ADMINISTRATION MADE EASY

[AUTOMATION OF THE ELECTRICAL / ELECTRONICS AND COMPUTER ENGINEERING DEPARTMENT OF THE FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA USING OBJECT ORIENTED PROGRAMMING (OOP)]

BY

TOKUNBO AKEEM AKINOLA

(97/592IEE)

DEPARTMENT OF ELECTRICAL / ELECTRONICS AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY

FEDERAL UNIVERSITY OF TECHNOLOGY

MINNA,

NIGER STATE

NIGERIA.

A PROJECT SUBMITTED TO THE DEPARTMENT IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF ENGINEERING (B.ENG) DEGREE IN ELECTRICAL / ELECTRONICS AND COMPUTER ENGINEERING

SEPTEMBER 2003

CERTIFICATION

This work was carried out by Tokunbo Akeem Akinola, 97/5921EE, of the Electrical / Electronics and Computer Engineering Department, School Of Engineering and Engineering Technology (SEET), Federal University of Technology, Minna, Niger State, Nigeria.

The standard requirements acceptable by the department and the School of Engineering are satisfied.

ENGR. (DR) Y. A. ADEDIRAN (PROJECT SUPERVISOR)	DATE
ENGR. M. N. NWOHU (H. O. D. ELECT. / COMP. ENG'G)	7/9/09 DATE
EXTERNAL EXAMINER	DATE

DECLARATION

I, Tokunbo Akeem Akinola, hereby declare that the idea and execution of this project was originated and carried out by me under supervision of Engr. (Dr) Y. A. Adediran of the department of Electrical / Electronics and Computer Engineering, Federal University of Technology, Minna.

TOKUNBO AKINOLA A.
(97/5921EE)

17/10/03
DATE

DEDICATION

I dedicate this project to all my friends who have always believed in me, encouraged me to work harder and by so doing, kept me going all through my stay in the university. Thus, this project is dedicated to the following people: Adeoti Akinbode, Tunde Olajide, Yemi Akinola, David Oyakhilome, Uncle Dayo Adebayo, Dapo Olanunji, Mr. & Mrs. Alabi and many others too numerous to mention. Most importantly, I dedicate this project work to my loved one, Esther Omeih, for the love, care and support shown to me in the course of this project work.

Finally, this project is dedicated to the Lord God Almighty, the One who provided everything I needed to make this project a success.

ACKNOWLEDGEMENT

I would like to first and foremost extend my profound gratitude to God the Almighty who in His infinite mercy has allowed me to reach my present position. His grace and mercy have been sufficient for me even through difficult times and I pray His love continues to see me through even till the end.

Secondly, I would like to thank my parents who have worked tirelessly through the rigors of life to provide the best for me and the rest of the family. To my dad, who encouraged me morally, financially and spiritually through the project, I say thank you so much. To my ever caring mother, I say I love you because you are the best. I couldn't have asked for better parents than you both. My thanks also go to my guardians, Professor & Mrs. Olatunji, for their undying support all through.

I wish to also seize this opportunity to express my profound gratitude to my project supervisor, Dr. Y. A. Adediran, for his support and supervision. Others who contributed immensely to the success of this project include Mr. Wole (NOVOTEL, Abuja), Mr. Segun Ogundairo, Mr. Chinedu, Mr. Tope Ajayi and Mr. Jonas. In addition to those mentioned, my gratitude also goes to the APTECH (Abuja) crew: Mr. Roy (Academic Head), Mr. Dennis, Counselors Joy & Diana and all other members of APTECH ABUJA.

Last but not the least, I would like to thank all those who gave me a listening ear and a shoulder to lean on when I had no one around, and all those who believed in me and hence encouraged my work. I am indeed grateful.

May the Lord God Almighty reward you all and guide you all in your endeavors

ABSTRACT

This project report presents a study of the automation / computerization of administrative jobs in the Electrical/Electronics and Computer Engineering department using an object oriented software (AMEzy 1.0) developed with Visual Basic 6.0, Microsoft Access 2000, and Structured Query Language (SQL).

It begins by highlighting on the departmental administrative tasks and the present situation of the department. It goes on to elaborate on the need for computerization / automation and the various ways to go about this. It also presents a viable and easy solution to several departmental issues by the use of developed software called AMEzy (Administration Made Easy).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 GENERAL INTRODUCTION

For any system to be effective and productive, it must embrace good organization from skilled, dedicated and experienced workers. Good products enhance output and this in turn brings about the need for more qualified workers as the bulk of work increases. It is therefore very important that quality output should be produced in as little time as possible with little or no deficiencies, using efficient and proper methods. It was man`s search for that method of perfection that brought about what is today known as automation.

Automation, by definition, refers to the use or introduction of automatic equipment to save mental and manual labour. It could also be defined as the automatic control of the manufacture of a product through successive stages. In all, the main goal of automation is to computerize manual tasks which are tedious and time consuming. In many establishments, machines have been embraced as a form of automation. But of recent the most popularly used automatic equipments for this purpose has been the microcomputer, and the automation process with computers has been termed computerization.

Computerization has made automation easy as the microcomputer can now be used to perform a lot of tasks which include word processing, computer aided design, hardware and software programming etc. Also this has led to the development of easy to use software packages which are in turn used to create other application software for specific purposes e.g. a billing and accounting software package for banks, traffic regulator simulated software for road users, etc. These application software can be modified to suit the needs of the users and hence bring about desired

results. The software packages used to create other application packages are actually high level languages which are up to date being modified to simplify coding. These languages have come a long way and now embrace an approach known as object oriented programming (OOP).

Object Oriented Programming is a method of programming in which an application implements entities as they are seen in real life and associates actions and attributes with them. Here code and data are merged into a single indivisible thing – an object. This helped to simplify complex systems and make them easier to manage and structure. With the help of OOP, programming became clearer and programs became more reliable and easier to maintain.

1.2 LITERATURE REVIEW

Electronic computers have existed only since around the middle of this century. The early computers (e.g. ENIAC – Electronic Numerical Integrator and Calculator in 1940, EDSAC – Electronic Delay Storage Automatic Computer in 1949, UNIVAC in 1950) vacuum tubes took over the functions that had previously been performed mechanically. Today, after several generations, miniaturization has brought about the microcomputer in a size compatible with other household furniture and at a cost within the reach of many family budgets.

Microcomputers are the smallest general processing systems that can execute program instructions to perform a variety of tasks. For such computers to play a part in problem solving, it was necessary that communication be established between them and the users. That bridge of gap in communications was closed with the advent of computer programming languages which translated programs written by users into machine codes.

Programming languages are basically of three types: machine language, low-level language and high level language. Machine language involves the use of binary codes; Low level languages use symbols in which instructions which corresponded to machine codes were translated using programs known as assemblers; High level language is a problem oriented language which is a restricted form of natural language used in programming and uses system software known as compilers (e.g. FORTRAN, BASIC, COBOL etc) to translate its codes into a machine language equivalent. While there were many important innovations in software development overall, perhaps the most significant over the past 25 years was the widespread adoption of object-oriented programming (OOP) using high-level language.

Object Oriented Programming (OOP) has many roots but can be traced back to the 1950s with Lisp and the late 1960s with Simula. Alan Kay, who coined the term "object-oriented programming" from his Smalltalk language, is described as the father of OOP. He established the fact that programs would be easier to write when they were modeled on things that were easier to understand. He also realized that graphical user interfaces were important as they improved the interaction between people and machines. Object-oriented languages like Smalltalk enabled the widespread adoption of these interfaces and this eventually led to more general software components wired together using prewritten objects and minimal coding. Most importantly, OOP helped lay the foundation for many advances in software engineering, from testing techniques to programming methodologies.

Today, unlike in older ways of coding, the OOP approach uses objects with actions and attributes to simplify coding. It offers a powerful model for writing computer software and allows for the analysis and design of an application in terms of entities or objects so that the process replicates the human thought as closely as

possible. In our present world, OOP compilers (e.g. Java, C++ and Microsoft Visual Basic 6.0) have been used to create all sorts of software ranging from desktop applications to distributed or client-server applications. Most of these software have been interfaced with relational databases to achieve desired results. These databases are simply a collection of interrelated tables which store and manage data in tabular formats using a relational database management system (RDBMS). Most importantly, these OOP compilers have been put to use in the educational sector by using them to create software for school management. A good example of such is the Administration Made Easy (AMEzy) 1.0 Software.

AMEzy Suite is a software application package developed using Microsoft Visual Basic 6.0 alongside Microsoft Access 2000 (an RDBMS package) for automation of the Electrical/Electronics and Computer Department of the Federal University Of Technology, Minna, Niger State. It handles administrative jobs (e.g. departmental registration, course registration, identification card creation, result compilation, transcripts etc) in the department and by so doing eases the jobs, manages data efficiently and simplifies administration.

1.3 THE ELECTRICAL & COMPUTER DEPARTMENT

The Department of Electrical and Computer Engineering came into being on 1st February 1983 as a department in the School Of Engineering and Engineering Technology (SEET) of the Federal University of Technology (FUT) Minna, Niger State. It aims at imparting a sound knowledge in Electrical, Computer and Electronics Engineering in students that will meet Nigeria's need for engineers is that field. It offers a Bachelor of Engineering programme for students admitted through UME and direct entry.

During the first academic semester all students admitted to this department are expected to register with the department and the Dean's office for all courses to be taken in the semester. If a course is not registered for by a student, he/she is not eligible to have examination results for that course. After registration, the students are to be given identification cards, via the department, from the school authority stating them as bonafide students of the university and the department. As part of the degree programme a one year Industrial Training (of which 24 weeks is to be done during the second semester of 400 Level) is compulsory for the students. At the end of the programme a point grading system is used to grade the students and give them a final assessment.

From the inception of the university, up till this present moment, the whole process of registration, creation of identification cards, transcript and result compilation have been done manually. Of recent however, attempts have been made to computerize these processes but not to automate or harmonize them.

1.4 BENEFITS AND RISKS OF COMPUTERIZATION

1.4.1 WHY AUTOMATE OR COMPUTERIZE?

With such rapid development in technological advancement in the present era, it is evident that computers are here to stay. Advancement in the software industry has also enabled programmers to easily write codes, reuse, package and deploy them on almost any platform available. This is a great advantage as many establishments cannot use off-the-shelf software due to variations in the format of design. A good example is that of currency symbols. Customized software enable establishments to specify parameters to be used solely for the establishment as computerization handles

the specific needs of that establishment. This is perhaps the most important reason for automation.

Other reasons for automation include:

- Quick, easy and improved access to critical and important information;
- To increase practice and business efficiencies of the establishment.
- Improved ability to share information locally and internationally;
- Remote access to information;
- Easier mode of security application and protection of information;
- To increase accountability and efficiency.

1.4.2 USE OF AUTOMATED SOFTWARE IN TODAY'S WORLD

In the present world today, OOP compilers have been used to create all sorts of software ranging from desktop applications to distributed applications.

Desktop applications refer to application software which run on a single system. Examples of such software created with OOP compilers are:

a. Accounting Software Packages e.g. Peachtree Accounting Package
b. Digital Camera Software e.g. Acer DC300 Package,
c. Computer clock monitors e.g. Clockwise

Distributed or client-server applications are applications which run on two-tier or three-tier architecture. In this case the whole software is distributed among several systems. For example the front end of the application (i.e. the user interface) could reside on one computer while the backend (the database or other components) could reside on other computers. Examples of such software are:

a. Cyber café Software e.g. CyberKlock, Cybercafé Pro
b. Antivirus Software e.g. Norton Antivirus Cooperate Edition
c. Peer-to-Peer Software e.g. Bearshare, Kazaa, Napster

1.4.3 FACTORS TO BE CONSIDERED IN AUTOMATION

Before an establishment or institution begins to automate its tasks it must consider many crucial factors. Some of these include:

i. **Cost:** The cost of acquiring such software as compared with already made (off-the-shelf) software should be compared and put into consideration before making a decision. Also to be considered is the cost of acquiring reliable computers for the software to be used if necessary.

ii. **Compatibility With Existing Computer Systems:** Automated software should be checked to ensure that it is compatible with the type of microcomputer being used or the operating system software existing on the computer. This step is also to be considered by the programmer before choosing the appropriate OOP software to use for automation.

iii. **Security:** This feature is of paramount importance in most establishments of the world today. An insecure software could ruin many establishments especially those that have sensitive information stored e.g. money, private data etc. The security features of the software should therefore be put into consideration before implementation of the software.

iv. **Architecture To Be Implemented:** In choosing to automate, the kind of network structure to be used is also very important. Client-server architecture might probably involve client-side software and server-side software to operate. A desktop system which is not on any network will only involve one software and is bound to be cheaper than a distributed application.

v. **Integrity And Reliability:** If a software is not reliable then its results cannot be relied on and therefore the software is of no good to any party. But software

whose results can be trusted reduces the scrutiny job that has to be employed by staff to ensure correctness and reliable results.

vi. **Staff Training:** A most important factor of consideration is staff training. For maximum efficiency and utility, staff have to be trained adequately to be able to handle automated software. This allows the staff to get familiar with the software and in turn increases productivity.

1.4.4 RISK INVOLVED

Even in its beauty and prestige, it must be stressed that automation with computerization has a few risks involved which would be worth noting.

The first and most important risk is that of security. A breach of the software's security features could be disastrous to the establishment as it could involve loss or corruption of valuable information, theft and even crashing of the whole system. It must however be noted that there is no software without errors but the security that can be offered by that software depends on how minimal the errors produced are.

Another risk to be considered is compatibility with upcoming operating software. For instance if a program has been developed to work on all existing platforms of Windows and a new package is released in the future, the software might no longer be able to run on that platform. In such a case an upgrade may have to be requested for and that could cost more money.

A firm, organization or establishment which decides to automate should also consider the risk of information loss or hindrances due to unsteady power supply as provided by the National Electric Power Supply (NEPA). If possible, there should be a backup plan so that vital information is not lost and work goes on smoothly.

1.5 PROJECT MOTIVATION AND OBJECTIVE

It is an ultimate dream that sometime in the near future automation of the Electrical and Computer Department would be used to compliment the presently employed manual method of performing administrative tasks in the university. If possible, this wave of automation should also spread to other arms of the university so that the entire community could benefit from its numerous advantages.

Although the software to be used here (AMEzy 1.0) was specifically developed for the Electrical and Computer Department of FUT Minna, it has numerous objectives:

a. To illustrate automation of an arm of an establishment;
b. To show how easily administrative tasks can be simplified;
c. To encourage networking of various departments in an establishment;
d. To spur up interest of young upcoming students especially in the area of computer programming;
e. To emphasize the benefits of automation in our world today.

1.6 PROJECT LAYOUT

Chapter 2: is a description of the theory of object oriented programming (OOP), its concepts and a brief look at some of the languages which make use of this approach. This chapter also discusses Relational Database Management System (RDBMS) and its design as well as previous database models. It will also discuss the connection between OOP and RDBMS.

Chapter 3: handles the description of the software development and design. This includes the database structure used as well as that of the software, and a breakdown

of the automated software (AMEzy 1.0). It also talks about the packaging and deployment of the software.

Chapter 4: deals with the testing of the automated software with real life case studies, collection and comparison of results with the manual method and discussion of the final results.

Chapter 5: contains the conclusion and recommendation for the automation process as a whole.

CHAPTER TWO

THEORY OF OBJECT ORIENTED PROGRAMMING AND RELATIONAL DATABASE MANAGEMENT SYSTEMS

2.1 OOP BASICS

In procedure based programming, when designing applications, the focus is on the procedures and functions in the program. The basic algorithm is designed first then implemented in procedures and functions. It can then be refined gradually by including more and more detail in these functions and procedures. This method of program development is known as stepwise refinement. In contrast, the focus of object-oriented programming (OOP) is on the data in the application and the methods that manipulate them. It consists of identifying the objects and what is to be done with them as steps in the solution to the problem. These objects have specific roles, communicate and work together to perform the functions of the program. OOP is basically a different way of organising and structuring programs and is modeled after the real world in which an object is usually made up of other smaller objects.

There are several properties, terminologies and concepts underlying object-oriented technology. The concepts of OOP listed ahead, have a one-to-one correspondence between the way we think about a problem and the implementation.

2.1.1 CONCEPTS OF OOP

Central to object oriented programming are the concepts of objects, classes, properties or attributes, messages, methods, interfaces and reusability of components.

- ***Objects:*** An object in OOP represents an entity in the real world. It is defined as a concept or thing with defined boundaries that is relevant to the problem being dealt with. Examples of objects include vehicles, electrical components

in a circuit design model, personnel files etc. An object is also said to be an instance of a class.

