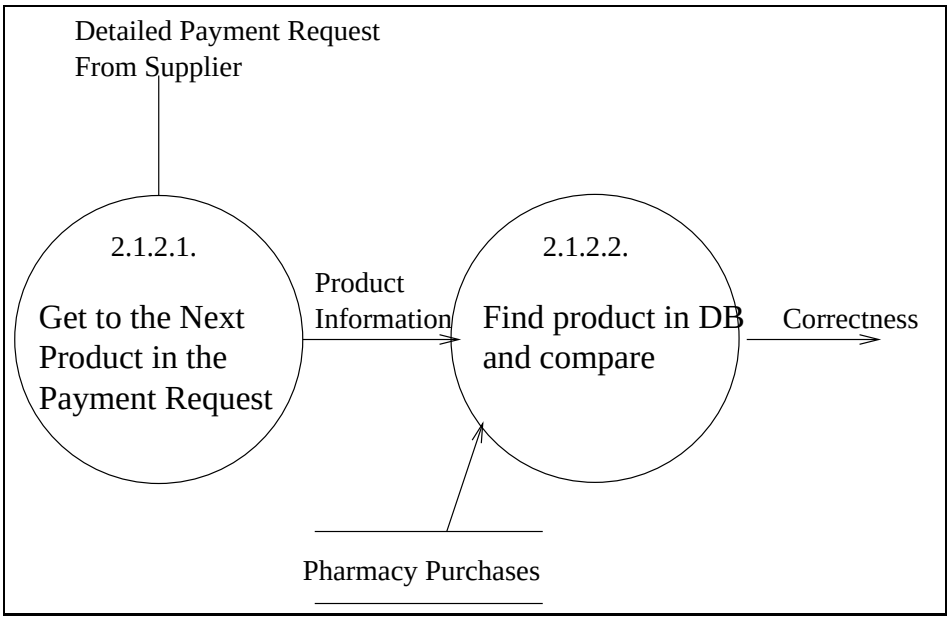

Figure 3.11: DFD 3 – Compare Request and purchases DB

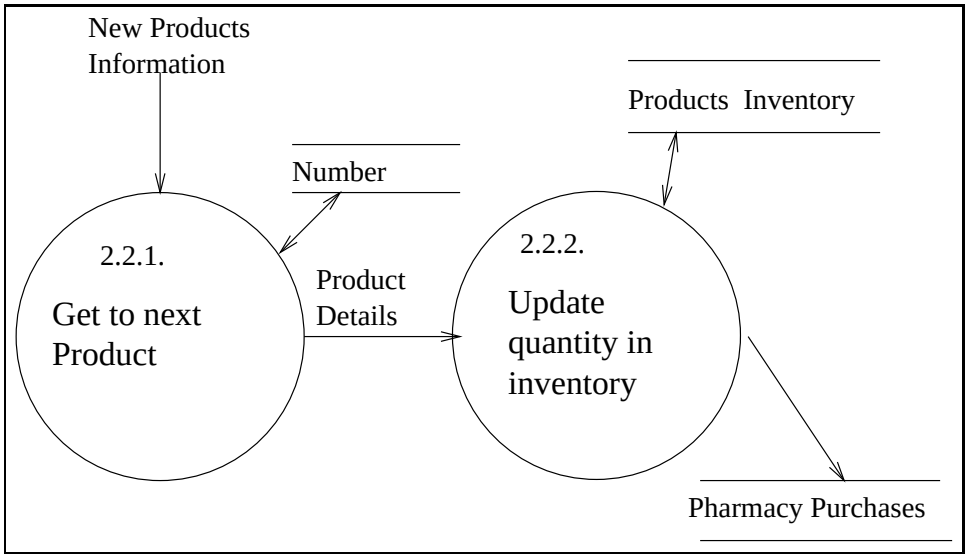

Figure 3.12: DFD 2 – Update Inventory

Data Flow Diagram 3 – Compare Request and purchases DB

1. **Data Entities**
   None New

2. **Pertinent Processes**

   (a) **Get Next product in the Payment Request**

   (b) **Find product in (purchases) DB, and compare**

Data Flow Diagram 2 – Update Inventory

1. **Data Entities**
   None New

2. **Pertinent Processes**

   (a) **Get to Next Product**

   (b) **Update Quantity in Inventory** – Updates the quantity in the inventory, and if the current product was pre-ordered, output the name and phone of the person who ordered to the screen (from now on it is manual operation).

