ADMINISTRATION SOFTWARE FOR THE DEPARTMENT OF ELECTRICAL/COMPUTER ENGINEERING, USING OBJECT ORIENTED PROGRAMMING AND SQL SERVER

BY

OYOM MAXWELL EZUKWA

(99/8346EE)

DEPARTMENT OF ELECTRICAL/COMPUTER ENGINEERING
SCHOOL OF ENGINEERING AND ENGINEERING TECHNOLOGY
FEDERAL UNIVERSITY OF TECHNOLOGY
MINNA, NIGER STATE
NIGERIA

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

NOVEMBER, 2005

CERTIFICATION

This work was carried out by Oyom Maxwell Ezukwa, 99/8346EE, of the Electrical/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.

Mr U.S. Dauda 6/12/05

(PROJECT SUPERVISOR) **DATE**

....

(H. O. D. ELECT./COMP. ENG'G) **DATE**

....

EXTERNAL EXAMINER **DATE**

LIST OF FIGURES

Page

Chapter One
Introduction

1.1 General Introduction

The use of computers by organizations or businesses in handling administrative tasks such as billing, appointments and computations of figures has grown dramatically over the years.

Computers can process information faster than the human brain and computers improve accuracy. A human repeating the same complex calculation one hundred times is likely to get a dozen different answers but a rigorously tested computer software repeating the same calculation one hundred times will get the same answer each time.

Computer software can be designed to undertake validity checks on data, rules can be added, comparisons made and reminders given. These help reduce human error and improve the accuracy of records, resulting in greater accuracy, improved communication and improved retrieval of information.

All that is required is a well thought out, carefully designed software with user education and a well managed computer system for the many proven benefits of computerisation to be achieved.

Programming with Object Oriented Programming (OOP) not only gives a simplified form of coding for developing software but also ensures reliability, reusability and easier maintenance of programs.

1.2 Brief Overview of the Electrical/Computer Engineering Department

The department which is one of the departments in the School of Engineering and Engineering Technology (SEET) of the Federal University of Technology, Minna, was established in 1983.

The Department offers a Bachelor of Engineering Degree in Electrical Engineering after five years to students who meet the criteria. These students are admitted into the department at 100 level based on their scores in the Universities Matriculation Examination (UME) or into 200/300 level through Direct Entry Program.

Students' registration is done in a per session basis and all courses to be offered by the students for a session are registered for at the beginning of the session with options for the student to add or drop a course (or courses) registered. The students sit for examinations at the end of the semester based on the courses they have registered.

There are two academic semesters for each of the five academic years and the students are expected to go for a 24 week Industrial Training by the second semester of the fourth year (400 level).

Presently, carrying out administrative tasks in the department, such as results compilation, issuance of students' transcripts and documentations of course registrations are quite tasking, since these tasks have not been fully automated with computer software.

1.3 Benefits of Utilising Computer Software in carrying out Administrative Tasks:

Many organizations/businesses are taking advantage of computerization to customise and simplify their administrative tasks and Electrical/Computer Engineering Department should not be an exception.

Some of the benefits of computerization include:

- Savings in energy, cost and time, the energy in this case being that of the staff rather than of the power required to drive the computer systems. By employing the use of software, quite a number of processes including manual form-filling, manual computation and storage of students results, drafting and re-drafting in manuscript and typescript, correction and re-typing and proof-reading have been obviated.
- 'Reduced unit costs' or 'reduced handling time' per transaction; for example in a case where utilising a computer software results in making students data available with few minutes as against the several minutes it would take to get the data manually.
- The third practical benefit from computerisation has been in opening up entirely new possibilities in information management. Most significantly, information can now be dynamic, growing continuously as data is accrued. Users of the software can be offered completely up-to-date student information through inter-active searches.

1.4 Examples of the use of computerisation by organisations in handling administrative tasks:

Commercially available software applications have been found entirely satisfactory by many public and private establishments for word processing applications and for creating simple databases.

