CHILD MEDICAL & HEALTH ADVISOR

(CMHA)

A report prepared by:

ANBARASAN A/L PALASUBRAMANIAM

(WEK 98044)

SESSION 2002/2003

Supervisor:

Dr. Rukaini Haji Abdullah

Moderator :

Cik Norisma Idris

A Final Year Project Report Submitted to the Faculty of Computer Science & Information Technology, Universiti Malaya Kuala Lumpur

As A Partial Fulfillment of the Requirements for the Degree of Bachelor in Computer Science.

Abstract

The Child Medical & Health Advisor (CMHA) is a stand-alone system that incorporates the design structure of an expert system. It has a knowledge base, an inference engine and the interface. The system is developed to serve the purpose as a system that provides information on child health issues. These issues are child illness, sickness care, treatment and home care. The system also provides a diagnostic capabilities to the user. It's able to provide diagnosis up to 30 diseases. The system is designed using forward chaining control strategy and having rules as its knowledge base. The knowledge was acquired from medical books on child health and the system was developed using Visual Basic v6.0. This system could be installed in a personal computer using windows operating software version 98 and above.

Acknowledgements

I would like to take this opportunity to thank my project supervisor, Dr .Rukaini Hj.Abdullah for all the help given to me while preparing for this project documentation. This project would also not be completed without the assistance of Cik Norisma Idris, whose valuable suggestions for the development of this system is greatly appreciated. I would also like to express my gratitude to my brother P. Kumarasan whose valuable assistance in this project will not be forgotten. This Project is dedicated to my family and God Almighty and Merciful. Thank You

TABLE OF CONTENTS

Contents	Page
Acknowledgements	i
Abstract	ii
List of diagrams, figures, and tables	v
Chapter 1: Introduction	1
1.1 The Problem	1
1.2 Objective	4
1.3 Motivation	5
1.4 Motivation	6
1.5 Project Planning	7
Chapter 2 : Literature Review	10
2.1 Overview	10
2.2 Findings	11
2.3 Expert systems	12
2.3.1 Heuristics	14
2.3.2 Representing Knowledge	15
2.3.3 Inference Engine	23
2.3.4 Explanation Faculty	28
2.4 Analysis	30
2.4.1 History of Mycin	30

2.5. Synthesis of Expert System	32
Chapter 3: Methodology	34
3.1 Assessment	37
3.2 Knowledge Acquisition	39
3.3 Design	40
3.3.1 Selecting Knowledge Representation Technique	40
3.3.2 Selecting Control Technique	41
3.3.3 Selecting Systems Development Software	42
3.3.4 Hardware Requirements	43
3.3.5 Prototype Development	43
3.3.6 Interface Development	45
3.4 Testing & Development	47
3.5 Expected Outcome	48
Chapter 4 : System Design	49
Chapter 5: System Implementation	62
5.1 Introduction	62
5.2 Software Development Tools	62
5.3 Visual Basic	63
5.4 Integrated Development Environment	64
5.4.1 Project Window	65
5.4.2 Toolbox	65

5.5 System Coding & Implementation	67
5.5.1 Coding Approach	67
5.6 Summary	70
Chapter 6: Testing	71
6.1 Introduction	71
6.2 Debugging & Testing in Visual Basic Environment	71
6.2.1 Using the Debugger	73
6.3 Types of testing	74
6.3.1 Unit Testing	74
6.3.2 Integration Testing	75
6.3.3 System Testing	75
6.4 CMHA testing	76
6.5 Summary	79
Chapter 7: Evaluation	80
7.1 Introduction	80
7.2 Problem Encountered	80
7.3 System Evaluation by User	81
7.4 System Strengths	82
7.5 System Limitation	83
7.6 Future Enhancement	84
7.7 Knowledge & Experienced Gained	84
7.8 Summary Report	85

7.9 Conclusion	86
Reference	
Appendix A Visual Outlook Of System Menus **Appendix B** User Manual	

List of Diagrams, Figures and Tables

Table 1.1	Project Plan Task	7
Table 1.2	Second Semester Project Plan Task	8
Diagram 2.1	Applications Area of AI	13
Table 2.1	Common Logic Symbols	18
Figure 2.1	Inference Process in Expert System	23
Table 2.2	Comparison between Conventional & Expert System	28
Figure 3.1	Phases in Expert System Development	35
Figure 4.1	System Modules	50
Figure 4.2	System Data Flow	51
Figure 4.3	Disease Reference Menu Data Flow	53
Figure 4.4	Data Flow for Diagnosis Menu	55
Figure 4.5	Data Flow for Health Reference Menu	56
Figure 4.6	The Main Menu for CMHA	58
Figure 4.7	The Disease Reference Menu	59
Figure 4.8	The Diagnosis Menu	59
Figure 4.9	The HealthCare Tips Menu	60
Table 5.1	Toolbox Control Summary	65

CHAPTER 1
INTRODUCTION

CHAPTER 1:INTRODUCTION

1.1 THE PROBLEM

Parenting is a challenging process that involves the parent and the child. In raising a child, the parent is always facing issues involving a child health. The Saying "Health IS WEALTH" is certainly something that should be pursued diligently by a caring and concerned parent. When a child is sick, every parent will understand the feeling of panic that strikes because the lack of knowledge in assessing of a given condition.

The Problem is, how many parents have the knowledge to assess and handle these situations. For critical cases, the parent will take the child to a clinic or hospital to get the medical attention it needs, but what about situation the needs minor treatment, such as minor fever, burns, and other conditions. How do you treat them? What are the precautions to take? What does a fever really mean? There are many questions that you can ask but what's the answer.

The Child Medical & Health Advisor (CMHA) being proposed works on the principle to inform and advise the concerned parent the best possible solution to deal with a medical issues during childhood. CMHA is a guide to the problems that parents most likely to encounter along with practical advice on recognizing, assessing and dealing with each problem. The system will be able to give the possible disease suffered by the child after being given sufficient information to the system. For example, if the child is infected with influenza, the key symptoms are

- Pain
- Fever

- Cough

While the minor symptoms are

- Vomiting
- Headache
- Diarrhea

With these symptoms given, the system will be able to identify possible disease and will pinpoint the sickness. Now, it will give the action, precaution and the medical treatment that can be given. The CMHA is not intended to replace the general Practitioner or the Pediatrician; its purpose is just to provide information and advice on everyday medical health problem afflicting a child. Thus the parent or any user who wants to know what the child is suffering from can query the system to get information on the disease. Its main purpose is to help the parent play an informed, practical and competent role in caring for the child. The advantage of having the CMHA at home is

- Availability

The system is available to user at any time; hence it's easier to give diagnosis, quick reference and health care tips when the user needs it. It will also save the time wasted searching for information on the disease through electronic and printed media

- Consistency

The system is always consistent in giving the decision, based on the input given by the user. Thus the same input will give the same decision every time. Thus the system is not affected by age or emotions that impairs decision making

- Easy to use

The user will find the interface easy to navigate and the system easier to use.

- Comprehensive

The system provides comprehensive information on child disease that is reliable, precise, and up to date

1.2 OBJECTIVE

The objective of developing CMHA:

- To help the user to diagnose and identify the disease suffered by the child.

- To give a detailed information on the disease being queried that covers symptoms, home care, precaution and medical treatment to be taken

- Provides health care tips that covers basic child health care knowledge that the parents will find very useful.

1.3 SCOPE

The scope of the project covers the problem to be solved and also the system requirements that should be fulfilled during the development of the system

- The system is limited to child health domain only. For the system, 30 child related diseases would be covered for the reference and the diagnosis for child diseases.

- System can perform its own logical deduction on the data provided by the user for he purpose of diagnosis, in accordance to the line of reasoning specified in the rules.

- The system will be modeled after the design of expert system thus it will have a knowledge base, inference engine and an interface. A separation of knowledge from control is to provide modularity in designing the rules and also ease in modification of rules and knowledge
- Uses rules as basis for knowledge representation, in other words the system is a rule based system
- Will use forward chaining as means of control strategy
- System is a " stand alone " system

1.4 MOTIVATION

The System to be developed was inspired by the need to introduce an intelligent substitute to child medical health book. This project is an effort to merge an information system with a diagnostic system. As a student of Artificial Intelligence (AI), the author was compelled to introduce an AI approach to merge these both system, thus a decision was made to design a system that will based on an Expert System. The reader might ask why an Expert System approach was used, well the answer to question is provided in detail in chapter 2. For the time being, to simplify the reason why an expert system based approach was taken, it was because an Expert System can perform the task that require special knowledge on certain domain and also have problem solving skills embedded in it. An Information System has a special knowledge on certain domains, while the Diagnostic System is able to solve the problem of diagnosis; thus an Expert System can perform both of this task. This is the reason why an Expert System based approach will be used.

Capturing an expert knowledge is a complex process and modeling it in a system is an even more challenging task. The domain of an expert system does not have to be as detailed as an expert for certain simple task. An Expert System can also be built to perform diagnosis based on information that can be found in medical books, journals and other reliable resources. It might be simple but must be useful to the user. The design of the system will pursue the objectives and scopes that were already mentioned before. The target audience for the system are the parents themselves and also anyone who is interested to increase their knowledge in child medical healthcare.

1.5 PROJECT PLANNING

The project plan is made to make sure that all the important activities and their relative task is executed in a proper and efficient manner, that the system is developed in between the time scale given. A plan is also important to make sure that the system designer can measure the progress of his work and work the plan that has been designed. It guides the designer through all the important steps in making of the system. The development of the system is in two phases. There is Projek Ilmiah Tahap Akhir 1 (WXES 3181) and Projek Ilmiah Tahap Akhir 2 (WXES 3182). The goal of WXES 3181 is to measure the student's competence in analyzing and designing the project. The development of the Child Medical & Health Advisor started in the 3rd semester, which only consist of 7 weeks. The following table illustrates the task undertaken for the seven weeks.

Week	Task Undertaken
1	Selecting Project Title
2	Collecting information from reference book and the internet
3	Collecting information and conducting

5	Designing code
6	Designing code.,writing initial code
7	Designing code.,writing initial code
8	Coding and testing system
9	Coding , testing and expanding the system
10	Coding, testing and evaluate the system
11	Coding, testing and evaluate the system
12	Documentation of system
13	Submission of system, and documentation

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2: LITERATURE REVIEW

2.1 OVERVIEW

A literature review is a very important stage that is undertaken for the purpose of this project paper. In this stage, a detailed study and research is conducted in the field that relates to the problem to be solved and also the system proposed to handle the problem. In literature review we have several important steps that should be undertaken to achieve a comprehensive literature review. The major steps are:

1. Findings
2. Summary
3. Analysis
4. Synthesis

All these areas will be covered in this chapter. The purpose of undertaking a literature review is to increase the knowledge and understanding of the problem to be solved and also the characteristics of the system to be developed. The analysis and synthesis will also give new insight and ideas to be used in designing systems and also highlights the problem that might be encountered during the development stage of the system.

2.2 FINDINGS

The Literature Review is impossible to conduct without having any information resource to base our study upon. For this project, the source of the knowledge were obtained from:

- Previous final year project undertaken by students of Faculty of Computer Science and Information Technology. The scope of the project selected was narrowed down to expert system that relates to diagnosis.

- Books on expert system development, artificial intelligence and child health books.

- Journals, research papers, websites that related to expert system.

2.3 EXPERT SYSTEMS

Medical software tools began to emerge during the 1980's, some became known as 'expert systems'. In contrast to conventional software which process data, expert systems process 'knowledge'. For this reason, expert systems are also called 'Knowledge Based Systems' (KBS). The most well known medical example is MYCIN.' This expert system was developed at Stanford University in 1976 to aid physicians in diagnosing and treating patients with infectious blood diseases caused by bacteria in the blood and meningitis. These diseases can be fatal if not recognized and treated quickly. Many other medical expert systems have followed the success of MYCIN.

An expert system is a program, which attempts to mimic human expertise by applying inference methods to a specific body of knowledge called the domain. Knowledge is different from data or information in that data is passive. Knowledge on the other hand is active in that it can be used to infer new information from what is already known about a problem. As will be seen later, this domain knowledge is frequently represented as rules.

Artificial Intelligence Expert systems owe their origin to the field of Artificial Intelligence (AI). One of the pioneers of AI, Dr. Marvin Minsky defined AI as "The field of study which is attempting to build systems which if attempted by people would be

considered intelligent". AI is a broad field, with some of the application areas shown in the diagram 2.1 below.

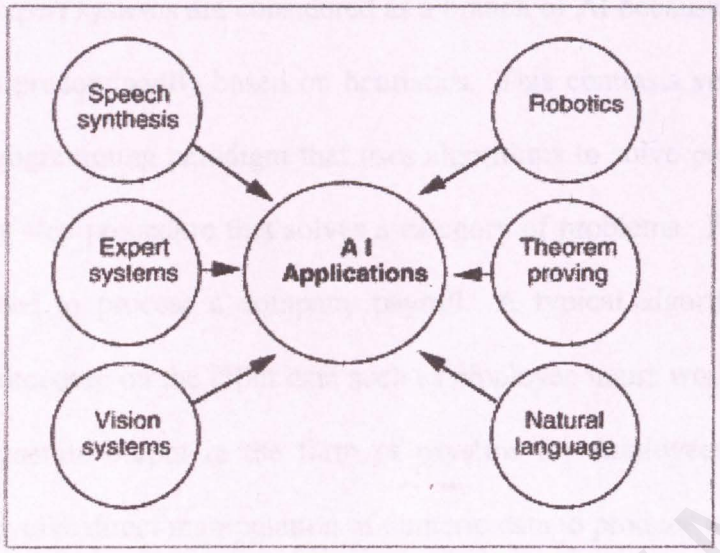


Diagram 2.1 Expert systems application

2.3.1 Heuristics

Expert systems are considered as a branch of AI because the method of problem solving is predominantly based on heuristics. This contrasts very much with the conventional programming paradigm that uses algorithms to solve problems. An algorithm is a step by step procedure that solves a category of problems. For example, algorithms may be used to process a company payroll. A typical algorithm would use a step by step procedure on the input data such as employee hours worked, overtime rate and so on, to generate output in the form of payslips for employees. The steps in this procedure involve direct manipulation of numeric data to produce information.

Heuristics, on the other hand, solve a problem by trial and error guided by some reference to a predetermined goal. There are many examples that we may encounter in our daily lives. For example, a motorist searching a multistorey car park for a parking space would not use an algorithm to find a space. There is no guarantee that whatever procedure is adopted a parking space will be found. The motorist may for instance, drive to the top-level first rather than searching each level in turn. Whilst this strategy may sound attractive there is no guarantee it will work: there may be no more spaces on the top-level available. The motorist then may have to try a lower level.

2.3.2 Representing knowledge

Knowledge is the fuel that powers the inference engine, without knowledge the expert system is just an empty shell without function. What is knowledge? Knowledge is a term that denotes the understanding of an individual on a given subject. For example, take the understanding of a doctor in the area of surgery. For the building of an expert system, we don't try to capture all the expert knowledge, we just focus upon a domain specific knowledge that will form the building block of our expert system, in this case for example, child related disease. After acquiring this knowledge from an expert in a well-focused domain, the next step is encoding the knowledge in the expert system. There are several types of method for structuring the knowledge in an expert system. Each technique has advantages and disadvantages for capturing efficiently the different types of knowledge. The important point is to choose the correct representation for a given application to produce a structure that supports effective problem solving. Two method of knowledge representation will be discussed for this review. These 2 methods were chosen because they strongly relate to the knowledge representation that will be used latter for the development of the system.

I. Rule based

Rules are knowledge structure that relates to some known information to other information that can be concluded or inferred to be known. A rule is a form of procedural knowledge. It associates given information to some action. This action may

be the assertion of new information or some procedure to perform. In this sense, a rule describes how to solve a problem. The rule's structure logically connects one or more **antecedent** (also called **premises**) contained in the **IF** part, to one or more consequents (also called **conclusions)** contained in the **THEN** part. For example,

IF The child has fever

THEN give child lots of water to drink

For this simple example, if the child has fever, then the rules infer that the child should be given lots of water to drink. In General, a rule can have multiple premises joined with **AND** statements (conjunctions), **OR** statements (disjunction), or a combination of both. Its conclusion can contain a single statement or a combination joined with an **AND**. The rule can also contain an **ELSE** statement, that is inferred to be **TRUE** if one or more of its premises are **FALSE**. The Following is an example of a general rule structure.