- ***Classes:*** Objects which have the same properties, common behaviour and common relationships are called classes. Examples of classes include class of polygons, class of animals etc. Classes help to define the characteristics that objects possess and normally consist of interfaces and implemented codes.
- ***Property/Attribute:*** The characteristics of the object are represented as the variables in a class and referred to as the properties or attributes of the class. For example in a class of polygons all objects have vertices and edges as part of their properties or attributes.
- ***Messages:*** Software objects interact and communicate with each other using messages. Sending a message to an object causes that object to perform an operation on itself. When a message is sent, the receiver object looks at the message, decodes the appropriate operation to perform, and executes it. The code corresponding to a particular message is known as a "method".
- ***Method:*** refers to an action required of an object or entity when represented in a class. All objects in a class perform certain common actions or operations, which become functions in the classes that define them. "Draw", "Erase" and "Move" are examples of the methods that are part of the polygon class.
- ***Interfaces:*** An interface is a contract in the form of a collection of method and constant declarations. When a class implements an interface, it promises to implement all of the methods declared in that interface. Most interfaces in OOP are visible to the user and help in interaction with the program.
- ***Reusability Of Components:*** In all object-oriented languages programs are broken down into extensible and reusable components. This allows

programmers to reuse them later or modify the existing component to create a new one.

2.1.2 PROPERTIES OF OOP

Every object oriented program has four important properties. These are: Data Abstraction, Inheritance, Encapsulation and Polymorphism.

I. Data Abstraction :

This is the process of examining all the available information about an entity to identify properties and methods that is relevant to the application. Grouping of objects into classes is a good example of data abstraction as this causes common definitions to be stored once per class instead of once per instance of a class.

II. Inheritance :

Inheritance is the property that allows the reuse of an existing class to build a new class. The new class created inherits all the behaviour of the original class and is called the subclass. It is said to be a specialization of the original class. The original class is called the parent class, or superclass, of the new class. It is said to be a generalization of its subclass. Inheritance helps to eliminate duplication of data and behaviors.

III. Encapsulation :

This is a process that allows selective exposing or hiding of properties and methods in a class. It involves providing access to an object only through its messages, while keeping the details private. It is a technique for minimizing interdependencies among modules by defining a strict external interface. This way, internal coding can be changed without affecting the interface, so long as the new implementation supports the same (or upwards compatible) external

interface. Encapsulation of code helps to protect code from accidental corruption and isolate errors to small sections of code to make them easier to find and fix.

IV. Polymorphism:

Object oriented languages try to make existing code easily modifiable without actually changing the code. Polymorphism makes this possible. With polymorphism the same functions may behave differently on different classes. This approach allows a programmer to maintain and revise code with less errors since the original object is not changed.

2.2 OOP LANGUAGES

Object oriented languages have come a long way. Today there are several of them in use for many purposes, especially automation. The choice of which language to use depends on the programmer and the features offered by the language.

2.2.1 TYPES OF OOP LANGUAGES

The language concepts started from concepts set forth in LOGO and in Simula67 programs. Using these concepts the first object oriented programming language to be developed was called Smalltalk. It was developed by Alan Kay in the mid 70's. It used run-time binding and a rich class library that could be easily reused via inheritance. It also had a dynamic development environment which allowed easily refined class definitions. Today there are almost two dozen major object-oriented languages in use. Some of these are: Ada, Clu, C++, Eiffel, Java, CLOS, F90 and Visual Basic. However the leading commercial object-oriented languages are C++, Java, and Visual Basic.

C++ was developed by Bjarne Stroustrup at AT & T Bell Laboratories using C language as a base but implementing the principles of object-orientation most effectively to give a powerful language. However it sacrifices some flexibility and trades off some of the power to reuse classes, in order to remain efficient

Java (originally called Oak) was created by James Gosling in 1991 at SUN Microsystems in the USA. Although it was originally intended for embedded applications, the development of the Internet shifted the aims of the project. Today java is the leading language for web applications.

Visual Basic, which was developed from the procedural BASIC language, was originally developed in 1987 by Alan Cooper under the name of 'Ruby'. At one time Visual Basic could produce code for both Disk Operating System (DOS) and Windows applications. Today, however, DOS is obsolete and Visual Basic is now a fourth generation programming language used for roughly two-thirds of all business applications programming on personal computers.

As earlier stated, all these languages vary in their support of object-oriented concepts. It should however be noted that there is no single language that matches all styles and meets all needs.

2.2.2 VISUAL BASIC 6.0: THE RIGHT CHOICE?

As the programmer, I have chosen to use Visual Basic 6.0 for creation of the Administration Made Easy (AMEzy) software for automation of the department. The reasons for these are not far fetched.

Visual Basic 6.0 (VB6) is the sixth version of Visual Basic created by Microsoft. Before Visual Basic 1.0 was released, developing a Microsoft Windows application required an expert C/C++ programmer with 10 kilogram worth of documentation for

the needed C/C++ compiler and the essential add-ons. Visual Basic 1.0 (which was first released on March 20th 1991) brought a stop to that as it involved easier ways of coding. Visual Basic 2.0 (November 1992) was faster, easier and more powerful than 1.0. Visual Basic 3.0 (1993) added simple ways to control the most powerful databases available using a technology known as OLE (Object Linking and Embedding). This was a very significant development in programming history. Visual Basic 4.0 (released in 1995) added support for 32-bit development and began the process of turning Visual Basic into a fully object oriented programming language. Visual Basic 5.0 (released in early 1997) added the ability to create true executables and even the ability to make your own controls. Visual Basic 6.0 (VB6) was introduced in 1998 and was included as part of a package known as Visual Studio 6.0. VB6 added new capabilities in the areas of data access, Internet features, controls, component creation, language features and wizards. To quote Microsoft's web site, "Visual Basic 6.0 features provide graphical, integrated data access to any ODBC or OLE DB data source, and additional database-design tools for Oracle and Microsoft SQL Server™-based databases. New Web development features bring the easy-to-use, component-based programming model of Visual Basic to the creation of HTML- and Dynamic HTML (DHTML)-based applications." Many organizations are still using this version today. The newest version of Visual Basic, sometimes referred to as VB7 or Visual Basic .NET, was released in February 2002. This product will be part of Microsoft's .NET software initiative, designed to produce Extensible Mark-Up Language (XML)-based applications for the Microsoft Internet environment. However, it has not been fully deployed by Microsoft to the .Net family yet.

In the country today, most establishments (FUT Minna inclusive) make use of "Windows" as their computers' operating system. With a beautiful Graphic User

Interface (GUI), an attractive desktop filled with easy to click icons, and programs that use easily accessible menus, the reason for choosing "Windows" over other operating systems can be easily understood. In the same vein, we find the most important reason for choosing VB6 over other versions of Visual Basic and other programming languages.

VB6 employs the use of drag and drop reusable components for easy programming. It lets you add common "Windows" components like menus, test boxes, command buttons, scroll bars etc. It lets the user communicate with other applications, control and access databases, create multi-tasking environments and most importantly, allows easy packaging and deployment of software using automated wizards. Although Visual Basic 5 has all these features, it does not incorporate any form of report creation. This had to be done with external software like Crystal Reports. VB6 has conquered that defect by providing an easy to use data report. Other modern languages (e.g. C++ and Java) have many of these features but are not as flexible and user friendly as VB6. This, in a nutshell, explains the choice of VB6 for the creation of AMEzy 1.0 Suite.

2.3 BENEFITS OF OOP WITH VISUAL BASIC 6.0

The numerous benefits which VB6 provide include the following:

- ✓ User friendly programming interface;
- ✓ It facilitates reuse of code and components and as a result of this, fewer and shorter iterations;
- ✓ VB6 allows for ease of maintenance and enhancement;
- ✓ Provides easy access, connection and control of different powerful databases (e.g. ORACLE and SQL Server) using Structured Query Language;

- ✓ Employs tools like "Watches", "Immediate Window" and many others for easy testing, debugging and optimization of programs;
- ✓ Contains an integrated data report for creating reports which could be printed out on hard copy or exported to the internet;
- ✓ Monitoring of mouse activity and file handling is made easier with VB6;
- ✓ Easy interaction of programs with other Windows applications;
- ✓ Contains an easy to use Package and Deployment Wizard for compiling, packaging and deploying software on almost any medium.

2.4 DATABASES

A database is a collection of data as well as programs required to manage that data. Earlier databases came in form of flat-file systems i.e. all data was stored in one table. However this type of system led to data redundancy (duplication of data) and inconsistency. Nowadays databases contain many more tables related to one another, are compact, accurate, secure, fast and easier to use than traditional databases. Also of importance is the fact that databases now allow multiple users to share data as common resources.

2.4.1 DATABASE MODELS

In the process of evolution, the database was designed on different database models, with each model adding more efficiency to the database. A database model is a description of the data container and a methodology for storing and retrieving data. Before the 1980's, the two most commonly used database models were the hierarchical and network systems. In the 80's (and up until now) the 'Relational Database Model' became the rage.

2.4.1.1 HIERACHIAL AND NETWORK MODELS

These models gave birth to the flat-file system of earlier databases. They involved a number of people, forms and books of records or simply 'registers'. Tables could not be automatically related and so people had to be employed to manually check relationships, sort and update records. This process was definitely tedious and not error free as it provided room for redundant and inconsistent data especially in large establishments

2.4.1.2 RELATIONAL DATABASE MODEL

This model was developed by a mathematician named Dr. E. F. Codd at IBM in the late 1960s. He specified a set of 12 rules that have become popular as Codd rules. A database that meets these rules is classified as a 'true' or 'complete' RDBMS' (Relational Database Management System).

While a database can be defined as a set of related records, a database management system (DBMS) can be simply described as a computerized record keeping system. A DBMS takes care of complex calculations and validations, sorting, data security and many other tasks that cannot be performed by using spreadsheets. It acts as an interface between the database and the user. When a DBMS contains tables that can be related, the system is called a Relational Database Management System or RDBMS

In the relational data model, the data in the database is arranged in 'relations' or 'tables' consisting of columns and rows referred to as fields and records in DBMS terms, or attributes and tuples in RDBMS terms. These tables store and manipulate

data according to user instructions. Tabular relationships help to prevent duplication and inconsistency of stored data.

Some major examples of RDBMS commercially in use today are Borland's Paradox, FoxPro, ORACLE, SQL Server and Microsoft Access. The preference of which to use is normally dependent on familiarity with the software and amount of data to be stored.

2.4.2 MICROSOFT ACCESS 2000 – AN RDBMS

Microsoft Access 2000, version 9.0, is an RDBMS component of Microsoft Office 2000 Suite. It comprises of a 'Back-End' and a 'Front-End'. The 'Back-End' part of the application is the one which takes care of storing and retrieving the data. The 'Front-End' on the other hand provides a user-interface or a means by which the user can interact with the back-end data. Most times however, Access is simply used as a 'Back-End' software as its front end is not as flexible as required.

As an RDBMS, Access offers many features and responsibilities. Some of these are as follows:

i. ***Large Data Storage Capability:*** Access has the ability to store, manipulate and maintain a maximum of about 65,536 records.

ii. ***Control of Data Redundancy:*** By storing data in several tables, data repetition or redundancy is reduced in Access 2000.

iii. ***Abstraction of Data:*** Access hides the actual way in which data is stored, while providing the user with a conceptual representation of the data.

iv. ***Multiple User Support:*** Typical of a true RDBMS, Access provides multi-user support by ensuring that several users can concurrently access data without affecting the speed of data access. It also employs a 'record-locking'

mechanism which ensures that no two users could modify a particular record at the same time.

v. ***Multiple Ways Of Interfacing To The System:*** Microsoft Access allows a variety of front-end tools to use the database as a back-end. For example data stored in Access can be displayed and manipulated using forms created in FrontPage 2000 or Visual Basic (as is the case with AMEzy Software).

vi. ***Restricts Unauthorized Access:*** By assigning rights and passwords to users, Access is able to implement data security effectively.

vii. ***Enforces Integrity Constraints:*** As an RDBMS, Access checks data entered into a table to ensure that it is valid so as to preserve data integrity e.g. primary keys are created and checked to ensure that data is not duplicated.

viii. ***Backup and Recovery:*** In spite of ensuring that the database is secure from unauthorized access as well as invalid entries, there is always the danger of losing the database. To guard the database, Access has an inbuilt backup and recovery technique that ensures protection.

In all, it can be seen that Microsoft Access 2000 is a very powerful and secure RDBMS to be used as a backend for any front end software.

2.5 STRUCTURED QUERY LANGUAGE

To be able to effectively and efficiently manipulate data, the application accessing the database must implement the Structured Query Language (SQL). SQL is actually a programming structure used to implement the mathematical way of formulating expressions for querying databases. The history of SQL and relational databases traces back to Dr. E.F. Codd, an IBM researcher who first published an article on the

relational database idea in June 1970. Codd's article started a flurry of research, including a major project at IBM. Part of this project was a database query language named SEQUEL, an acronym for Structured English Query Language. The name was later changed to SQL for legal reasons, but many people still pronounce it SEQUEL to this day. IBM published many articles in technical journals about its SQL database language, and in the late 70's two other companies were started to develop similar products, which became Oracle and Ingres. By 1985 Oracle claimed to have over 1000 installations. In the late 80's and early 90's SQL products multiplied and became virtually the standard for database management in medium to large organizations, especially on UNIX and mainframes. Over the past few years SQL has rapidly emerged as both a de facto and as an official standard language for database management. Today there literally are dozens of vendors providing SQL for everything from the smallest PC to the largest mainframe.

The SQL "language" allows anyone with a computer terminal to access and use relational databases. SQL uses about 30 simple "english like" commands like Open, Close, Select, and Update to manipulate the database. For example, the SQL command shown below could be use to select all database records in the "Western" region.

SELECT * WHERE Region="Western"

Although SQL can be used directly by simply typing in commands like this, the SQL language is tricky for non-programmers to learn. AMEzy 1.0 software however allows you to access the database using a standard Windows graphical interface. AMEzy software then translates your mouse clicks and keyboard taps into the SQL language and passes them on to Microsoft Access.

One of the major benefits of SQL is that it provides a more or less standard way to access and use database systems from a variety of vendors. For example, the SELECT statement listed above would work exactly the same on Butler, Oracle, Sybase, DB2, or any other SQL based database on any host machine. However SQL standardization is not complete. Although an ANSI standard for SQL was adopted in 1986, then revised in 1989, no commercial SQL product available today exactly conforms to it. Each product has a slightly different dialect of the SQL language (most have extensions to the standard SQL language). Like dialects of human languages, these different SQL dialects are basically similar but incompatible in their detail. Because of these dialects it is very difficult to write a software that runs with all versions of Access databases.

2.6 INTERACTION OF VB6 WITH DATABASES

Visual Basic 6 has several methods of interfacing or connecting to databases. This is done with an efficient approach known as Universal Data Access (UDA). UDA is a philosophy whereby a developer should be able to use one tool to access any type of a data from any data source. This dream is presently being realized through two database access technologies namely ODBC (Open Database Connectivity) and OLE-DB (Object Linking and Embedding – Databases).

ODBC is a standard protocol for accessing relational databases based around SQL. It (ODBC) is an older and less advanced method of connecting to a database. A more advanced way to connect to a database is to use OLE-DB. OLE-DB is a database architecture that provides universal data integration over an enterprise's network, from mainframe to desktop, regardless of the data type. It is a more generalized and more efficient strategy for data access than ODBC, because it allows

access to more types of data (both relational and non-relational) and is based around Microsoft COM (Component Object Model). Using OLE-DB Microsoft offers an array of technologies for accessing data sources from applications. They include Data Access Objects (DAO), Remote Data Objects (RDO), and ActiveX Data Objects (ADO). Of all these, Microsoft Access 2000 supports two data access models: the traditional Data Access Objects (DAO) and ActiveX Data Objects (ADO).

DAO targets the Microsoft Jet database engine to enable quick and easy database programming. Access 2000 is the first version of Access to also support ADO for manipulating Jet databases. Instead of being based on a single database engine, ADO uses a common programming model to deliver access to universal data. It combines the best features of RDO and DAO. It relies on OLE-DB providers for low-level links to data sources. It is speculated that OLE-DB technologies will eventually make their ODBC predecessors obsolete, much as ADO will replace DAO. Therefore, the connection invested for the compilation of AMEzy 1.0 is ADO.

ADO can be implemented in VB6 using a data control or using any of its libraries. The data control used it is referred to as ADODC (ADO Data Control). Also the most used ADO library is the ADODB library. Using this library, the process of connection follows a certain hierarchy which is worth noting. The connection process is as follows:

- First a "*Connection*" or link to the database is created;
- Secondly a "*Command*" must be stated or issued stating what is to be queried for, using Structured Query Language (SQL);
- Following the "*Command*", a "*Recordset*" has to be created as a container for the SQL results;

* Lastly, the collected records can be grouped under "*Fields*" in the "*Recordset*". This process is however optional.

A good knowledge of this connection process is required to understand the way the connection is made in the AMEzy software.

CHAPTER THREE

AMEzy SOFTWARE DEVELOPMENT AND DESIGN

3.1 THE BACK-END DATABASE

The Administration Made Easy Software (AMEzy) uses interfaces created in Visual Basic 6.0 as the front-end and a database created with Access 2000 as the back end. The database used is saved as **FUTdb.mdb** in the "C:\Common Files" folder of the desktop computer to be used. (If saved in another location e.g. over a network, the network path would have to be supplied before the program is executed).

3.2.1 STRUCTURE OF THE DATABASE AND TABLES

The database consists of 10 tables namely: AllStudents, CarryOver, CourseReg, OfferedCourses, PersonalRes, Results, Session1, TempTable, TempTable1 and Users.

