CERTIFICATION OF APPROVAL

Automated Visual Basic Application for Zipping and Backup

By

Hema Latha Nantha G.

A project dissertation submitted to the
Information System Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION SYSTEM)

Approved by,

(Mr. Ahmad Izuddin)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

HEMA LATHA NANTHA G.

ACKNOWLEDGEMENT

First and foremost, my highest gratitude and endless thank you to the God above for giving me the people around me at the right time and the right places in doing and completing this project.

I would like to express my sincere thanks to my supervisor Mr Ahmad Izuddin for his guidance, advice and encouragement throughout the length of this project.

I also wish to thank my friends that have helped me by giving me useful ideas, pointing out my mistake and giving me comforting advice during the process.

Never to forget my deepest gratitude and never ending thanks to Mr. Jesudass Thomas for his perseverance and effort in making sure that I have no problem completing this project. I also wish to record my sincere appreciation for his patience and assistance in editing this report..

May all efforts and hard work will be fully appreciated and rewarded accordingly.
Thank You

TABLE OF CONTENTS

ABSTRACT

This study focuses on the visual basic zipping and backup application to improve and develop the way of life to a more convenient style. This approach can be applied to various areas such as different working platform or environment to develop a more secure project. The main target is to design an application to interface between the VBPZip with the visual basic project files working on the Microsoft Visual Basic 6.0 and Microsoft Access 2002 terminal with data compression and backup functionality. The dictionary data compression technique and LZW method used to zip the project files before backups. It will act as a synergy between user and application with the server that complies with the overall application.

The VBPZip application development involves several stages. Defining the methodology, there will be four phrases, which are analyzing, designing, coding and testing. The cores of the project are the data compression and backup functionality in the VBPZip source code, database development and its interface design. Data structure of the application is obtained through research and detailed assessment. Entirely the back end of the application is concerning the source code and its interface. To achieve those with fine results, there are tool required during the whole process.

Thus, the result will be concluded based on the objective set. The application comprises of several form which will act as the interface between the user and database. The form encompass the VBZipping, File Type Options, Auto Zipping, date and time setting and related project files data. All of them have the same purpose which is to ease and create simplicity for the current visual basic platform applied at most development areas. Suggested works for further enhancement and realization are also stated.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

In this technological development era, Visual Basic should be well manageable and effective in term of quality of application. This is to ensure an effective project files management, cost reduction of the maintenances and time saving. Thus, visual basic project files compression (zipping) and backup considered as elemental and very important for the future programmers or developers. In general, this application will have an end program with a function of linking the VBPZip with the Visual Basic project files that complies with the Visual Basic 6.0 and Microsoft Access (Figure 1.1). Apart from crucial secure backup, the VBPZip should guarantee the integrity of all the updated files or codes.

The three main components in this project would be the VBZipping, File Type Options, and Auto VBBackup Setting. The area of study is more towards data compressing (zipping) and backup coding in the Visual Basic environment. Besides human computer interaction (HCI) understanding, author must also get used to graphical user interface (GUI) designing method. These subjects will have main function as an intermediary between human and machine. However, the foundation of this project would be data compressing algorithms or method and visual basic programming. Author has to familiarize with advanced programming language besides disseminating perceptive on the particular visual basic application architecture. Technically, author will encounter diverse new scope of study associated with technology knowledge.

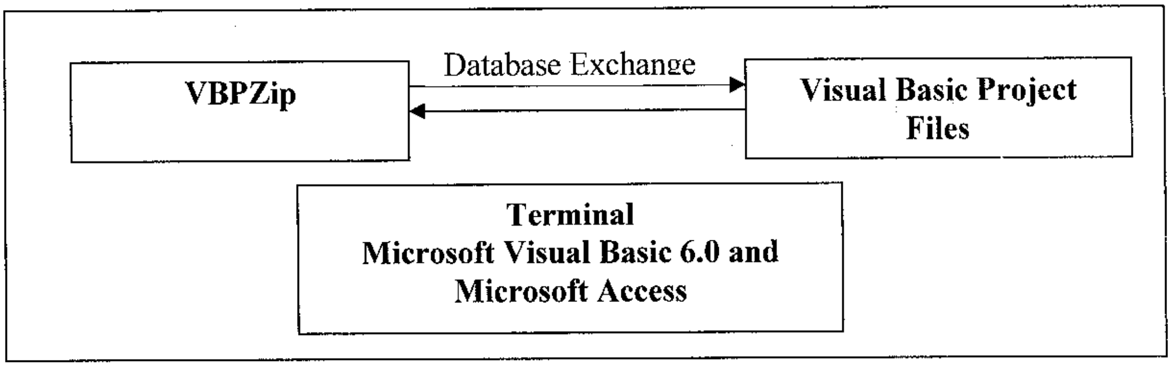

Database Exchange
VBPZip
Visual Basic Project Files
Terminal
Microsoft Visual Basic 6.0 and Microsoft Access

Figure 1.1 Theoretical Framework of VBPZip

1.2 Problem Statement

Working on major applications would need the user to save all their Visual Basic projects files and references (.DLL/OCX). By right, information storage might encounter problem besides demanding manpower during peak hours. Over the study, author encounter that system developers and programmers face similar problem as losing many project files while upgrading codes or overwriting files by mistake. Thus, author sees that the introduction of VBPZip would provide a proper solution for efficient work ethic. This is a part of technique to achieve effective file management, efficient workforce and organized system.

1.2.1. Problem Identification

There are several problems identified at the Visual Basic environment which necessitate this project to be accomplished:

a) Overwriting visual basic project files by mistake.

b) Lost of many project files while upgrading codes.

c) Large project files capacity leads to poor file management and limited storage space.

1.2.2. Significance of the Project

The significance of this project to overcome the problem is to act exactly as an intermediary between four parties, which are VBPZip, Visual Basic Project Files, Microsoft Visual Basic 6.0 and Microsoft Access. Existing visual basic environment issues are the elements in prompt to develop this application. It will definitely create an effective confidential communication environment. This project also reliable enough for further enhancement as it provide a suitable interface and stable database system. If this project is a complete success, it can be applied in real industry, as long as it is stable and secured enough.

1.3 Objective and Scope of Study

1.3.1. The relevance of the Project

This project need less theory on past course to be practiced, which mean author has the opportunity to go deep into technical knowledge. Yet, author must familiarize with the particular visual basic application, for instance data compressing algorithm (zipping) and methods. Upon completing this project, there will be only one approach involved, which is delivered in one semester:

a) Final semester – Design and develop preliminary interface layout and exact information to be in the application, and prepare tools for Visual Basic programming. Develop and debug the prototype, its database and terminal Microsoft Visual Basic 6.0 and Microsoft Access for successful implementation.

For a long term plan, the main objective of this project is to compress (zip) and backup the existing visual basic project files. That is to create a smart, effective, productive and efficient environment for the subject. However, it has to be realized partially, by applying a test run the VBPZip and particularly a single project file. Author had successfully achieved the entire respective objective during the semester. Thus, to realize VBPZip objectives are:

a) To gather information and perform case study on several compressing method (zipping) to determine the best approach.

b) To compress the visual basic project files in a zip format.

c) Allow maximum project zipping and backup options for a single file, backup folders, and backup file types and restore backup.

d) To complete the source code development for the VBPZip.

e) To design the data structure and interface coordination appropriately.

f) To design VBPZip that is able to store data.

1.3.2. Feasibility of the Project within Scope and Time Frame

The application is VBPZip. Upon completing the whole prototype, the objectives are vital. The scheduled tasks and milestones for the final year project are summarized in the Gantt chart [Appendix A1]. It covers the entire schedule from doing the preliminary design and constructing data structure, coding, testing, and also report writing. The scope of project seems to be feasible for author to complete on time with the possible outcomes, and the time allocated will be used efficiently for developing the whole end product.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2. LITERATURE REVIEW

2.1 Introduction

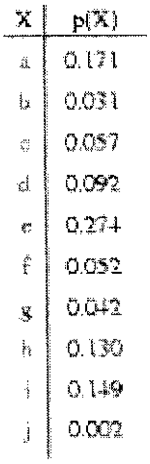
Any programmer working on mini or microcomputers in this day and age should have at least some exposure to the concept of data compression. In MS-DOS world, programs like ARC, by System Enhancement Associates, and PKZIP, by Pkware are ubiquitous. ARC has also been ported to quite a few other machines, running UNIX, CP/M, and so on. CP/M users have long had SQ and USQ to squeeze and expand programs. Unix users have the COMPRESS and COMPACT utilities. Yet the data compression techniques used in these programs typically only show up in two places: file transfers over phone lines, and archival storage. Deciding on what is good or not is very subjective. It depends on the user very own perspective of applying the subject. Compressing and decompressing methods can vary from different individual perception. Technologically, this application can as well be considered as evolution to the Visual Basic environment.