Example of such software are:

- UNESCO's **CDS/ISIS** and its derivatives (with users in countries like Hungary, Nigeria, Canada and China, the Soviet Union and Portugal).
- One of the best known systems, because it was widely demonstrated to archivists attending the ICA Congress in Paris, 1988, is the **PRIAM 3** application operating at the Centre des Archives Contemporaines, Fontainebleau.

1.5 Problems Involved/Factors to be Considered In Computerisation:

1.5.1 Training:

There is a great need for in-depth training, not just familiarization, of staff intending to use the software.

1.5.2 Finance:

Finance remains a problem everywhere, and for all the cost benefits that can be marshaled in its defence, there is no contesting that computerisation can be a very expensive process.

1.5.3 **Identifying the user:**

One of the most fundamental issues to be considered is the user community. For whom is the computer software designed? For level advisers, Heads of Department, or for Departmental secretaries? Can they all be served by the same system or are their needs sufficiently different to justify different approaches for different user groups?

1.5.4 **Security:**

Any kind of user-access to live data, even by the department's own staff, must be properly controlled to ensure that only authorised persons may enter and amend or delete information. To a large extent provision for this kind of security is written into the software itself, and is no more difficult to arrange than the 'menus' or questions designed to help any user to reach the desired information, but it must be competently set up and maintained.

1.6 The Risk Involved in Computerisation:

Computerised data is vulnerable in quite different ways from that contained in earlier means of data storage. A determined vandal with a magnet may be capable of wiping clean a database. Power surges can do the same. Computer fires are by no means unknown. It is therefore essential to maintain adequate back-up facilities such as up-to-date duplicate disks or tapes stored away from the location of the main data.

More common practical problems arise when for any reason the computer is 'down' or 'has crashed', and it is important to have a strategy for

recovery and for servicing user demand in the meanwhile. But many would find this an unworkable solution on account of the bulk or form of data and the expense involved.

Not all softwares come complete with a readily intelligible manual. In developing software like this for the department rather than using standard software packages, there is a risk of the software becoming complex and difficult for new generations of programmers to understand.

1.7 Aims and Objectives of the Project:

The purpose for carrying out this project work and the benefits that are expected to be achieved are outlined below:

1.7.1 Aim of the Project:

The aim of this project is to deliver a working software that solves students administration problems in the Department of Electrical/Computer Engineering. The software will allow for future expansion and meet departmental objectives.

The software will provide database facilities for the storing, updating and retrieval students` personal information, course registration and results.

Another aim of the project is to demonstrate an interaction with external computer hardware, such as a printer, through the use of the software.

Additional aims are to allow for better understanding of the administration problems on ground, apply existing skills and develop new skills in solving the problem.

1.7.2 Personal Objectives:

Listed below are the objectives I believe I will gain from the project process. In satisfying these objectives, I believe I will be more aware of the problems and successes of managing and producing a project of this size:

- Gaining experience in a project like this which involves managing a software life cycle form start to finish.
- To apply knowledge acquired over the duration of the Electrical/Computer Engineering degree course, especially in areas relating to computer programming and communication with external computer hardware, like a printer.
- To develop and build on those skills already acquired though practical operations of similar software.
- To understand more about OOP and software development

1.7.3 Objective of the Project:

- To produce a simple, yet suitably effective user friendly interface, and a system that is easily maintainable, efficient and practical.
- To understand and appreciate the process of software development

- To better understand interaction of a computer program with an external computer hardware, such as a printer.

- To apply knowledge gained throughout my course of study, in areas such as systems design and analysis.

- The effectiveness of the system in both managing the Electrical/Computer Engineering Department administrative tasks in the department.

1.8 Project Layout:

Chapter 2: is the literature review of the project. The chapter gives an overview of object oriented programming. The chapter goes on to discuss Relational Database Management Systems, Structured Query Language (SQL) and Java Database Connectivity.

Chapter 3: Talks about the back-end of the database, structure of the database and tables in the database. The chapter also looks at the Front-end of the XManager software, divided into the students personal information form, students course registration, students academic information and the transcripts.