Data Flow Diagram 2 – Perform Special Order

1. **Data Entities**
   None New

2. **Pertinent Processes**

   (a) **Connect to supplier through modem** – Connects to the supplier's computer, and transfer the order using the modem.

   (b) **Call the supplier and Use Voice Synthesis** – call the supplier and use voice synthesis software to read the order.

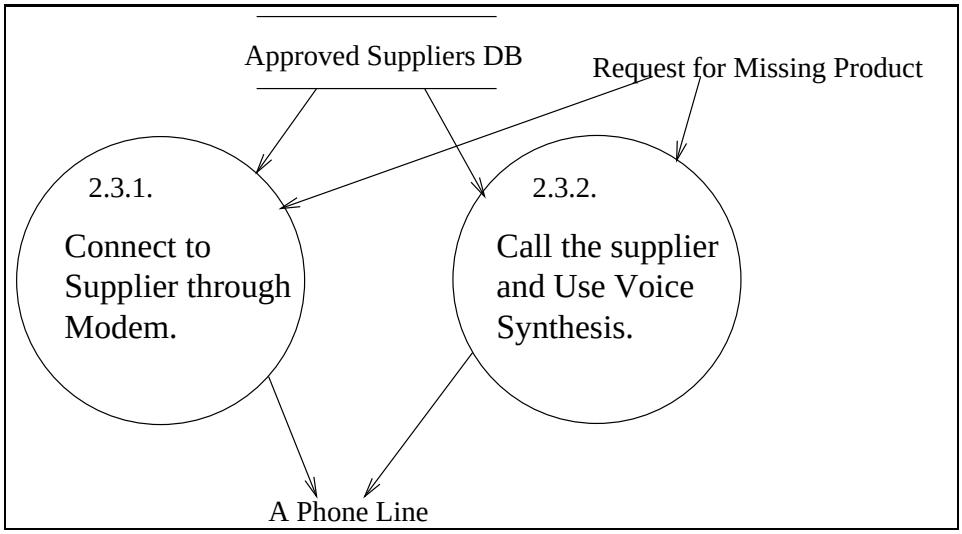

Figure 3.13: DFD 2 – Perform Special Order

3.2.2 Process Description

The following section explains the operation of every process in the DFDs (starting from DFD1).

Update Product Information – 1.1

1. **Input Data Entities**
   This process receives the serial number of a product, and the number of items of which to take off of the inventory and the number or items to order and for the customer.
2. **Algorithm or Formula of processing**
   If the quantity of items of this product for purchase is greater then 0, then call process 1.1.1. (Decrease Product Quantity From Inventory).

   If the quantity of the items of this product for order is greater then 0, then call process 1.1.2. (Manage Customers Orders)
3. **Affected Data Entities / Output**
   Updates the inventory if needed, and calls the process that calls the supplier.

Decrease Product Quantity From Inventory – 1.1.1.

1. **Input Data Entities**
   The quantity to decrease from the inventory.
2. **Algorithm or Formula of processing**
   Retrieves the product by the product serial number, then decreases the quantity to decrease from the total quantity of this product, and insert it back to the database.
3. **Affected Data Entities / Output**
   The Inventory Database.

Manage Customers Orders– 1.1.2.

1. **Input Data Entities**
   The quantity to order from the inventory, and the serial number of the product.

   The details of the customer.

2. **Algorithm or Formula of processing**
   Updates the product record, such that the information of the customer, and the number of items from that products are at the end. (The inventory database should include a table of waiting customers that has the name and the phone, and acts as a linked list – that is one customer that ordered the product is linked from the customer before him that ordered the product.

   After it, the process signals the calling process the information about the product.

3. **Affected Data Entities / Output**
   The Inventory Database, and the signalled request for missing product.

Manage Customer's Payments – 1.2

1. **Input Data Entities**
   Customer's Details, this purchase, and the regular customers database.

2. **Algorithm or Formula of processing**
   First it finds the customer (either in from the regular-database, and return serial number, or from the health care database, and then we have the I.D., or the customer may not be in any database.

   If the Customer is a regular one, check whether s/he wishes to add this purchase to the account, or pay all the account.

   If there is potential (that is, the customer is part of the health-care system, and is chronic), a new account is opened.