IF Child has Headache

AND Vomiting

AND Sleepiness

AND High Fever

OR No Fever

THEN suspect Child suffers INFLUENZA

ELSE test for ENCEPHALITIS

In a rule-based expert system, domain knowledge is captured in a set of rules and entered in the system's knowledge base. The system uses these rules along with information contained in the working memory to solve a problem. When the **IF** portion of the rule matches the information contained in the working memory, the system performs the action specified in the **THEN** part of the rule memory. When this occurs, the rule fires and its **THEN** statements are added to the working memory. The new statements added to the working memory can also cause other rules to fire.

II. Logic

The oldest form of knowledge representation in a computer is logic. The most often used in intelligent system have been propositional logic and predicate calculus. Both techniques use symbols to represent knowledge and operators applied to the symbols to produce logical reasoning. They offer a formal well-founded approach to knowledge representation and reasoning.

Propositional logic represents and reasons with proposition; statements that are either true or false. Proposition logic assigns a symbolic variable to a proposition, such as

A = The child has fever

In prepositional logic, if we were concerned with the truth of the statement "The child has fever", we would check the truth of the variable. For many problems we are concerned with the truth of logically related propositions. Consider the following rule:

IF	The child body is hot	→	A
AND	Temperature is higher than 98.6°F	→	B
THEN	Child has fever	→	C

Propositional logic provides logical operators such as AND, OR, NOT, IMPLIES, EQUIVALENCE, that allow us to reason with various rule structures. Table 3.1 list the propositional logic operators and their common symbols.

Operators	Symbols
AND	∩, ^
OR	V
NOT	!, ~, ¬
IMPLIES	→
EQUIVALANCE	≡

Table 2.1 Common logic symbols

Using the symbols of table 2.1, we can write the rule as

$A \cap B \rightarrow C$

Propositional logic offers technique for capturing facts or rules in symbolic form and then operates on them through the use of logical operators. This formal logical approach provides an exact method for managing statements that are either true or false.

Predicate calculus is an extension of propositional logic that provides a finer representation of the knowledge. For example, instead of representing an entire proposition with a single symbol, such as **A = Child has symptom** for fever, the predicate calculus permits a representation that describes the relationship of the knowledge in form of **symptom(child,fever)**. This representation enhances knowledge processing by allowing the use of variables and functions.

The predicate calculus like propositional logic, uses symbols to represent knowledge. These symbols may represent either constants, predicates, variables or functions. Predicate calculus also permits you to operate on these symbols using the propositional logic operators.

- Constants

Constants are used to name specific objects or properties about the problems. In general constants are symbols that begin with a lowercase letter

- Predicates

In predicate calculus, a fact or proposition is divided into two parts: a predicate and an argument. The argument represents the object or objects of the proposition and the

predicate an assertion about the object. For example, to represent the proposition "Child has Fever symptoms" we would use

symptom(fever,child)

In predicate calculus representation, the first word of each expression (i.e. symptoms is a **predicate** denoting some relationship between the arguments within the parentheses. The arguments are symbols of objects within the problem. The standard convention represents predicates with the first character in lowercase.

- Variables

Variables are used to represent general classes of object or properties. Variables are written as symbols beginning with an uppercase letter. For example, to capture the proposition "Symptom is fever" and " Symptom is not headache" we can write

symptom(X,Y)

In this case, variables assume or instantiate the values X=fever, headache, Y=yes,no. In predicate calculus, variables can be used as arguments a predicate expression or a function.

- Functions

Predicate calculus also permits symbols to be used to represent a function. A function denotes a mapping from entities of a set to a unique element of an another set.

- Operations

Predicate calculus uses the same operators found in propositional logic. To illustrate their use, consider the following

Jack has fever symptoms(fever,jack)

Jack has headache symptom(headache,jack)

These two predicates indicates that jack has fever and headache suppose these symptoms imply jack has flu, we could write this

symptom(fever,Y) ∩ symptom(headache,Y)→sickness(flu,Y)

This is a general implication that captures the knowledge:

"*If a person has headache and flu, then the person has the flu*"

Applying the implication produces the result: sickness(flu,jack).

The predicate calculus also introduces two other symbols called **variable quantifiers** that you can use to define the range or scope of the variables in a logical expression. The two quantifiers are ∀, the universal quantifier, and ∃, the existential. The universal quantifier ∀ indicates that the expression is **TRUE** for all values of the designed variable. The existential quantifier ∃ indicates that the expression is true for some values of the variable. i.e, at least one value "exist" that makes the statement true.

The prior sections illustrated how to use the predicate calculus to represent actual knowledge in symbolic form. However, an intelligent system also needs a way to reason with this knowledge. Through the use of predicate calculus operators, this reasoning ability can be provided to the system. Reasoning requires the ability to infer conclusions from available facts. One simple form of inference is **modus ponens**. Modus ponens states that if some proposition **A** is true, and **A implies B** is true, then propositions **B** is true.

IF A is true

AND A→B is true

THEN B is true

2.3.3 The Inference Engine

The real forte of expert systems is their capacity to make inferences or the drawing of conclusions from premises. This is precisely what makes an expert system intelligent. Even when it is possible to represent domain knowledge as rules, a human expert would not only have to know how to apply these rules but in which order they should be applied to solve a particular problem. Similarly, a computer expert system would need to decide which, and in what order, the rules should be selected for evaluation. To do this, an expert system uses an inference engine. This is a program that interprets the rules in the knowledge base in order to draw conclusions. It combines facts contained in the working memory with knowledge contained in the knowledge base. From this action it is able to infer new information that it then adds to the working memory. Figure 2.1 illustrates this concept

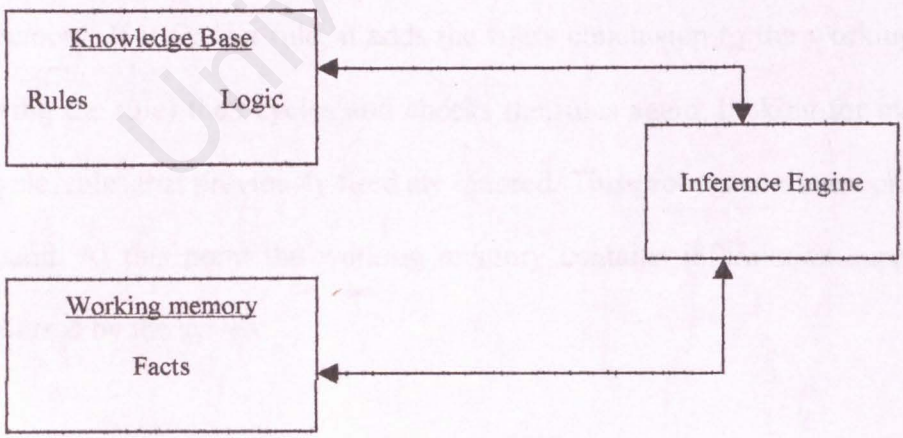


FIGURE 2.1 Inference process in expert system

Two alternative strategies are available: backward chaining and forward chaining. A particular inference engine may adopt either or both.

I. Forward Chaining

The solution process for some problems naturally begins by collecting information. This information is then reasoned with to infer logical conclusions. For example, a doctor normally begins patient diagnosis by first asking the patient about his or her symptoms. Sore throat, high temperature, or coughing are typical responses. The doctor then uses this information to infer a reasonable conclusion or to establish a hypothesis to further explore. This style of reasoning is modeled in an expert system using data driven search, it is also called forward chaining. Forward chaining is similar to modus ponens discussed earlier. The simplest application of forward chaining in a rule based expert system proceeds as follows. The system first obtains problem information from the user and places it in the working memory. The inference engine then scans the rules in some predefined sequence looking for one whose premises match the contents in the working memory. If it finds a rule, it adds the rule's conclusion to the working memory (called firing the rule) then cycles and checks the rules again, looking for matches. On a new cycle, rules that previously fired are ignored. This process continues until no matches are found. At this point the working memory contains information supplied by user and inferred by the system.

Advantages of Forward Chaining

- A major advantage of forward chaining is that it works well when the problem naturally begins by gathering information and then seeing what can be inferred from.
- Forward chaining provides a considerable amount of information from a small set of data.
- Forward chaining is an excellent approach for certain types of problem solving such as diagnosis, monitoring, control and interpretation

Disadvantages of forward chaining

- One of the major disadvantages of forward chaining is that it may have no means of recognizing that some evidence might be more important than others. The system will ask all possible questions, even though it may only need to ask a few questions to arrive at conclusion.
- The system may ask unrelated question. Though the answers to the questions may be important, it is disconcerting o users to answer questions on unrelated subjects.

II. Backward Chaining

By contrast, a backward chaining inference engine starts from the other end. It begins with a hypothesis and then attempts to prove it by gathering supportive information. For example, a doctor may suspect some problem with a patient, which he then attempts to prove by looking for certain symptoms. This style of reasoning is modeled in an expert system using goal driven search is called backward chaining.

A backward chaining system begins with a goal to prove. It first checks the working memory to see if the goal has been previously added. This step is necessary since another knowledge base may have been previously proven, the system searches its rules looking for one or more that contains the goal in the THEN part of the rule. This type of rule is called a goal rule. The system then checks to see if the goal rule's premises are listed in the working memory. Premises not listed then become new goals (also called subgoals) to prove, that may be supported by other rules.This process continues in this recursive manner, until the system finds a premise that is not supported by any rule - a **primitive**.When a primitive is found, the system is asks the user for information about it. The system then uses this information to help prove both the subgoals and the original goal. The backward-chaining process is similar to hypothesis testing in human problem solving.

Advantages of Backward chaining

- One of the major advantages of a backward chaining system is that it works well when the problem naturally by forming a hypothesis and then seeing if it can be proven

- Backward chaining remains focused on a given goal. This produces a series of questions on related topics, a situation that is comfortable for the user. Backward Chaining system searches only that part of the knowledge base that is relevant to the current problem

- Backward chaining is an excellent approach for certain types of problem solving tasks, such as diagnostics, prescription and debugging

Disadvantages of Backward Chaining System

- The principle disadvantage of a backward chaining system is that it will continue to follow a given line of reasoning even if it should drop it and switch to different one.

2.3.4 Explanation facilities

The ability to explain their reasoning processes are another key feature of expert systems. Such explanation facilities provide the user with a means of understanding the system behaviour. This is important because a consultation with a human expert will often require some explanation. Many people would not always accept the answers of an expert without some form of justification. For example, a medical expert providing a diagnosis and treatment of a patient would be expected to explain the reasoning behind his/her conclusions: the uncertain nature of this type of decision may demand a detailed explanation so that the patient concerned is aware of any risks, alternative treatments, and so on.

The characteristics that distinguish expert systems from conventional systems are summarised in table 2.2

Characteristic	Expert System	Conventional system
Underlaying Paradigm	Heuristic. Usually implemented using state space search.Solution steps implicit (not determined by programmer).Solution if found, not always guaranteed correct. Usually declarative problem solving paradigm	Algorithmic. Solution steps explicitly written by programmer. Correct answers given. Procedural problem solving paradigm.
Method of operation	Reasons with symbols. For example, infers conclusion from known premises in order to diagnose patient ilness.Inference engine is used to decide the order in which premises are evaluated	Predominantly manipulates numeric data. For example, sorting, calculating and storing data processed to produce information such as payslips for a company payroll system
Processing Unit	Knowledge. This may be represented in the form of rules.Knowledge is active in that an expert system can reason with knowledge to infer new knowledge form given data	Data. Typically represented in the form of Arrays or records in languages like C or COBOL. Data is passive in that it does not give rise to further generations of data
Control Mechanism	Inference engine usually separate from domain knowledge	Data or information and control usually integrated together
Fundamental components	Expert system = Inference+ Knowledge	Conventional system=Algorithm + Data

Explanation capacity	Yes. An Explicit trace of the chain of stops underlying the reasoning process. Would typically enable a user to find out how the system arrived at its conclusions or perhaps why the system is asking answer to a particular question	No

Table 2.2 Comparison between conventional and expert systems

2.4 ANALYSIS

Analysis is a study of a subject by examining its part and their relationship. For the purpose of analysis on expert system, this section starts with MYCIN.

2.4.1 History of MYCIN

MYCIN was developed at Stanford University to aid physician in diagnosing and treating patients with infectious blood disease caused by bacteraemia and meningitis. These diseases can be fatal if not recognized. The system was developed during the mid-1970s and took approximately 20 person years to complete. The system was written in INTERLISP a dialect of the LISP programming language.

Major features of MYCIN

- **Utilizes a Backward Chaining System**

  MYCIN works in a backward chaining fashion to work with the 500 hundred rules in order to identify the nature of the infection

- **Separates Knowledge from Control**

  This is a trademark of all expert system. The separation of knowledge from control allows the modification and the addition of the existing knowledge to become a simple task. This is possible because the rules are independent pieces of knowledge that are used on an as needed basis.

- **Incorporates Meta Rules**

Meta Rules are included because at times, simple backward chaining search scheme was inadequate. At times, the program seems to explore an issue that seems invaluable instead of considering other areas. To enable the system to redirect its search, meta rules were incorporated.

- **Employs Inexact Reasoning**

MYCIN can also work with inexact or incomplete information. This inference technique is managed with a numeric type of value called **certainty factor** (CF), a reflection of the degree of belief in the answer. This numeric is based on a scale of -1 to +1, where -1 represents definitely false and +1 definitely true.

- **Remembers Prior Session**

MYCIN remembers information from prior sessions concerning a given patient. This capability is important in those situations where new information may become available at a later time.

- **Provides Explanations**

MYCIN can explain why it's asking a question and how it derived a conclusion. MYCIN can also explain why it found other results implausible. These features provide a more natural interaction and a transparency to the system's reasoning.

2.5 SYNTHESIS OF EXPERT SYSTEM

Since the advent of MYCIN there have been numerous other successful expert system that have been developed. Expert system are not only applied in areas of medicine but also in other areas. MYCIN was developed 26 years ago, since then many Expert systems have been built, that were successful in their applications. Some of the expert system that were successful are PUFF, CENTAUR, INTERNIST, PERFEX and many others. From the study that was made, most of the issues of building the expert system were around the areas of knowledge representation technique, control strategy of the inference mechanism and the interface design. In the beginning of this chapter, only 2 forms of knowledge representation technique were discussed in order to narrow down the scope of the discussion. There are also other forms of knowledge representation technique such as frame based technique which is an extension of the schema idea first proposed by Bartlett. It also borrows the similar ideas of OOP (Object Oriented Programming) paradigm, like instances and hierarchies. In the reasoning process also, expert systems now contain the power to handle inexact reasoning by incorporating Bayesian Probability, Fuzzy Logic, Neural Networks, Belief Networks and others. These issues were not explained thoroughly in this chapter because they are not going to be implemented in the Child Medical & Health Advisor system because of the complexity involved and also because the author does not have a deep understanding of fuzzy and bayesian system. From the research that was made, only certain systems was very helpful in giving ideas on designing the Child Medical & Health Advisor. These systems were previously built by the former students of the Faculty of Computer Science Universiti Malaya. One system that was particularly helpful was the Horse Expert

system developed by Kamarul Arrifin bin Jumaein. This expert system had a domain focused on horse illness. It took a multimedia-based approach in the development of the expert system. The interface provides an easy to use approach to the user. It implemented a " point and do" interaction technique for the horse diagnosis. Instead of typing yes or no, the user just had to click the respective button to complete the diagnosis session. The Horse Expert system used a forward chaining approach to diagnose the horse illness. It also used rules as means of knowledge representation technique.