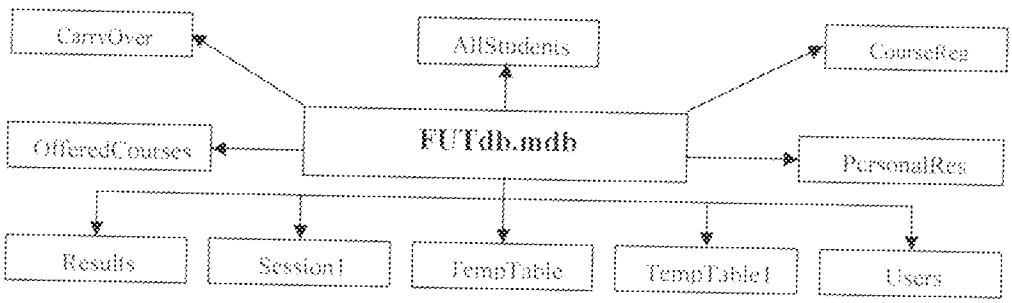


Figure 3.1: *The Database and its Tables*

Each of these tables in turn is made up of several fields having several data types depending on the nature of the data to be entered. A brief description of each field is also given in the table design view for ease of recognition. A detailed list of all the

tables, fields, data types and descriptions is given in Appendix 1. In brief, the following table describes what each database table is used for:

Table in Database	Description
AllStudents	This table contains general information about all students as filled in the students' information form.
CarryOver	This table stores a list of all student carry overs until the students involved pass the courses carried over.
CourseReg	This table keeps a record of which courses have been registered by students.
OfferedCourses	This table contains a list of all the courses offered by students in the Electrical/Electronics department.
PersonalRes	This table keeps record of each individual's semesterial points and aggregate points.
Results	This table stores the grades obtained by students in the courses registered for.
Session1	This table stores the sessional date.
TempTable	This table is a temporary table used to store an individual's data till it is fully processed and moved to its destination.
TempTable1	This is also a temporary table used to store data.
Users	This table stores the usernames and passwords to access the AMEizy software programs.

Table 3.1: *Table showing a brief description of tables existing in the FUTdb.mdb database*

3.2.2 RELATIONSHIPS BETWEEN TABLES

Apart from being in the same database, some of the tables have links known as relationships that enables data to be accessed from the tables simultaneously. This relationship always involves linking two tables on their common fields. The link between these two tables could be either of the following:

- One to One (represented as 1 : 1);
- One to Many (represented as 1 : ∞);
- Many To Many (represented as ∞ : ∞);

Two major relationships are used in this database. The first one is a One to One relationship between the "RegNo" fields of both the `PersonalRes` and `Results` tables as both fields are unique in both tables. The second is a One to Many relationship which exists between the "RegNo" field of the `PersonalRes` table and the `AllStudents` table.

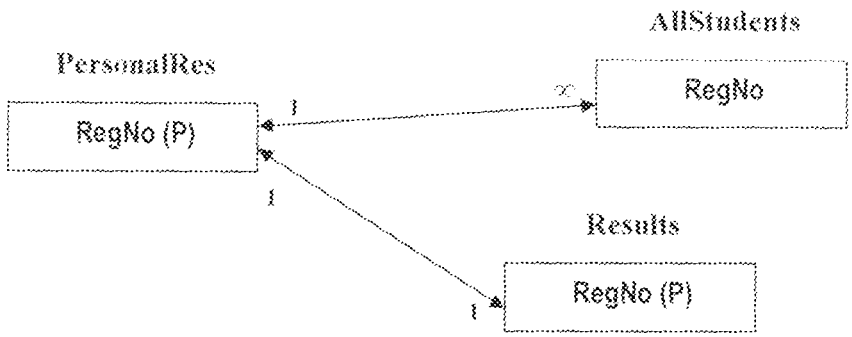


Figure 3.2a: *Diagram illustrating the relationship between the PersonalRes, AllStudents and Results tables*

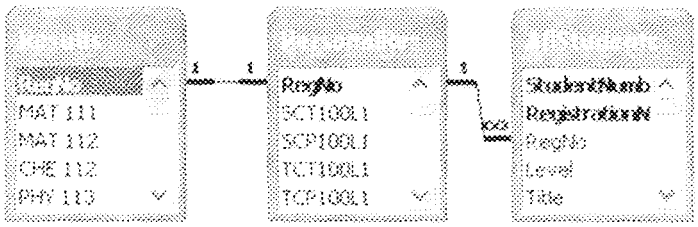


Figure 3.2b: *Pictorial representation of the existing relationship between the PersonalRes, AllStudents and Results tables, as shown in Microsoft Access 2000*

3.2.3 CONSTRAINTS, REFERENTIAL INTEGRITY AND SECURITY

Constraints in RDBMS vocabulary refers to a restriction placed on a field or record to ensure or verify correct entries into that row or column. Constraints help to ensure data integrity and are of numerous types. One very important constraint used in

Access is the 'Primary Key' constraint. A 'Primary Key' is a field or a group of fields that uniquely identify records in a table. It ensures that each record is unique. Every table can have only one primary key and no two records in that table can have the same primary key. The 'Primary Key' fields of each table appear in bold and can be distinguished pictorially by using a **(P)** notation next to the field name. A field in a table that refers to a primary key field in another table is known as a 'Foreign Key'

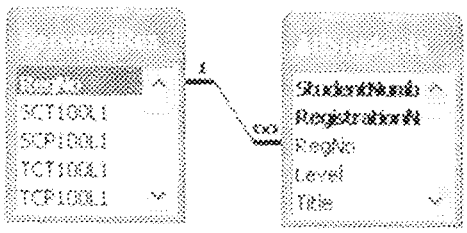


Figure 3.3: *Pictorial view of the Primary Key-Foreign Relationship used between the PersonalRes and AllStudents tables in the FUTdb.mdb database*

RDBMSs like Microsoft Access also provide a feature called 'Referential Integrity' which is a set of rules, which ensure that every value that is entered into the 'Foreign Key' field is validated to check that it exists in the 'Primary Key' field of the corresponding table. 'Referential Integrity' also specifies that when a user deletes or modifies a record from a table, any records that are linked to the deleted record will also be deleted or modified accordingly. This helps to ensure consistency of data and should be taken into consideration when interfacing with external software as in the case of AMEzy 1.0.

Also for security reasons the original database created in Access 2000 is converted to Access 97 format for security reasons. The Access 97 format not only produces a compact database but also produces a read only database which cannot be edited unless converted back to its original format. Alternatively, the database can be protected with a password set by the database administrator.

3.3 THE FRONT-END SOFTWARE

The front-end part of AMEzy uses interfaces created in VB6. This part of the software is divided into four representing the four basic administrative tasks to be automated in the department. These tasks are: Student Registration departmentally, Identification Card Creation, Course Registration and Result Compilation. Before an individual can access any of the programs for these tasks, he/she has to be given a user name and password by the database administrator, which has to be stored in the 'User' table of the FUTdb database. This is done for security reasons.

3.2.1 THE STUDENT REGISTRATION PROGRAM

The Student Registration Program is responsible for registering students departmentally. Its operation is relatively easy. The basic steps involved here are:

(i.) The authorized personnel logs in.

(ii.) He/She selects one of three basic options to proceed:

- **"Register A Student"**: Choosing this option enables the user to input student details and register the student in the department.

- **"View Registration Details"**: This displays details of all registered students in either of two views: Tabular View or Individual View.

- **Update Registered Students"**: This option loads an update form which could be used to alter information for already registered students;

(iii.) Also the user may decide to perform alternative tasks e.g. automatically update the school session and student information, print reports or search for student(s) based on certain criteria, export student information etc;

(iv.) The user exits.

FLOWCHART:

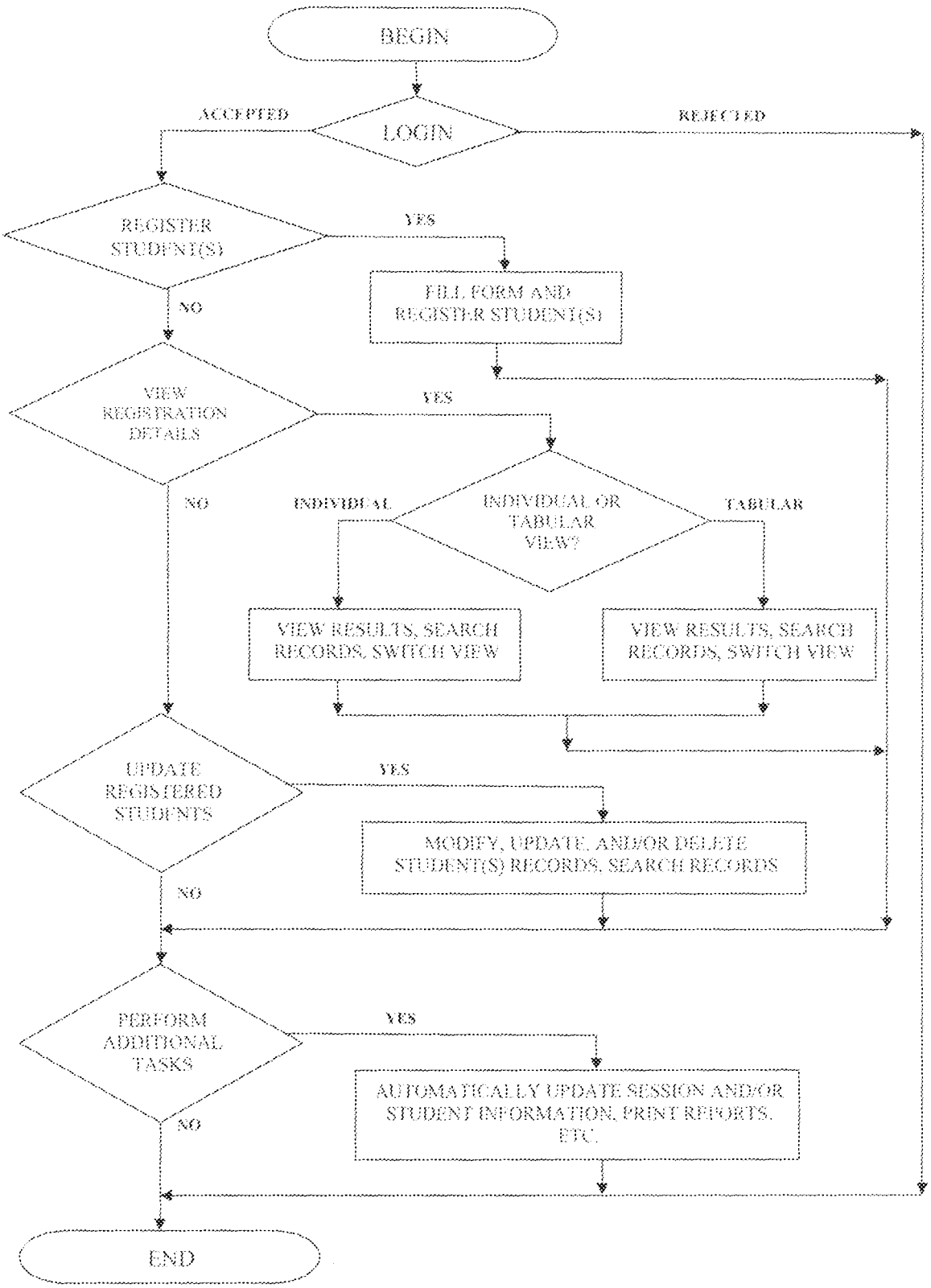


Figure 3.4: *Flowchart of the Student Registration Program*

Software Operation:

When executed the software first displays a splash screen for about 2 seconds, after which the login form is displayed. Here an authorized user is at liberty to select his user name, type (or change) his/her password and gain entry to the main program. Once authorized, the user must agree to the terms on the next form before continuing. On entry a welcome screen appears and one of three options as regards student registration can be chosen.

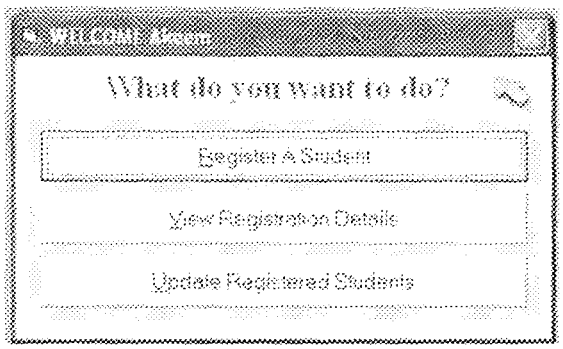


Figure 3.5: *Pictorial View of the Student Registration Welcome Screen*

- Registering Students:

To register a student, the "Register A Student" button is clicked and the student details are filled out on the Student Registration Form. Certain validations have been placed on the form to guide the user and mininize errors. Also the student number in the database is automatically generated so that a record count of students in the department is monitored. On completion of the form, the "Register Student" button can be clicked to register the student and move to the next student.

- Viewing Student Details:

Student details can be viewed in an Individual Student Detail Viewer by clicking on the "View Registration Details" option. Here the user is presented with two views: Tabular or Individual. The Tabular view displays a tabular result of all registered details in the database while the Individual view displays all results one student at a time. The user can switch views

at will by clicking on the "Switch View" Button. Also students can be searched for using the "Search for Records" button on both views, and displayed in any of the two available views.

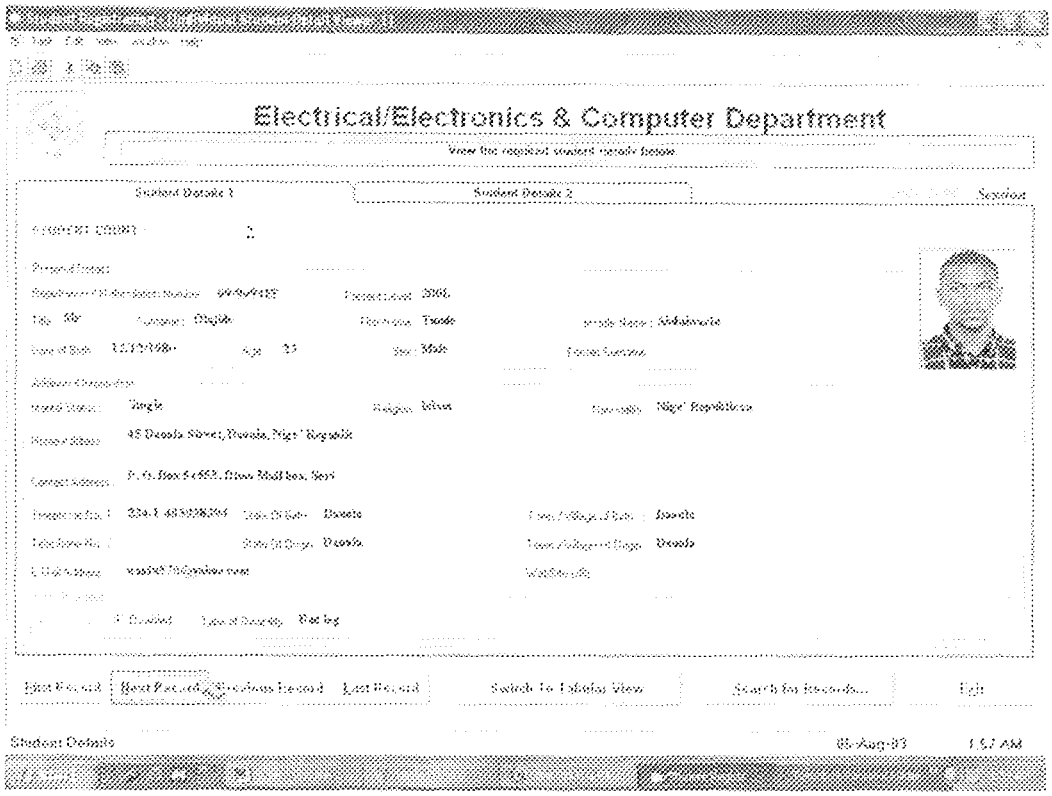


Figure 3.6: *Pictorial View of an Individual Student Detail Viewer Screen*

- Update Registered Students:

The Update form is very similar to the registration form except for few button inclusions e.g. Modify Student Details, Update and Delete Record buttons. Here student records can be modified or deleted as the case may be.

- Other tasks:

Apart from the basic tasks highlighted, other tasks that can be performed here include:

- Update ages and levels of all students automatically;
- Update the school session to the present session;
- Print out the completed student information form for registered students.

3.2.2 THE IDENTIFICATION CARD WIZARD

The Identification Card Wizard is the part of AMEzy Suite which can be used to generate identification cards (ID Cards) for all registered students in the department. Unlike the registration program, it requires little or no input from the user to generate the cards, as the cards are generated based on a simple template. It should however be noted that the wizard can only generate identification cards for students that have been registered using the Student Registration Program. This is to say that the student information for which the ID Card is to be generated, must exist in the database otherwise the card cannot be generated.

The steps involved in using this wizard are:

(i.) The authorized personnel logs in.

(ii.) He/She selects one of three options to proceed for the card generation:

- **"All Registered Students"**: Choosing this option automatically generates identification cards for all registered students existing in the database.
- **"One Student"**: This option requires the matriculation/graduation number (e.g. 98/4567EE) of the student to be entered before generating the card for the student.
- **"Multiple Students"**: Choosing this option loads another interface which allows the user to choose based on two options: "Multiple Registration Numbers" and "Other Criteria" e.g. age, sex, level etc. After a choice is made, the required information is entered and the cards are generated accordingly.