   If the user wishes to pay, a bill is printed, and the amount to be paid is transferred to the cash register.

   At the end the *this purchase* database is cleared.

3. **Affected Data Entities / output**
   The this purchase database, the regular-customers database, and the values of

Search Customer – 1.2.1

1. **Input Data Entities**
   The customer either by name or by I.D or by serial Number.

2. **Algorithm or Formula of processing**
   Goes over the databases and search for the customer.

3. **Affected Data Entities / Output**
   If the customer is a regular-customer , the output is the serial number of the customer. If the customer is a health care network patient the output is his/her I.D, and if neither, the output is that it is not in the database.

Search By Name – 1.2.1.1.

1. **Input Data Entities**
   Customer information, by name.

2. **Algorithm or Formula of processing**
   Since all the names in the database are stored in UPPERCASE, we should first uppercase the name, and then look it up first in the regular-customers database, and then in the health-care system patients database.

3. **Affected Data Entities / Output**
   Same as 1.2.1.

Search by Serial Number – 1.2.1.2.

1. **Input Data Entities**
   Customer Information, by serial number.

2. **Algorithm or Formula of processing**
   We check if the serial number exist, and if it is return the same argument, else change it to not-in-the-database.

3. **Affected Data Entities / Output**
   Either a serial number if the customer is a regular-customer, and not present if it is not in the regular-customer database.

Search by I.D. – 1.2.1.2.

1. **Input Data Entities**
   Customer Information, by I.D..

2. **Algorithm or Formula of processing**
   Query first the regular-customers database, and then the patients database, and return the answer, as in 1.2.1.1.

3. **Affected Data Entities / Output**
   The same of 1.2.1.

Handle Regular-Customer Account – 1.2.2.

1. **Input Data Entities**
   Regular-Customer, sorted by serial number.

2. **Algorithm or Formula of processing**
   The system now asks (using the GUI)whether the customer wishes to add the current purchase to the sum, to be paid later, or to pay all his/her debts now.

   If decided to pay all the debts now, it reads the needed records from the regular-customer and add them to the "this purchase" database.

   If decided to pay later, it adds the current sum to the records of the regular-customer.

3. **Affected Data Entities / Output**
   Changed the regular-customer database, and the this purchase database.

Produce Bill – 1.2.3.

1. **Input Data Entities**
   This purchase database, and information whether to pass the information to the cash register or not.

2. **Algorithm or Formula of processing**
   Prints the bill, taken from the This Purchase database, and if the sum is not added to the account, also pass the amount to the cash register.

3. **Affected Data Entities / Output**
   A printer bill and (maybe) a payment.

Handle Regular Customer Information – 1.2.4.

1. **Input Data Entities**
   The user Details. Either full-name or serial, and the needed option from the GUI.
2. **Algorithm or Formula of processing**
   Either Updates the information , adds a new customer or removes one from the system.
3. **Affected Data Entities / Output**
   The regular-customer database.

Add Regular Customer – 1.2.4.1

1. **Input Data Entities**
   User details that include all the needed (cleartext) details.
2. **Algorithm or Formula of processing**
   Adds another record to the database. Before adding, we should check the health care system patient DB to verify that the customer is indeed a patient, and is indeed chronic.
3. **Affected Data Entities / Output**
   The regular-customer database changes. The serial number of the new customer is displayed on the screen.

Remove Regular Customer – 1.2.4.2.

1. **Input Data Entities**
   The Serial number of the customer.
2. **Algorithm or Formula of processing**
   The record is removed from the system.
3. **Affected Data Entities / Output**
   The regular-customer database.

Update Regular Customer – 1.2.4.3.

1. **Input Data Entities**
   All the information needed in Text form.
2. **Algorithm or Formula of processing**
   Retrieve the record (first uppercase the name, or look it up with the serial number), update it, and insert it back to the database.
3. **Affected Data Entities / Output**
   The regular-customer database.

Search Product – 1.3

1. **Input Data Entities**
   The Details of the product.
2. **Algorithm or Formula of processing**
   The product is then looked up in the database, the location on the shelf is shown, and the pharmacist is prompt if the product needs doctor's recipe.
3. **Affected Data Entities / Output**
   Extended product details are outputted (the numbers or products to currently purchase, and number of the product the pre-order for the customer).

Search By Barcode – 1.3.1.