In the coming 2 chapters, the system methodology and design will be discussed. The selection of the knowledge representation technique along with the control strategy will be addressed.

CHAPTER 3
METHODOLOGY

CHAPTER 3:METHODOLOGY

From the Oxford Advance Learner's Dictionary of Current English (5th edition), methodology carries the meaning, a set of methods used in a particular area of activity. The methodology for developing an expert system is different from normal System Development Life Cycle (SDLC). The main difference is that SDLC is more reliant of data while expert system relies more on knowledge. The main paradigm of designing, testing and maintaining is the same. For the SDLC the main methodology for system design is

I. Requirement Analysis & Specification

II. Designing the system

III. Designing the code

IV. Implementing the code

V. Unit Testing

VI. Integration Testing/System Testing

VII. Validation Testing

VIII. Maintenance

While the phases for the development of an expert system is

I. Assessment

II. Knowledge Acquisition

III. Design

IV. Test

V. Maintenance

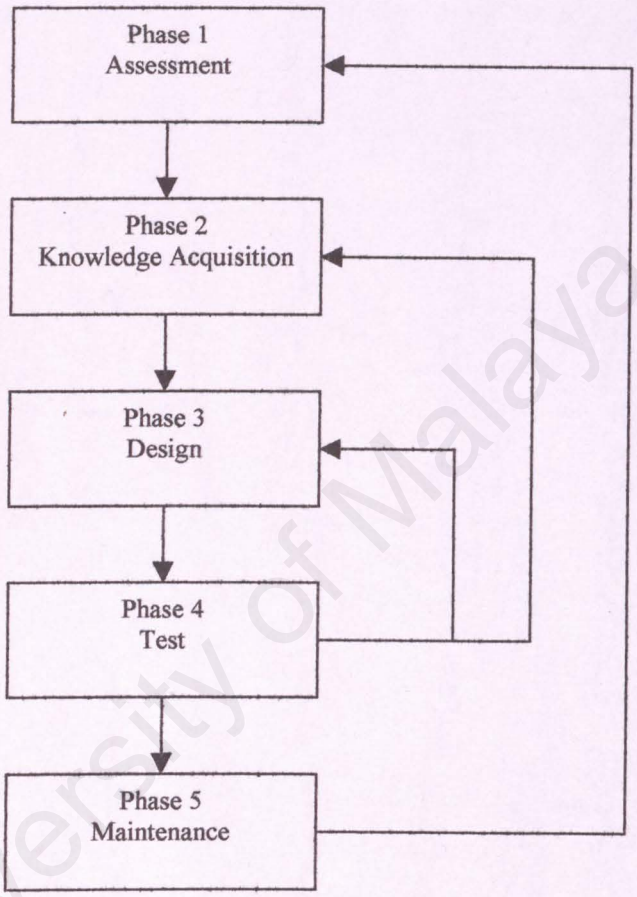


Figure 3.1 Phases in Expert system development

These phases of expert System development is also collectively known as Knowledge Engineering. The development effort is not as neat and clean as shown in figure 3.1.though the task are in sequence as shown, in practice there is considerable overlap in their execution. In addition, the process is highly iterative. As information is gained from the execution of later tasks, there will be a need to return to the earlier ones. As these tasks are cycled through, the system will begin to take shape. It gradually evolves

from one with limited knowledge to one that becomes more capable due to its improved knowledge and problem-solving skills

3.1 ASSESSMENT

Before the solution for the problem could be proposed, an assessment of the problem was made to make sure it is suitable to solve the problem using an expert system based approach. These following issues were considered when making the assessment:

- **Expert knowledge needed** - The problem requires expertise to solve, which includes both expert knowledge and expert problem solving skills.

- **Problem solving steps are definable** - The major steps used by the expert when solving the problem can be clearly defined.

- **Symbolic knowledge used** - The type of knowledge used by the expert is symbolic in nature, rather in the numeric form that is more commonly found in conventional programs.

- **Heuristics used** - The expert uses rules of thumb gained from past experience to guide the problem solving.

- **Problem is solvable** - Expert systems are not intended to address new or research issues, but rather to solve problems that can currently be solved by human experts.

- **Problem is well focused** - The overall scope of the problem is manageable, focused on an issue.

- **Problem is reasonably complex** - The problem is reasonably complex, not too easy where the effort may not be justified, or too difficult, where the problem may not be manageable.

- **Solution more of a recommendation** - The problem does not require an exact answer, rather an educated recommendation.

- **Problem is stable** - The knowledge and the approach to solving the problem are stable.

From the points given above, the diagnosis of a child illness can be solved using an expert system paradigm because there are experts and knowledge to solve this problem using heuristics and symbolic knowledge. The problem has already a narrow scope that is child disease, is well focused and most important solvable. The systems proposed have characteristics like of an information system that provides reference on child illness. It has also an added capability of diagnosing child illness and also dispenses health care related advice. It is not called an information system because of the way the system will be designed, which will follow the designs of a forward chaining expert system. As mentioned in chapter 2, it will have the main characteristics of the expert system that is a separation of knowledge from control, a knowledge base, an inference engine and an interface. The knowledge will be encoded symbolically using predicate logic. It will require data on the problem to give a diagnosis on the illness. It can also give information obtained from the knowledge base to give reasoning why the following decision was made.

3.2 KNOWLEDGE ACQUISITION

After an assessment of the problem was undertaken, the next phase is acquiring knowledge. This is a critical phase of the system development because the knowledge acquired will be used to model the knowledge base. For the development of the Child Medical & Health Advisor, a decision was made to acquire the knowledge from books, journals and websites. This decision was made because of the time constraints in developing the system. There are several difficulties in having to elicit this knowledge from an expert and this was made more difficult because of the time limit given for WXES 3181. Getting an expert knowledge was a task, which was difficult to accomplish for this project because the expert has to make himself available in order to get the information, which is not a task that can be accomplished in a single session. However, an alternative approach exists to getting the knowledge required for this system. There are books written in child illness that are written by doctors to help parents to identify child illness and also give excellent guide to address the health issues that children often get inflicted. Thus an expert system can also be developed by using the knowledge from these books and also from numerous websites on the Internet as an intelligent substitute for the book themselves. One of the primary source knowledge will be obtained from this book Your Child: A Medical Guide written by Ira J. Chasnoff, with the Editors of Consumer Guide. This book gives an excellent reference to childhood related disease and also how to diagnose these illnesses. For the purpose of validating the knowledge that will be provided by the book, the help of a real physician will be used to verify and to improve the knowledge when testing the system.

3.3 DESIGN

After the required knowledge has been obtained, an understanding of the problem will be constructed from this knowledge and thus it will be sufficient to begin the design of the system. These are the several tasks that will be performed for the design stage.

3.3.1 Selecting Knowledge Representation Technique

A knowledge representation technique should be chosen that matches the problem's knowledge. The Knowledge representation technique proposed for this problem is a rule-based approach. This technique was chosen because for recognizing the diseases, the knowledge used was in the form of rules. Thus it is a very suitable form of knowledge representation. For example in order to diagnose the symptoms of asthma

IF WHEEZING

AND DIFFICULTY BREATHING OUT

AND SHORTNESS OF BREATH

AND SENSATION OF AIR HUNGER

THEN SUSPECT ASTHMA

It is easier also to convert these rule based to form of logic based predicate calculus in order to code this knowledge.

3.3.2 Selecting Control Technique

The control technique that will be used will influence the way the inference engine will work with the given data to get the relevant rules and establish the goal. For this system, Forward Chaining will be used as the control technique. This is because, for the problem of diagnosis of child disease, initial information on the problem must be collected and then reasoning with it to come up with a conclusion. The data is the driver behind the reasoning process. Furthermore, the number of diseases to be diagnosed is 30 while the total number of symptoms that will be needed to identify these diseases is 17. Because the amount of data to consider is smaller than the number of solutions, forward chaining is a better choice. Only 30 disease were selected because to reduce the complexity of diagnosing the disease and also these are the common ailments that strikes during childhood

3.3.3 Selecting Systems Development Software

For this task, attempts are made to match features of the problem with the capabilities of available software. Other important issues that should be considered are the practical issues of available resources and programming skills. For the systems development, the software that was chosen is **Visual Prolog v5.2**. The history of prolog dates to the early 70's. Prolog is an abbreviation of (Programming Language), it was developed by the combined team of the Artificial Intelligence Group at the University of Aix-Marseille, together with the Department of Artificial Intelligence at the University of Edinburgh. The primary components are a method of for specifying predicate calculus propositions and a restricted from of resolution. It has a built in inference mechanism, which makes it easier to write logical deducing statements. The syntax of the program makes it easier to write the code for facts statements, rule statements, and goal statements. Prolog easily fulfills the basic needs of an expert system, using resolution as the basis for query processing, using its ability to add facts and rules to provide learning capabilities, and using its trace facility to inform the user the reasoning behind a given result. Furthermore, the rules for the database can be expressed in forms of predicate calculus, which makes it easier to code in prolog language. Visual Prolog is one of the new additions to the different dialects of Prolog. The name Visual was added because in addition of the normal prolog capabilities, visual prolog offers the capabilities of adding Graphical User Interface components such as windows, buttons forms and etc. This is gives an opportunity to move from text based interface to a graphical interface.

3.3.4 Hardware Requirements

The minimal hardware to run the once completed system is:

- A personal computer (PC)
- Windows 98 operating system
- 233 MHz processor speed
- 60 MB hardware space
- 64 Mb Random Access memory

3.3.5 Prototype Development

After the systems development software has been identified, building a small system prototype can start out the system development. A prototype is a model of the final system. Its basic structure, in terms of the way it represents and processes the problem's knowledge, is the same as expected in the final system. Though the prototype is only a small version of the final system and has limited ability, its serves the following purposes:

- Validates the expert system approach
- Confirms the choice of the knowledge representation technique and control strategies

- Provides a vehicle for knowledge acquisition

There are several issues that should be considered when building the system prototype

- **Defining global strategy**

This strategy is a series of high level tasks that the system will need to perform. It provides a general approach to the problem, and a high level view of the problem solving approach. The system designer must get a general insight into structuring the problem solving approach. The global strategy represents a goal for the system to achieve. In this phase, a flowchart is created and agendas formulated.

- **Defining knowledge structure**

During prototype development, a framework that accommodates future changes should be crafted. A well-structured prototype will ease the development and maintenance efforts. Plans for maintaining the expert system begins in this stage. The system developer must think how to maintain the system following the overall development effort. A good way to begin forming the knowledge is to list the problem's major goals and attributes. This information represents the knowledge about the problem that was compiled during the knowledge acquisition phase. This information serves as a source for encoding the rules or decision tables.

3.3.6 Interface Development

Interface specifications should be defined at the beginning of the project with the prototype development of the expert system. The interface is an integral part of the expert system development process. Some of the important general issues that should be considered:

- **Consistent screen format**

The presentation of the interface is as important as the information it contains. A good interface must display consistency. Each screen might contain several types of materials to present such as title, questions, area for answers, and control functions. All screens should be designed so that similar material is placed in the same locations. The user will be able to develop a mental mode of where to find the expected information. Moving onto the next screen, the user is prepared to find the needed information in the expected place.

- **Clarity of presented material**

The material presented in the screen must be presented in a clear manner so that the user will be receptive to the system and the reliability of the exchange of information between user and system is enhanced. Question design is an important issue here. The question must not be confusing or complex, they must be clear and simple. Clear explanations and results must also be addressed in this task. The contents of the screen must simple and uncluttered. Only the

material that is necessary to accomplish the purpose must be included in the screen

- **Screen Control**

The system must make the user feel that he is in control when consulting the system. The user must also not be afraid to make mistake. These two requirements place additional demands on the user interface. The system must be easy to start and exit. The exit system should be available in every screen. This must be achieved through function keys or control keys within the interface. The user must also be prevented from making serious mistake. One way is to do this is to give the user an interface that is restrictive and error catching interface. This type of interface allows the user to point to parts on the screen-using mouse, to select appropriate action. This avoids the problems that can occur from typing errors with text based interface.

3.4 TESTING & DEVELOPMENT

The development and the testing of the system will be conducted in the second phase of the project alongside the building of the prototype. The testing strategies used and the validation of the system will be covered in more details in chapter 6 and 7. The details that should be covered are the decisions made on what to be tested, how and when the test will be conducted, who will involved are some of the questions need to be answered. This segment is more concerned with validation and user acceptance. These issues will be discussed in the latter part of the project report.

3.5 EXPECTED OUTCOME

After the system design & analysis phase is over .The coding of the system will be started and further system testing and validation will be conducted in a highly iterative and self-correcting process. All these activities are scheduled to take place in the second part of the system development. There are several outcomes that are expected once the second phase of the system development has started, they are:

- System is able to handle all the functions that have been specified in the system objectives and scope. A well functional system is able to gain trust among the user and ensures its acceptability if its well operational, available and reliable all the time

- The system is able to help the user as it was intended. The main reason this system was proposed is to help parents to gain knowledge and understanding on child health related issues. If the system is able to inform and give knowledge to parents who are in need of this kind of information, the system has achieved its objectives.

- The system has a user-friendly interface and easy to understand. This is important because too much fancy artwork and poor planning lets down the system. The user must feel comfortable when using the system. The interface is important in obtaining and displaying information.

CHAPTER 4
SYSTEM DESIGN

CHAPTER 4: SYSTEM DESIGN

System design is a creative process of transforming the problem into a solution. A description of a solution is also called a design. In this chapter, how the system will be designed is explained. A forward Chaining system starts with problem data and fires rules to infer new information. The problem data is the fuel of the system. Some of the initial task in designing the system was:

- Defining the problem & input data

The system has 3 major goals to achieve. First, it must be able to give information on childhood diseases. As mentioned in chapter 3, 30 diseases that were selected is given in Appendix A. Also in Appendix A, a list of symptoms that will be used to diagnose the sickness is also provided. These 17 symptoms will be the data that will fire the system rules to get the diagnosis of the illness. Hence the system will be using these symptoms as input data for the diagnosis of the system. The system must also be able give information on basic child healthcare information. For these 30 diseases, not only they have their symptoms that should be coded; they have their own home care, precautions, and medical treatment.

- Planning the modules of the CMHA

The Child Medical & Health Advisor is a system that consists of smaller subsystem. There are 3 sub-system that make up the whole CMHA they are

I. The Disease Reference module

II. The Diagnosis module

III. The Healthcare Tips module

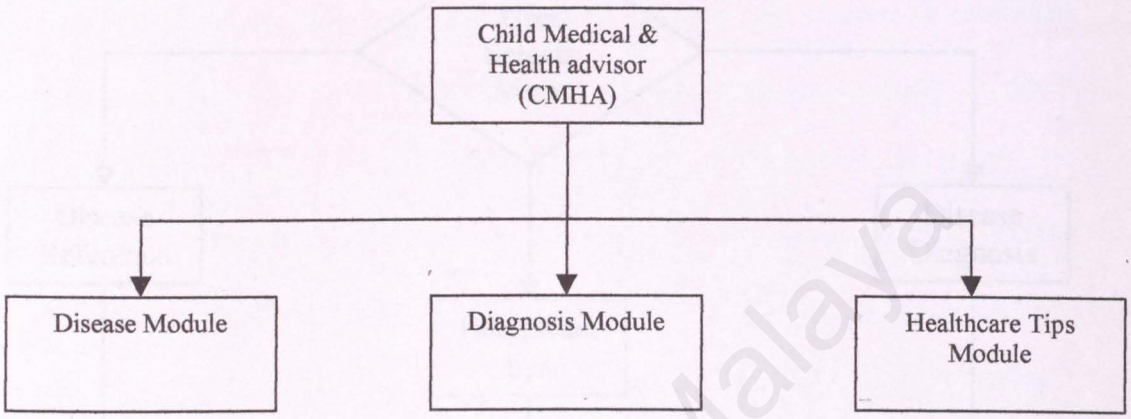


Figure 4.1 Systems Module

The figure 4.1 shows the modules involved in the systems. Each of these modules has their own interface, and coding issues that should be dealt with. These three modules must than be linked together to make a working system.