(iii.) The user exits.

FLOWCHART:

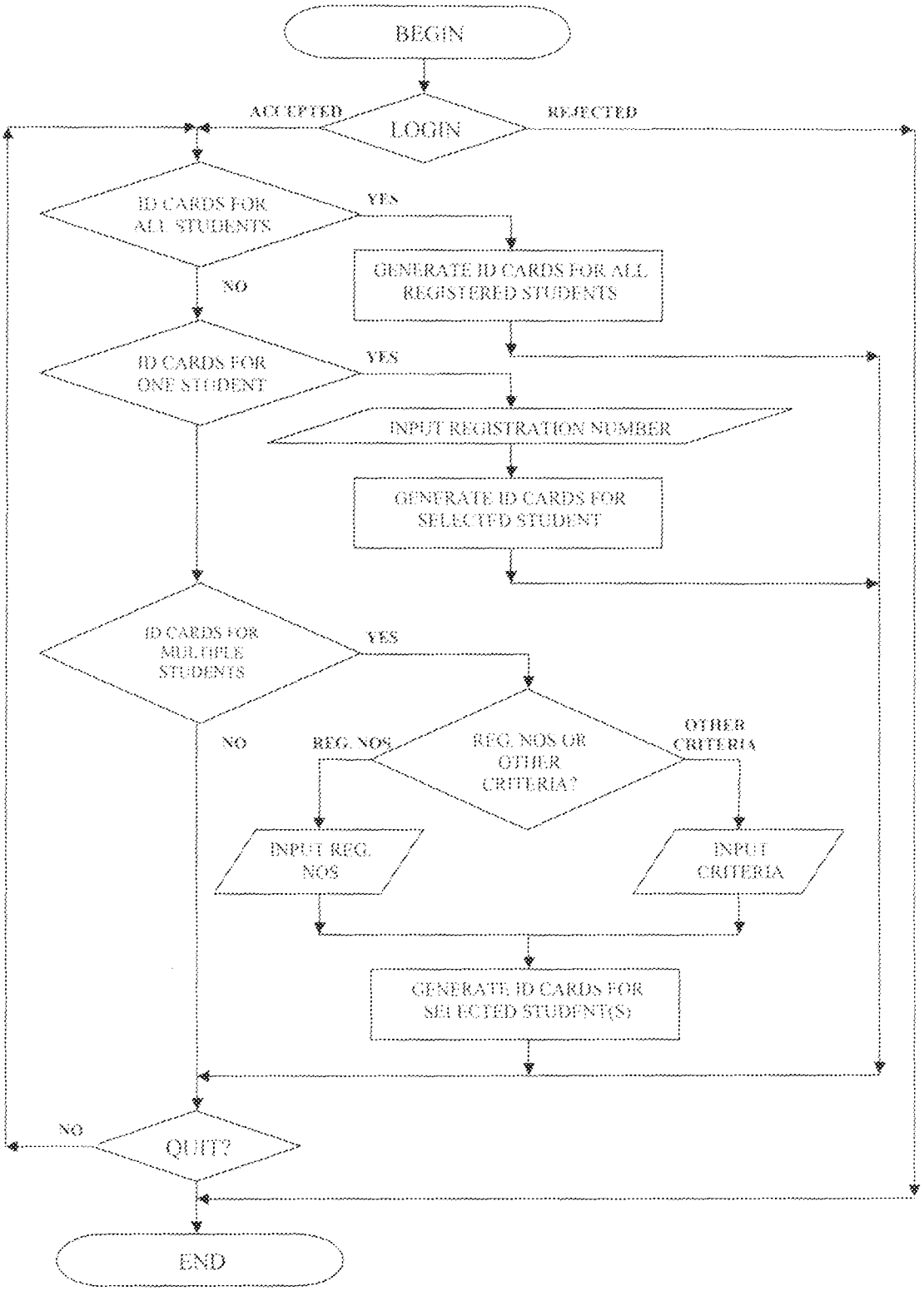


Figure 3.7: *Flowchart of the Identification Card Wizard Program*

Software Operation:

After displaying the splash screen the user is required to login and agree to the terms of the programmer before starting the wizard. On entry a welcome screen appears and one of three options as regards the students to generate the identification cards for is to be chosen.

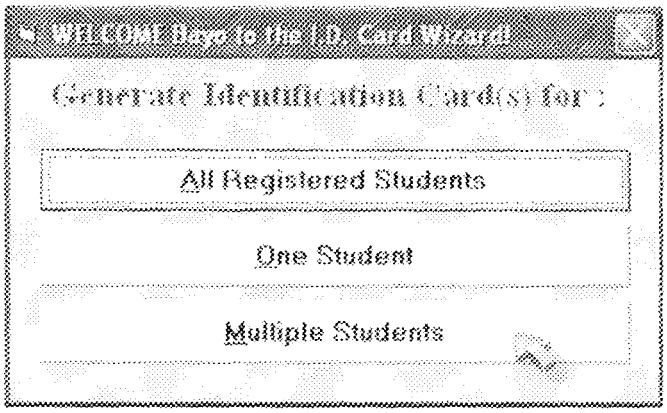


Figure 3.8: *Pictorial View of the I. D. Card Wizard Welcome Screen*

- Generating Identification Cards for All Students:

Identification cards for all students can be generated at the click of the "All Registered Students" button. However the form results shown on the screen are not the printable results. They only show a preview of what the results will look like. To print the identification cards, the "Print Page" button has to be clicked. The report can then be previewed and printed on paper (as it will appear when printed) and then sent to the printer to produce a hard copy.

- Generating Identification Card for One Student:

AMEzy I. D. Card Wizard also offers the user the ability to generate identification cards for an individual student. This option can be used where one identification card is to be printed at a time. Once this option is chosen, the user must supply the full matriculation / registration number of the student concerned (e.g. 99/9995EE). When this is done, the program automatically searches the database of registered students and informs the user if the

student was located or not. If not found, the user can supply another number. If found however the user can preview the card before finally printing the result.

Figure 3.9: *Pictorial View of a student's identification card generated with the AMEzy Identification Card Wizard*

- Generating Identification Cards for Multiple Students:

When this option is chosen, another interface comes up with two separate options. The first of those options assumes that the user has a list of students' registration numbers for which he/she wants to generate identification cards. If this option is selected, a form opens which allows the user to insert those numbers and cross check the student names when the program matches the names with the registration numbers. At completion the user clicks on the "Print Page" button to preview and finally print the results.

The second option is used when the user needs to search for the student's profile with certain criteria e.g. Registration Number, Name, Level, Sex, Age etc. When the search criteria are filled and the search is performed, the wizard automatically uses the results to generate previews which can in turn be printed.

3.2.3 THE COURSE REGISTRATION PROGRAM

This program is responsible for registering courses to be offered by students semesterially. As shown in Appendix 1.2, all available courses offered by students in the department are stored in the CourseReg table of the FUTdb.mdb database. The Course Registration Program connects to this database to load semesterial courses for registration. Therefore in the event of future changes all that needs to be modified is the database and not the program. The program also connects to the CarryOver table shown in Appendix 1.5 for carry over students.

The steps involved in registering students for courses using this program are:

(i.) The authorized personnel logs in.

(ii.) He/She selects one of three options to proceed for the course registration:

- **"Fresh Registration Of Student(s) Courses"**: This option registers students taking courses for the first time.

- **"Add / Drop Courses For Student(s)"**: For students who have formely registered courses but have to drop one or all of them for one reason or the other, this option can be selected.

- **"Register Carry Over Courses For Student(s)"**: Choosing this option enables the user to register courses which are presently being carried over by the student or students concerned.

(iii.) The user exits.

FLOWCHART:

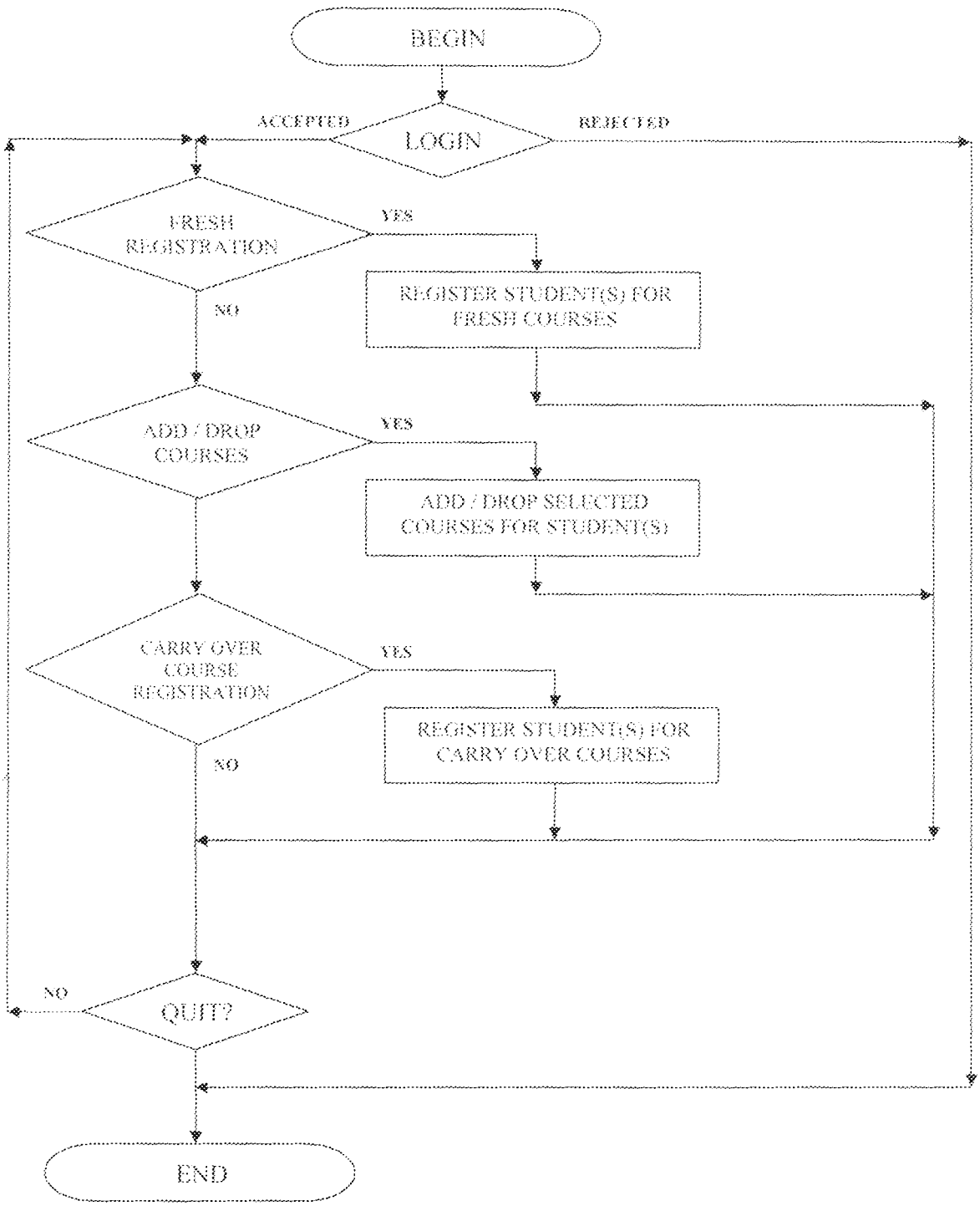


Figure 3.10: Flowchart of the Course Registration Program

Software Operation:

Like the previous programs, the user has to login and agree to the programmer's terms before gaining access to the welcome screen. There he/she is confronted with 3 options as shown below.

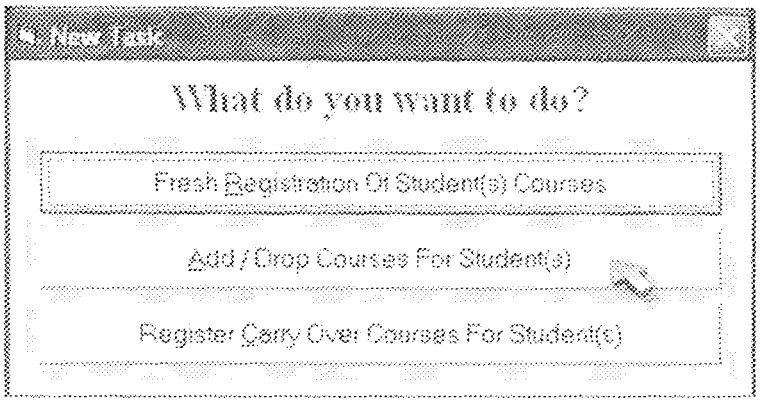


Figure 3.11: *Pictorial View of Welcome Screen for the AMEzy Course Registration Program*

- Fresh Registration Of Courses:

This option is used when students are registering courses for the first time. This registration is made easy as all courses have been stored in the database and are automatically grouped semesterially for ease of registration.

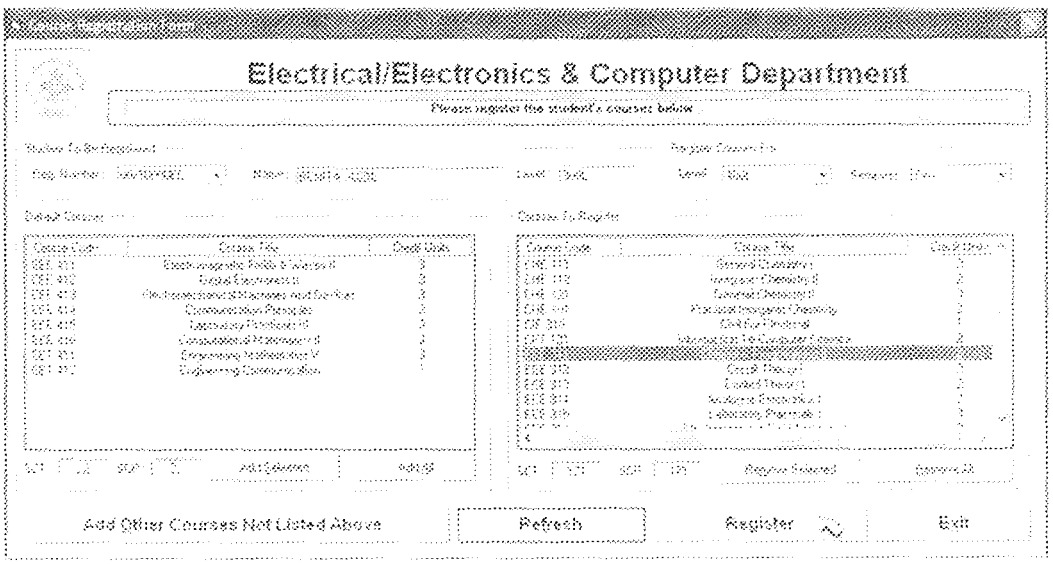


Figure 3.12: *Pictorial View of the Course Registration Form for fresh course registration*

Apart from the ease by grouping, the program also sets certain validations to ensure rules are enforced. For instance the student is not allowed to register more than 24 credit units for any semester. If however the user tries to register more than that, the program gives an error message and confirms the action. At the user's discretion, he/she can decide whether to allow this or not. Another example is that of registration during Industrial Attachment (400 Level Second Semester). Since all students are expected to be on industrial attachment, no courses are provided by the software for registration for that semester. If however a student is allowed to register for that semester, the "Add Other Courses" button can be used to do this. Many other validations of this sort exist throughout the AMEzy program.

- Adding / Dropping Courses:

In a case where a student has over registered (i.e. exceeded his maximum credit load unit), under registered or feels he/she cannot handle a certain credit load, AMEzy Course Registration Program provides the "Add / Drop Courses For Student(s)" option for that. When this part of the program is entered the student's number is supplied and his courses for that semester are shown. He/She can then add or drop courses as the case may be.

- Registering Carry Over Courses:

Carry over courses are also considered as a separate section in the program. Any student carrying over a course is supposed to register it like any other course being offered. As a check measure, the Course Registration program checks and displays courses carried over by the concerned student so as to ensure that the student is actually allowed to carry over the course(s). The course can then be registered and the student is given an entry in the CarryOver table of the FUTdb database. This entry will remain there until the individual passed the course or drops it.

3.2.4 THE RESULT COMPILER PROGRAM

The Result Compiler Program is the last part of the AMEzy Suite and probably the most tasking as it involves storing, converting, calculating, display and printing of students' status and results. However the beauty of the program is that only entry of the students' grades is done manually. The rest of the work is handled by the program.

The steps involved here are as follows:

(i.) Authorized personnel logs in.

(ii.) On entry, the user selects an option depending on what section he/she is interested in. There are 3 sections namely: "Fresh Results", "Carry Overs" & "View Grade Points".

- **"Fresh Results" Section:** This section provides the options to input results (based on the semester or on the courses), view results (semesterially view or tabular view) and/or update results based on the semesters.