2.2 Statistical Techniques

Statistical compression method use a statistical models of the data, so the quality of compression they depends on how good that model is. The most basic solutions for compressing files revolve around computing how often particular patterns can be found in the file and then replacing the most common patterns with shorter patterns and replacing the least common patterns with longer ones. In the end, the tradeoff is worth it. These solutions are usually described as applying to letters or bytes. The statistics can be gathered for any collection of characters, bytes, tokens, or patterns of bits. The first approach is commonly known as "Huffman encoding" and named after David Huffman **[Huf52].**

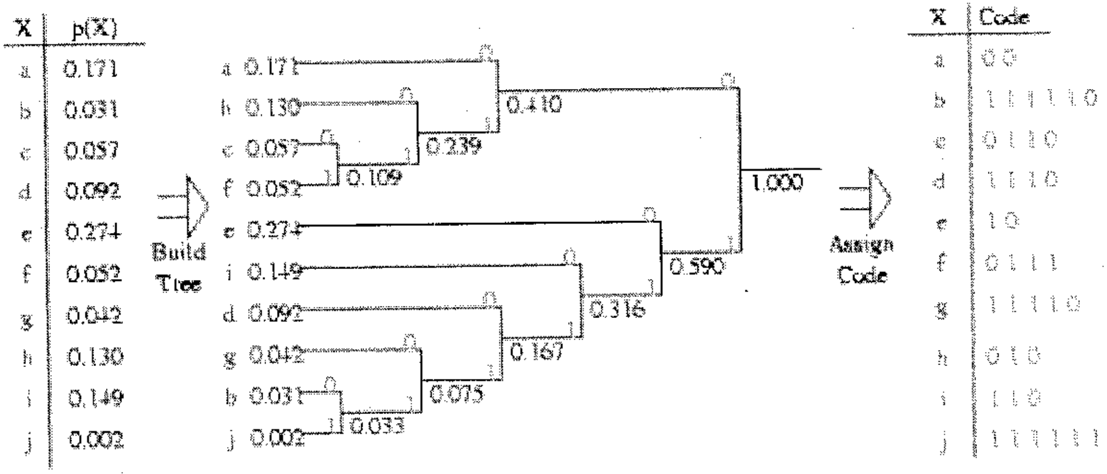
It provides a simple way of producing asset of entire placement bits. The algorithm is easy to describe, simple to code, and comes with a proof that it is optimal, at least in some sense. The algorithm finds the strings of bits for each letter by creating a tree and using the tree to read off the codes. The common letters ends up near the top of the tree, while the list common letters end up near the bottom. The paths from the root of the tree to the node containing the character are used to compute the bit patterns.**[Fin85]** The biggest problem is that both the compression and decompression routines must have access to the same list or tokens or bits. **[Muk89].** The basic idea in Huffman coding is to assign short codeword to those input blocks with high probabilities and long codeword to those with low probabilities. This concept is similar to that of the Morse code.

A Huffman code is designed by merging together the two *least probable* characters, and repeating this process until there is only one character remaining. A code tree is thus generated and the Huffman code is obtained from the labeling of the code tree. An example of how this is done is shown below.



X	p(X)
a	0.171
b	0.031
c	0.057
d	0.092
e	0.274
f	0.052
g	0.042
h	0.130
i	0.149
j	0.002

The final static code tree is given below:



X	p(X)
a	0.171
b	0.031
c	0.057
d	0.092
e	0.274
f	0.052
g	0.042
h	0.130
i	0.149
j	0.002

X	Code
a	00
b	111110
c	0110
d	1110
e	10
f	0111
g	11110
h	010
i	110
j	111111

- It does not matter how the characters are arranged. I have arranged it above so that the final code tree looks nice and neat.
- It does not matter how the final code tree are labeled (with 0s and 1s). I chose to label the upper branches with 0s and the lower branches with 1s.
- There may be cases where there is a *tie* for the two least probable characters. In such cases, any tie-breaking procedure is acceptable.
- Huffman codes are not unique.
- Huffman codes are optimal in the sense that no other lossless fixed-to-variable length code has a lower average rate.
- The rate of the above code is 2.94 bits/character.
- The entropy lower bound is 2.88 bits/character.

2.3 Dictionary Techniques

Dictionary based compression methods do not use statistical model, nor do they use variable codes. Instead they select strings of symbols and encode each string as a *token* using a dictionary. **[Mil89, Weg89]**

Dictionary techniques utilize groups of symbols, words, and phrases with the corresponding abbreviations; the abbreviations are used during data transmission and the receiver translates them back to the original form using the same dictionary as the sender. The dictionaries can be static or dynamic. A *static dictionary* remains the same during the course of encoding the data. This can be as simple as utilizing commonly used acronyms (CEO, UN, ASCII) abbreviations (dr., Penna, ca), or coined abbreviations (M—male, 192 – acidfree paper, #$-- load instruction). More systematically an explicit dictionary is created that list frequently used symbol patters and their codewords. One such dictionary is not applied with equal efficient in all situations. **[Snow86]**

One of the most common solutions for compressing data is to create a list of the most common words or phases in a document and then replace each of these blocks of letters with a short number representing the position in the dictionary. This can be quite powerful if the dictionary is the right size and there are plenty of common patterns in the file. **[Mol83]**

First, a dictionary should be language specific. One technique uses n-grams—that is, frequently used pattern of n consecutive symbols. If three-grams (trigams) are used, then in an English text such *trigams as the, que, ome, or neu* can be found much more often than *sch, hin, or wer,* but in a German text it would be the other way around.

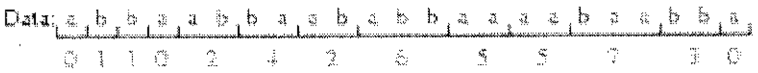
Second, dictionaries should be domain specific. If commonly used words are to be encoded in few bits, then in a dictionary pertaining to gardening texts and databases such as a *rake, soil, foliage, or parsley* would be found more often than the words *hardbound, Press, index, or pages,* which in turn would have a very high frequency of occurrence in the library setting.

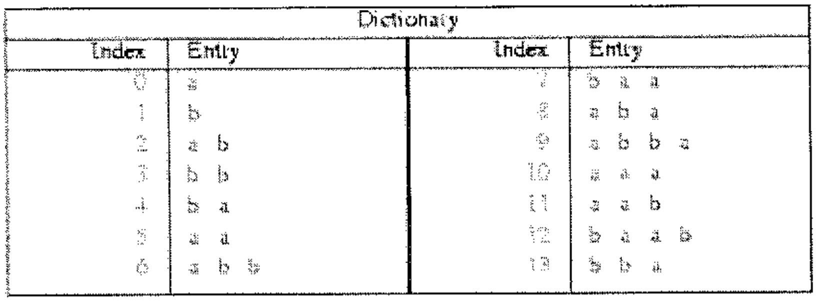
The best-known version of these dictionary schemes, which are sometimes called substitution compression, is called "Lempel-Ziv" after Jakob Ziv and Abraham Lempel. The algorithm, first described in 1977 **[ZL77],** is used in many different compression programs in many different versions. Some of the later versions have been patented, and the patents have been the basis for some bitter battles over rights and ownership. The Lempel-Ziv algorithm is a variable-to-fixed length code. Basically, there are two versions of the algorithm presented in the literature: the theoretical version and the practical version. Theoretically, both versions perform essentially the same. However, the proof of the asymptotic optimality of the theoretical version is easier. In practice, the practical version is easier to implement and is slightly more efficient. We explain the practical version of the algorithm as explained in the book by Gersho and Gray and in the paper by Welch. The basic idea is to parse the input sequence into non-overlapping blocks of different lengths while constructing a dictionary of blocks seen thus far.

Encoding Algorithm

- Initialize the dictionary to contain all blocks of length one (D={a,b}).
- Search for the longest block W which has appeared in the dictionary.
- Encode W by its index in the dictionary.
- Add W followed by the first symbol of the next block to the dictionary.
- Go to Step 2.

The following example illustrates how the encoding is performed.





Dictionary			
Index	Entry	Index	Entry
0	a	7	b a a
1	b	8	a b a
2	a b	9	a b b a
3	b b	10	a a a
4	b a	11	a a b
5	a a	12	b a a b
6	a b b	13	b b a

- Theoretically, the size of the dictionary can grow infinitely large.
- In practice, the dictionary size is limited. Once the limit is reached, no more entries are added. Welch had recommended a dictionary of size 4096. This corresponds to 12 bits per index.