1. **Input Data Entities**
   The Serial number of the products and the number of needed units.
2. **Algorithm or Formula of processing**
   The record is looked up, and the serial number and quantity is found.
3. **Affected Data Entities / Output**
   The product details ad a serial number.

Search by Name – 1.3.2.

1. **Input Data Entities**
   The name of the product, and the quantity.
2. **Algorithm or Formula of processing**
   The name is uppercased, and looked up. Then the output is the serial number of the product.
3. **Affected Data Entities / Output**
   The product details in serial number form.

Show Location on shelf – 1.3.3.

1. **Input Data Entities**
   The product details in serial number form (and the quantity).
2. **Algorithm or Formula of processing**
   The record is fetched from the database, and the location description is presented on the screen.
3. **Affected Data Entities / Output**
   No actual processing is done (only side effect).

Check Whether a Recipe is needed – 1.3.4.

1. **Input Data Entities**
   The product details in serial number form, and the quantity.
2. **Algorithm or Formula of processing**
   The record is fetched, check whether the product needs a recipe, and prompt to the screen.

   Now Check whether the quantity is larger then the available quantity of the product. If it fill all the available e quantity in the "for buying" section, and the rest in the "for order" section. Else all the quantity is filled in the "for buying" section.
3. **Affected Data Entities / Output**
   Extended Product Details.

Create Monthly Reports – 1.4.

1. **Input Data Entities**
   The product inventory.
2. **Algorithm or Formula of processing**
   Go over all the products and print the quantity. Before printing, a person with accounting dept. permission that is logged should authorize the print job. If non is logged in, the system will wait until the first one does,
3. **Affected Data Entities / Output**
   Output to the printer.

Pay the Suppliers – 2.1.

1. **Input Data Entities**
   Detailed Payment Requirement from Supplier, and the pharmacy purchases database.

2. **Algorithm or Formula of processing**
   This process go over every line in the payment requirement and checks whether there is a corresponding line in the pharmacy purchases database.

3. **Affected Data Entities / Output**
   If the requirement is correct, the printer will print the cheque for the supplier.

Get Next Supplier – 2.1.1.

1. **Input Data Entities**
   The Detailed Payment Requirement from Supplier. The number of lines already processed placed in the Number database.

2. **Algorithm or Formula of processing**
   The process should skip Number lines, and then output that line to the next process. Then Increment Number by one. When the requirement is over, The details of the supplier and amount are outputted.

3. **Affected Data Entities / Output**
   The line from the Payment Requirement, or the details of the supplier.

Compare the Request and the Purchases DB – 2.1.2.

1. **Input Data Entities**
   The line from the requirement.

2. **Algorithm or Formula of processing**
   The product is given by barcode. Then the product is looked up in the purchases DB, when the record returns, the quantity in the DB is compared to the quantity in the payment requirement. If everything is good, then logical anding TRUE to the correct DB.

   When the line with the suppliers details in inputted, we check the Correct DB, and if it is true, we pass the Supplier's details are passed to the next process. Else some predefined value is outputted.

3. **Affected Data Entities / Output**
   The details of the supplier and the amount to be paid.

Print Cheque – 2.1.3.

1. **Input Data Entities**
   Supplier and amount details.

2. **Algorithm or Formula of processing**
   A cheque with the details is printed.

3. **Affected Data Entities / Output**
   No output besides the printer.

Update Inventory – 2.2.

1. **Input Data Entities**
   New Products Information.

2. **Algorithm or Formula of processing**
   Update the inventory and pharmacy purchases DBs when new products are arriving every month. It should set the Number DB to 0 when first run (the Number DB is encapsulated within it)

3. **Affected Data Entities / Output**
   Update the inventory and pharmacy purchases databases.

Get to Next Product – 2.2.1

1. **Input Data Entities**
   New Products information, Number DB.

2. **Algorithm or Formula of processing**
   It should skip the first Number entries of the New Product Information, and then increment the Number DB, and output the Number+1 entrie.

3. **Affected Data Entities / Output**
   Product Details, in serial number form.

Update Quantity in Inventory

1. **Input Data Entities**
   Product Details in serial number form.

2. **Algorithm or Formula of processing**
   It should update the quantity in the inventory DB, and add a new record for this product/quantity/time at the pharmacy purchases DB.