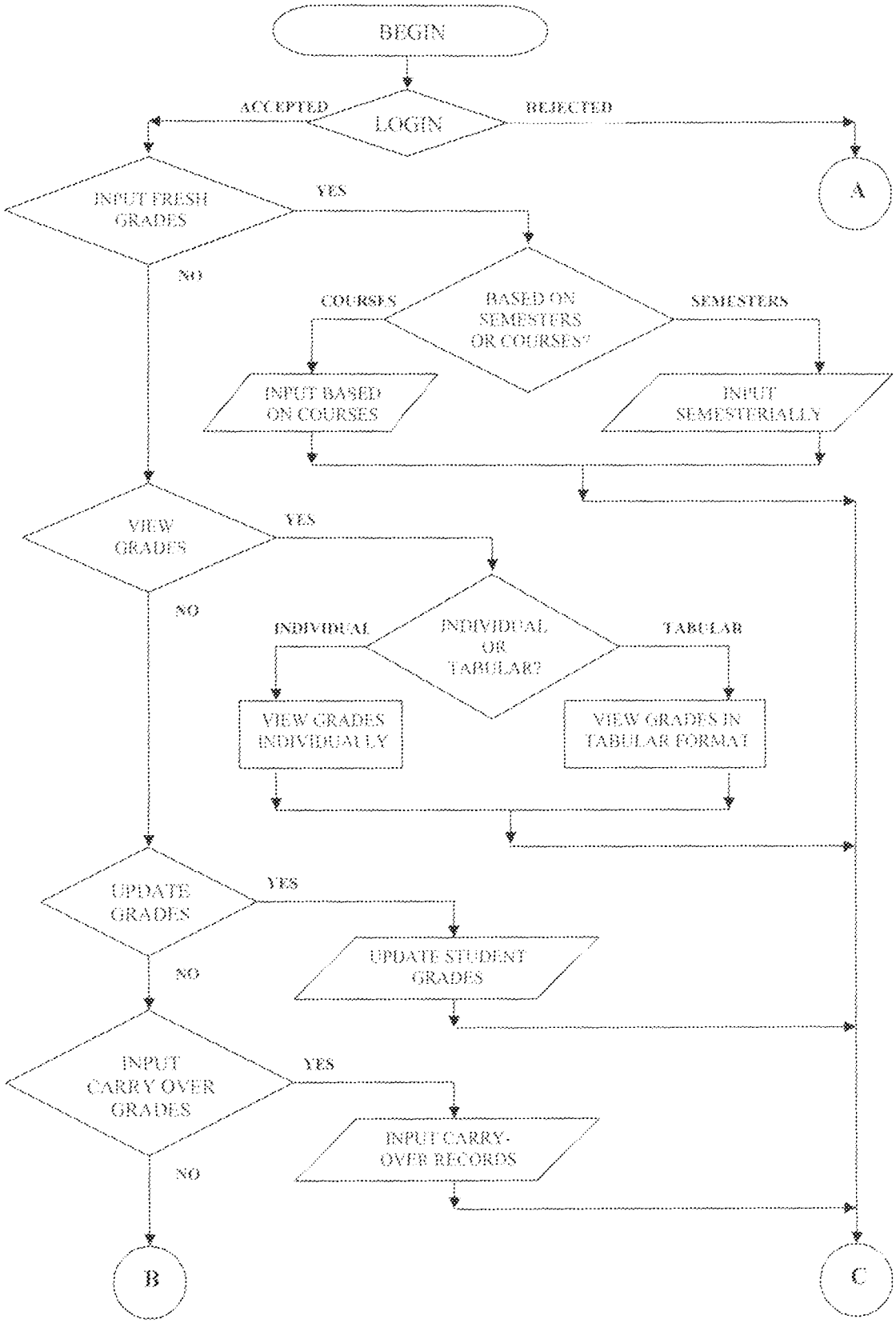
- **"Carry Overs" Section:** In this section the user has the opportunity to store carry over results of concerned students.

- **"View Grade Points" Section:** Choosing this option enables the user to view the grade points of all students in two views: Individual View or General / Total View. However before viewing the user is advised to run the Automatic Student Result Updater so as to obtain the correct results when viewed.

(iii.) Also the user may decide to perform additional or alternative tasks e.g. automatically update student results, print reports for student(s) based on individuals, levels or semesters, etc

(iv.) The user exits.

FLOWCHART:


BEGIN
LOGIN
ACCEPTED
REJECTED
A
INPUT FRESH GRADES
YES
NO
BASED ON SEMESTERS OR COURSES?
COURSES
SEMESTERS
INPUT BASED ON COURSES
INPUT SEMESTERIALLY
VIEW GRADES
YES
NO
INDIVIDUAL OR TABULAR?
INDIVIDUAL
TABULAR
VIEW GRADES INDIVIDUALLY
VIEW GRADES IN TABULAR FORMAT
UPDATE GRADES
YES
NO
UPDATE STUDENT GRADES
INPUT CARRY OVER GRADES
YES
NO
INPUT CARRY-OVER RECORDS
B
C

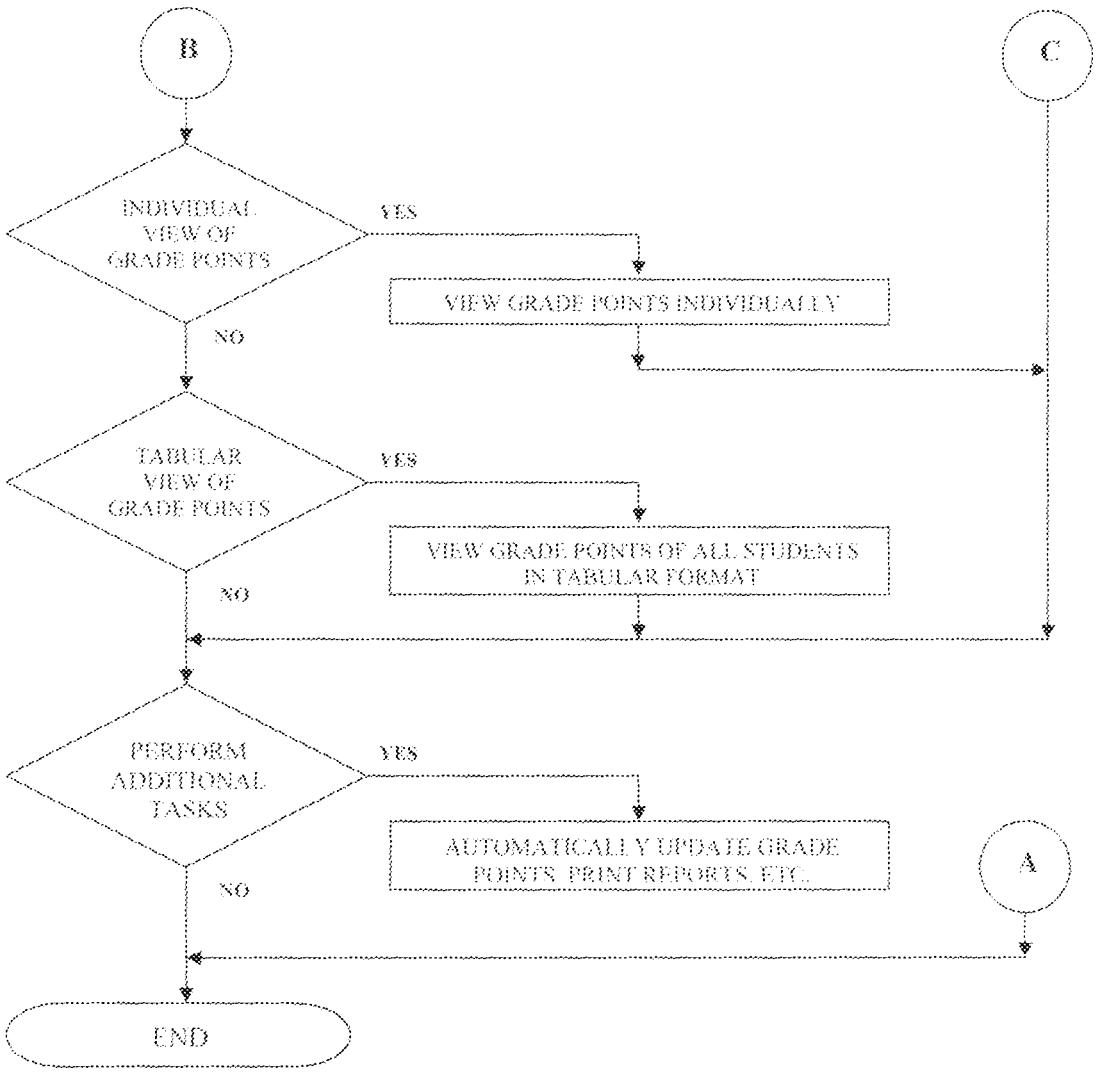


Figure 3.13: *Flowchart of the Result Compiler Program*

Software Operation:

The user, as in the other cases, has to log in and accept the program terms before coming across a welcome screen with a couple of tasks. The tasks that can be performed here include: Storing / Recording, Updating and Viewing of Results.

- Storing / Recording of Grades:

Students' grades can be stored according to semesters or according to courses. For courses carried over, there is also a special form for storing the student's present

or new results. Before results are accepted the program checks to make sure that the course has been registered and that no grade exists initially. This program also ensures that the semester for industrial attachment is highlighted so that results are not wrongly filled into that semester. All these checks help prevent accidental recording of grades and ensure correctness in results.

- Updating of Grades:

The Result Compiler allows the user to alter already stored grades in the database. This is very important as grades can be accidentally entered or filled in for the wrong student. This program can be used to correct such errors.

- Viewing Grades / Results:

In this program, the user can view grades entered as well as calculated / computed grade points for all registered students. The individual view of grade points also allows the user to see the status of the student e.g. First Class, Second Class (Upper) etc.

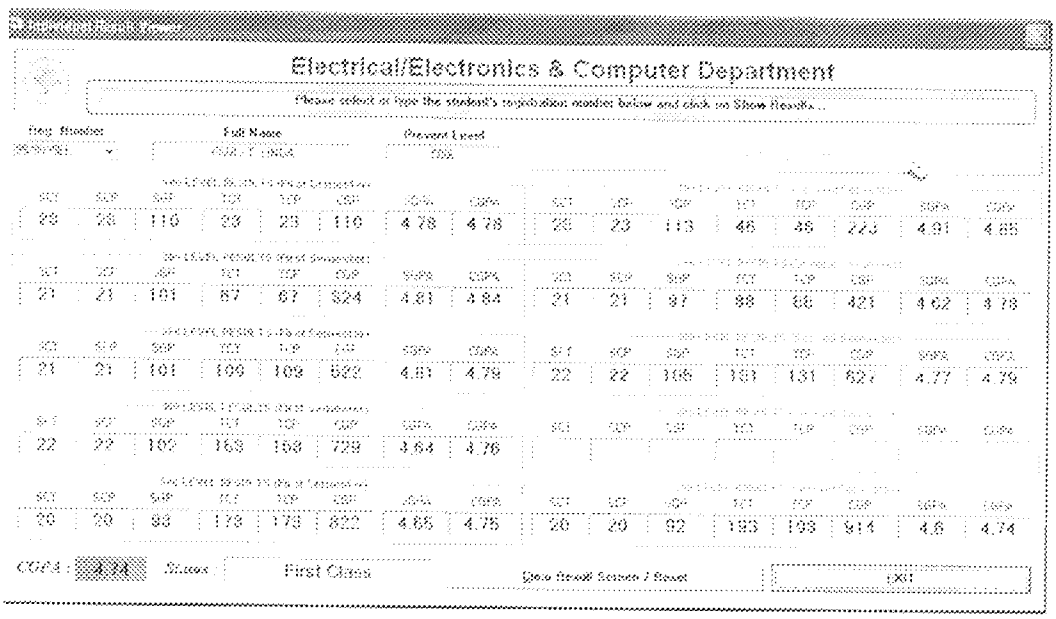


Figure 3.14: *Pictorial View of an Individual Result Viewer Form*

- **Other tasks:**

Apart from the basic tasks highlighted, other tasks that can be performed in this program include the following:

- Automatic update of all results by running the 'Automatic Result Updater'.
- Printout of student results semesterially and individually.
- Exporting of student results to other file formats etc.

3.3 COMPILING, PACKAGING & DEPLOYMENT

After all the programs have been written and tested, they undergo three final processes. The first of these, which is done in the Visual Basic Integrated Desktop Environment (VB IDE), is compilation into executables (i.e. EXE). This enables the program to be run as an executable on any microcomputer.

The next two steps (Packaging and Deployment) can be performed with an add-on in the VB IDE or with a stand-alone component of the Visual Studio Suite called 'The Packaging and Deployment Wizard'. Packaging is the act of creating a package of one or more cabinet files that contain the compressed project files, dynamic link libraries (DLLs), and any other files the user must have to install in order to run the application. This creates the installation program for the software.

After packaging, the package has to be deployed using the same wizard. Deployment is the process of sending application components and related files from one computer to another or from one location to another so that the components may be available to other users. The components of the AMEzy program in this case are packaged as cabinet files and are then deployed on a compact disc for distribution. This is the final stage after which the software is ready for use.

3.4 THE AMEzy SUITE INSTALLER

Like any other software suite, the AMEzy Suite Installer is just an easy way to run the AMEzy programs that have been compiled, packaged and deployed by Visual Basic. Unlike the other programs previously highlighted the AMEzy Suite Installer is only used to launch and install the deployed programs. It is not a functional part of the AMEzy Suite. It is loaded automatically when the compact disc carrying the AMEzy Suite is inserted into the microcomputer. A sample of the interface for the installer program is shown in Appendix 2.5. By clicking on the appropriate options on the interface shown, the required setup programs are executed, installed and can be opened like any other installed software.

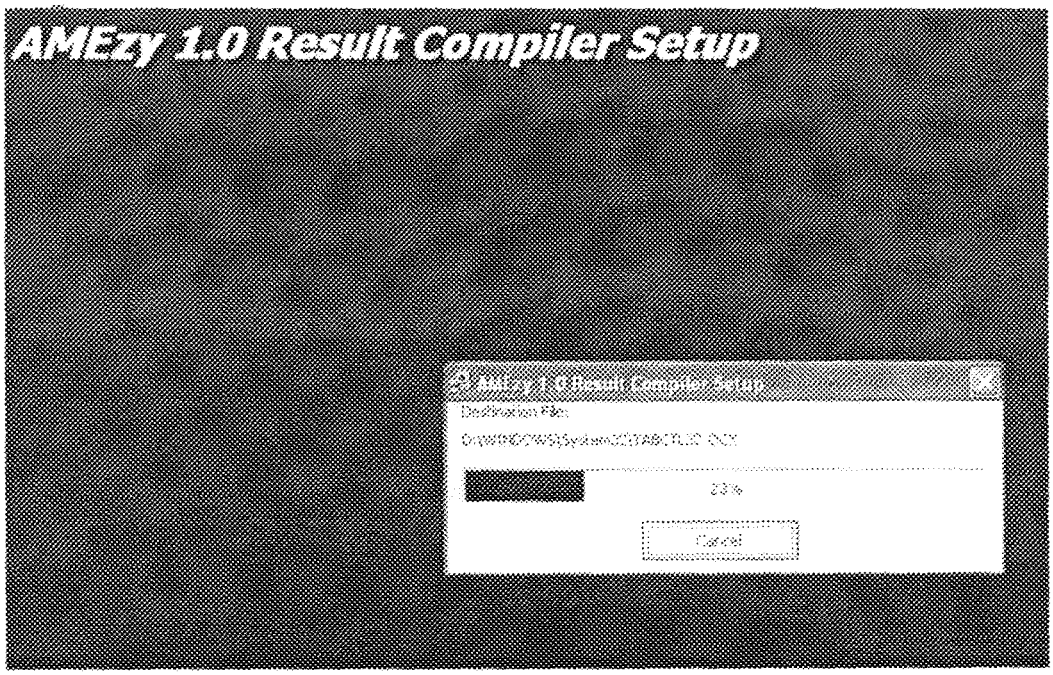


Figure 3.15: Pictorial Clip of the installation process of the Result Compiler Program

CHAPTER FOUR

TESTS, RESULTS AND DISCUSSION OF RESULTS

This chapter contains the various test operations carried out in order to ascertain the reliability and efficiency of the software.

4.1 TESTS

Testing is a major part of a software development life cycle (SDLC). The two main types of tests carried out here are Alpha Test and Beta Test.

4.1.1 ALPHA TEST

Alpha test is always the first test done on software. It is normally done by the developer. It involves about three stages:

- **Unit Testing:** This is the testing of individual components of the system by the author to see if they are functioning properly. In this case each of the four programs is considered to be a unit and all their individual components (e.g. combo boxes, command buttons, shortcut keys etc.) are tested one after the other to ensure that they are functioning correctly.

- **System Testing:** Integrated testing of the system with emphasis on interfaces between configuration items and modules is the essence of System Testing. The emphasis is on testing the functionality of the system, its security, recovery and restart procedures. Here all four programs are tested together to ensure proper functioning.

- **Regression Testing:** When errors are found during testing, they are corrected and tested and testing proceeds as planned. While this happens care has to be taken to ensure that new errors are not introduced into the system. This is the essence of Regression Testing.