- The length of the index may vary. For example, in the *n*-th block, the current dictionary size is $n+1$ (assuming that the limit has not been reached). Therefore, we can encode the index of block *n* using $[log_2(n+1)]$ bits (rounded up to the nearest integer). For example, the first index can be represented by 1 bit, the 2nd and 3rd by 2 bits each, the 4th, 5th, 6th, and 7th by 3 bits each, and so on. This is the variable-to-variable length version of the Lempel-Ziv algorithm.
- For a maximum dictionary size of 2^m, variable-length encoding of the indices saves exactly 2^{m-1} bits compared to fixed-length encoding.
- The above example, as most other illustrative examples in the literature, does not result in real compression. Actually, more bits are used to represent the indices than the original data. This is because the length of the input data in the example is too short.
- In practice, the Lempel-Ziv algorithm works well (lead to actual compression) only when the input data is sufficiently large and there are sufficient redundancy in the data.
- Decompression works in the reverse fashion. The decoder knows that the *last* symbol of the most recent dictionary entry is the *first* symbol of the next parse block. This knowledge is used to resolve possible conflict in the decoder.
- Many popular programs (e.g. Unix compress and uncompress, gzip and gunzip, and Windows WinZip) are based on the Lempel-Ziv algorithm.

2.4 Basic Lemper-Ziv-Welch

Many program use a version of Lemper –Ziv created by Terry Welch in 1984. This version is pretty simple and easy to code, so it is frequently included in many simple compression schemes. It is also relatively good, although it takes its time building up the dictionary. **[Hen93]**

From author point of view, the application of VBPZip now is becoming a necessity for the developers and programmers with the increasing project. People are now living in a fast pace and demanding high lifestyle. People will insist for a firm and quick response in every aspect of life. This is where the VBPZip application provides solution with guaranteed security.

2.5 Conclusion

A Huffman encoder takes a block of input characters with *fixed* length and produces a block of output bits of *variable* length. It is a fixed-to-variable length code. Lempel-Ziv, on the other hand, is a variable-to-fixed length code. The design of the Huffman code is optimal (for a fixed block length) assuming that the source statistics are known a priori. The Lempel-Ziv code is not designed for any particular source but for a large class of sources. Surprisingly, for any fixed stationary and ergodic source, the Lempel-Ziv algorithm performs just as well as if it was designed for that source. Mainly for this reason, the Lempel-Ziv code is the most widely used technique for lossless file compression. Data compression has an undeserved reputation for being difficult to master, hard to implement, and tough to maintain.

In fact, the techniques used in the previously mentioned programs are relatively simple, and can be implemented with standard utilities taking only a few lines of code. This article discusses a good all-purpose data compression technique: Lempel-Ziv-Welch, or LZW compression. The routines shown here belong in any programmer's toolbox. For example, a program that has a few dozen help screens could easily chop 50K bytes off by compressing the screens. Or 500K bytes of software could be distributed to end users on a single 360K byte floppy disk. Highly redundant database files can be compressed down to 10% of their original size. Once the tools are available, the applications for compression will show up on a regular basis. In this studies author will be using the Lempel-Ziv-Welch method.

CHAPTER 3
METHODOLOGY

3. METHODOLOGY

The process of completing this project involves several components and distinct stages. Mainly, the 3 parts of the system are the VBPZip, Visual Basic Project files and Microsoft Visual Basic 6.0 and Microsoft Access.

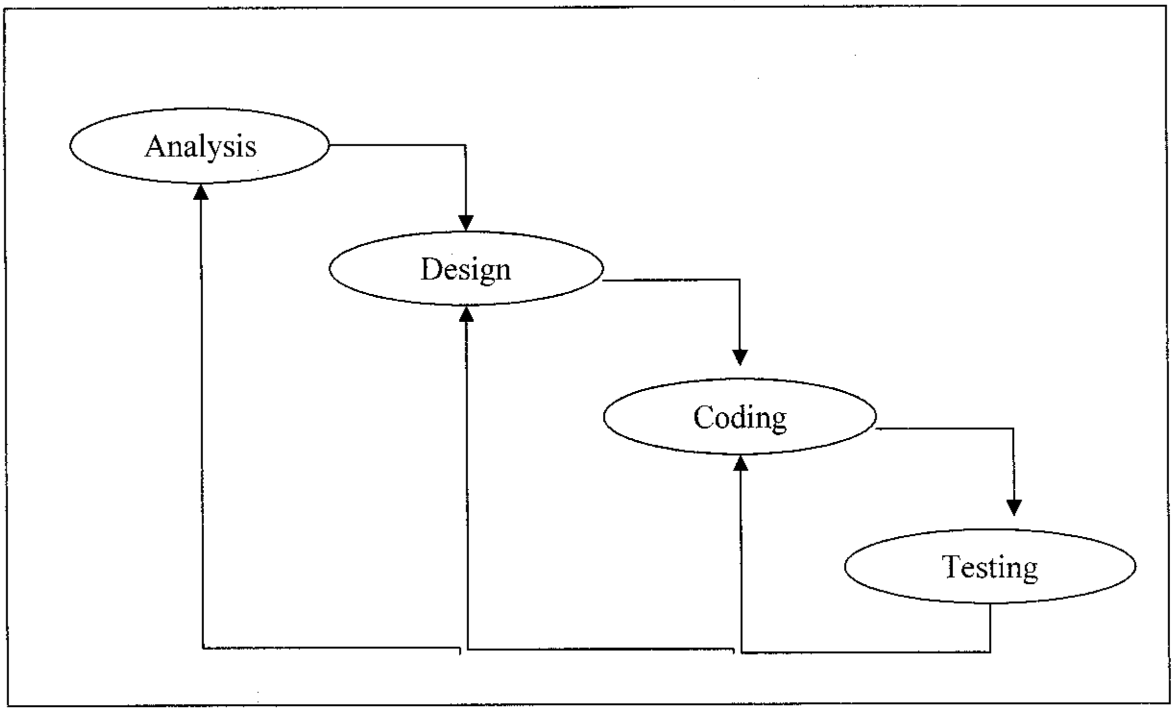


Figure 3.1 Software Engineering Waterfall Model

Applying the software engineering knowledge, the classic waterfall model is applied to design the system. Three stages are referred to as system development life cycle (SDLC). This methodology was chosen because it provides systematic approach in solving any encountered problems. Cycle in SDLC refers to the natural tendency for the system to cycle through these activities, which can be viewed in Figure 3.1

3.1 Procedure Identification

3.1.1. Analysis

This phase is to study the problems, causes and effects. Next, there will be problem identification and list of requirements in providing the solution. These are two elements in the preliminary analysis, which are the system comprehensive data and Visual Basic Application. These two provide ample information and details proceeding to the next designing stage.

a) System Comprehensive Data

Several methods of gathering useful data are used throughout this preliminary period. There had been interview, literature reviews, internet research, journals and books to clarify the problem statement. The data compiled will be used for creating comprehensive coverage of VBPZip database and interface of the system.

b) Visual Basic Application

This is done by performing literature reviews and doing research on the internet. The information obtained is not directly involves with the project, but it provides more understanding of the visual basic interior system.

For this project, visual basic platform and visual basic programming is used. The latest techniques and method used to determine the best compression for the files zipping, the sub classing instances and support files download to the application, and supports database.

3.1.2. Designing

The designing of system layout and GUI for the application needs a good HCl knowledge. Therefore, during this stage, the three main factor of HCI are the study of:

a) Anthropology – science of the origin and development of human.
b) Psychology – science of the mind and human nature.
c) Sociology – science of organization of human society.

This knowledge of system analysis is applied. At this step, the designing started with Unified Modelling Lauguage(UML) diagram. Object- oriented analysis was performed using UML. Object-oriented design is a design strategy where system designers think in terms of 'things' instead of operations or functions.

The execution application is made up interesting objects that maintain their own local state and provide operations on the state information. An object oriented design process involves designing the object classes and the relationships between these classes. When the design is realized as an executing program, the required objects are created dynamically using the class definitions. UML is an integration of the different notations for describing object-oriented designs. There are several diagrams produced using UML.

3.1.3. Coding

Upon the completion of analysis and designing stages, there is the coding of the whole source code for the system. The main source codes to be developed are for the VBPZip programs to support the Visual Basic project files in the Microsoft Visual Basic 6.0 application and integrate with the Microsoft Access database support. This interface created using Microsoft Visual Basic 6.0 terminal. This stage consumes most of the time because the main constraint is to learn in detail of the programming language. Coding stage also will mostly necessitate author to be well literate in computer programming software.