- System Flow Design

The CMHA system is started by having the user select from the main interface, the disease reference, diagnosis or healthcare tips menu. Please refer to the figure 4.2 to see the data flow of the Child Medical & Health Advisor. In the following pages a brief description will be given on the 3 menu of the system.

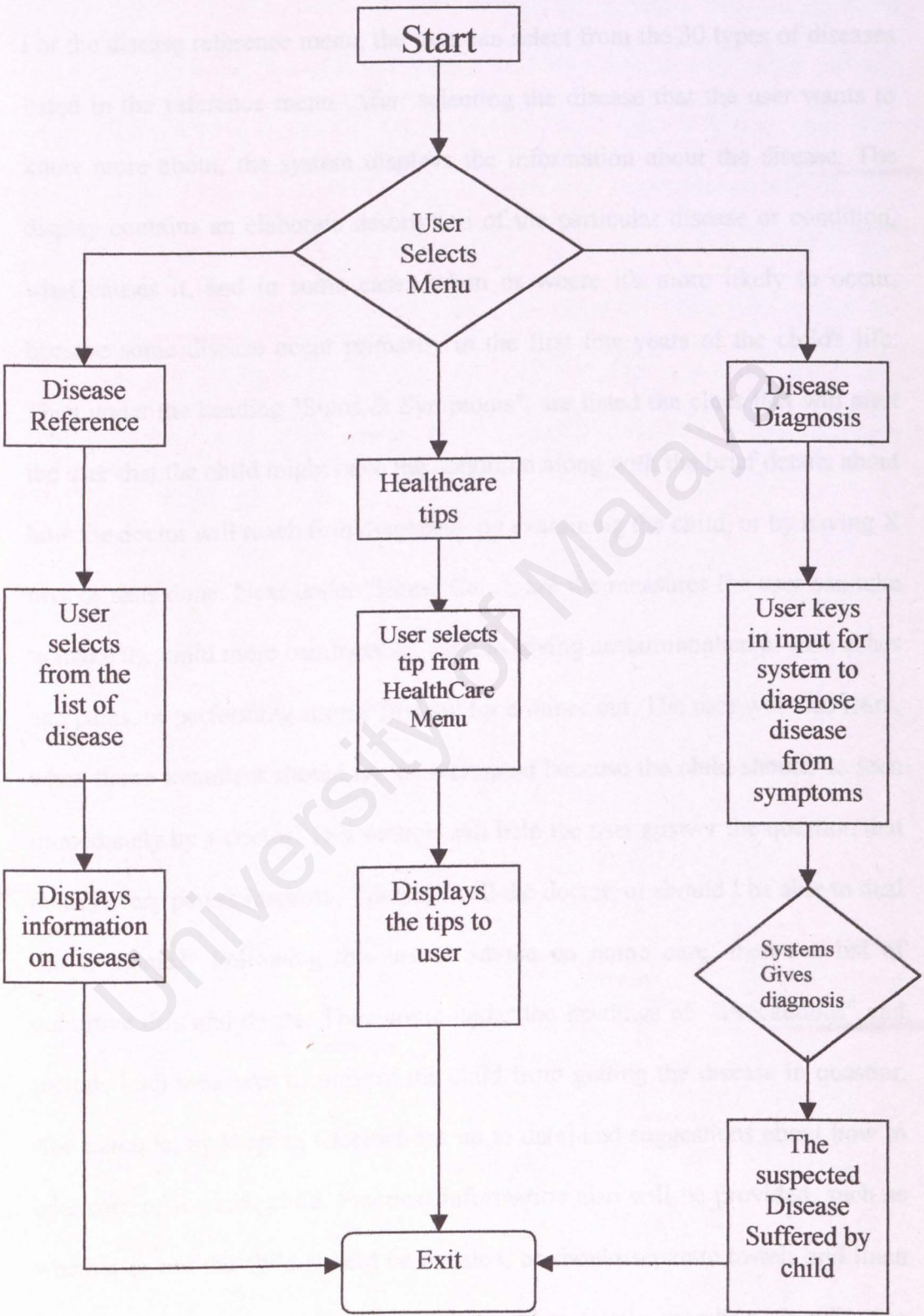


Figure 4.2 Systems Data Flow

- **Disease Reference Menu**

For the disease reference menu, the user can select from the 30 types of diseases listed in the reference menu. After selecting the disease that the user wants to know more about, the system displays the information about the disease. The display contains an elaborate description of the particular disease or condition, what causes it, and in some cases when or where it's more likely to occur, because some disease occur primarily in the first few years of the child's life. Then under the heading "Signs & Symptoms", are listed the clues that will alert the user that the child might have the condition along with the brief details about how the doctor will reach firm diagnosis- by examining the child, or by having X rays or tests done. Next under "Home Care", are the measures the user can take to make the child more comfortable, such as giving acetaminophen to ease aches and pains, or performing simple first aid for a minor cut. The user will also learn, when home treatment should not be attempted because the child should be seen immediately by a doctor. This section will help the user answer the question that makes every parent insecure, "Should I call the doctor, or should I be able to deal this at home?" Following this useful advice on home care, there's a list of definitive do's and don'ts. They come under the headings of "Precautions" and include both measures to prevent the child from getting the disease in question (for example, by keeping vaccinations up to date) and suggestions about how to take care of the sick child. Practical information also will be provided, such as whether or not the child should be isolated, or should separate towels and linen provided to avoid spreading infection to other family members. It will also

recommend whether other family members should be treated at the same time even if they do not have definitive symptom of the condition. This section alerts to the possible complications that require medical attention, and points out any situations or developments that are normal and don't need a doctor's care. The final section, "Medical Treatment" deals with medical treatment and tells the user how a doctor goes about treating this condition. This section is designed to help the user understand the doctors order and carry them out accurately so that the child gets well again as fast as possible. Here the user will also learn if any follow up care is necessary once the child is well. The figure 4.3 shows the data flow in this menu.

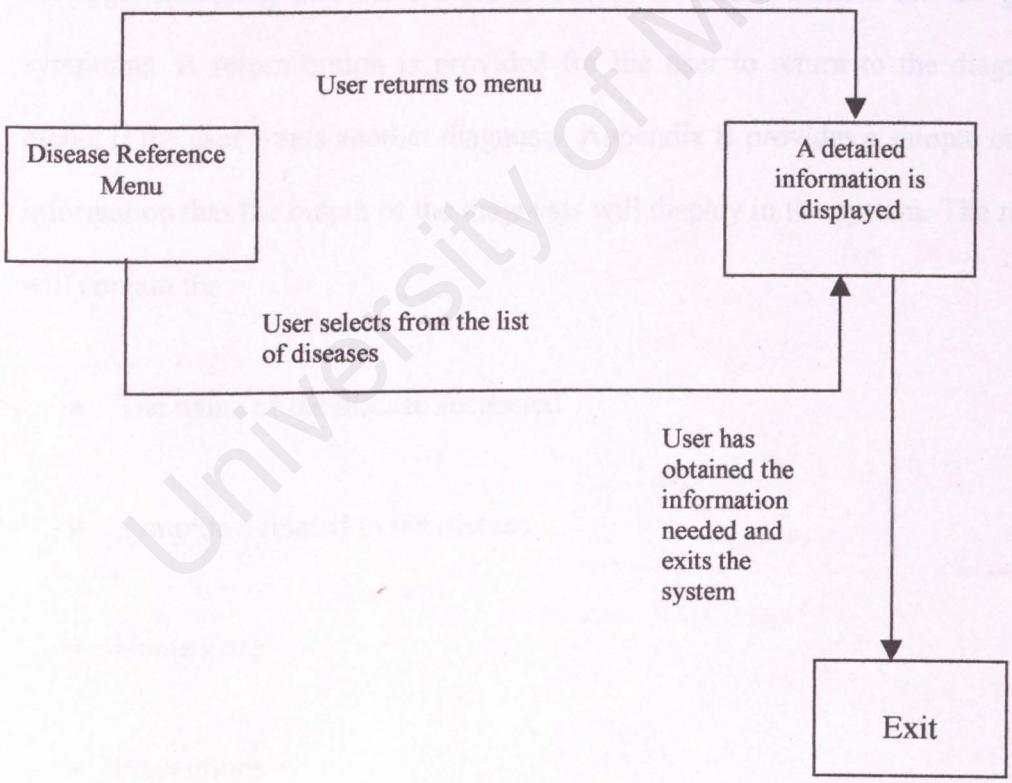


Figure 4.3 Data Flow for Disease Reference Menu

- **Diagnosis menu**

The diagnosis menu will handle the diagnosis of the child disease. For the purpose of reducing the complexity of the problem within the given time limit. The diagnosis will only cover 30 diseases. Please refer to Appendix A for the list of the diseases that have been selected. This interface will practice a "point and do" concept. This type of interface allows the user to use the mouse button to click on the yes and no button. After asking the user for the 17 symptoms listed in the Appendix A, the system should be able to pinpoint the disease that the child suffers. If the system is unable to figure a disease to match it will display a message indicating that the system is unable to find a disease for the given symptoms. A return button is provided for the user to return to the diagnosis menu, if the user wants another diagnosis. Appendix B provides a sample on the information that the output of the diagnosis will display in the system. The result will contain the

- The name of the disease suspected
- Symptoms related to the disease
- Home Care
- Precautions

Figure 4.4 displays the control flow for the Diagnosis Menu.

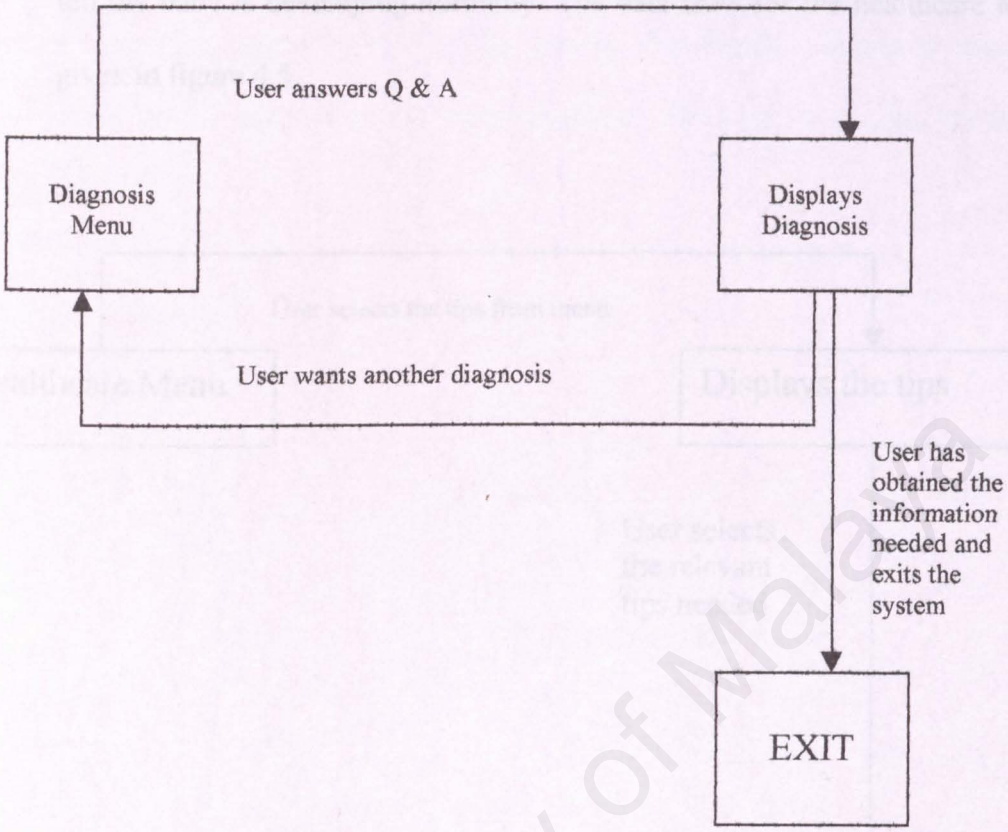


Figure 4.4 Data Flow For Diagnosis Menu

- **Healthcare tips menu**

This section covers basic information that every parent needs to know. There's sensible advice on how to choose a doctor, what to expect from the doctor, and how the user can make sure and the child's doctor can work together to take good care of your child's health. The user will find out why vaccinations are vital; why routine check ups are so important and what the doctor will look for in the course of a medical examination; what's the need to have a medicine chest at home; what kinds of medications the doctor will order and others. There is also a

reassuring section for parents on what to expect from a newborn baby and how to tell the baby is developing normally. The data flow for the healthcare Menu is given in figure 4.5

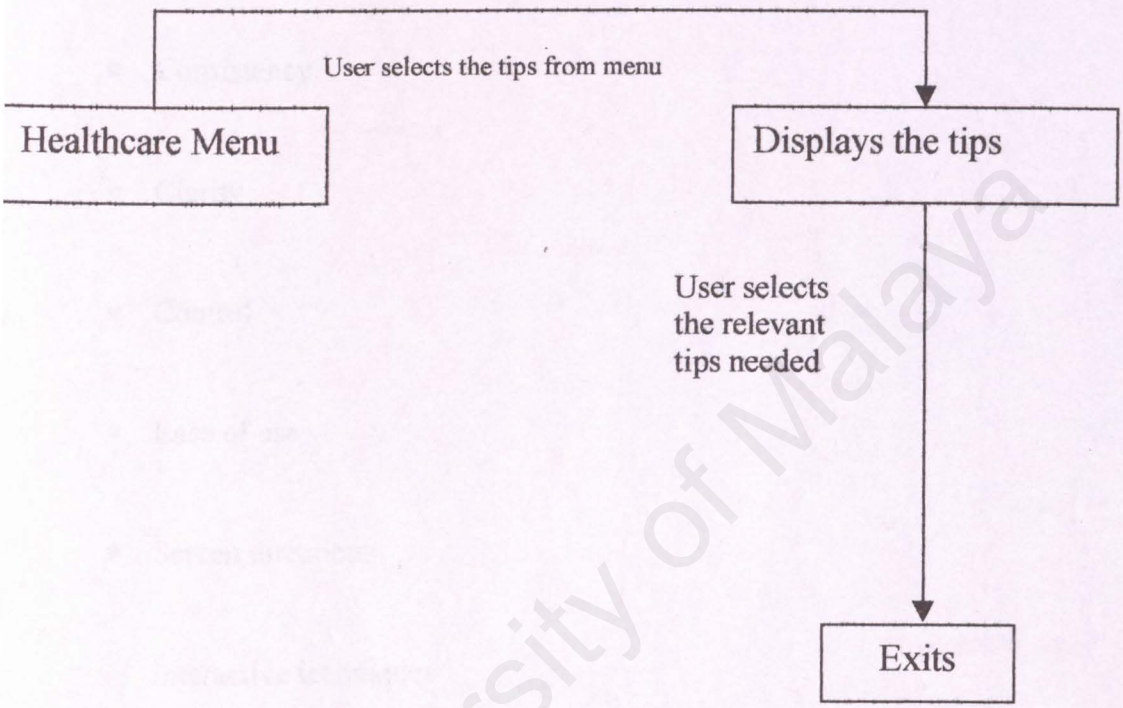


Figure 4.5 Data Flow For Health Reference Menu

- **Designing the Child Medical & Health Advisor (CMHA) interface**

Interface is an important component of the system. The interface should be designed in parallel with the development of the knowledge base. The keys for an effective interface design are:

- Consistency
- Clarity
- Control
- Ease of use
- Screen directions
- Interactive techniques

The CMHA has 4 menus that will make up the system. The first is the main menu. This is the interface that will greet the user. The user can choose from the menu, the Health Reference menu, Diagnosis Menu, HealthCare Tips menu, or the option to exit the system. A brief description of the system will also be provided in the main menu for the user to get an overall understanding of the system. Please refer to the figure 4.6.for the Main Menu. For the Health Reference, the design of the menu will follow the figure 4.7, because there are 30 diseases to recover. There will be two interfaces in this section, the first interface will cover 15 diseases and the second 15 diseases, buttons will be provided

so the user can switch to either menu. Figure 4.8 is a design on the outlook of the diagnosis menu. The final figure, figure 4.9 displays how the HealthCare Tips menu will look in the system.