Chapter 4: This chapter is on the tests carried out on the XManager software, comparison of values obtained using manual computations and using the software to compute students results, using the software to retrieve specific information regarding specific students and discussion of these results.

Chapter 5: Gives the conclusion and recommendations for the automation of the department using the XManager software.

Chapter Two

Literature review

2.1 Theory of Object Oriented Programming:

Objects are key to understanding object-oriented technology. Real-world objects like dogs, desks, television sets and bicycles share two characteristics. They all have *states* and *behaviours*.

Software objects are modelled after real-world objects in that they too have state and behaviour. A software object maintains its state in one or more *variables*. A variable is an item of data named by an identifier. A software object implements its behaviour with *methods*. A method is a function (subroutine) associated with an object [6]

Definition: An object is a software bundle of variables and related methods

OOP *encapsulates* data (*attributes*) and methods (*behaviors*) into *objects*; the data and methods of an object are intimately tied together. Objects have the property of *information hiding*. This means that although objects might know how to communicate with one another across well-defined *interfaces*, objects normally are not allowed to know how other objects are implemented - implementation details are hidden within the objects themselves [6].

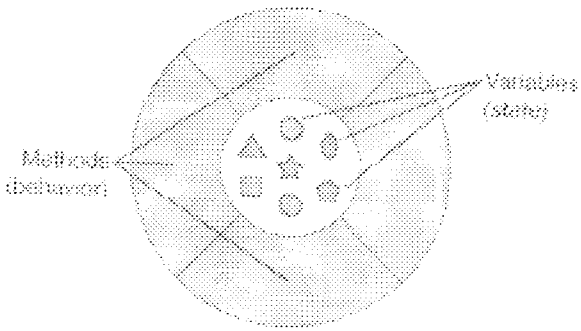


Figure 2.1: Visual representation of a software object [6]

2.2 Object Oriented Programming Concepts:

Some of the concepts of Object Oriented Programming are: Object, Message, Class, Inheritance, Interface etc [6].

2.2.1 Objects:

Already explained above, represent entities found in the real world.

2.2.2 Messages:

Software objects interact and communicate with each other by sending messages to each other. When object A wants object B to perform one of B's methods, object A sends a message to object B (see fig 2.2 below).

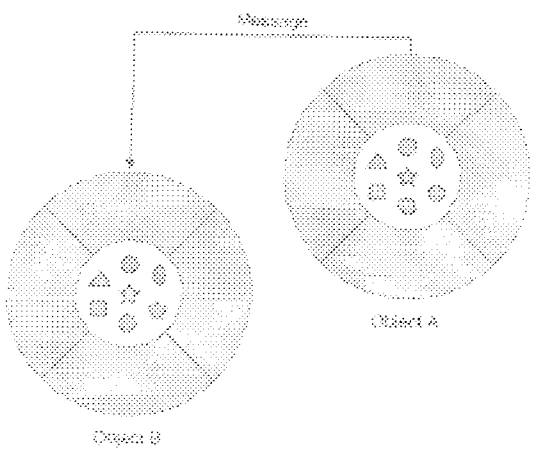


Figure 2.2: Communication between two objects[6]

The three parts of a message are:

- The object to which the message is addressed
- The name of the method to perform
- Any parameters needed by the method

2.3 Databases

A *database* is a structured, integrated collection of meaningful information stored over a period of time in machine-readable form for subsequent retrieval. By this definition, any file or collection of files can be considered a database. However, to be useful in practical terms, a database must form part of a system that provides for the management of the data it contains. Seen from this perspective, a database must be more than a mere collection of files. It must be a complete system. There are many different strategies for organizing data to facilitate easy access and manipulation of the data [2].