During this stage, the process of developing the source code for this particular VBPZip program is important. There are stages and exact practices to be preformed such as its requirements, library path setting and sub classing instances.

a. Method to be included

An applet must include the installation, selection and processing methods. All these methods were to be declared in the applet's primary class. The purpose of installation method is to create the class object needed by the applet. The select method is to prepare the applet to receive and process the commands.

b. Library Path

To ensure source code can be debugged, the host system's environment variable path must be set to the class library directory and also to the Visual Basic class files in author's development environment. In this case, the path to the class library in Microsoft Visual 6.0 must be set properly.

c. Instructions

There are several orders to make the program as efficient as possible to minimize the requirements and also to reduce the instances needs. Each instantiation of an object or array declared with a local scope allocated memory. Each call to the local method allocates memory until all the Visual Basic resources are exhausted. For variables, the static and final modifiers are used in the declaration statements so that the variables act as constants. This practice minimizes the instance's size through compression and improves performance. Each new variable introduced consumes additional visual basic resources. So, it is better to reuse variables.

Local variables also should not be used frequently since they occupy space in the Visual Basic platform. Class hierarchy also must be as simple as it can since calling classes within classes takes up program space. It can be done by using the class hierarchies in the prototype program, and the hierarchy is compressed before converting the code to an applet. Arguments also were to be used as few as possible. Reviewing of source code is also important since it helps a lot in editing process. Unused variable definitions and operations must be removed. In terms of data types, short is more preferred rather than bytes whenever possible because short consumes the same amount of memory as bytes do but offer a wider range of values.

d. *SSubTmr* Implementation

SSubTmr implementation is to produce a stable sub classer technique which can used regardless of how many instances created and how many controls want to attempt to subclass the same ***hWnd***. The implementation is very similar to the Sub Timer component from Hardcore Visual Basic, but with some improvements.

The sub classing consists of three parts:

- A bas module (***subclass.bas***) which contains the logic to associate object pointers with window handles, and manage the adding and removing of the sub classed proc.
- An Interface, (***ISubClass.bas)***, defining the properties and methods which an object must support in order to take part in sub classing with subclass.bas. To ensure all these interfaces are present, subclass.bas will not accept any object which doesn't implement this interface.
- A global class (***GSubClass.Cls***) which exposes functions to initiate and control the sub classing procedure.

Here are the window properties ***SSUBTMR*** uses to manage the sub classing process:

C[hWnd]

Store the number of instances using the subclass. When the property is 0 and you add to the subclass, it installs the WindowProc. Subsequent additions just increment the counter and use the existing WindowProc. When items are removed, the count is decremented, until it gets to zero, when the WindowProc is removed again.

[hWnd]#[Message]C

The count of how many times the message [Message] is attached to the hWnd [hWnd]. This allows subclass the same message on the same hWnd more than once. For example, if a control wants to subclass its parent, and place two controls on the same parent, need both controls to be able to receive that message.

[hWnd]#[Message]#[Number]

Store a pointer to the object which wants to receive notification for the hWnd or Message combination.

e. Adding Files to a Zip

Adding files to the zip is used using the ***AddFileSpec*** method. A file specification can either be a fully specified path name (e.g. C:\Stevemac\HTML\ssite\index.html), a wildcard specification (e.g. C:\Stevemac\HTML\ssite\*.htm?) or a relative path (e.g. index.html). Relative paths are taken relative to the ***BasePath*** property of the zip

AddFileSpec adds additional file specifications to be used during operations. To clear the buffer, use ***ClearFileSpecs.*** To modify existing file specifications, ***FileSpecCount*** can be used to return how many specifications have been set up and the property ***FileSpec*** to read or write the property.

f. Progress and Cancel

As Zipping operations are performed, the class will raise the ***Progress*** event, which will display status messages about the directory operation, and ***Cancel*** an event, which allows stopping the directory or unzipping operation.

g. Options

The most important zip options are ***RecurseSubDirs*** and ***StoreFolderNames***. When set ***RecurseSubDirs***, the zip DLL will check for all files starting at the ***BasePath*** property and below for matches against each of the ***FileSpecs.*** So, for example, if add "*.*" as a specification, all files in the ***BasePath*** folder and below will be added to the zip. The ***StoreFolderNames*** option determines whether the zip will include the folder names as well as the file names. The default operation of this option is to store the full folder name, however, if ***FileSpec*** items are relative paths at or below the ***BasePath*** then the zip will include the relative path name instead

h. Deleting Files

Deleting files from a Zip is accomplished in exactly the same way as zipping, except to use the ***Delete*** method instead of the Zip method.

i. Two types of incremental backups

Differential incremental backup

- Archives all files changed since the last incremental or complete backup. To restore files, restore all archives: at first the last complete backup, and then all incremental backups.

Cumulative incremental backup

- Archives all files changed since the last complete (or full) backup. To restore files restore two archives: at first the last complete backup and then the most recent incremental backup.

Daily Backup uses differential incremental backups, but speeds up the restoration by reverse retrieval:

It restores at first the most recent incremental backup, and continues till the last complete backup. It stores the retrieved files in a list, and does not retrieve them again. This trick speeds up the restoration. The normal way to restore with an incremental backup scheme is to restore at first the last complete backup, and then all incremental backups. In this way a file might be overwritten many times.

Decompressing algorithm recognize that table is growing larger and allocate more space to each code. The decompression loop is summarized with this pseudo code:

```
read a character x from compressed file;
write x to uncompressed version;
    word=x;
while not end of compressed file do begin
    read x
    look up dictionary element corresponding to x;
    output element
    add w + first char of element to the dictionary
    w = dictionary element
endwhile
```

The two algorithms maintain the same version of the dictionary without ever transmitting it in a separate file.

3.1.4. Testing

Testing would be the final stage of the whole process of this project. Basically testing is a process of after compiling and debugging the source code. There are some syntax errors. This is an evaluation period of the project development where run time error testing will be executed to manipulate real time situation.

3.1.5. Data Collection

To ensure the project can adapt with real time purpose, the updated data and path from the original project files are required. Therefore, the feedback with several users is vital. This is performed in normal procedure where the author had to flow all formality, protocols and established contact with all significant personnel. Several developers and programmers were selected to obtain the data required.

3.2 Tools Required

This project should have a reliable prototype and it does not cost high budget. Since it is visual basic application project, therefore it requires some additional hardware. For the next coding and testing phases, Microsoft Visual Basic 6.0 and Microsoft Access are used as the platform to build the system.

3.2.1 Hardware Used

a) Processor – Pentium3 or higher

b) RAM – 64 Megabytes or higher

c) Disk space – 1 Gigabytes or higher

d) Monitor – 14 inch with 1024 x 768 resolution

e) Graphic card – any standard

f) Mouse/ Keyboard – any standard

3.2.2 Software Used

a) Operating System – Windows 98 or higher

b) Microsoft Visual Basic 6.0

c) Microsoft Access 2002

CHAPTER 4
RESULT AND DISCUSSIONS

4. RESULT AND DISCUSSIONS

This is the most critical part in the project. Research output is presented in this section. The results must meet the requirement of the target audience to fulfill their expectation of a system. Methodology applied within the timeframe of this project should get accurate result. Findings and results obtained during this coding stage.

4.1 Finding

The output for the end product is the system requirement functionalities. It mainly engaged with the interface, data structure and database tables.

4.1.1. Interface and Data Structure

The interface is designed and developed by using Microsoft Visual Basic 6.0. There are 2 main components in the interface design which author declares them as application menu and data modules.

a) Application Menu

The application starts with a main menu which will lead to three other sub menus, which are VBZipping, File Type Option, AutoZip. Each of them has different functionality and interface design.