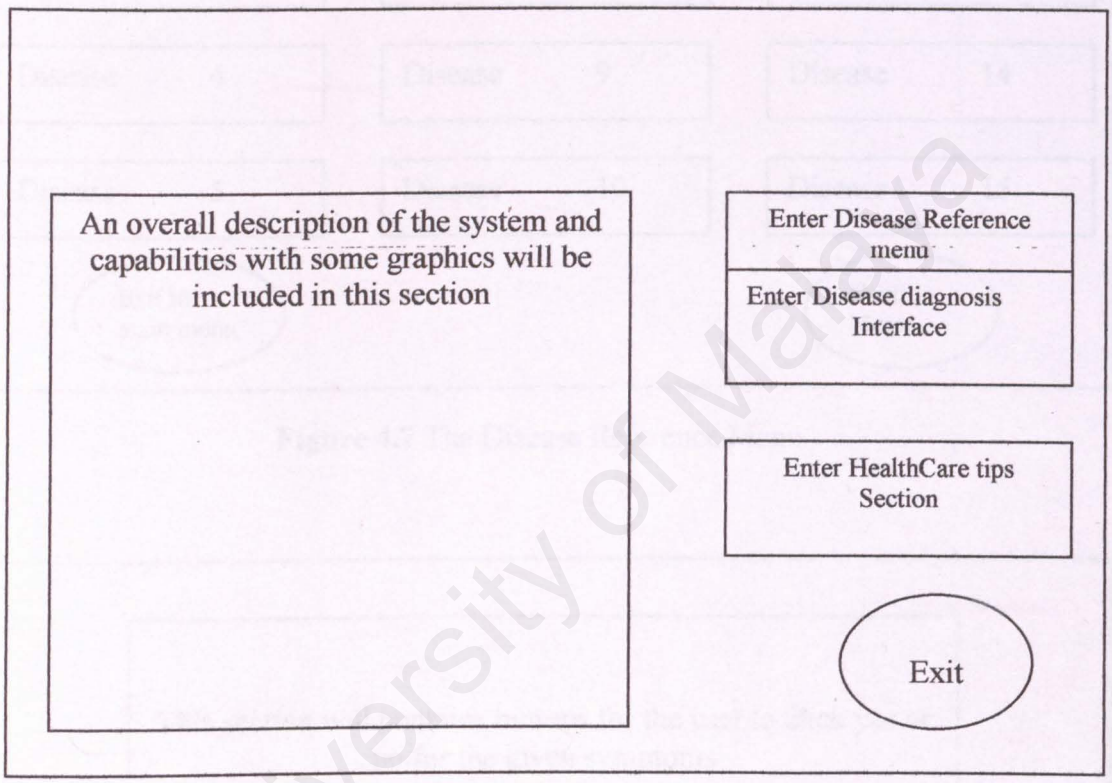


Figure 4.6 The main menu for CMHA

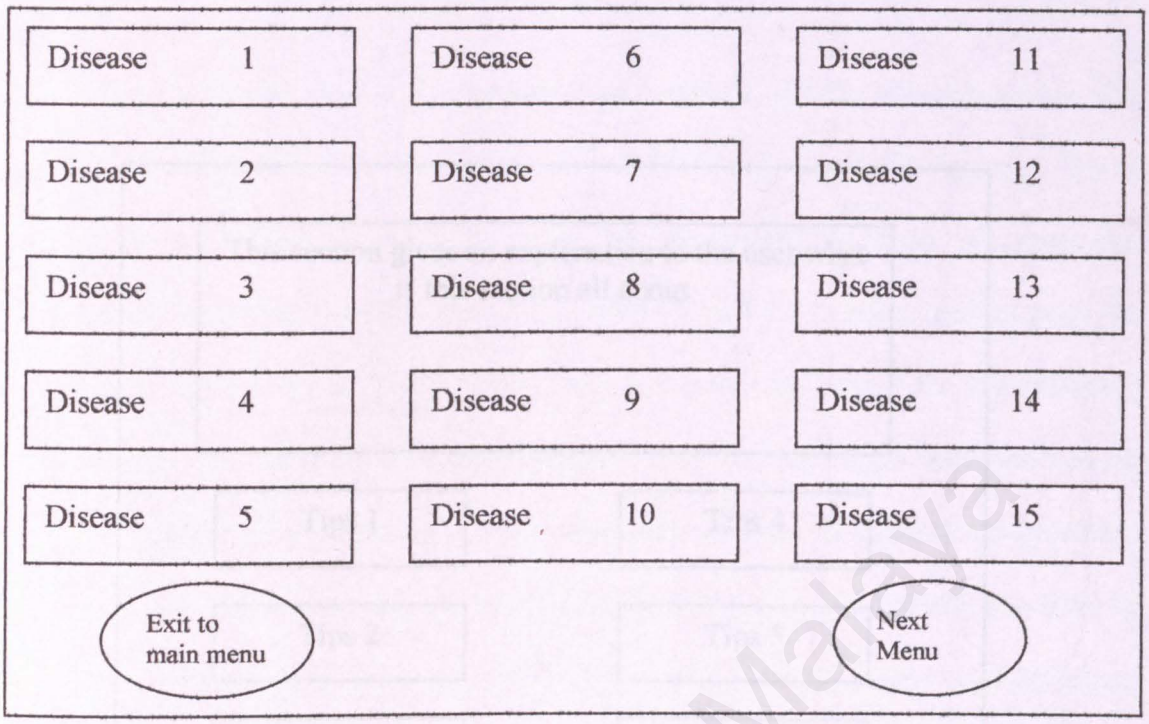


Figure 4.7 The Disease Reference Menu

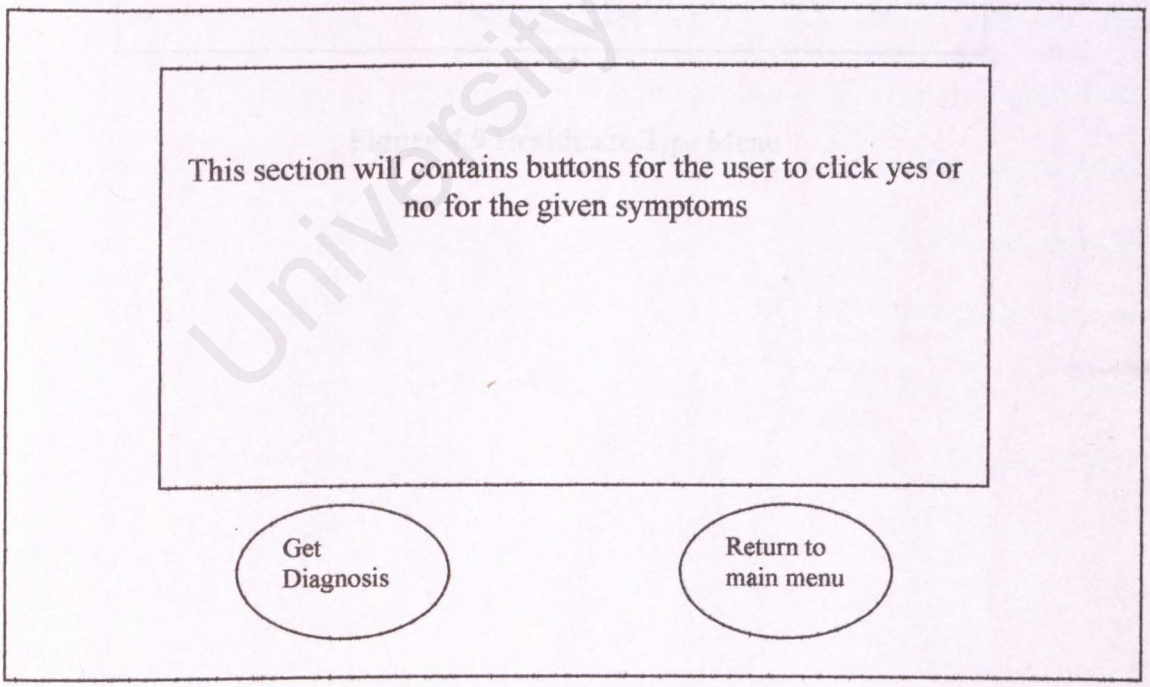


Figure 4.8 The Diagnosis Menu

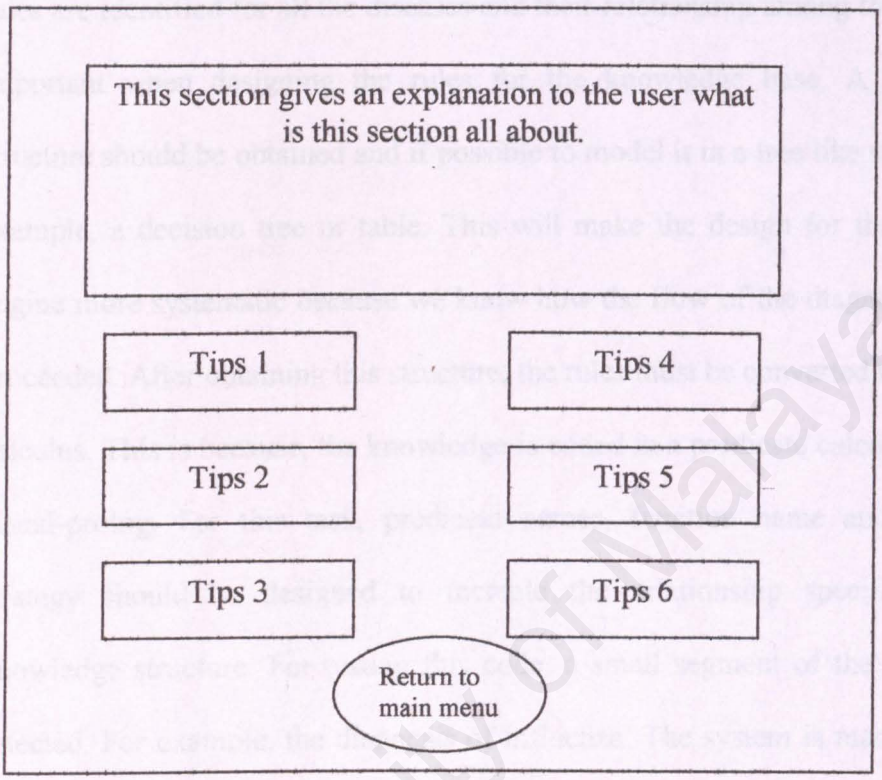


Figure 4.9 Healthcare Tips Menu

- Knowledge Base & Inference Engine design

The knowledge Base is a rule-based expert system. For the knowledge base, the rules are identified for all the diseases and their relationship among them. This is important when designing the rules for the knowledge base. A knowledge structure should be obtained and if possible to model it in a tree like structure for example, a decision tree or table. This will make the design for the inference engine more systematic because we know how the flow of the diagnosis will be proceeded. After obtaining this structure, the rules must be converted to predicate calculus. This is because, the knowledge is coded in a predicate calculus form in visual-prolog. For this task, predicate names, function name and inference strategy should be designed to recreate the relationship specified in the knowledge structure. For testing this code, a small segment of the problem is selected. For example, the diagnosis of influenza. The system is made sure that its able to diagnose the sickness when the required input data is given. After this, the system is able to give the required signs and symptoms, home care, precaution and medical treatment. Please refer to Appendix B for the required Output for diagnosis. The System should be able to access its knowledge base; to give the information on the disease suffered when needed by the user. All these points must be checked when designing the knowledge base, inference engine and writing the code.

CHAPTER 5
SYSTEM IMPLEMENTATION

CHAPTER 5: SYSTEM IMPLEMENTATION

5.1 INTRODUCTION

System implementation is the phase in which the all the design specification and methodology for the system will be converted into a visible idea. The system will be built according to the specified design in chapter 4 with the appropriate tools necessary. The 3 modules that which is the heart of the system will be implemented along with the main menu that will be the link to these 3 modules. The Modules will be built using specific software tools and will be tested and debugged. This chapter is focused on the building of the system with the specified application tool so that the readers get an impression on how the system was constructed.

5.2 SOFTWARE DEVELOPMENT TOOLS

A major change was made in the tool that will be used to implement the system. In the previous chapter, Visual Prolog was mentioned as the tool to build the system. Upon commencing the system implementation, the language Visual Basic was chosen to be tool of implementation for the Child Medical & Health Advisor (CMHA). This was because:

- Visual Prolog is suitable if the system emphasizes a lot on intelligence based solutions, logical problem, and mathematical problem and vice versa. For the CMHA, the requirements were more to point and click to get the needed information by the user of the system. 2 of the modules place an importance on this aspect to get

the display of the information. The system will use a lot of these display windows to display the information and the result of the diagnosis. Thus the windows will have a lot of text embedded in them and the display of the text is also important. The usage of print line statement will make it more difficult to place the text in the way it is wanted. Visual Basic language allows writing on the eventual window itself during the coding itself so that we could see for ourselves where we want the text to be included in the window. Different size of fonts will be used, thus it will be more efficient to use visual basic to fulfill these need and to keep track the system implementation process.

- The author had encountered problem in getting the appropriate expertise in Visual Prolog. The author also had problem in getting help from people who had earlier programming experience in Visual Prolog to help him solve problems encountered. This problem was even compounded by the fact that there are virtually no books on Visual Prolog compared to Visual Basic. This hampered the author's learning experience. The Visual Basic language was chosen to replace the Visual Prolog because of the lower Learning Curve compared to Visual Prolog in order to complete the system in the given time frame.

5.3 VISUAL BASIC

Visual Basic is a Microsoft Windows programming language. Visual Basic programs are created in an Integrated Development Language (IDE). The IDE allows the programmer to create, run and debug Visual Basic programs conveniently. IDEs allows a programmer to create working programs in a fraction of the time it would normally take

to code programs without IDEs. The process of rapidly creating an application is typically referred to as Rapid Application Development (RAD).

Visual Basic language is derived from the BASIC (Beginners All-purpose Symbolic Instruction Code) which has been around for 35 years. Visual Basic provides powerful features such as graphical user interfaces, event handling, structured programming, access to WIN32 API, object-oriented features and other functions. Visual Basic is an Interpreted language that lets the program run as it is written. Interpreted Languages make good learning platform because of their quick feedback. Visual Basic is more than just a programming language. It is a Visual Experience of programming. There are 3 version of Visual Basic. For this system, Visual Basic 6 Enterprise Edition was used.

5.4 INTEGRATED DEVELOPMENT ENVIROMENT (IDE) OVERVIEW

When Visual Basic is loaded, the New Project window dialog is shown. This NewProject dialog allows the programmer to choose what type of Visual Basic program to create. **Standard EXE**, which is highlighted by default, is a standard executable is the program that uses the most common Visual Basic feature. For the system implementation, Standard Execution is used. A project is a group of related files. Collectively, the project files form a Visual Basic program. ProjectFile contains predefined features for designing Windows program.

When a **Standard EXE** is opened, it contains these following windows **Project1-Form1 (Form), Form Layout, Properties-Form1, Project-Project1** and the **toolbox**.

The **Project1-Form1 (Form)**, window contains a form named **Form1**, which is where the program's Graphical User Interface will be displayed. **Form Layout** window enables the user to specify the form's position on the screen when the program is executed. The **Properties-Form1** window displays form attributes or properties. The **Project-Project1** window groups the project's files by type.

5.4.1 Project Window

A project window is also called the **Project Explorer** and contains the project files. The Project window toolbar contains three buttons, namely View Code, ViewObject and Toggle Folders. View code button displays a window for writing visual Basic Code. View Object displays the form. The Toggle Folders Button toggles the Form folder. The forms Folder contain the listing of all forms in the current project.

5.4.2 ToolBox

The Toolbox contains controls used to customize forms. Controls are prepackaged components that are reused instead of writing them each time. This makes the writing of programs faster. List 5.1 summarizes the toolbox controls.