   Then it should be checked whether the product has a waiting list. If it has, the list should be outputted to the screen. The pharmacist, then should put the needed quantity or products on a special shelf with the details of the customers that ordered it, and call those customers.

3. **Affected Data Entities / Output**
   Updated Inventory and Pharmacy Purchases.

Perform Special Order – 2.3.

1. **Input Data Entities**
   The details of the special order. (Request for missing product)

2. **Algorithm or Formula of processing**
   The algorithm is to call external programs to perform the order. But first we have to look up the phone number and whether the supplier chose modem or voice interaction.

3. **Affected Data Entities / Output**
   A phone line.

Connect to Supplier through Modem – 2.3.1.

1. **Input Data Entities**
   Request for missing product.

2. **Algorithm or Formula of processing**
   Looking up the information as explained in 2.3. and then calling the 3rd party modem controlling software/library.

3. **Affected Data Entities / Output**
   A phone line.

Call the Supplier and Use Voice Synthesis – 2.3.2.

1. **Input Data Entities**
   Request for missing product.
2. **Algorithm or Formula of processing**
   Looking up the information as explained in 2.3. and then calling the 3rd party voice synthesis for phone line controlling system.
3. **Affected Data Entities / Output**
   A phone line.

3.2.3 Data Construct Specification

Here is the ERD for this project:

3.2.4 Data Dictionary

Please note that I do not use the data dictionary entries that are proposed in the IEEE 830 document (that is Name, Representation, Units/Format, Precision/Accuracy, Range) but a bit different format that I find more useful. The Name of each entree is the title for that entree.

Customer Details

1. **Aliases:** None.
2. **Where Used:** The Customer Details data item is used throughout the purchasing processes.
3. **How Used:** It is used to store all the information we'll need about the customer. It has several different ways to keep the information, by the serial number, I.D. number or name.
4. **Description:** Customer Details = FormSpecifier+[SerialNumber — IDNumber — FullName+(PhoneNumber+IDNumber)]
5. **Format:** FormSpecifier : (specify whether we deal with the serial, ID or name form.) Can be either S, I or N.
   SerialNumber : $\{0-9\}^9$
   IDNumber : $\{0-9\}^9$
   FullName : $\{a-zA-Z.\}^100$
   PhoneNumber: $0-9^16$

Product Details

1. **Aliases:** None.
2. **Where Used:** The Product Details is used throughout all of the system, since this is the basic data structure of the system.
3. **How Used:** It is used to store all the information we'll need about the Product. It has several different ways to keep the information, by the serial number, barcode or name.
4. **Description:** Product Details = FormSpecifier+[SerialNumber — Barcode — Name]+Quantity.
5. **Format:** FormSpecifier : (specify whether we deal with the serial, barcode or name form.) Can be either S, B or N.
   SerialNumber : $\{0-9\}^9$
   Barcode number : $\{0-9\}^20$
   FullName : $\{a-zA-Z.\}^100$
   Quantity: $0-9^16$