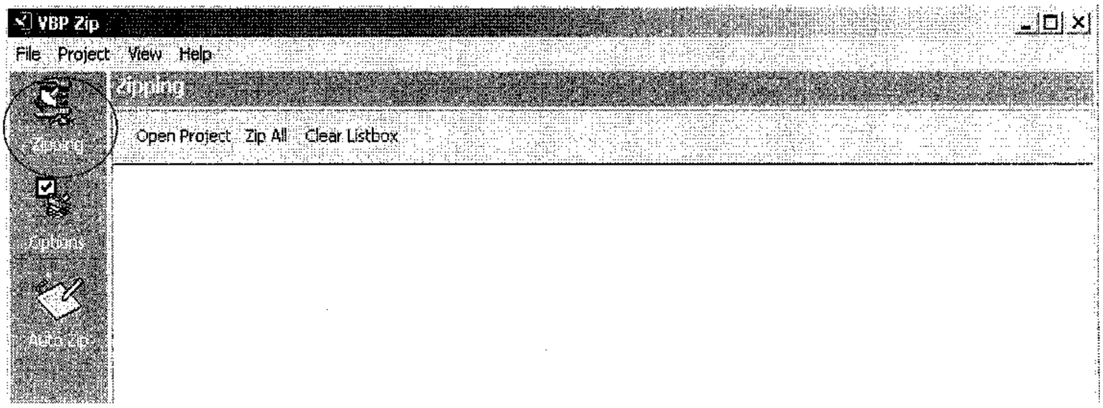
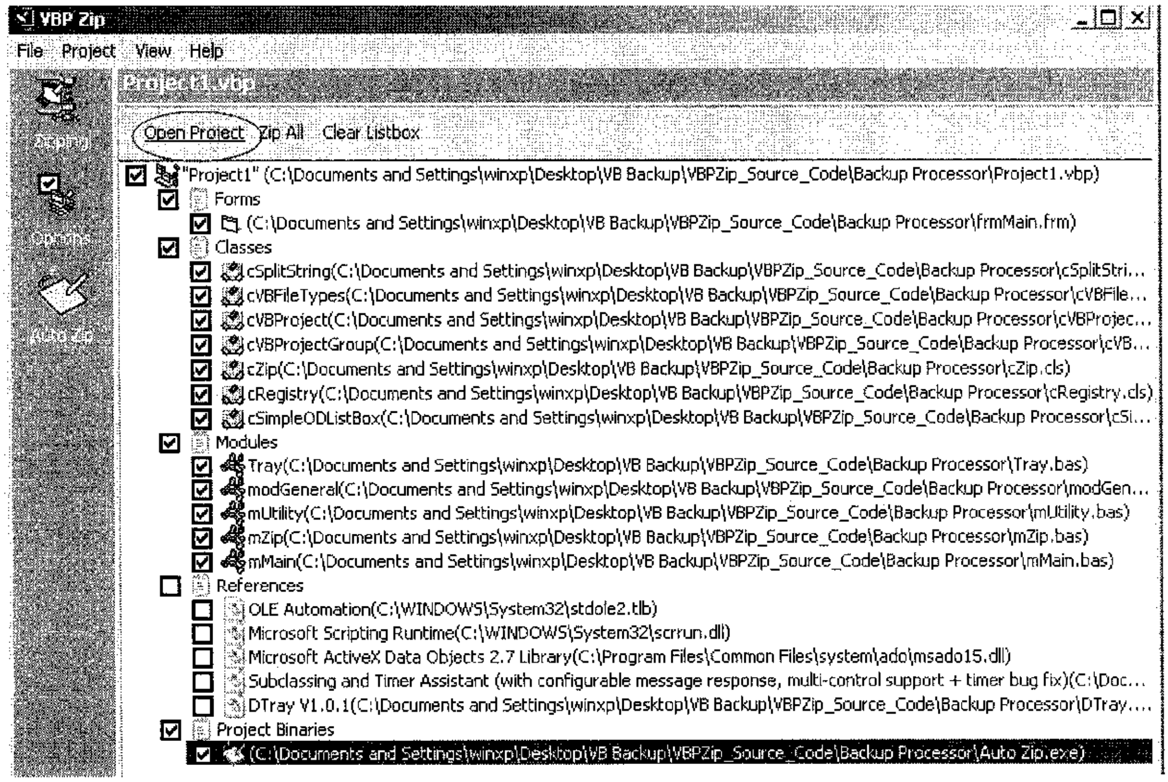


Figure 4.1 VBPZip Application Main Menu

The menu also contains the connectivity to the visual basic project files and library path setting. There are also window taskbar features with open file, zip project, view full mode, status bar, zip page and help on the vbaccelerator on the web with the F1. Other addition features with the clear list box to close the open project files. The source code is vital in associating the VBPZip, VB project files and the Microsoft Visual Basic 6.0 and Microsoft Access Platform.



```
☑ "Project1" (C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\Project1.vbp)
   ☑ Forms
      ☑ (C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\frmMain.frm)
   ☑ Classes
      ☑ cSplitString(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cSplitStri...
      ☑ cVBFileTypes(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cVBFile...
      ☑ cVBProject(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cVBProjec...
      ☑ cVBProjectGroup(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cVB...
      ☑ cZip(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cZip.cls)
      ☑ cRegistry(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cRegistry.cls)
      ☑ cSimpleODListBox(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\cSi...
   ☑ Modules
      ☑ Tray(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\Tray.bas)
      ☑ modGeneral(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\modGen...
      ☑ mUtility(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\mUtility.bas)
      ☑ mZip(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\mZip.bas)
      ☑ mMain(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\mMain.bas)
   □ References
      □ OLE Automation(C:\WINDOWS\System32\stdole2.tlb)
      □ Microsoft Scripting Runtime(C:\WINDOWS\System32\scrrun.dll)
      □ Microsoft ActiveX Data Objects 2.7 Library(C:\Program Files\Common Files\system\ado\msado15.dll)
      □ Subclassing and Timer Assistant (with configurable message response, multi-control support + timer bug fix)(C:\Doc...
      □ DTray V1.0.1(C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\DTray....
   ☑ Project Binaries
      ☑ (C:\Documents and Settings\winxp\Desktop\VB Backup\VBPZip_Source_Code\Backup Processor\Auto Zip.exe)
```

Figure 4.2 VBZipping Menu

VBZipping menu contains the link button for the administrator personnel to go to the project files contents and select the required file to be zipped. The VBZipping menu allow the administrator personnel to select from the automated or default file types of forms, classes, modules, references and project binaries. The zip all buttons allow to zip the selected file types to be zipped. The clear list box will clear the opened project file from the interface.

The administrator personnel can only upload single project file at a time to be zipped. To add another project file, the opened project file must be cleared.

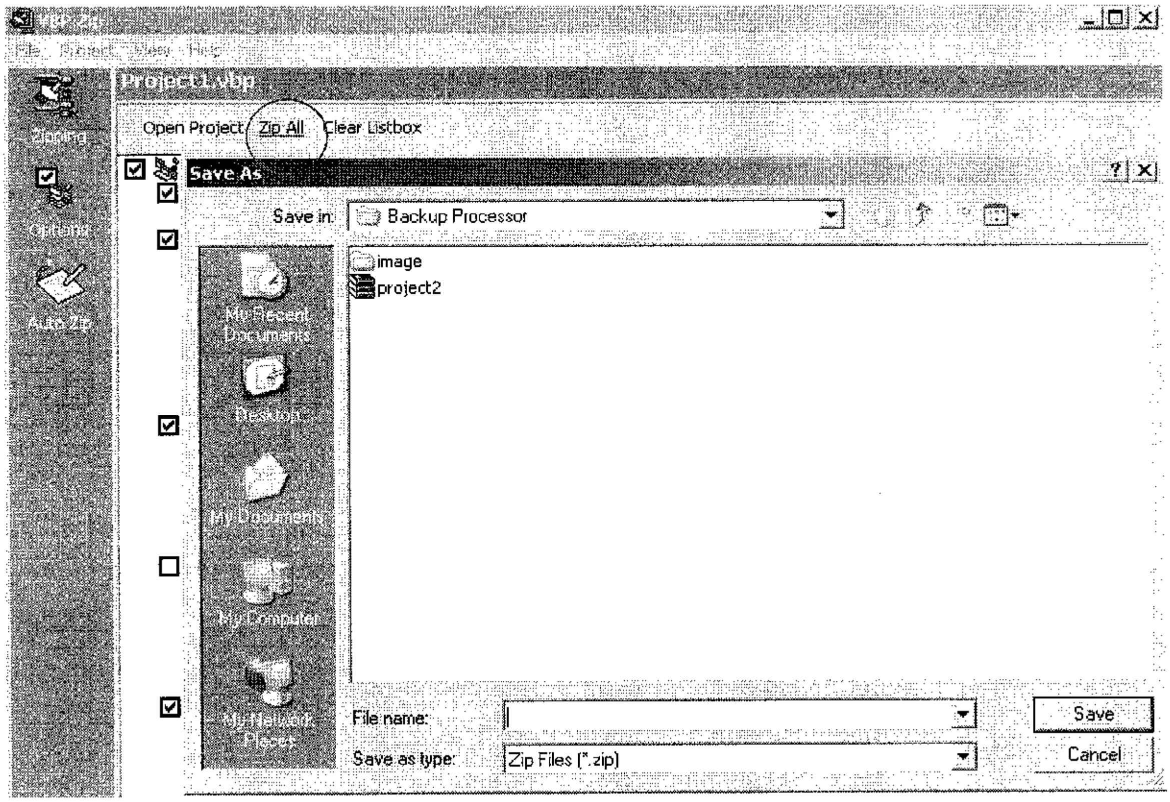
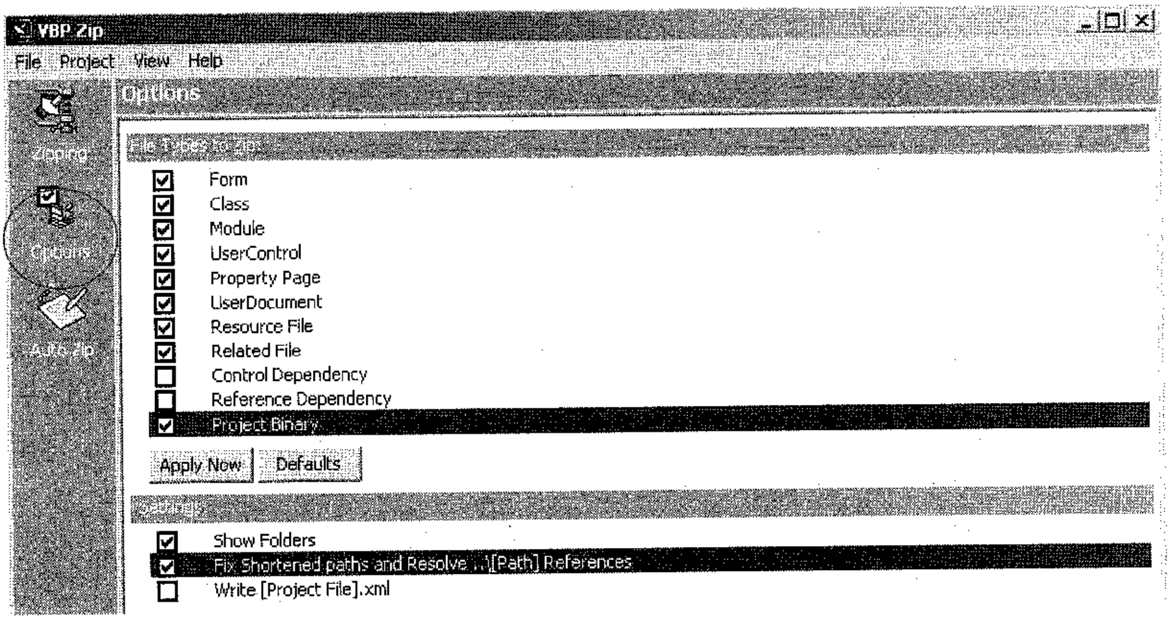