4.1.2 BETA TEST

Unlike alpha testing, this type of test is usually done by users outside the development life cycle. Based on the errors found in the system, the users have to accept or reject the system. This type of testing is therefore also known as Acceptance Testing. This is the last round of testing and must therefore be done in depth.

4.2 CASE STUDIES

The best way to illustrate the outcomes of the tests done is by using real life examples (case studies). Here the results from manual methods will also be compared with the results of the software.

CASE 1:

Mrs. Linda Fanis has just graduated from the department and requests for a printout of her grade points at the end of her degree for an interview holding the next day. Her grades had been manually recorded against her courses as follows:

100 LEVEL			200 LEVEL			300 LEVEL		
FIRST SEMESTER			FIRST SEMESTER			FIRST SEMESTER		
COURSE CODE	CREDIT UNIT	GRADE	COURSE CODE	CREDIT UNIT	GRADE	COURSE CODE	CREDIT UNIT	GRADE
CHE 111	3	A	EET 211	3	A	EET 311	3	A
CHE 112	2	A	EET 212	2	A	EET 312	2	A
GST 104	2	A	EET 213	2	A	EEE 311	3	A
GST 110	3	A	EET 214	2	A	EEE 312	3	A
MAT 111	3	A	EET 215	2	B	EEE 313	2	A
MAT 112	3	B	EET 216	2	A	EEL 314	2	C
PHY 100	2	A	EET 217	3	A	EEP 315	3	A
PHY 113	3	A	EET 218	1	C	EEP 316	3	A
TCD 111	1	C	EET 219	3	A	CHE 318	1	A
WKS 110	1	A	EET 231	1	A			
SECOND SEMESTER			SECOND SEMESTER			SECOND SEMESTER		
CHE 121	3	A	EET 221	3	B	EET 321	3	A
CHE 191	2	A	EET 222	2	A	EEE 321	3	A
CPE 121	2	A	EET 223	2	A	EEE 322	2	A
GST 103	2	B	EET 224	2	A	EEP 323	3	A
GST 121	2	A	EET 225	2	A	EEE 324	2	A
MAT 121	3	A	EET 226	2	A	EEE 325	2	A
MAT 127	3	A	EET 227	3	A	EEP 326	2	A
PHY 123	2	A	EET 228	1	A	EEE 327	2	A
PHY 126	3	A	EET 229	3	A	EEE 328	2	A
TCD 121	1	A	EET 241	1	D	MEE 352	1	B

400 LEVEL			500 LEVEL					
FIRST SEMESTER			FIRST SEMESTER					
COURSE CODE	CREDIT UNIT	GRADE	COURSE CODE	CREDIT UNIT	GRADE			
EET 411	3	A	EET 512	3	B			
EET 412	3	C	EFE 511	3	A			
EFE 411	3	A	EFE 512	2	B			
EFE 412	3	B	EFE 513	2	A			
EFE 413	3	A	EFE 514	2	A			
EFE 414	3	A	EFE 516	2	A			
EFE 415	3	A	EFE 517	2	A			
EFE 416	3	A	EFE 526	2	A			
			EFE 527	2	B			
SECOND SEMESTER			SECOND SEMESTER					
S. I. W. E. S. PROGRAMME			EFE 521	2	A			
			EFE 522	1	A			
			EFE 523	1	A			
			EFE 524	2	A			
			EFE 525	8	B			
			EFE 528	2	A			

Table 4.1: *Table showing grades recorded for the Mrs. Linda Fians*

Manual computations indicate the following grade points:

	100 LEVEL		200 LEVEL		300 LEVEL	
	FIRST SEMESTER	SECOND SEMESTER	FIRST SEMESTER	SECOND SEMESTER	FIRST SEMESTER	SECOND SEMESTER
SCT	23	23	21	23	23	22
SCP	23	23	21	23	23	22
SGP	110	113	101	99	101	109
TCT	23	46	67	88	109	131
TCP	23	46	67	88	109	131
CuP	110	223	324	423	524	633
SGPA	4.7826	4.9130	4.8095	4.7143	4.8095	4.9545
CGPA	4.7826	4.8478	4.8358	4.8068	4.8023	4.8321

	400 LEVEL		500 LEVEL	
	FIRST SEMESTER	SECOND SEMESTER	FIRST SEMESTER	SECOND SEMESTER
SCT	22		20	20
SCP	22		20	20
SuP	105	SIWES	83	92
TCT	183		173	193
TCP	183	PROGRAMME	173	193
CGP	738		821	913
SGPA	4.7727		4.15	4.6
CGPA	4.8235		4.7457	4.7306

Table 4.2: *Table showing manually computed results for Mrs. Linda Fians*

As a test the AMEzy software is required to accept her grades, compute her results and produce a printout of her result. The final printout is to be compared with the manual computation.

CASE 2:

The senate of the university, in investigating the cause of the last violent demonstrations embarked upon by students, requests all available information about two students in the department having registration numbers 99/9994EE and 99/9997EE. The senate has also requested that a printed copy of their identification cards should also be attached to their information forms.

CASE 3:

Mr. Chinedu Sanusi (00/10009EE), a student of the department, fell sick shortly before exams and was unable to write his continuous assessment test for EET 321 (Engineering Mathematics IV). As he was not given a second chance, he felt he could not cope with the course as part of his 300 Level Second Semester courses and so decided to drop the course.

Thus his result at the end of his 300 Level was reflected as shown below:

300L SECOND SEMESTER						
COURSE CODE	CREDIT UNITS	GRADES			VALUES	
EET 321	3	F		SCT	19	
EEE 321	3	A		SCP	19	
EEE 322	2	A		SGP	76	
EEE 323	2	A		TCT	131	
EEE 324	2	A		TCP	131	
EEE 325	2	A		CGP	630	
EEE 326	2	A		SGPA	4.1579	
EEE 327	2	A		CGPA	4.8697	
EEE 328	2	A		REMARKS	L.C.S.	
MEE 329	1	B				

Table 4.3: *Table showing computation of results for Mr. Chinedu Sanusi*

He decided to re-take the course as part of his 500 Level Second Semester courses and registered it for 500 Level second semester. At the end of his degree, he passed the course with a distinction and was awarded a second class upper. Chinedu disputed this as his manual calculations showed a first class. Assuming his grades were the same as Mrs. Fanis in CASE 1 (except that he scored E's in EET 321 in and EEE 525), use the AMEzy software to prove or disprove Chinedu.

4.3 RESULTS OF TESTS USING CASE STUDIES

The results obtained from the tests using the case studies are as follows:

CASE 1 RESULTS:

As AMEzy required, Mrs. Linda Fanis was registered departmentally, and likewise she was registered for all the required courses. As her grades had already been stored, the software handled the rest. A printout of her results was eventually made which tallied with the manually computed results. It was then handed over to her on the same day of request. A sample of that printout is shown in Appendix 4.1.

CASE 2 RESULTS:

With the help of the AMEzy search engine, the matriculation numbers of both students was used to search for and print out their information forms and identification card samples. This was done in the record time as the students had initially been registered departmentally. Samples of these are shown in Appendix 4.2.

CASE 3 RESULTS:

By dropping EET 321, Chinedu's credit unit load was reduced to 19 units and his CGPA became 4.83 (as shown by AMEzy Software). Further computation till his final result gave a CGPA of 4.54, which is inevitably a first class. Observation showed that his result was probably miscalculated as he was scored an *F* for EET 321 in 300 Level and his CGPA was therefore miscalculated. With the help of the AMEzy Software, that has however been resolved as an individual cannot be awarded a grade for a dropped course using the software.

4.4 DISCUSSION OF RESULTS

From the case studies presented previously, it could be seen that the functionality of the AMEzy software can be vastly utilized to solve everyday issues faced by the department.

In the first case the student's results were compiled and released by the click of a few buttons. Also the computerized results tallied with manual results (which are more prone to errors). In the second case, the easy to use search facility provided by AMEzy helped locate required students and print out required information. Contrary to normal semesterial delay in identification card delivery, AMEzy software enables the department to make such identification cards available with little or no effort. Pictures of the required individuals can also be scanned and printed out as part of the card so that there is no case of cutting of pictures. The third case demonstrated the case of adding, dropping and recomputing results with AMEzy software.

In all the results have been favourable and show a high level of output and efficiency from the AMEzy Suite.

CHAPTER FIVE

RECOMMENDATION AND CONCLUSION

5.1 CONCLUSION

With the help of AMEzy 1.0 Suite, departmental tasks have been automated and performed with ease. The software has been used to successfully register students departmentally, process and print their identification cards, register students' courses, compute and print results for all registered students. All these have been accomplished in just a few minutes (unlike the manual method which would take days) with no processing errors and has therefore released reliable and accurate results.

This enables us to conclude that the AMEzy software is the best form of automation for the Electrical / Electronics and Computer Engineering Department.

5.2 RECOMMENDATION

I recommend that the automation of the Department of Electrical / Electronics and Computer Engineering with the Administration Made Easy (AMEzy) software to ease and enhance administrative tasks in the department. Future designs of the software with more functionality should also be encouraged.

I also recommend that other departments in the University be automated with such object oriented software in the long run so as to allow easy sharing of information, data storage, synchronization and communication.

REFERENCES

1. Francis Scheid (1983) "Computers And Programming", Schaum's Outline Series, McGraw-Hill Book Company, Singapore, International Edition.

2. Design Team (2002) "Visual Basic 6.0 Desktop", APTECH WORLDWIDE, U. S. A., First Edition.

3. Gary Cornell (1998) "Visual Basic 6 From The Ground Up", Osborne / McGraw-Hill, U. S. A., First Edition.

4. Design Team, H. O. (1999) "Object Oriented Programming With C++", APTECH LIMITED, India, First Edition.

5. Ravi R. & Shekhar S. (1997) "SOFTWARE ENGINEERING Modules 6 & 7: Software Project Management and Quality Assurance (SPM & SQA)", APTECH LIMITED, India, First Edition.

6 Design Team. (2003) "Object Oriented Analysis and Design With Unified Modeling Language", APTECH WORLDWIDE, U. S. A., First Edition.

7. Design Team (2000) "Database Design With MS Access 2000", APTECH WORLDWIDE, U. S. A., First Edition.

8. John Smiley (2000) "The History Of Visual Basic", Smiley and Associates INC., U. S. A., Revised Edition.

9. Microsoft Corp. (2001) "Microsoft Digital Network (MSDN) Library", Microsoft Corporation. U. S. A, Revised Edition

APPENDIX

APPENDIX 1

SAMPLE STRUCTURES OF THE TABLES IN THE FUTDB.MDB DATABASE

Field	Data Type	Description
StudentNumber (P)	Number	Serial Number of Students (Primary Key)
RegistrationNumber (P)	Text	Registration Number of the Student (Primary Key)
RegNo	Text	Full Registration Number
Level	Text	Level of the Student
Title	Text	Title of the Student
Surname	Text	Student's Surname
FirstName	Text	Student's First Name
MiddleName	Text	Student's Middle Name (if any)
FullName	Text	Student's Full Name
DateofBirth	Text	Date of Birth of the Student
Age	Number	Age of the Student
Sex	Text	Sex of the Student
FormerSurname	Text	Former Surname (if any)
MaritalStatus	Text	Status Of The Student
Religion	Text	Present Religion
Nationality	Text	Nationality
HomeAddress	Text	Home/Permanent Address of the Student
ContactAddress	Text	Contact Address
TelNo1	Text	Telephone Number 1 Of Student
TelNo2	Text	Telephone Number 2 Of Student
StateOfBirth	Text	State of Birth
Town/VillageofBirth	Text	Town of Birth
StateOfOrigin	Text	State of Origin
Town/VillageOfOrigin	Text	Town of Origin
E-mailAddress1	Text	E-mail address of the Student (if any)
WebsiteURL	Text	Website Address (if any)
HealthStatus	Text	Student's Health Status
TypeOfDisability	Text	Type of Disability (if any)
ModeOfEntry	Text	Mode Of Entry Into The University (for Undergraduates only)
ModeOfStudy	Text	Mode of Study
Programme	Text	Programme in the University
YearOfEntry	Text	Year Of Entry
PreviousUniversity	Text	Previous University (if transferred)
HighestQualification	Text	Highest Qualification obtained
InstitutionWhereObtained	Text	Institution Where Qualification was Obtained
SubjectOfFirstDegree	Text	Subject Of First Degree (for Postgraduate Only)
NameOfNextOfKin	Text	Next Of Kin Name
RelationshiptoNextOfKin	Text	Relationship to Next Of Kin
AddressOfNextOfKin	Text	Address of Next Of Kin
TelNoOfNextOfKin	Text	Telephone Number Of Next Of Kin
E-mailAddress2	Text	E-mail Address of Next Of Kin
Sponsor'sName	Text	Name of Sponsor
AddressofSponsor	Text	Address of Sponsor
OtherDetails	Text	Any Other Information

Appendix 1.1: *Structure of the AllStudents Table*

Field	Data Type	Description
CourseCode (P)	Text	Code of the Course (Primary Key)
CourseTitle	Text	Title of the Course
CreditUnits	Number	Credit Units of the Course
Semester	Text	Semester Taken
Level	Text	Level to be taken

Appendix 1.2: *Structure of the OfferedCourses Table*

Field	Data Type	Description
RegistrationNumber	Text	Registration Number of the Student
CourseCode	Text	Code of the Course
Grade	Text	Grade Obtained in the Course

Appendix 1.3: *Structure of the CarryOver Table*

Field	Data Type	Description
StudentNumber	Number	Serial Number of Students
RegistrationNumber	Text	Registration Number of the Student
RegNo	Text	Full Registration Number
Level	Text	Level of the Student
FullName	Text	Student's Full Name
Nationality	Text	Nationality
Sponsor'sName	Text	Name of Sponsor

Appendix 1.4: *Structure of the TempTable Table*

Field	Data Type	Description
Session	Text	Present Session

Appendix 1.5: *Structure of the Session1 Table*

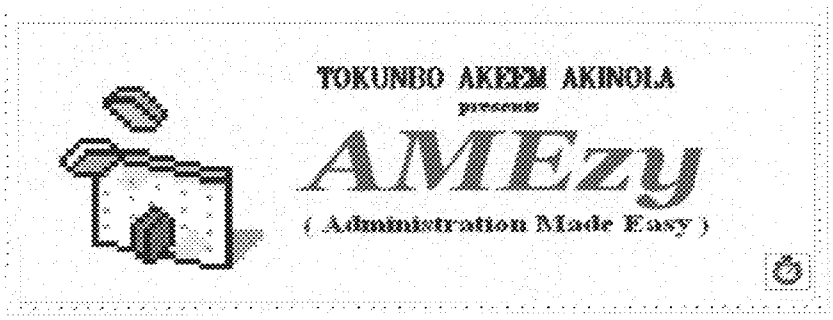
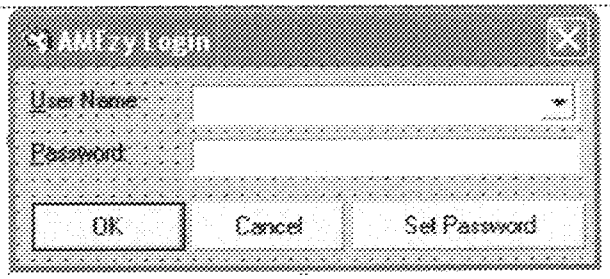
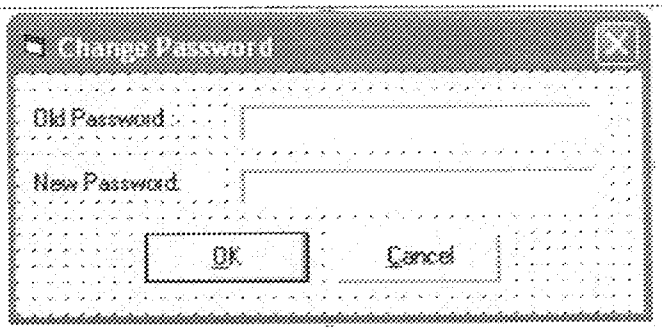
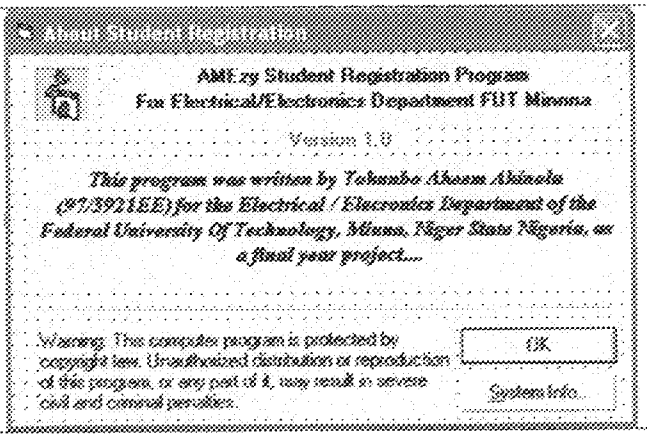
Field	Data Type	Description
UserName (P)	Text	User Name (Primary Key)
Password (P)	Text	Password of users (Primary Key)