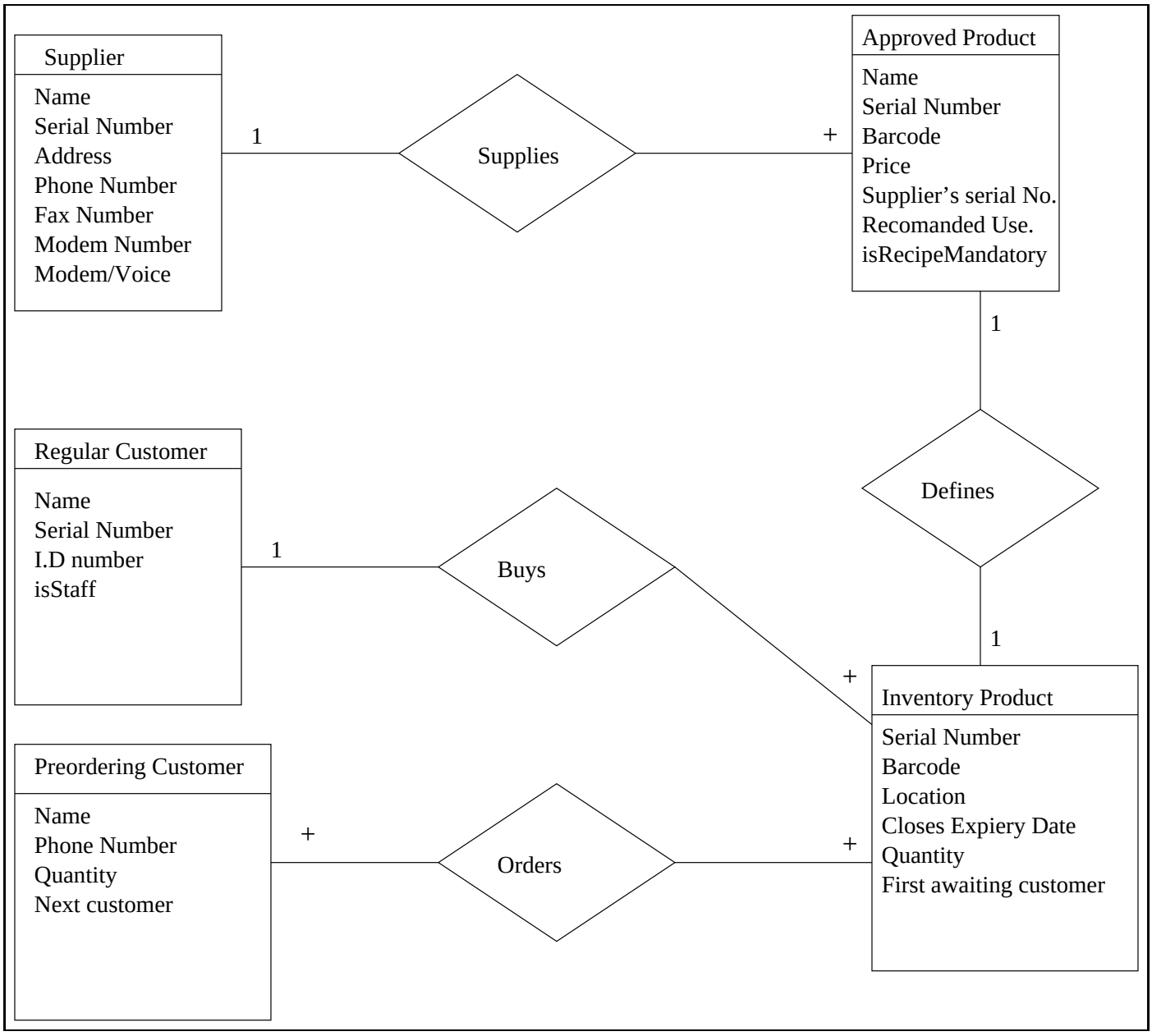

Figure 3.14: Entity Relationship Diagram

Supplier Details

1. **Aliases:** None.
2. **Where Used:** The Product Details is used throughout all of the system, since this is the basic data structure of the system.
3. **How Used:** It is used to store all the information we'll need about the Product. It has several different ways to keep the information, by the serial number, barcode or name.
4. **Description:** Supplier Details = FormSpecifier+[SerialNumber — Name]
5. **Format:** FormSpecifier : (specify whether we deal with the serial, barcode or name form.) Can be either S or N.
   SerialNumber : $\{0-9\}^{9}$
   Name : $\{a-zA-Z.\}^{1}00$

Request for missing Product

1. **Aliases:** None.
2. **Where Used:** Used in order to store together product information and customer details.
3. **Description:** Request For Missing Produst = Product Details + Customer Details.

New Product Information

1. **Aliases:**
2. **Where Used:** To keep a lot of product information together.
3. **How Used:** A lot of products one after the other.
4. **Description:** $NewProductInformation = \{ProductDetails\}^{*}$

Detailed Payment Requirement from Supplier

1. **Aliases:** None.
2. **Where Used:** Used when the supplier wants to retrieve his/her money.
3. **How Used:** It is just a "New Product Information" with the name of the supplier at the end.
4. **Description:** Detailed Payment Requirement from Supplier = New Product Information + Supplier Details.

Extended Product Details

1. **Aliases:** None.
2. **Where Used:** To support a product with two quantities. The regular and for the preordering.
3. **Description:** Extended Product Details = Product Details + PreOrderQuantity.
4. **Format:** PreOrderQuantity = $\{0-9\}^{1}0$

3.3 Performance Requirements

3.3.1 Static Requirements

1. The system shall support up to 20 terminals. Up to 10 of those terminals should be able to connect to a cash register, and all of them have a barcode reader.