Figure 4.3 Zip All Menu

Once the administrator personnel select the project file type, the Zip Menu would enquire the file name for the new zip.



Options

File Types to Zip:

☑ Form
☑ Class
☑ Module
☑ UserControl
☑ Property Page
☑ UserDocument
☑ Resource File
☑ Related File
□ Control Dependency
□ Reference Dependency
☑ Project Binary

Settings

☑ Show Folders
☑ Fix Shortened paths and Resolve ..\[Path] References
□ Write [Project File].xml

Figure 4.4 VB Manual File Type Option

The VB Manual File Type Option allows the administrator personnel to select manually from the file types to zip. The type file options are forms, class, module, User Control, Property Page, User Document, Resource File, Related File, Control Dependency, Reference Dependency and Project Binary. The administrator personnel can either select the file types and apply to the project file or select the default option of the file types. This selection will be automatically updated in the VBZipping menu of any opened project files. Other setting provides the show folders, fix shortened path and resolve and write [Project file] xml. These are addition features to the application allow to partition the project file according to the type for easy viewing.

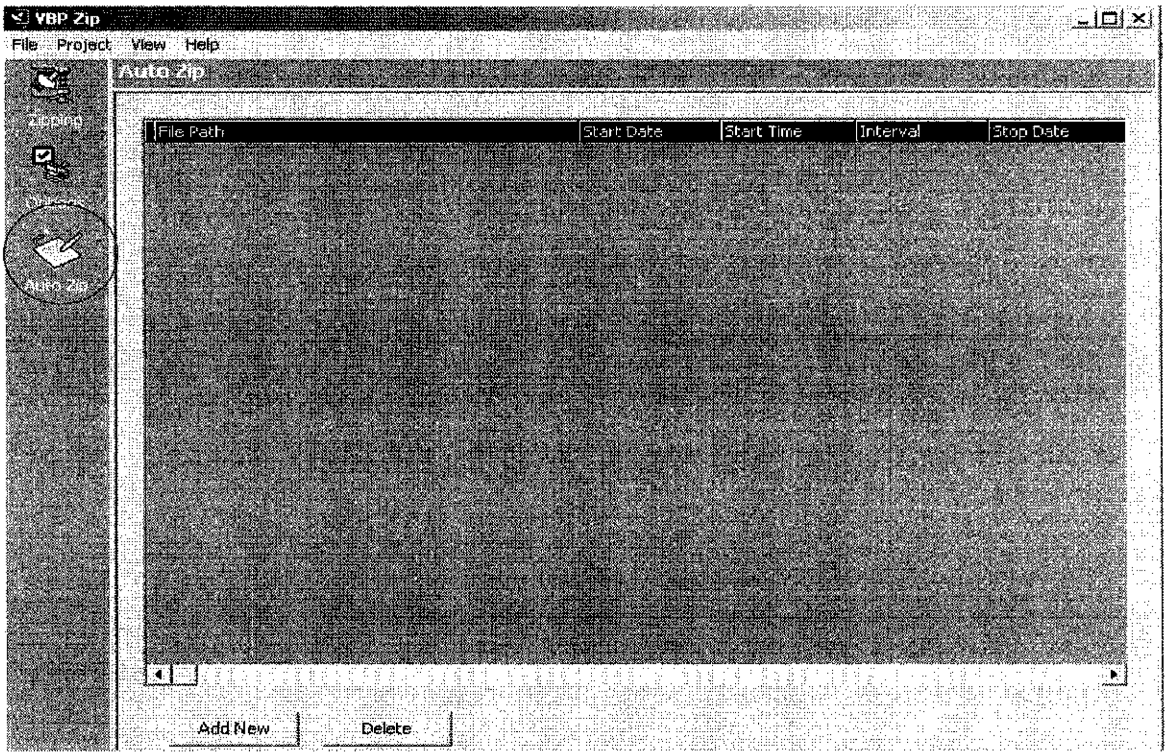
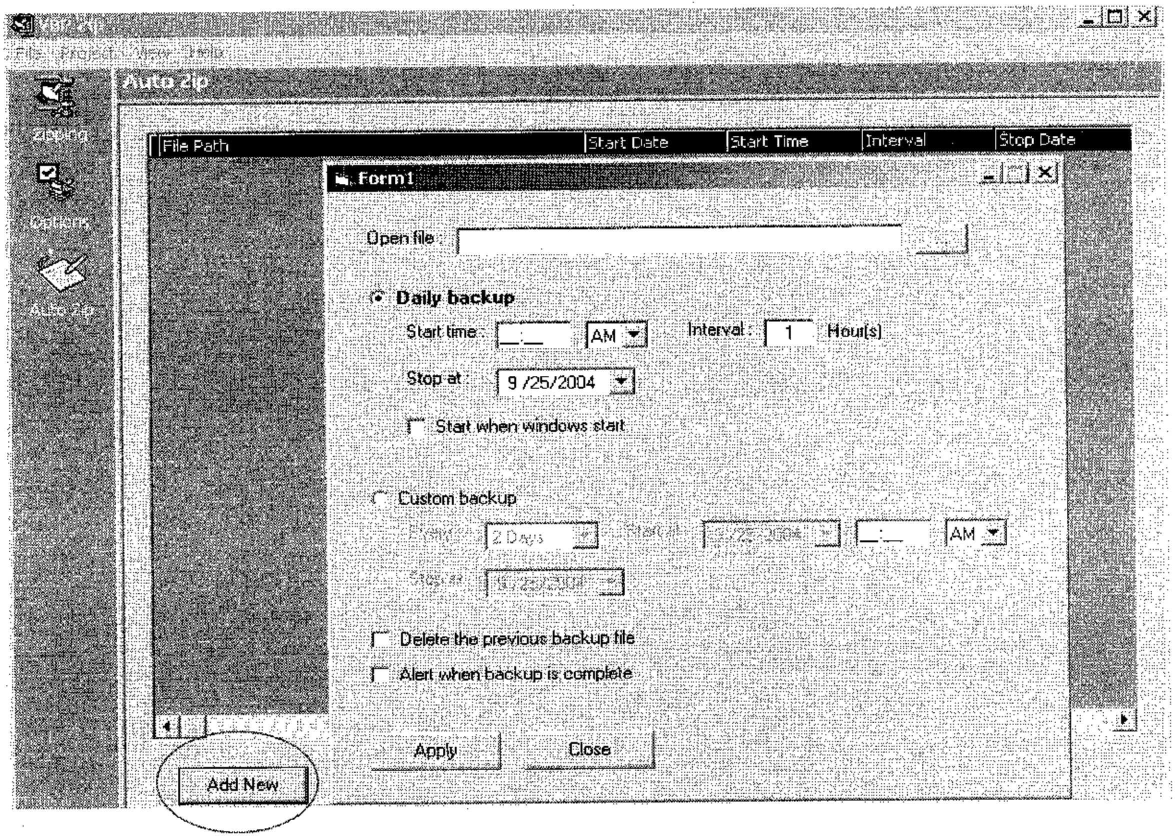


Figure 4.5 Auto Zip Menu

The Auto Zip menu contains add new button linked to the file directory of the zipped project. The interface shows the file path, start date, start time, interval and stop date of the updated database based on administrator personnel preferences. The delete button pops out a message box to the user.