Appendix 1.6: *Structure of the Users Table*

APPENDIX 2

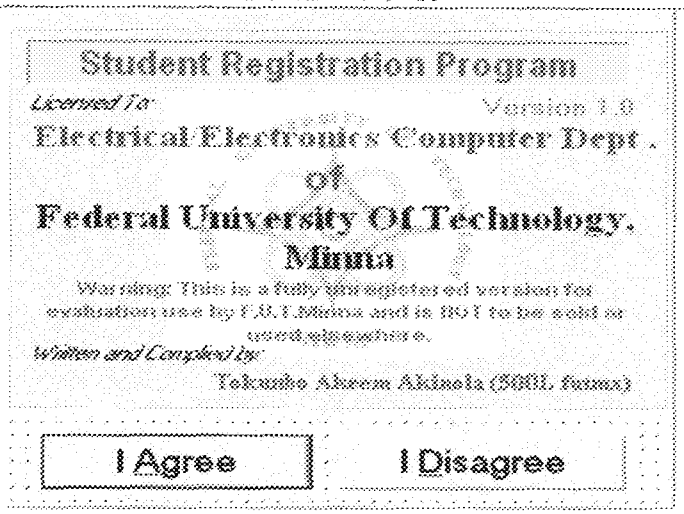
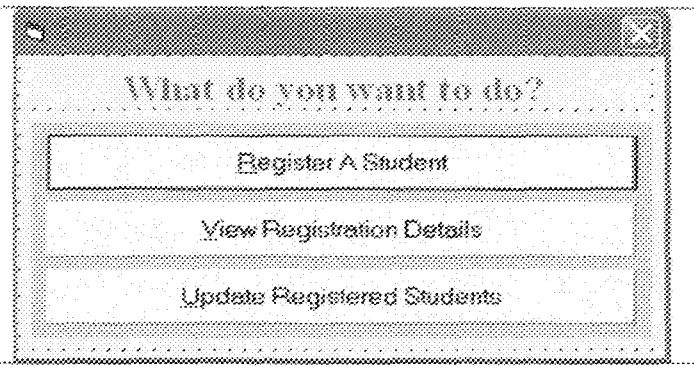
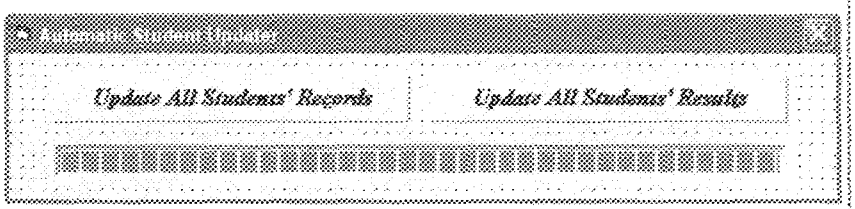
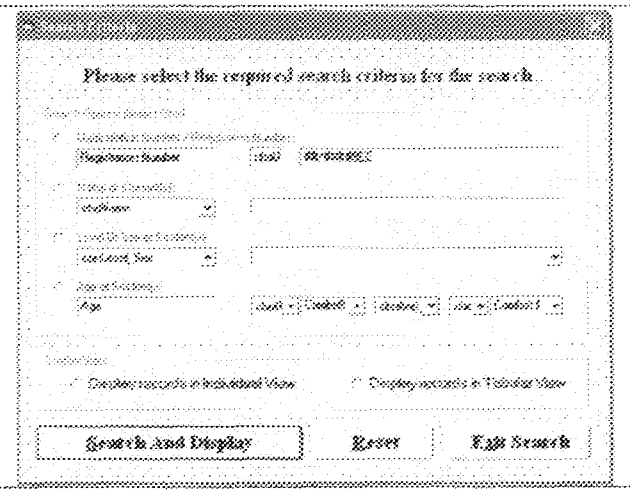
FORMS USED IN THE FRONT-END SOFTWARE

Common Forms:

Details	Form Pictures
Form Name: - *frmSplah1* Saved As: - *frmSplash1.frm* NOTES: - *AMEzy Splash screen.*	
Form Name: - *frmLogin* Saved As: - *frmLogin.frm* NOTES: - *Login Form.*	
Form Name: - *frmPassChange* Saved As: - *frmPassChange.frm* NOTES: - *Used to change the user password.*	
Form Name: - *frmAbout* Saved As: - *frmAbout.frm* NOTES: - *Displays the details of the AMEzy Student Registration Program*	

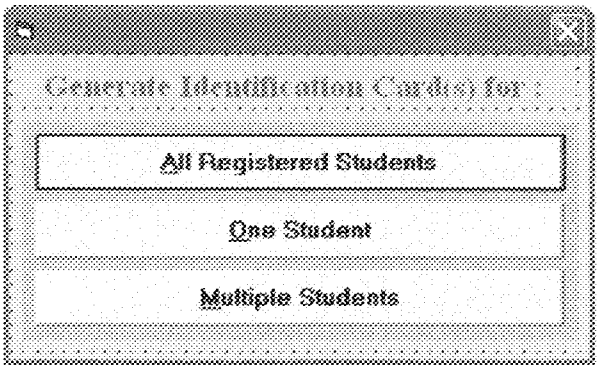
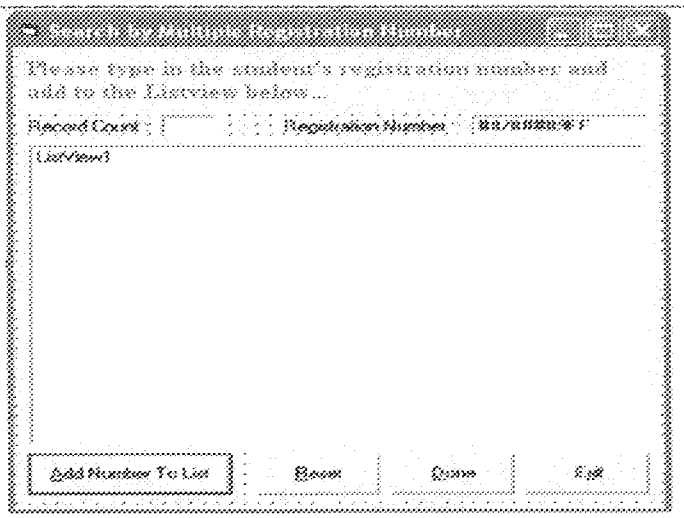
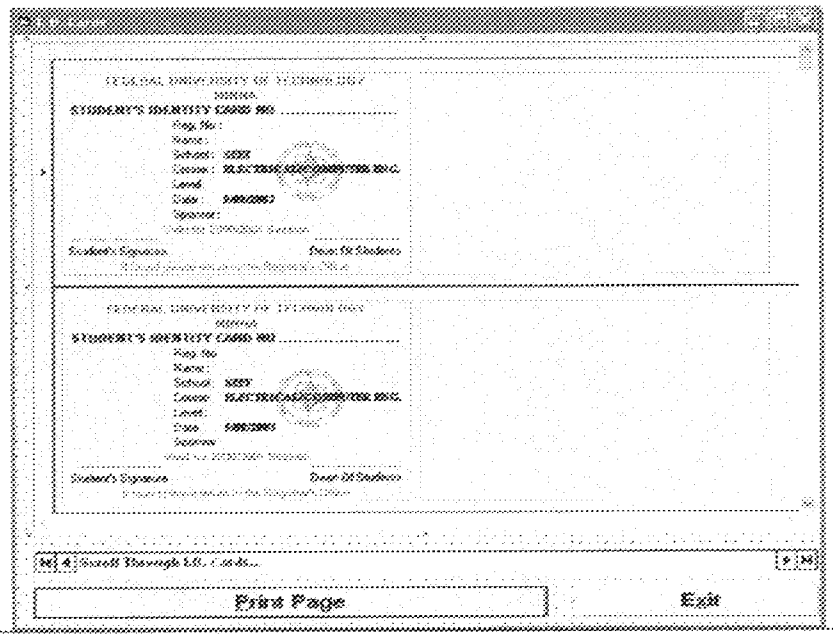
Appendix 2.1: *Common forms used in the AMEzy Student Registration Program*

The Student Registration Program:

Details	Form Pictures
Form Name: - *frmSplash2* Saved As: - *frmSplash2.frm* NOTES: - *Form showing terms of use of the program to be agreed on before proceeding.*	
Form Name: - *frmWelcome* Saved As: - *frmWelcome.frm* NOTES: - *Used to gain access to basic tasks.*	
Form Name: - *frmStudUpdater* NOTES: - *Used to automatic update information every session.*	
Form Name: - *frmSearch* Saved As: - *frmSearch.frm* NOTES: - *Search Criteria Form used to search for student records based on selected criteria.*	

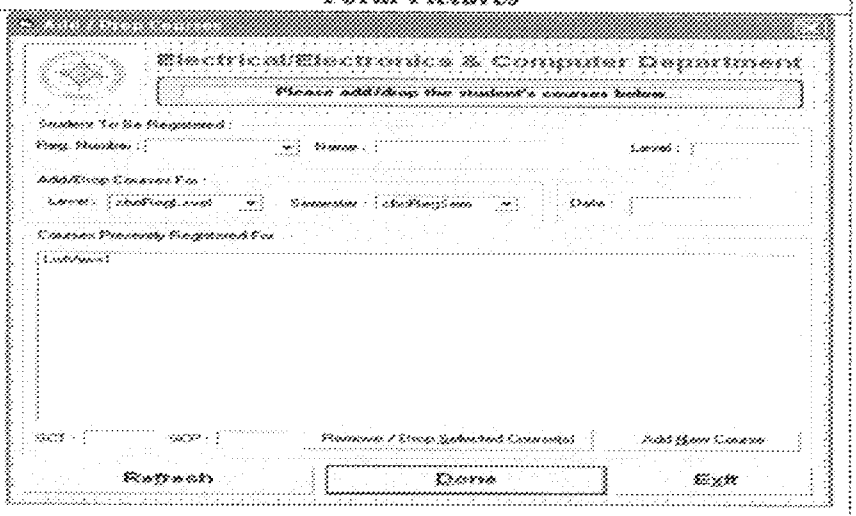
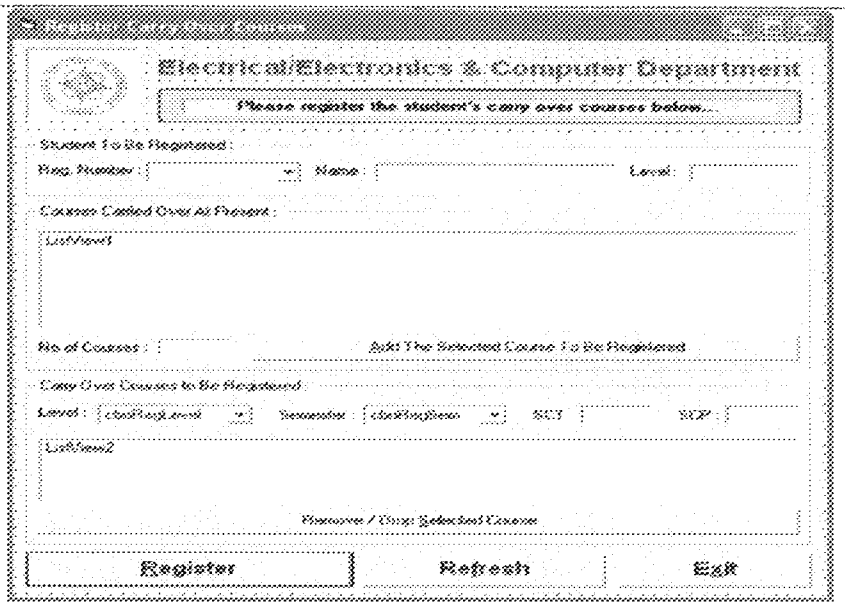
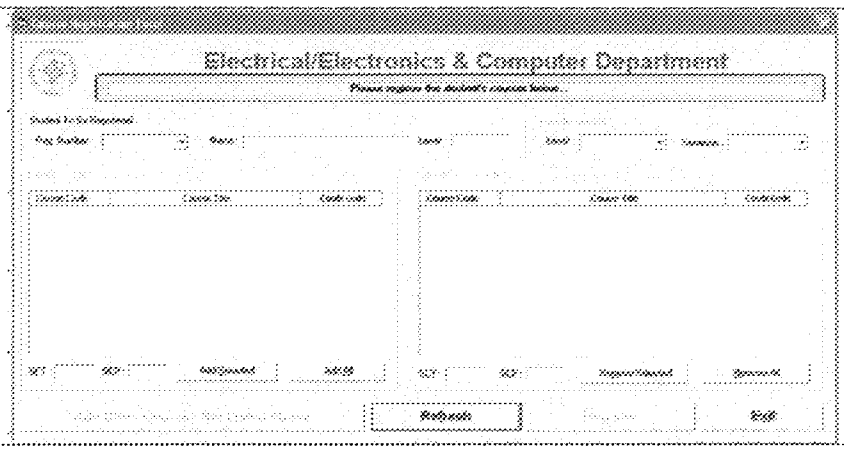
Appendix 2.2: *Sample Forms used in the AMEzy Student Registration Program*

The Identification Card Wizard:

Details	Form Pictures
Form Name: - *frmWelcome1* Saved As: - *frmWelcome1.frm* NOTES: - *Used to gain access to how the cards should be generated.*	
Form Name: - *frmMultiple* Saved As: - *frmMultiple.frm* NOTES: - *This form is used to search for and generate identification cards for students based on their registration numbers.*	
Form Name: - *frmRepeater / frmRepeater1* Saved As: - *frmRepeater.frm / frmRepeater1.frm* NOTES: - *These forms display identification cards to be generated for all students or multiple students (searched for with the search criteria).*	

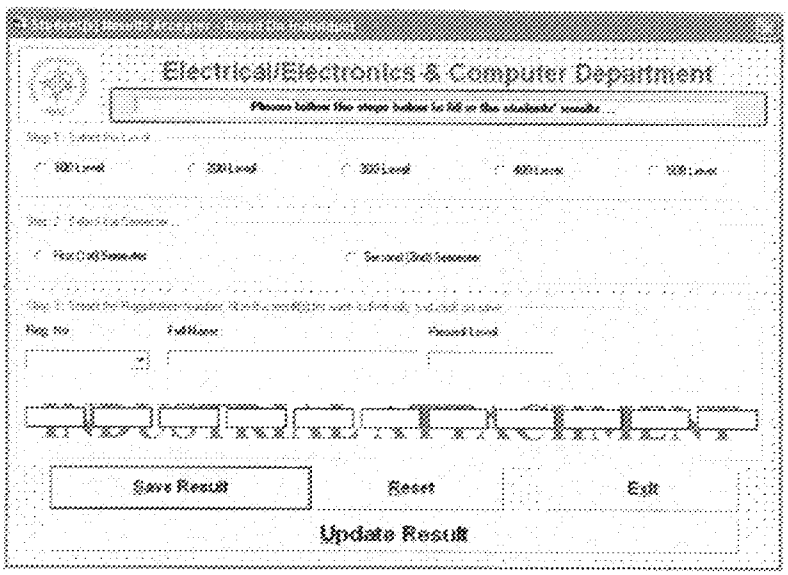
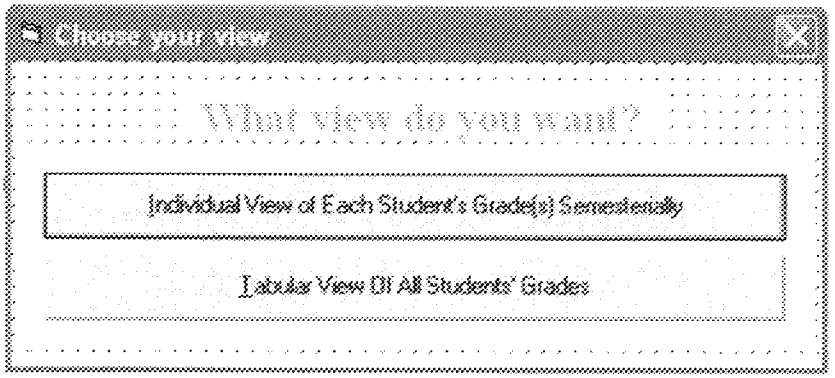
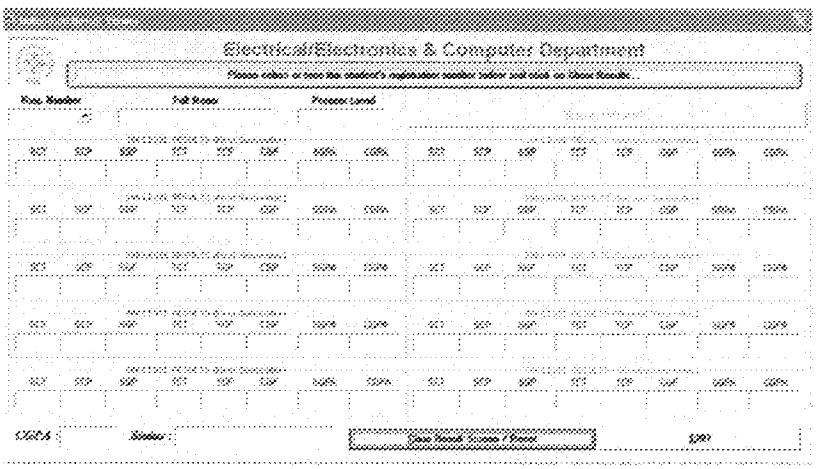
Appendix 2.3: *Sample Forms used in the AMEzy Identification Card Wizard*