2. The system shall support a database of up to 2000 products, and may upgrade this requirement when needed. About 10 new products are added to the system every month.

3. The system shall support a database of up to 200 different suppliers. This number may be increased in the future. About 1 supplier is added to the system every two months.

4. The system shall support a database of up to 5000 regular customers. This number might be upgraded. About 100 new customers are joining the branch every year.

3.3.2 Dynamic Requirements

1. The system shall be able to perform even if either of the other two systems it connects to are not functioning. The system will not be able to provide all the functionality, of course, but should support at least:

   - Purchases by customers.
   - Purchases by regular-customers (including paying old purchases, and adding the current one to the account.)
   - Inventory Updates.

2. 95% of the transactions should take less then 2 seconds. The rest should complete after at most 8 seconds.

3.4 Design Constrains

Design constrains may result from such things as the prescribed use of certain standards or hardware.

3.4.1 Standard Compliance

Since in its current state the system will only be used for the internal management of the pharmacy, and does not have to output any reports or any other information to other systems, there are no standards to follow. This might change if we choose to add more functionality in the future.

3.4.2 Hardware Limitations

The server should be supplied with enough memory to store all the current open connections with the terminals in its physical memory (no need to swap in order to manage the connections).

3.5 Software System Attributes

In this section, particular attention is paid to quality aspects.

3.5.1 Reliability

The system should be fully debugged when shipped to the client. The server should not crash because of any component of our system, and the terminals should work without problems too.

3.5.2 Availability

The server should continue working even if some of its clients crashed, or stopped responding. The server will report the home-base when a terminal is not functioning well.

Since the server availability is crucial for the system to operate we should minimize its downtime. All administrative work should be able to be done while the server is running (that is, adding more disk space, backing the data up, enlarging the tables of the database, etc.). Since the data integrity is also important all the interaction with the database will be through rollback transactions. The filesystem of the RAID should be either a transaction (log) based filesystem to avoid data loss, or a raw partition (the database will manage the filesystem).

If the server crashed, it should be able to boot within 5 minutes.

3.5.3 Security

Before the terminals are functional the user has to login. The system should provide three levels of security:

1. **Pharmacist** – Should be able to query the products database, sell products to customers and perform special orders.
2. **Accounting personnel** – Should be able to produce the monthly reports, pay the suppliers, and perform special orders.
3. **Inventory Personnel** – Should be able to update inventory information.

It should be noted that in small pharmacies one person might perform more then one task, and the system should support it.

An additional user type is the administrator, which is the only person that can log into the server. The administrator is not part of the regular pharmacy personnel.

3.5.4 Maintainability

The complexity of a program is related to the number of ways that its components can interact. It is desirable to reduce the interrelation of the system as much as possible.

3.5.5 Turnkey System

This system should operate as a turnkey system. That is, when the computers boot the system is the first and only thing that the user can see and has access to. It is required that regular users will not be able to access the operating system of the machines. Moreover, it is desired that regular users will even not be able to recognize the underlying operating system.

Appendix A

Possible Product Evolution

This chapter provides a list of features that are expected to either change or be added to the application in the future. The application's design should be constructed to help reduce the effort required to integrate these features when they are incorporated into the system.

1. Merge the cash register into the system.
2. Enable a regular customer to perform an order in a branch of the network that is not his/her "home branch", and have the amount of money added to his/her account at his/her "home branch".
3. Save information on regular customers (either by their credit card number or by other means), and then analyze the information gathered to better understand the market.
4. Provide a way to perform orders from the internet (using a WWW interface).
5. Since we deal with a network of shops, there is a possibility to pay the suppliers only from a central controlling location (the home-base) , and thus the workers at the pharmacies won't have to waste their time.
6. Combine the cash register into the system. This will provide the following improvements over the current implementation:
   - The monthly report will be able to contain more extensive information.
   - The workers of the pharmacy will have only one computerized interface to deal with.
   - Once all the payments information is passed through our system, we will be able to keep track of (non-regular) customers by their credit card, and use the information to perform statistical studies to enhance the service.

Bibliography

[1] IEEE Std 610.12-1990. *IEEE Standard Glossary of Software Engineering Terminology.* Approved September 28, 1990.

[2] IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications.* Approved 25 June 1998.

[3] IEEE Std 1233, 1998 Edition. *IEEE Guide for Developing System Requirements Specifications.* Approved 8 December 1998.