Open file :

● Daily backup

Start time : __:__ AM Interval : 1 Hour(s)

Stop at : 9 /25/2004

□ Start when windows start

○ Custom backup

Every : 2 Days Start at : 9/25/2004 __:__ AM

Stop at : 9/25/2004

□ Delete the previous backup file

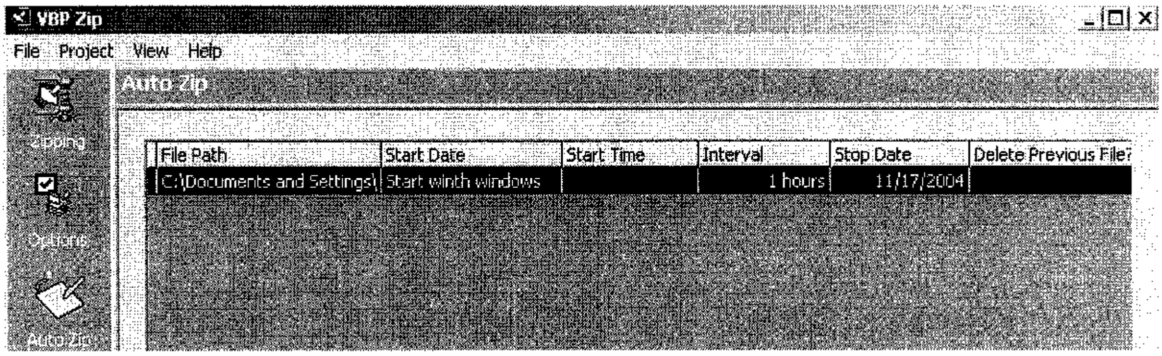
□ Alert when backup is complete

Figure 4.6 Add New Menu

The add new menu form is for administrator personnel to select the backup option for a single zipped project file. It contains the file directory to the zipped project. The administrator personnel can choose from two options of backup of daily backup or custom backup.

The daily backup contains the start time to start the backup, interval between the backup and the stop date to terminate the backup. The start when windows start allow set project file to be backed up once the windows start.

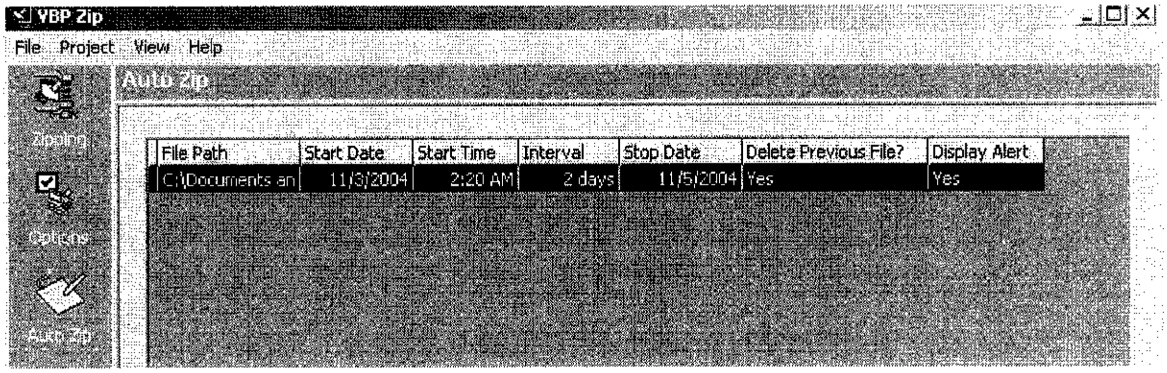
The custom backup contains day backup, start date, time and stop date. The delete the previous backup file allow the program to delete the previous backups after each updated backup. The alert when backup is complete will notify the administrator personnel that backup is saved and completed.



File Path	Start Date	Start Time	Interval	Stop Date	Delete Previous File?
C:\Documents and Settings\	Start winth windows		1 hours	11/17/2004	

Figure 4.7 Daily Backup Set Menu

The daily backup set menu show the updated administrator personnel preference to daily backup the project file set on start time, interval and stop date.



File Path	Start Date	Start Time	Interval	Stop Date	Delete Previous File?	Display Alert
C:\Documents an	11/3/2004	2:20 AM	2 days	11/5/2004	Yes	Yes

Figure 4.8 Custom Backup Set Menu

The custom backup set menu show the updated administrator personnel preference to custom backup the project file set on start date, start time, interval, stop date, delete previous file? And display alert.

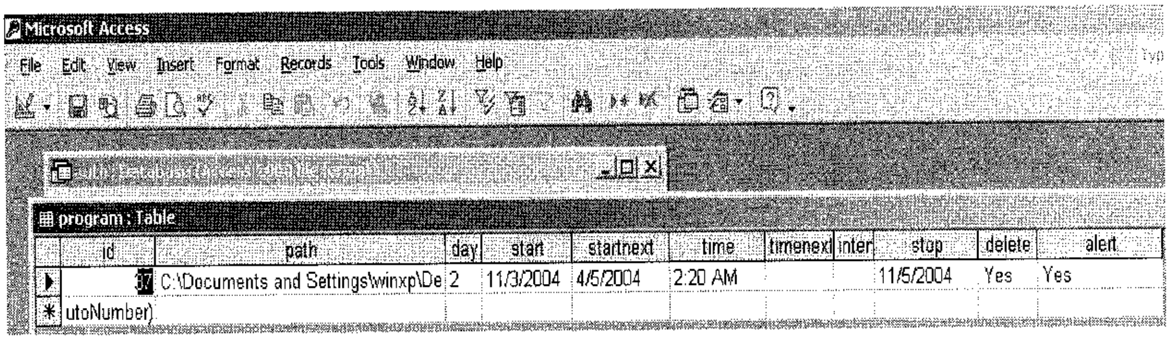
b. Data Modules

All the data for the system is placed in one directory. By using the Microsoft Access for the directory, data structure is arranged in a proper way besides having it looks more real in sense of application wise. Moreover, it reduces the possibility of forms disorder.

There only one single folder that captures the entire backup updated and all data saved in the same database of the directory.

4.1.2 Database

The database is designed by using the Microsoft Access 2002. This is one program responding to the data modules interface designed. The database function is simply to store and retrieve backup setting and data depending on the task performed on the interface. The database is linked to the interface by applying the Microsoft ADO Data Control 6.0 (OLEDB)



	id	path	day	start	startnext	time	timenext	inter	stop	delete	alert
▶	37	C:\Documents and Settings\winxp\De	2	11/3/2004	4/5/2004	2:20 AM			11/5/2004	Yes	Yes
*	utoNumber)										

Figure 4.9 Program List Table

There will be tables of database included inside the interface for the purpose of viewing and monitoring by the authorized personnel. These particulars are linked using project component Microsoft Data grid Control 6.0 "(OLEDB). The particulars are selected because they are vital information and in need have quick and firm response in case of emergency. Thus, authorized personnel could access the database system in a whole.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5. CONCLUSION AND RECOMMENDATIONS

As a conclusion, this project is a success since it achieves the six objectives stated earlier. Gradually, author can now make the project as significant as it can be and the relevance of the matter towards engineering prospect are becoming understandable and apparent.

5.1 Relevancy to the Objectives

The first objective is to compress the visual basic project files in a zip format. This is completed by selecting the best compressing technique /method and by doing some modification and revision from the source provided from vbaccelerator. The dictionary compression technique with the LZW method is used to compress the visual basic project files.

The second objective is to design the maximum project zipping and backup options for a single file, backup folders, and backup file types and restore backup. This is the more crucial part since this act as a synergy between the user and the application to interact and it is successfully achieved.

The third objective is to complete the source code development for the VBPZip. The completion leads to the testing phase, besides establishing the connectivity database of the system.

The fourth objective is to design the data structure and interface coordination appropriately. This was successfully achieved to create the good interaction between the user and the application. It is a user-friendly application.

The fifth objective is to design VBPZip that is able to store data. This is in occurrence with the third objective but the key is only the authorized personnel able to view and modify basic information for the application to retrieve backup.

The last objective is to gather information and perform case study on several compressing methods to determine the best approach. Upon this succession, it leads to the testing process which can detect run time error. This is vital consecutively to manipulate real time application.

5.2 Expectation of the Application

The application should provide data that allow justification of the project files, high level of trust in data precision during backup, storage space and short time of data interpretation. Moreover, VBPZip ensure the basic function to act in emergency cases to backup important files. In addition, the application provides mutual information among the developers and programmers who are implied in the long term project developments.

5.3 Problems Encountered

Throughout the process of developing this application, author encountered some minor problems which affected the flow of the progress.