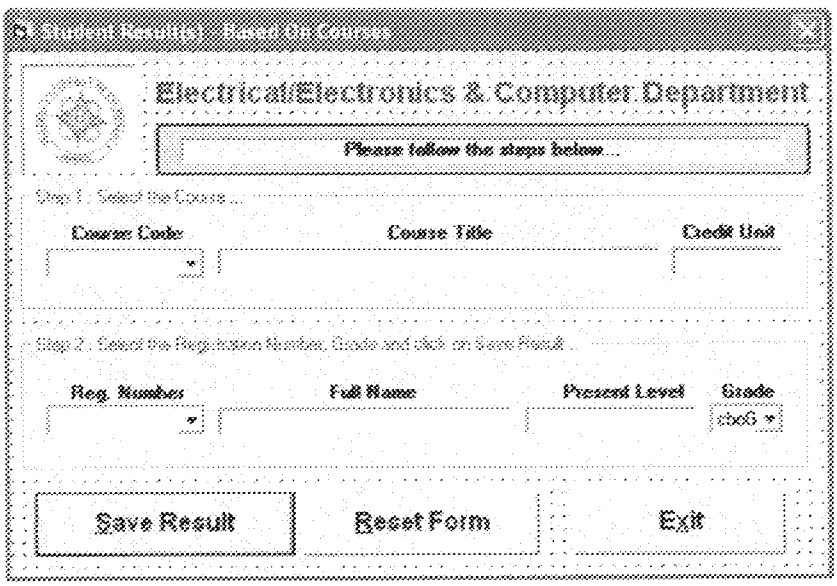
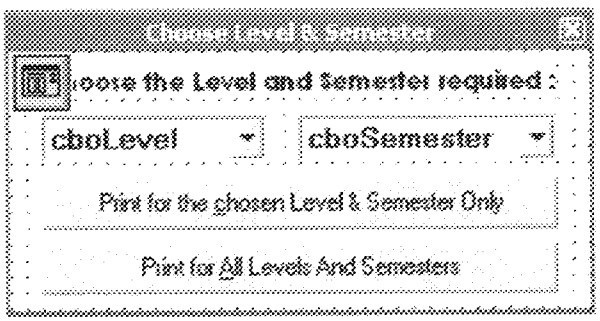
The Course Registration Program:

Details	Form Pictures
Form Name: - *frmAddDrop* Saved As: - *frmAddDrop.frm* NOTES: - *Add / Drop Courses Form for adding extra courses or dropping already registered courses.*	
Form Name: - *frmCarryOver* Saved As: - *frmCarryOver.frm* NOTES: - *This form is used to register courses carried over at present by students.*	
Form Name: - *frmCourseReg1* Saved As: - *frmCourseReg1.frm* NOTES: - *This form is used for fresh course registration.*	

Appendix 2.4: *Sample Forms used in the AMEzy Course Registration Program*

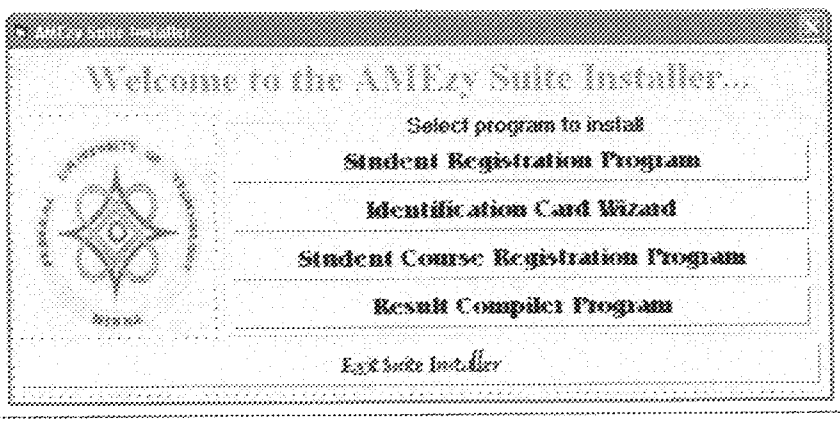
The Result Compiler Program:

Details	Form Pictures
Form Name: - *frmResults* Saved As: - *frmResults.frm* NOTES: - *This Result Acceptor form is used to update results stored for courses taken and to display results stored.*	
Form Name: - *frmViewType1* Saved As: - *frmViewType1.frm* NOTES: - *Used to select which view to see user results.*	
Form Name: - *frmSemesterial* Saved As: - *frmGP.frm* NOTES: - *This is the individual details viewer form used to view details of compiled results for registered students.*	

Details	Form Pictures
Form Name: - *frmResults1* Saved As: *frmResult1.frm* NOTES: - *Used to record student(s) results based on the courses taken.*	
Form Name: - *frmLevSem* Saved As: - *frmLevSem.frm* NOTES: - *Here, the user can choose a category of results to print.*	

Appendix 2.5: *Sample Forms used in the AMEzy Result Compiler Program*

The Suite Installer:

Details	Form Pictures
Form Name: - *frmMain* Saved As: - *frmSuiteIns.frm* NOTES: - *Suite Installer Form used to launch and install the AMEzy programs.*	

Appendix 2.6: *Form used for the AMEzy Suite Installer Program*

APPENDIX 3

SAMPLE CODES FOR MODULES USED IN THE FRONT-END SOFTWARE

The Student Registration Program:

- **Module Name:** Module1
- **Saved As:** Module1.Bas
- **Codes:**

```
Public cnn1 As ADODB.Connection, Report As Boolean, rsx3 As ADODB.Recordset
Public rs1 As ADODB.Recordset, rs121 As ADODB.Recordset, rs2 As ADODB.Recordset
Public OK As Boolean, F1 As Boolean, Search As Boolean, F2 As Boolean
Public num1 As Integer, Remember As integer, rs12 As ADODB.Recordset
Public Criteria, Value As String, rs11 As ADODB.Recordset
Public TempName As String, rsx2 As ADODB.Recordset
Public counter As Integer, rsx1 As ADODB.Recordset
Public rs3 As ADODB.Recordset, res1 As String, butt As integer
Public fMainForm As New frmMain, rsx As ADODB.Recordset


Sub Main()
'Declare new Login
   Dim fLogin As New frmLogin

'New Connection1
   Set cnn1 = New ADODB.Connection
   cnn1.Provider = "Microsoft.Jet.OLEDB.4.0"
   cnn1.CursorLocation = adUseClient
   cnn1.Open "C:\Common Files\PUTdb.mdb"

'New Recordset1 for Users
   Set rs1 = New ADODB.Recordset
   rs1.CursorLocation = adUseClient
   rs1.Open "Users", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

'New Recordset for Students Dtbase
   Set rs12 = New ADODB.Recordset
   rs12.CursorLocation = adUseClient
   rs12.Open "AllStudents", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

   Set rs2 = New ADODB.Recordset
   rs2.CursorLocation = adUseClient
   rs2.Open "AllStudents", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

   Set rsx = New ADODB.Recordset
   Set rsx1 = New ADODB.Recordset
   Set rsx2 = New ADODB.Recordset
   Set rsx3 = New ADODB.Recordset

   rsx.CursorLocation = adUseClient
   rsx1.CursorLocation = adUseClient
   rsx2.CursorLocation = adUseClient
   rsx3.CursorLocation = adUseClient

   rsx.Open "Results", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable
```

```
    rsx1.Open "CourseReg", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable
    rsx2.Open "PersonalRes", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable
    rsx3.Open "Session1", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

Set rs3 = New ADODB.Recordset
Search = False

'Load Splash Screen1
    frmSplash1.Show vbModal
    Unload frmSplash1

'Load Login Screen
    fLogin.Show vbModal
    If Not OK Then
        'Login Failed so exit app
        End
    End If
    Unload fLogin

'Load Splash Screen2
    frmSplash2.Show vbModal
    If Not frmSplash2.OK1 Then
        'Login Failed so exit app
        End
    End If

'Test Screen Size & Resolution
    'Set up the screen values
    Xtwips = Screen.TwipsPerPixelX
    Ytwips = Screen.TwipsPerPixelY
    Ypixels = Screen.Height / Ytwips 'Y Pixel Resolution
    Xpixels = Screen.Width / Xtwips  'X Pixel Resolution

    If (Xpixels < 1024) Or (Ypixels < 768) Then
        MsgBox "Please set your Screen Resolution to 1024 by 768 pixels or a higher resolution to " + _
        "continue!!", vbOKOnly, "Change Screen Resolution"
        Exit Sub
    End If

    fMainForm.Show

'Show welcome Form
    frmWelcome.Caption = "WELCOME " & rs1!UserName
    frmWelcome.Show vbModal
    'GoTo Leave


End Sub

Public Function LoadNewDoc()
Select Case num1

    Case 1
        Static DocNo As Integer
        Dim frmDoc1 As New frmDocument1
        frmDoc1.Show
        DocNo = DocNo + 1
        frmDoc1.Caption = "Student Registration Form " & DocNo
```

```
    Case 2
        Static DocNo1 As Integer
        Dim frmDoc2 As New frmDocument2
        frmDoc2.Show
        DocNo1 = DocNo1 + 1
        If Search = False Then
            frmDoc2.Caption = "Individual Student Detail Viewer " & DocNo1
        Else
            frmDoc2.Caption = "Search Results for " & Criteria & " " & Value
            Search = False
        End If

    Case 3
        Static DocNo2 As integer
        Dim frmDoc3 As New frmDocument3
        frmDoc3.Show
        DocNo2 = DocNo2 + 1
        If Search = False Then
            frmDoc3.Caption = "Tabular Student Detail Viewer  " & DocNo2
        Else
            frmDoc3.Caption = "Search Results for " & Criteria & " " & Value
            Search = False
        End If

    Case 4
        Static DocNo3 As Integer
        Dim frmDoc4 As New frmDocument4
        frmDoc4.Show
        DocNo3 = DocNo3 + 1
        frmDoc4.Caption = "Student's Update Form " & DocNo3

End Select
End Function

Public Sub RecRefresh()

If rs2.BOF Then
   Exit Sub
Else
   rs2.MoveFirst
End If

End Sub
```

Appendix 3.1: *Module information for The AMExy Student Registration Program*

The Identification Card Wizard:

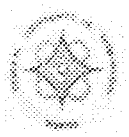
- **Module Name:** Module2
- **Saved As:** Module2.Bas
- **Codes:**

```
Public cnn1 As ADODB.Connection, Numb As String, counter As Integer, res1 As String
Public rs1 As ADODB.Recordset, rs2 As ADODB.Recordset, Playa As Boolean
Public OK As Boolean, F1 As Boolean, Search As Boolean, F2 As Boolean
Public num1 As Integer, rs4 As ADODB.Recordset, rs3 As ADODB.Recordset
Public Criteria, Value As String, BOOM As Boolean, TempName As String

Sub Main()

'Declare new Login
   Dim fLogin As New frmLogin

'New Connection1
   Set cnn1 = New ADODB.Connection
   cnn1.Provider = "Microsoft.Jet.OLEDB.4.0"
   cnn1.CursorLocation = adUseClient
   cnn1.Open "C:\Common Files\FUTdb.mdb"

'New Recordset1 for Users
   Set rs1 = New ADODB.Recordset
   rs1.CursorLocation = adUseClient
   rs1.Open "Users", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

'New Recordset for Students Dbase
   Set rs2 = New ADODB.Recordset
   rs2.CursorLocation = adUseClient
   rs2.Open "AllStudents", cnn1, adOpenDynamic, adLockPessimistic, adCmdTable

'Load Splash Screen1
   frmSplash1.Show vbModal
   Unload frmSplash1

'Load Login Screen
   fLogin.Show vbModal
   If Not OK Then
      'Login Failed so exit app
      End
   End If
   Unload fLogin

'Load Splash Screen2
   frmSplash4.Show vbModal
   If Not frmSplash4.OK1 Then
      'Login Failed so exit app
      End
   End If

'Show welcome Form
   frmWelcome1.Show vbModal

Playa = False
End Sub
```

Appendix 3.2: *Module information for The AMEzy Identification Card Wizard*

APPENDIX 4

SAMPLE PRINTOUTS FOR CASES SOLVED WITH AMEzy SOFTWARE



Electrical / Electronics & Computer Department
Federal University of Technology
Minna, Niger State, NIGERIA.

Student Report Sheet

Date Printed: Monday, October 13, 2003

Registration Number: 97/9905EE

Title: Miss. Full Name Of Student: FANIS T. LINDA

1998 RESULTS

First Semester	Second Semester
SCE: 23	SCE: 23
SCP: 23	SCP: 23
SGP: 110	SGP: 113
TCE: 23	TCE: 46
TCP: 23	TCP: 46
CGP: 110	CGP: 223
SGPA: 4.78	SGPA: 4.91
CGPA: 4.78	CGPA: 4.85

2000 RESULTS

First Semester	Second Semester
SCE: 21	SCE: 21
SCP: 21	SCP: 21
SGP: 101	SGP: 99
TCE: 87	TCE: 88
TCP: 87	TCP: 86
CGP: 324	CGP: 423
SGPA: 4.81	SGPA: 4.71
CGPA: 4.84	CGPA: 4.81

2001 RESULTS

First Semester	Second Semester
SCE: 21	SCE: 22
SCP: 21	SCP: 22
SGP: 101	SGP: 109
TCE: 108	TCE: 131
TCP: 108	TCP: 131
CGP: 524	CGP: 633
SGPA: 4.81	SGPA: 4.95
CGPA: 4.81	CGPA: 4.83

2002 RESULTS

First Semester	Second Semester
SCE: 22	SCE: 6
SCP: 22	SCP: 6
SGP: 105	SGP: 0
TCE: 153	TCE: 153
TCP: 153	TCP: 153
CGP: 736	CGP: 736
SGPA: 4.77	SGPA: 0
CGPA: 4.82	CGPA: 4.82

2003 RESULTS

First Semester	Second Semester
SCE: 20	SCE: 20
SCP: 20	SCP: 20
SGP: 83	SGP: 82
TCE: 173	TCE: 193
TCP: 173	TCP: 193
CGP: 821	CGP: 913
SGPA: 4.15	SGPA: 4.1
CGPA: 4.75	CGPA: 4.73

Printed by The Electrical/Electronics Department of Federal University Of Technology, Minna, Niger State

Appendix 4.1: *Sample Printout of Mrs. Linda Fanis' Result*

Electrical / Electronics & Computer Department
Federal University of Technology
Minna, Niger State, NIGERIA.

Student Information Form

Date Printed: Monday, October 13, 2003

Student Number:	2	Title:	Mr
Registration No:	99/9994EE	Full Name:	OLAJIDE A. TUNDE
Level:	200L	Nationality:	Nige' Republican
Date of Birth:	12/12/1980	Sex:	Male
Religion:	Islam	Entry Mode:	D - Direct

Marital Status:	Single	Former Surname:	
Programme:	F - First Degree	Year Of Entry:	1999
State Of Birth:	Daoula	Town/Village Of Birth:	Daoula
State Of Origin:	Daoula	Town/Village Of Origin:	Daoula
Mode Of Study:	E - Evening	Highest Qualification:	HSC/GCE A/L
E-mail Address:	wasini5"6@yahoo.com	Website URL:	
Telephone No. 1:	234-1-4859383394	Telephone No. 2:	
Home Address:	45 Daoula Street, Daoula, Nige' Republic		
Contact Address:	P. O. Box 54555, Dmac Mail box, Kerr		

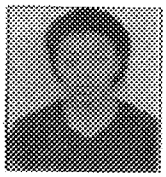
Next Of Kin:	Mr Gang Moly	Relationship to Student:	Uncle
Next Of Kin No:	0802-2348977789	Address Of Next Of Kin:	45 Daoula Street, Daoula, Nige' Republic
Sponsor's Name:	Mr Gang Moly	Address of Sponsor:	45 Daoula Street, Daoula, Nige' Republic
Other Details:	Bad Boy		

Student's Signature

H. O. D's Signature

Printed by The Electrical/Electronics Department of Federal University Of Technology, Minna, Niger State

FEDERAL UNIVERSITY OF TECHNOLOGY
MINNA
STUDENT'S IDENTITY CARD 1



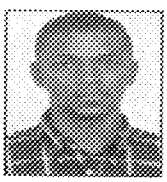
Reg. No: 99999TEE
Name: BENITA J. OKOYE
School: SEET
Course: ELECTRICAL/COMPUTER ENG
Level: 100L
Date: 13/10/2003
Sponsor:

Valid for 2003/2004 Session

Student's Signature

Dean Of Students

FEDERAL UNIVERSITY OF TECHNOLOGY
MINNA
STUDENT'S IDENTITY CARD 2



Reg. No: 99999EE
Name: OLAJIDE A. TUNDE
School: SEET
Course: ELECTRICAL/COMPUTER ENG
Level: 200L
Date: 13/10/2003
Sponsor: Mr Gang Moly

Valid for 2003/2004 Session

Student's Signature

Dean Of Students

Appendix 4.2: *Sample Printout of Information Form and Identification Cards*