Table 5.1:ToolBox control summary

Control	**Description**
Pointer	Used to interact with the controls on the from
PictureBox	A control that displays images

Label	A control that displays uneditable text to the user
Textbox	A control for accepting user input.TextBoxes can also display text
Frame	A control for grouping other controls
CommandButton	A control that represents a button. The user presses or clicks to iniatiate an action
CheckBox	A control that provides the user with a toggle choice
OptionButton	A radio button.Optionbuttons are used in groups where only one at a time can be True
ListBox	A control that provides a list of items
Combobox	A control that provides a short list of items
HscrollBar	A horizontal bar
VscrollBar	A vertical bar
Timer	A control that performs a task at programmer specified intervals. A Timer is not visible to the user
DriveListBox	A control for accessing the system disk drives
DirListBox	A control for accessing directories on a system
FileListBox	A control for accessing files in directories
Shape	A control for drawing circles, rectangles, squares or ellipses

Line	A control for drawing lines
Image	A control for displaying images. The image control does not provide as many capabilities as a PictureBox
Data	A control for connecting to a database
OLE	A control for interacting with other window application

5.5 SYSTEM CODING & IMPLEMENTATION

CMHA was implemented using top down approach. The System's main interface and front menu was built first before moving on to the lower modules. As stated in the chapter 4 system design, the system has 3 modules. They are The Disease Information module, Health tips module and the Diagnostic module. Each module is developed through coding and then tested for correctness before proceeding to the other modules. The integration of the system was uptaken during the coding stages itself because each form is linked to the main then and there itself.

5.5.1 Coding Approach

In the development of CMHA using Visual Basic 6.0, the system consisted of forms that had GUI components such as CommandButton, CheckBoxes and also Labels. The starting point of CMHA, is Form1, which was built using the Standard EXE form. Using the Functions of the ToolBox, the appropriate tool where then selected to fill the forms. This is the procedure for all the forms in the system. For the Greetings Menu and the

Main Menu, the fonts that were used had 3D visual properties to it to make it more attractive to the User. These fonts were taken from the Xara Software that can be downloaded at http://www.club-xara.com/referrer/headmaker_trial.asp.

Simple commands are used to call and hide the form. For example, the greeting menu (Form1) is connected to the main menu. When the user clicks enter, the user is taken to the main menu. The previous form is hidden. Codes can be written that responds to the accordingly by the action needed in the form. An example using Form1, the command button labeled "ENTER" will lead to the action of leading to the main menu. After creating a command button on the Form1, double click on the command button and the view code window is visible, and then this command is typed

Private Sub Command1_Click()

Form1.Hide

Form2.Show

End Sub

This command means that during the running of the program, when the user clicks button"ENTER", it will close the form1 and displays a new form (form2.frm). This command applies to all form to call and hide single or multiple forms at one time. In CMHA Form2 (the main menu), has 4 types of command button (Tips,Diagnosis,Info,About). If the user clicks the about button, it displays information about the system. The display of the Diagnosis menu is made when the user clicks the Diagnosis button. The diagnosis menu is made out of a form that contains checkboxes that corresponds to a particular symptom (17 symptoms). Each of this checkboxes has

their own name accordingly, which is used to identify the checkboxes and their corresponding symptoms. These boxes are labeled chk1 until chk17. For example, for the appendicitis form(Form 17) to be shown. These are the codes

If Chk1.Value = 0 And Chk2.Value = 1 And Chk3.Value = 0 And Chk4.Value = 1 And Chk5.Value = 0 And Chk6.Value = 1 And Chk7.Value = 0 And Chk8.Value = 0 And Chk9.Value = 0 And Chk10.Value = 0 And Chk11.Value = 1 And Chk12.Value = 0 And Chk13.Value = 1 And Chk14.Value = 0 And Chk15.Value = 0 And Chk16.Value = 0 And Chk17.Value = 0 Then

Form5.Show

Please note that chk2 refers to the symptom vomiting ,chk4 (tenderness), chk6(Pain), chk11(fever) and chk13(Diarrhea). In other this rules translates that if the user test positive for vomiting, tenderness, pain, fever, diarrhea that he is might have appendicitis. The Diagnosis menu has 30 forms in it. They all correspond to the 30 illness that has been specified. These are Form4 until Form33.

The Disease reference menu (Form34) is related to the Diagnosis menu because the Disease reference menu contains detailed information on the illness that have been diagnosed. These are more in-depth information about the respective illness that contains information on diagnosing the symptom, and the course of action to take. There are 30 command buttons labeled with the illness names. The characteristic of this menu is just point to the respective button labeled with the disease and then clicking on it to get the information. Then a new window will be displayed for the user to read. This utilizes the **Form.hide** and **Form.show** codes. Form35 until Form64 are all the information about the illness.

The Health Tips Menu is the final module of this system (CMHA), the Health tips menu is Form65. This menu displays basic health care knowledge on child health. The Are 8 commandbuttons created with the toolbox, they are **Parent/Physician Partnership**, **Fever**, **Medications**, **The Medicine Chest**, **Immunizations**, **The Physical Examination**, **Medical Tests** and **The Normal Newborn Baby**. They correspond to Form 66 to Form 83. For this forms containing these information, command buttons labeled print had been inserted to the forms so that the user might be able to print these useful information. For Example to print information about Parent/Physician Partnership (Form67.frm) the command is **Form67.PrintForm**.

5.6 SUMMARY

System implementation a coding phase where the design specification are converted to a working system. The original programming language proposed Visual Prolog was replaced to Visual Basic because of the reasons specified in the beginning of this chapter. Because of the nature of visual basic that incorporates Rapid Application Development (RAD), it was able to reduce the time involved in building this system. The implementation phase is an important phase where the system is built to ensure its functionality.

CHAPTER 6
TESTING

CHAPTER 6: TESTING

6.1 INTRODUCTION

After the coding of the CMHA, there is another important phase it must go through before it can be used. The CMHA must be tested to ensure that every function in the system works correctly. The Main objective is carrying out testing is to find out faults in the earlier stage before the system is actually released to the user. A series of tests are run to find out any bug or error ant then proceeding to fix the bug

6.2 DEBUGGING AND TESTING IN VISUAL BASIC ENVIRONMENT

All applications need testing. Bugs can find their way into application during the programming stages. When an application is tested, a series of test cases are executed. During testing, random variables with extreme values are entered in user controls to ensure that the application can handle values outside the typical range. Bugs might appear during the testing phase. Debugging is a three step routine:

1. Determine the problem bugs and their locations

2. Correct the bugs

3. Retest the application to ensure that the bugs are eliminated

Bugs range from mild errors, such as misspellings or text alignment mistakes, to serious errors, such as when an application terminates the entire Windows session and loses

data. To a user, a bug is anything that doesn't match the expected program results or prevents the application from running.

Programmers might face many debugging problems when looking for bugs. Programmers must find as many as much of bugs as possible, then the programmers must test and retest to ensure that the bugs are gone and don't reappear. Careful planning before, during and after the coding process helps to reduce the time it takes to debug the application. The testing and debugging is made in the Visual Basic environment. This environment contains debugging tools that help to track and locate errors. Visual Basic might find error during compilation or preparation for the programs execution. There are three types of errors

1. Syntax Error

2. Runtime error

3. Logical Error

Visual Basic can highlight syntax error with an error message, we can use functions like Automatic Syntax Error checkboxes that automatically scan syntax errors. A runtime error is much difficult to spot just like logical errors. The programmer must track down the problem. To track the problem, a search must made through program codes looking for traces where such runtime and logical error might reside and then continue to fix the problem. If the problem involves the forms or a control's appearance onscreen, a trace to all reference must be made. Visual Basic can locate some logic errors if the logic error results in a request for Visual Basic to do something Impossible.

6.2.1 Using The Debugger

Visual basic's development environment includes a debugging tool that becomes part of the development environment when a requesting help for debugging. The following task can be done using the debug tools

- Analyze variable contents at runtime
- Stop the program at any statement and restart when ready
- Set breakpoints throughout the code that automatically stop the program execution when one is reached
- Change Variables during the execution of a program to different values from their current state to test the application
- Set watch variables that halt the program's execution when received a particular values or range of values
- Skip statements that does not need to be executed
- Use the Debug object's output window to print values during a program's execution. The debug window lets the programmer to capture output, such as variables output, without disturbing the normal form window

6.3 TYPES OF TESTING

There are 3 major types of testing that can be implemented. They are Unit Testing, Integration Testing and System Testing.

6.3.1 Unit Testing

Unit testing verifies that each component within module functions properly. It tests individual components independently to ensure that they operate correctly and accurately before the components are combined together to form a module. Each of the components is tested on its own without other system components. Unit testing tests for the following conditions in most of the system development.

- **Module Interface** is tested to ensure that the information flows properly into and out of the program unit

- **Boundary Condition** is carried out to ensure that the module operate properly at boundaries values

- **Independent Paths** test is aimed to make sure that all independent paths in a modules are tested at least once

- **Error Handling Paths** test is to confirm that the system is able to handle all expected and unexpected errors and to prompt out appropriate error message.

Normally, unit testing was done after accomplishing the coding of a component. All components were tested thoroughly for its functionality, flow of data, buttons and much

more minor checking. If there was an error found during testing, the cause of error is identified and eliminated soon by debugging the code. That component is tested again until is error free before the development of another component. Normally test cases are used to test the units.

6.3.2 Integration Testing

Integration testing is the process of verifying that the system components work together as described in the system program specification. There are several types of integration techniques that can be used they are

1. Bottom Up Integration

2. Top Down Integration

3. Big Bang Integration

4. Sandwich Integration

6.3.3 System Testing

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that the system elements have been properly perform the allocated functions.

System testing is done to ensure that the system fulfills the systems requirements. The Integrated modules must be operated with other system elements such as hardware, database and end-user. The objectives of system testing are to ensure the developed

system functions properly without faults or errors, process incoming data correctly and output accurately, expected data, guarantee that the user interface is intuitive so that the end user can easily interact with the system.

6.4 CMHA TESTING

The testing of CMHA started during system implementation itself. As mentioned in the previous chapter, the system was built top down. First the greeting interface was built, then the main menu and after this the 3 modules. Because of this the lines between system, integration and unit were blurred. For example, when the 2 early modules were built which were the main interface and the diagnosis menu, tests were carried to make sure the command buttons on each module were functions. This was already an integration testing because the modules were linked. Then the diagnosis menu was tested. All the buttons were tested to make sure correct display was made. Already a path exists between main interface module and the diagnosis menu now it's been extended until the display of the diagnosis results. There was already a path between system, unit and integration that had been explore and made sure it functions very well. This step was repeated for all the systems modules. There are 4 modules or interfaces that testing were focused one these were

- **The Main Menu**

This is the menu that will link the 3 main modules. For this interface testing, it was based on functional testing. The command buttons linking to the Disease Reference Menu, Diagnostics menu, Health Tips menu and the About Message window were tested. The Objective of the testing was to make sure that the buttons when clicked would produce the correct interface that corresponds to the label on the button. After this testing had been done, and some minor linking mistakes had been overcome. The Testing proceeded to the other menus.

- **The Diagnostics Menu**

In this menu, the user has to click on the checkboxes according to the symptoms experienced. After this, the user will click on the diagnose button. Then using these symptoms checked by the user, the system would produce a window that will display the illness suspected by the CMHA. The testing was made to makes sure that the accurate and correct window form will be produced when the user clicks the diagnose button. There are 30 diseases that have their own symptoms. Testing was made to make sure all the 30 disease symptoms had been covered by the system. Then the display information that had been written were tested to make sure that there had been no factual error by comparing this information with the reference book. The forms were made sure they had all the relevant information needed as specified in the system design. After making sure the linking from the checkboxes to the illness display was correct and smooth, the testing was then done to the other remaining modules.

- **Disease Reference Module**

The disease reference menu emphasizes on point, click and display function. Again the same 30 diseases that had been covered in the diagnosis menu are displayed in the disease reference module. Testing were focused on the smooth linking between the command button and the information that will be displayed in the corresponding window. Accuracy is the main objective of testing in this menu. The system must make sure that the correct information is displayed once the user clicks on the button. Any textual errors were also corrected in the forms that displayed the information. The link button to the main was also tested to make sure they linked to the main menu. After repeating this testing to the 30 illness labeled command buttons and also the return button. The testing was focused to the remaining module.

- **The Health Tips Menu**

The testing objective of this module is similar to the Disease Reference menu. Both of this menus emphasize on point, click and display function. So the testing was similar to the Disease reference. There are 8 command button that links to the respective windows that will display the information as the user clicks. Test was made to ensure that the correct window was displayed and there were no runtime error such as window stalls or the computer stalls. For the information display, a button was added to able the user to print his form. This function was also tested to make sure it prints properly. All the links were tested from the links leading to the Tips menu and to the next display from. After repeating this

process for the entire buttons. The system testing was completed and was waiting for evaluation.

6.4 SUMMARY

Many types of tests are done before a system can be released to the customer. The objective of the software testing is to detect errors (bugs), debug and generate functional programs. To achieve this objective, three types of testing namely unit, integration and system tests are carried out.

Unit Testing is done to test functional specification of individual components. Integration checks the incorporated components with the interface built. It is the process of verifying that the system components work together as described in the system. Next, system testing validates software once it had been incorporated to a larger system.

During the testing phase of CMHA, many types of errors occurred which causes some minor glitches. The process of debugging was done to eliminate this error.

CHAPTER 7
EVALUATION

CHAPTER 7: EVALUATION

7.1 INTRODUCTION

System evaluation is the final phase in the System Development Life Cycle (SDLC) of this system (CMHA). This chapter will cover the stages of evaluation of the CMHA system. It will also cover the problems faced and their solution as well, system strengths, weakness and the proposed future enhancements. At the end of this chapter a brief summary on all the previous chapter will also be given.

7.2 PROBLEMS ENCOUNTERED AND THE SOLUTION

Various problems were encountered during the development of CMHA. Most of the problems encountered were due to the lack of experience and ignorance on the side of the author himself. The problems were overcome successfully, listed below were the some of the major problems that were encountered during the system building

- **Failure in Learning Visual Prolog**

  Problem: This was the major problem that was a responsibility in the part of the author. Because of this, it made some major changes in the system. The Author did manage to acquire the necessary knowledge needed in Visual Prolog to build the system. Visual Prolog is different from normal text based prolog that the author knows. In Visual Prolog, there must be some knowledge need in working the Integrated Development Environment of the Visual Prolog to create a Visual Prolog based interface. There were not many

books around and the books in the library are books based on text based prolog. The author also could not manage to find any one knowledgeable in teaching him visual prolog, because most of his colleague have graduated and working. Time was running short and it seemed that the system could not be managed to be built in time using visual Prolog language using the author's knowledge.

Solution: Because of these factors, the language was replaced with Visual Basic. The Language was easier to learn and the author could get help from friends because most of them have prior knowledge using this language.

- **Lack of Time**

Problem: Around 8 weeks of project time was wasted on learning Visual Prolog. Time was a crucial factor because the system had to be developed in 4 weeks time to meet the deadline.

Solution: The work on the project had to be doubled in order to complete it on time. Time was sufficiently managed to ensure the work was finishes before the deadline

7.3 SYSTEM EVALUATION BY USER

After the completion of the system, the system was evaluated by the author, and the author's friends along with some family friends. This was done informally, but their

feedback was taken notice and from this session, the useful information was used to find out the system's strength and limitations.

7.4 SYSTEM STRENGTHS

I. Easy to Use

The system is easy to use. From reading the manual, the user will get a picture on how the system works. From a minimal number of usage, the user will learn to navigate the system and use the functions easily. This is because the system utilizes point and click function.

II. Printing Capability

This feature was not in the original design specification because it was originally suppose to be developed in Visual Prolog. Because of the ease of this Visual Basic, a simple command button was added and the code supplied to print the form. Although this capability is only for the Health Tips, in future it will be extended to the other menus.