Communication was a problem during the process of gathering real issues faced by developers and programmers. This is because author has to go to different developers and programmers before getting the exact information of the application. The protocol and level of management seems to be the main factor of this drawback. However, author coped with the situation well and obtained the data as expected but in a longer period.

From the technical aspect, author encounter problem on the backup folders due to the lack of testing. For this reason, author has to set by backup according to the appropriate intervals.

5.4 Suggested Future Work for Expansion and Continuation

The author hopes that this project will be a major success, useful and practical through different versions to be used in the future. In the meanwhile, the author focuses on to cater the user for a user-friendly system and also the design constraints.

5.4.1. Development of the System

a) Improvisation in terms of its interface to ease user interaction.
b) Usage of a more stable database to adapt with real time environment.
c) Clarification of the definite memory size of the backup storage.

5.4.2. Application Wise

a) Security is guaranteed with more identifier such as biometrics and image scanning to identify the authorized personnel.
b) The application is standardized for all visual basic project files and it continues to be enhanced in the future.
c) The application has the ability to store multiple backups of different project files at the same time with less accuracy issue and more storage capacity through file zipping.
d) The application could backup single file working on different platform of developers in the same directory.

As a conclusion, the author confirmed that this project meet the expectation and successfully in achieving the long terms objective of the project. The exposure and experience gained throughout the project will be a very good base for working environment. Certainly, the knowledge and benefits gained will be implied in daily life.

REFERENCES

Books and Journals

[Huf52] David A.Huffman. A methodfor the construction of minimum redundancy codes. Proceedings of the IRE. 40(9): 1098 – 1101 September 1952

[Fin85] Steven G. Finn. Data compression:. Dec. 24, 1985.

[Muk89] Amar Mukherjeee. Code converter for data compression/ decompression: Aug, 1, 1989.

[Mil89, Weg89] Victor S.Miller and Mark N.Wegman. Data compression method: March 21, 1989.

[Snow86] Craig A.Snow. System for compressed storage of 8-bit ASCII bytes using coded string of 4 bit nibbles: June, 24, 1986.

[Mol83] Edward W.Moll. System for minimizing space requirements for storage and transmission of digital signals. Oct 25, 1983.

[ZL77] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, IT-23(3): 337 – 343, May 1977.

[Hen93] N.Ranganathan Selwyn Henriqies. Method and apparatus for the compression and decompression of data using Lempel-Ziv based techniques: Jun 12, 1993.

David Solomon, A Guide to Data Compression Methods. Springer Professional Computing.

Adam Drozdek, Element of Data Compression, Brooks/Cole Thomson Learning.

Peter Wayner, Compression Algorithms for Real Programmers, Morgan Kaufmann.

A Technique for High Performance Data Compression, Terry A. Welch, IEEE Computer, 17(6), June 1984, pp. 8-19

J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," IEEE Transactions on Information Theory, Vol. IT-23, No. 3, May 1977, pp. 337-343

A. Gersho and R. M. Gray, Vector Quantization and Signal Compression.

D. A. Huffman, ``A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE*, Vol. 40, pp. 1098--1101, 195

Ziv and A. Lempel, ``A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, Vol. 23, pp. 337--342, 1977.

J. Ziv and A. Lempel, ``Compression of Individual Sequences Via Variable-Rate Coding," *IEEE Transactions on Information Theory*, Vol. 24, pp. 530--536, 1978.

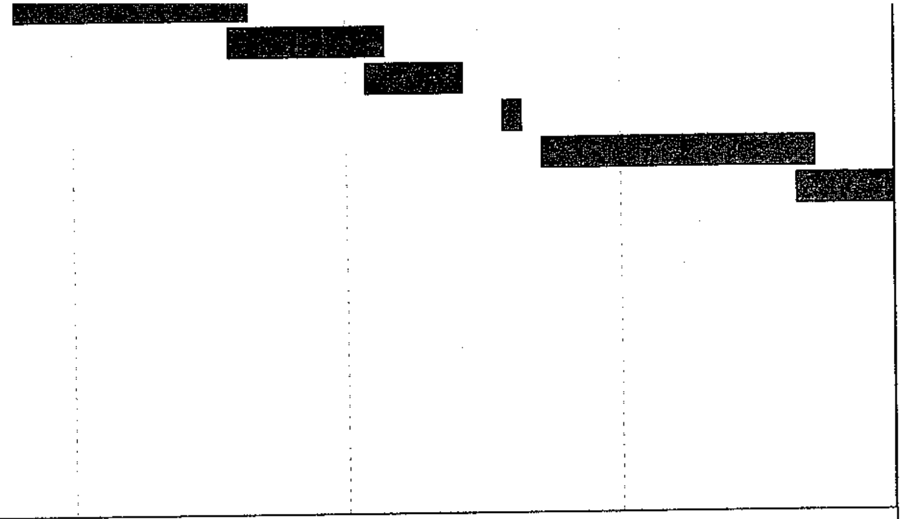
T. A. Welch, ``A Technique for High-Performance Data Compression," *Computer*, pp. 8--18, 1984.

Websites

VBAccelerator, http://www.vbaccelerator.com

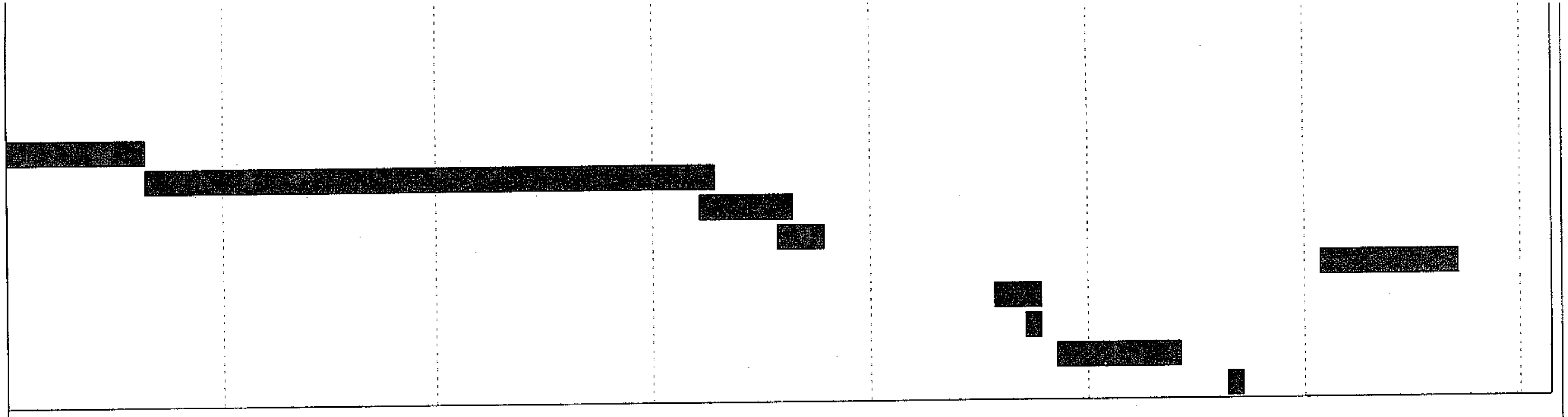
PlanetSourceCode, http://www.planetsourcecode.com

2	▦	Data Gathering	8 days	Thu 7/1/04
3	▦	Research of Usability	6 days	Mon 7/12/04
4	▦	Preparation of preliminary report	5 days	Mon 7/19/04
5	▦	Submission of preliminary report	1 day	Mon 7/26/04
6	▦	Database Design	10 days	Wed 7/28/04
7	▦	Screen/User Interface Design	10 days	Tue 8/10/04
8	▦	Application Construction	27 days	Tue 8/24/04
9	▦	Unit Testing	4 days	Wed 9/29/04
10	▦	Integration Testing	3 days	Mon 10/4/04
11	▦	Full Application Testing	7 days	Mon 11/8/04
12	▦	Preparing project Report	3 days	Mon 10/18/04
13	▦	IT/IS Exhibition	1 day	Wed 10/20/04
14	▦	Edit Final Draft Report	6 days	Fri 10/22/04
15		Final draft Submission	1 day	Tue 11/2/04



Project: Hema FYP Gantt Chart
Date: Tue 12/14/04

Task	▨	Milestone	◆	External Tasks	▨
Split	┆ ┆ ┆ ┆ ┆ ┆ ┆	Summary	▬	External Milestone	◆
Progress	▬	Project Summary	▬	Deadline	⇩



Project: Hema FYP Gantt Chart Date: Tue 12/14/04	Task	▨	Milestone	◆	External Tasks	▨
	Split	' ' ' ' ' ' '	Summary	▬	External Milestone	◆
	Progress	▬	Project Summary	▬	Deadline	⇩