III. Provides Useful Information

The system provides useful information to the user who accesses this system according to the need specified in the system design and specification. This information is taken from Health related books that are accurate.

IV. Availability, Reliant & Helpful

The system is available all time to the user once it is installed in the home computer. The user does not have to refer to other sources of information through the Internet. The information provided on the topic is reliant and helpful to the user. The diagnosis is consistent and does not change through time.

7.5 SYSTEM LIMITATION

I. Lack Of Multimedia Features

This was the frequent opinion expressed by the users. They all agreed the system was simple and easy to use. The information displayed was comprehensive they commented. But they said, the overall system was lacking in multimedia features. The system could have used images or sounds to spruce up the system. This point was noted and was included for future enhancement.

II. Limited Scope

The diseases that affect children and infant are many. For the practical implementation of this system, only 30 diseases were selected. This was done because initially, the system was to be developed with Visual Prolog, When the language was changed, it seemed much more information on diseases could be added to the system. The time limit factor made this more difficult to implement. The initial objectives had to be fulfilled, and this was the major task to be accomplished.

7.6 FUTURE ENHANCEMENT

I. INCORPORATION OF MULTIMEDIA FEATURES

The system is currently lacking in multimedia features. It is proposed that image pictures are included in the later version of this system. Images in jpeg format that show the visual condition of the symptom will be useful to the user.

II. BETTER DIAGNOSTICS TECHNIQUE

A better diagnostics technique could be implemented to the system. This can be achieved by getting an expert doctor in child illness. The methods used to get a diagnosis can be simulated by finding the pattern the physician use to get the diagnosis.

7.7 KNOWLEDGE AND EXPERIENCED GAINED

Throughout the development of the CMHA, a lot of knowledge and experienced were gained. The fundamental of system development was learnt especially in terms of going through the whole system development life cycle from the analysis phase until the evaluation phase. The software engineering techniques were applied and the theories that had been learned were put into practical use. The Author also learned the importance of proper planning, resource and time management. Furthermore, knowledge on Visual Basic language was also gained during the completion of this project.

After accomplishing this project, a clearer view of the respective jobs of a computer analyst and programmer were obtained. Apart from this, their responsibilities

ranging from the analyzing the system requirements, coordinating project, coming up with solution and other critical measure were gained from completing this project. Overall, this project development benefits in all aspects in preparing the undergraduate to become a system developer in the future.

7.8 SUMMARY OF REPORT

This report was started of with Chapter 1 titled Introduction. In this chapter, the discussion is about the motivation and the objective of the system that will be designed. What is the problem to be solved was explained to the reader in this chapter.

Chapter 2 touches on Literature Review. Literature review is a study that must be undertaken to get information and ideas on concepts relating to the development of the system. Throughout this exposure, the reader will gain a general idea about the nature of the system.

Chapter 3 and chapter 4 are interrelated. They are System Methodology and System Analysis and Design (SAD) respectively. The issues that were discussed what is the method that will be pursued for this system implementation. The SAD chapter complements the other chapter by giving the reader the interface designs of the modules, system data flow and it ends with the expectant results.

Chapter 5 is on System Implementation. This chapter informs the reader, the measures taken to implement what we had specified in the previous chapter. Samples of coding are given so that the reader knows the important codes that are used by the author system. Chapter 6 is about the testing that was used to identify the bugs and other errors

in the system. It explains the need of testing before the system is released to the user. The final chapter on this report is Evaluation. This is the last stage of the SDLC before the system is released to the user.

7.9 CONCLUSION

The Child Medical & Health Advisor (CMHA) had been completed successfully. From chapter 1, the objectives of the system were:

- To help the user to diagnose and identify the disease suffered by the child.
- To give a detailed information on the disease being queried that covers symptoms, home care, precaution and medical treatment to be taken
- Provides health care tips that covers basic child health care knowledge that the parents will find very useful.

From the finished system, the CMHA had achieved its main objectives. Even though the system was not developed with Visual Prolog, the main functions and design that was imagined and idealized by the author had been successfully achieved although by using Visual Basic. It is hoped that the system is able to provide parents and other user the knowledge and information they need to achieve a healthy and happy family.

REFERENCES

REFERENCES

1. Chasnoff,Ira.J.(1991). *Your Child : A Medical Guide* . Illinois, Publishing International Ltd.

2. Bowen, Kenneth A,(1991). *Prolog & Expert System* . New Jersey, McGraw Hill.

3. Covington, Micheal A; Nute, Donald; Vellino, Andre1991. *Prolog Programming in Depth* .Upper Saddle, New Jersey ,Prentice Hall.

4. Lucas, Robert.(1993) . *Application Programming in Quintus Prolog* . Alfred Waller Ltd.

5. Bratko, Ivan. (1993) . *Prolog : Programming for Artificial Intelligence.* Addison-Wesley.

6. Rich, E ; Knight, K.(1993) .*Artificial Intelligence.* McGraw Hill.

7. Durkin, John. (1994) . *Expert Systems : Design & Development* .New York, MacMillan .

8. Fort, Nigel.(1991). *Expert System & Artififcial Intelligence* .Library Association Publishing Ltd, London 1991.

9. Sebesta., Robert W.(1999). *Concepts of Programming Languages* . Reading , Massachusetts, Addison Wesley.

10. Pfleeger, Shari Lawrence.(1998) . *Software Engineering : Theory & Practice".* Prentice Hall.

11. Perry,Greg. (1998). *SAMS Teach Yourself Visual Basic 6 in 21 days.*SAMS Publishing

12. Deitel,H.M. (1999). *Visual Basic 6 how to program*. Prentice Hall.

13. http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/expert/part1/faq.html

14. http://www.bkng.demon.co.uk/

15. http://www.cee.hw.ac.uk/~alison/ai3notes/

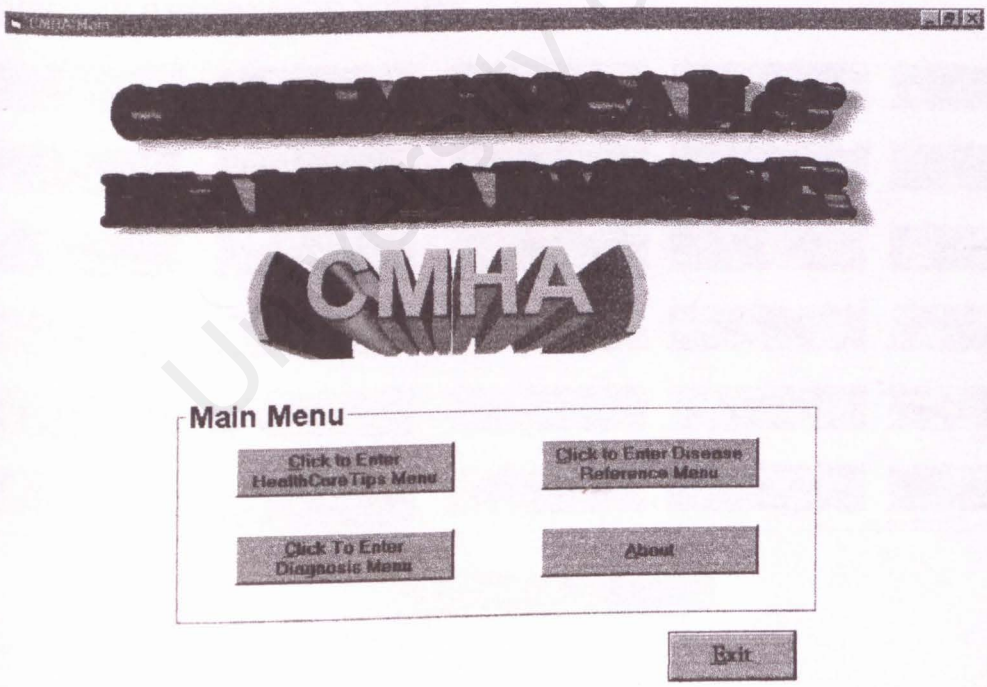
16. http://www.cis.unisa.edu.au/~cisjrw/

17. http://jitta.org/

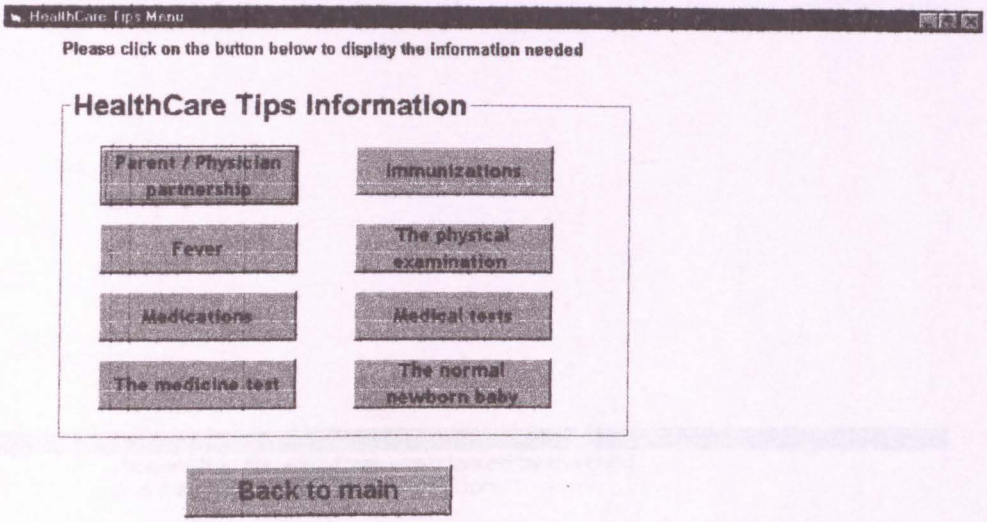
18. http://www.iit.nrc.ca/subjects/Expert.html

APPENDIX A
VISUAL OUTLOOK OF SYSTEM MENUS

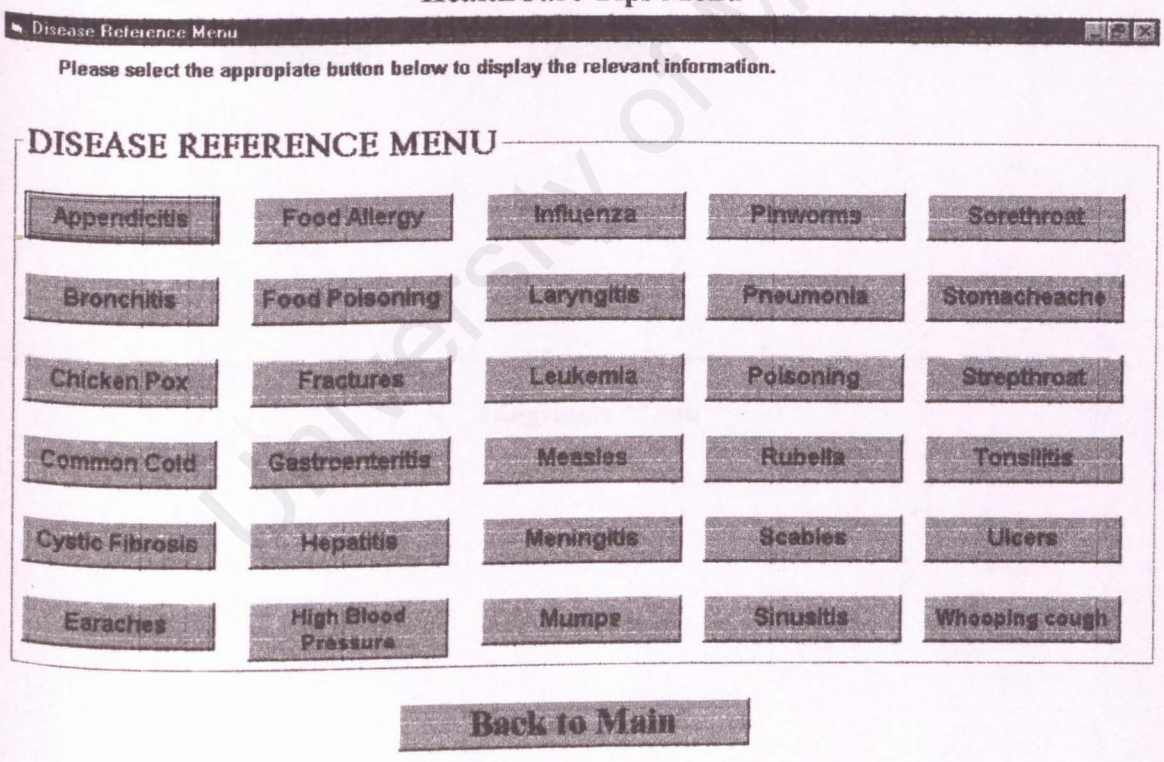
The Welcome To CMHA interface



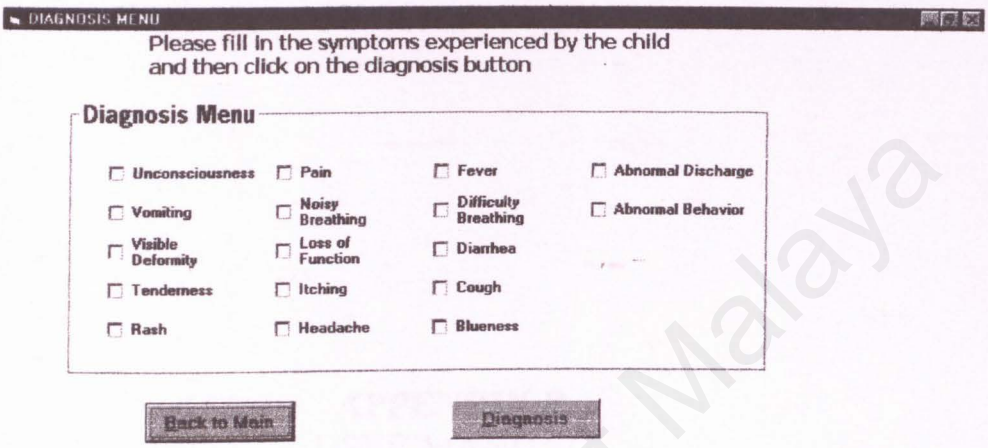
The Main Menu



HealthCare Tips Menu



Disease Reference Menu



Diagnosis Menu

APPENDIX B
USER MANUAL

INTRODUCTION TO CMHA

The Child Medical & Health Advisor was inspired by the need to give information to parents and people who are concerned with the health issues affecting children and infants. This system is intended to be installed in the home computers of user so that is available to the user at anytime. It is to be used whenever the user needs to get the information on a particular disease, to run a quick time diagnosis and also to learn more about basic child health care. The user will encounter 4 main menus when using this system. They are:

1. HealthCare Tips menu

2. Diagnosis Menu

3. Disease Reference Menu

The navigation procedures and usage will be given later in this manual.

ABOUT THIS MANUAL

This manual is a guide to help the user of Child Medical & Health Advisor. This user manual gives clear and step by step instructions. They user just have to follow the instructions to use the system.

SYSTEM MINIMUM REQUIREMENTS

- Windows 95, 98, 2000, ME or XP
- 233Mhz Pentium 2 or a faster Processor
- 64Mb RAM
- 4× CD ROM
- 5MB HardDisk Space
- Keyboard, Mouse

INSTALLING THE SYSTEM

1. Insert the CMHA CD into the CD drive.

2. Double Click on the icon represents the CD drive on the respective Computer

3. Once you have double click on the icon, an icon named CMHA.exe will seen

4. Right button click on it, and send it to the destination the user intends. For example, my documents folder

5. Then go to the My Documents folder and open it.

6. If the CMHA.exe is seen then the installation is complete.

GETTING STARTED: RUNNING THE CMHA

STARTING THE CMHA

1. First, Click open the folder you have installed the CMHA.exe file. In this instance it is in My Documents.

2. Click on the icon CMHA.exe .

3. The figure 1 below will then be displayed

Figure 1:Welcome Interface

4. Click on the Enter button on the left to enter to The Main Menu Interface. Please proceed to the next page for further instructions. Thank you

5. The User can click the Exit button to immediately exit this system.

THE MAIN MENU INTERFACE

1. Once you have clicked the enter button, this interface (figure 2) below will greet the user.

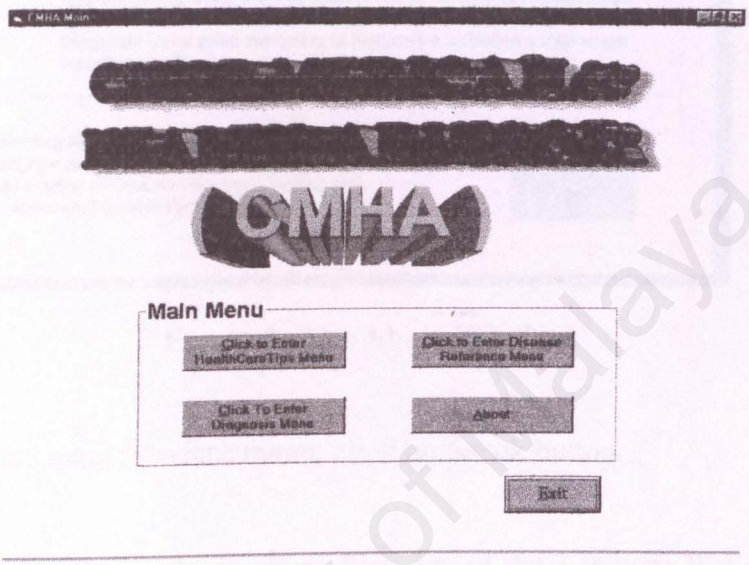


Figure 2: CMHA main interface menu

2. This interface has 5 buttons. 4 are grouped together. They are HealthCare tips button, Disease reference button, the Diagnosis Menu and the About Button. The final button is the Exit button.

3. The about button is an introduction on this system and also information on the version and copyright. When the about button is clicked, the figure 3 on the next page is the window displayed to the user.

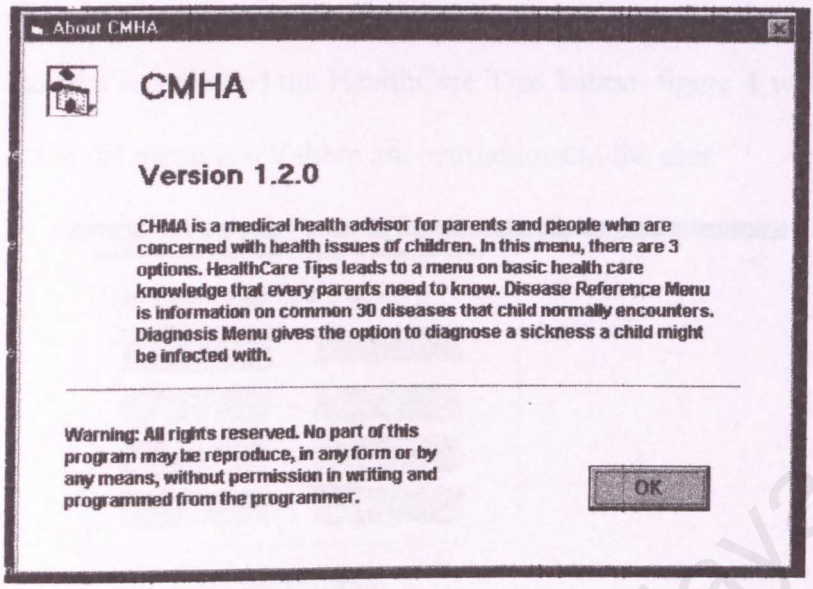


Figure 3: The About Window

4. To return to the main interface menu, click on the ok button.

5. Now the user has the option to chose from any of the 3 buttons that will lead to the respective Menu listed on the button.

6. Please proceed to page 7, if the user has clicked to enter the HealthCare tips. If the user has clicked on the Disease Reference Menu, please go to page 9 to get the guidance on this section. If the user has chosen Diagnosis Menu please proceed to page 11.

7. To exit this system, please click the **"Exit"** button.

THE HEALTHCARE TIPS MENU

1. When the user has clicked the HealthCare Tips button, figure 4 will be displayed to the user. On the menu it self there are instructions to the user

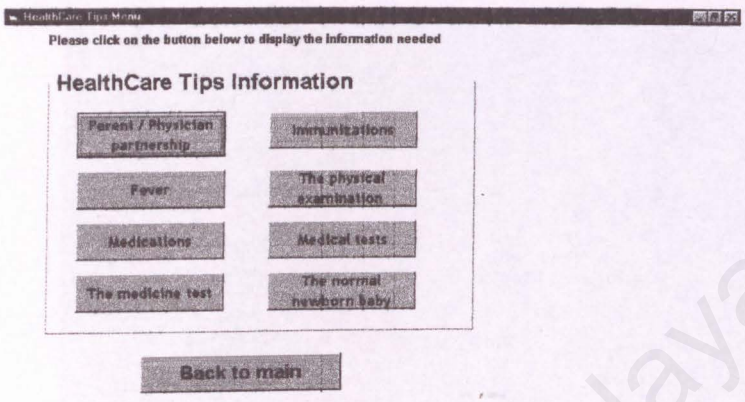


Figure 4:HealthCare Tips Menu

2. The user can choose from the 8 buttons, the information needed. When a button is selected and clicked, the information will be displayed. In this case if the user has selected Fever, the information displayed will be as in figure 5.

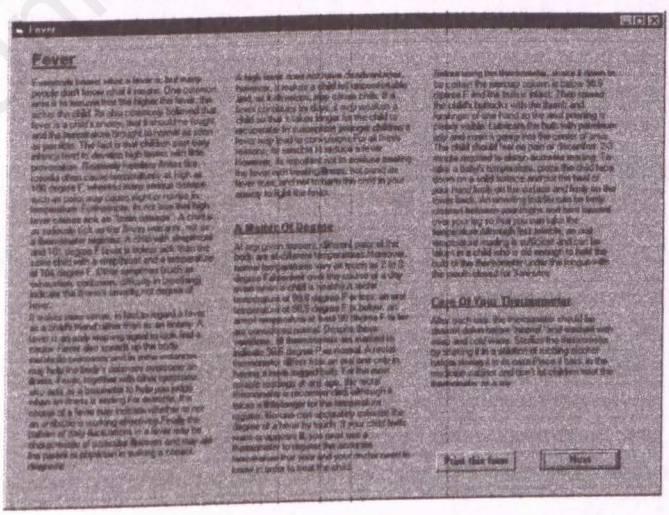


Figure 5: Information Display on Fever

3. The user can read the information they need on this display. The information on these menu are long, because of this they have a continuation to the next display. If the button next is clicked, a second window will be displayed as in figure 6.

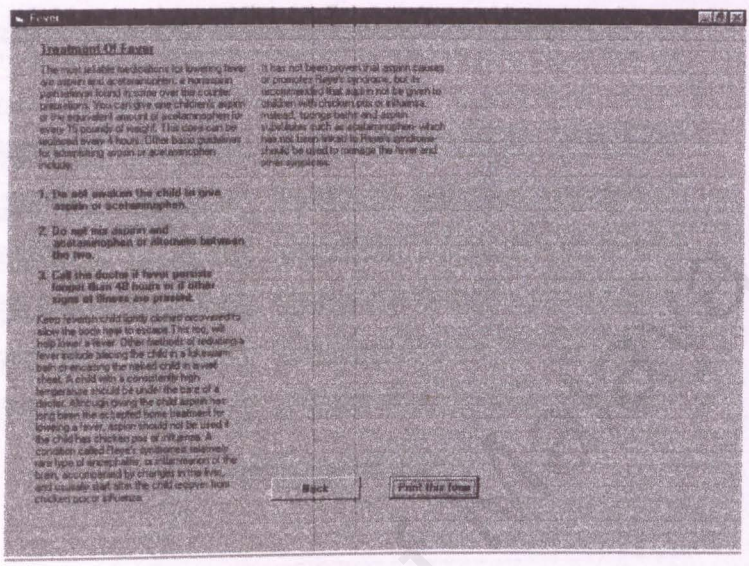


Figure 6: Display on Fever (continued)

4. The user can click on this button to move from one display to the other. The user also has an option to print each form.

5. To print click on the button **" Print this Form".**

6. Once the user has finished reading the information needed. To exit, click on the button on the right hand corner above. Clicking on this will instantly bring the use to the HealthCare Tips Menu (Figure 4).

7. Use the same steps as mentioned before, to access the other tips on this menu.

8. To go back to the main menu, click on the button **"Back to Main"**.

DISEASE REFERENCE MENU

1. When the user clicks the Disease Reference button, figure 7 will be displayed to the ser. On the menu it self there are instructions to the user.

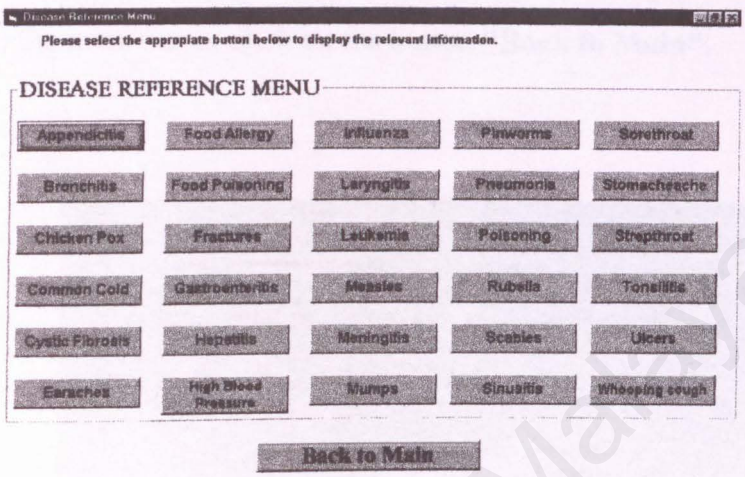


Figure7: Disease Reference Menu

2. On this menu there are 30 buttons. The name of each button is a disease. Clicking on these buttons will display the information window to the user.

3. For example, the action of clicking on the button "Appendicitis" will display this figure 8.

4. The Information on all the Disease in this menu follows the same format. On the top left is name of the disease, Signs **& Symptoms** of this disease, the **Health Care** that should be given, Precaution and the Medical Treatment given by the doctor.

5. To exit from this window click on the on the top most icon on your right marked X.

6. Clicking on this icon will bring the user straight to the Disease Reference Menu refer to figure7.

7. Please use the same procedure described to access the information on each disease.

8. To return to the main menu, click on the button **"Back to Main"**.

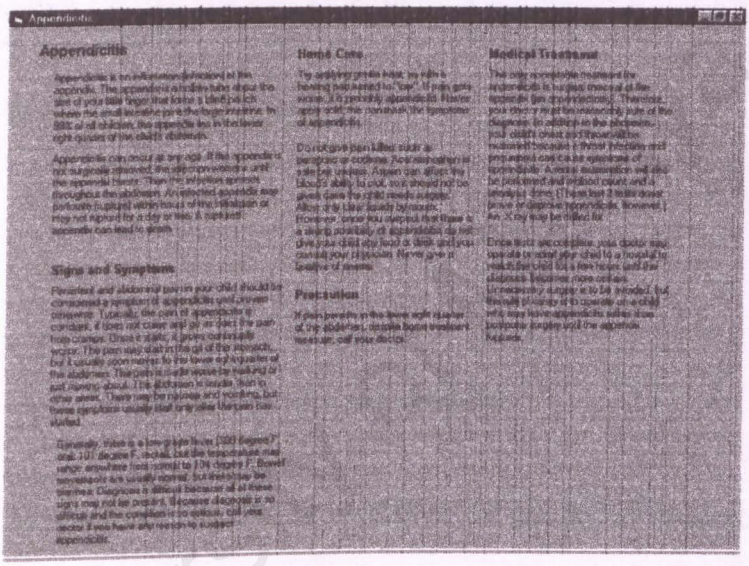


Figure 8:Display Window of "Appendicitis"

DIAGNOSIS MENU

1. When the user clicks the Diagnosis Menu button, figure 9 will be displayed to the user.

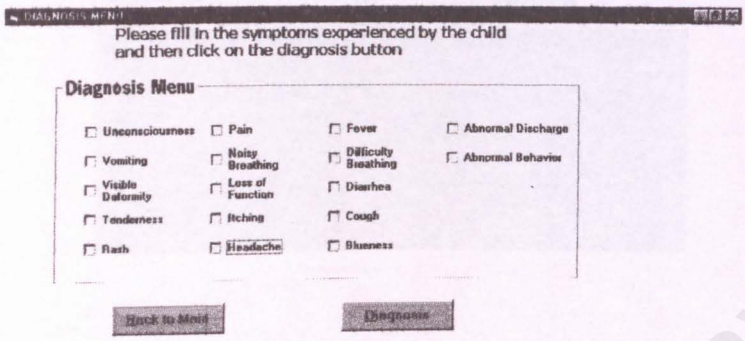


Figure 9: Diagnosis Menu

2. To use this menu, please check in the symptom or symptoms experienced by the affected child or infant.

3. Then click on the button to get your diagnosis. For instance, when you check the symptom **"rash"** and click the diagnosis button a window like figure 10 will be displayed.

4. Please note that the in the figure 10, the smaller window displays the output of the system. The user can resize, hide and exit this form using the buttons on the top right of this form.

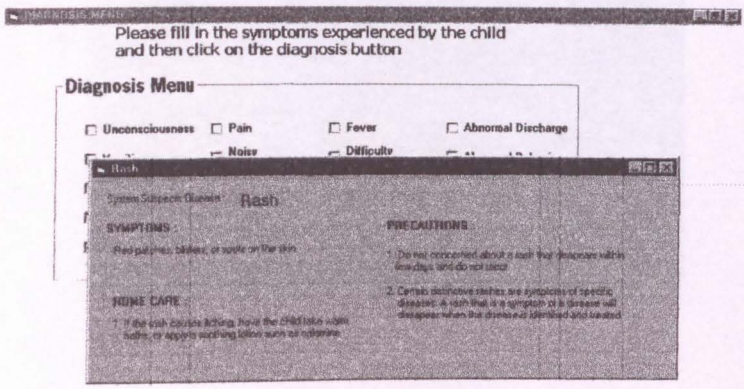


Figure 10: Output of Diagnosis

5. To exit this window, click on the button marked X to return to the Diagnosis Menu.

6. Please uncheck the previous symptom to reuse the diagnosis menu.

7. The user can also check multiple symptoms. For instance, if the user checks the symptoms "Rash" and "Itching" and clicks Diagnosis button. The window for the disease Scabies will be displayed like in the example Figure 11 in the next page.

8. Repeat step 5 to return to the Diagnosis menu

9. Repeat the same procedures to get the diagnosis for other sickness.

10. To return to the main menu, click on the button **"Back to Main"**.

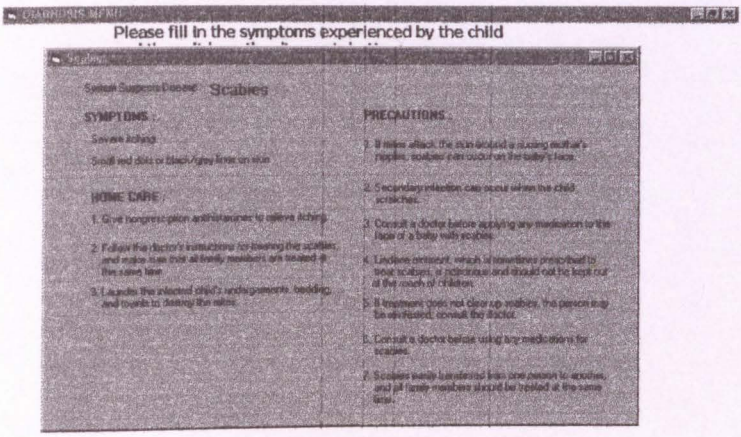


Figure 11:Output for Scabies Diagnosis