2.4 Database Management Systems

A *Database Management System (DBMS)* provides mechanisms for storing and organizing data in a manner consistent with the database's format. Database management systems allow for the access and storage of data without worrying about the internal representation of databases. Such a system must support the following tasks:

- Creation and management of a logical data structure
- Data entry and retrieval
- Manipulation of the data in a logical and consistent manner
- Storage of data reliably over a significant period of time

2.5 Database Models

To provide the required durability, data is stored in physical storage devices. These files are stored in different logical formats depending on the database model selected by every particular database management system.

Some of the many database models in the database market are: Flat files, Hierarchical, Networked, Relational, Object, Object-relational and Document models [5]. The Relational Model is of primary concern in this project work.

2.6 The Relational Database Model

The big step forward in database technology was the development of the *relational database model*. The relational database derives from work done in the late 1960s by E.F. Codd, a mathematician at IBM. His model is based on the mathematics of set theory and predicate logic. In fact, the term *relational* has its roots in the mathematical terminology of Codd's paper entitled "A relational model of data for large shared data banks," which was published in *Communications of the ACM*, Vol. 13, No. 6, June 1970, pp. 377-387. In this paper, Codd uses the terms *relation*, *attribute*, and *tuple* where more common programming usage refers to *table*, *column*, and *row*, respectively.

The importance of Codd's ideas is such that the term "database" generally refers to a relational database. Similarly, in common usage, a Database Management System, or DBMS, generally means a Relational Database Management System. Codd's model covers the three primary requirements of a relational database: structure, integrity, and data manipulation. The fundamentals of the relational model are as follows:

- A relational database consists of a number of unordered tables.
- The structure of these tables is independent of the physical storage medium used to store the data.
- The contents of the tables can be manipulated using nonprocedural operations that return tables [2]

Some popular enterprise-level relational-database systems are Microsoft SQL Server, Oracle, Sybase, DB2, Informix and MySQL. The volume of data to be stored, security requirements and number of simultaneous access to the data determine the software chosen for data storage [3].

2.7 Structured Query Language

Structured Query Language (*SQL*) is used almost universally with relational-database systems to perform *queries* (i.e., to request information that satisfies given criteria) and to manipulate data.

The Structured Query Language (SQL) was first developed by IBM in the 1970s and was later the subject of several ANSI standards [3]

2.71 Basic 'SELECT' Query

Several SQL queries that extract information from a sample database, **school** are considered. A typical SQL query "selects" information from one or more tables in a database. Such selections are performed by ***SELECT*** *queries*. The simplest format of a **SELECT** query is

SELECT * FROM *students*

In this query, the asterisk (*) indicates that all rows and columns from the **students** table of the **school** database should be selected.

To select specific fields from a table, the asterisk (*) is replaced with a comma-separated list of the field names to select. For example, to select only the fields **studentID** and **lastName** for all rows in the **student** table, the query below is used:

SELECT *studentID*, *lastName* **FROM students**

This query returns the data listed below:

studentID	LastName
001	Fred
002	Maxi
003	Durst
004	Chester

Figure 2.3 studentID and lastName from the students table [3].

2.72 'WHERE' Clause:

In most cases, it is necessary to locate records in a database that satisfy certain *selection criteria* .Only records that match the selection criteria are selected. SQL uses the optional ***WHERE*** *clause* in a **SELECT** query to specify the selection criteria for the query. The simplest format of a **SELECT** query with selection criteria is:

SELECT *studentID*, *lastName* **FROM** *students* **WHERE** *studentID = 002*

studentID	lastName
002	Maxi

Figure 2.4: studentID and lastName for studentID 002 [3]

2.8 Microsoft SQL Server 2000- An RDBMS

Microsoft SQL Server is a scalable database system whose primary purpose is to serve as a back-end database for a client program, such as a Web browser, an accounting program, or a human resources application —anything that makes use of the data. In the most common usage scenario, a client program connects to SQL Server and requests some information, whereupon SQL Server processes the request and returns results [4]. The client must then interpret and display these results (for example, a school's departmental application displaying a list of students and their matriculation number).

Unlike text editors or games, which do not require any additional components in order to be useful, MS SQL Server does not make much sense as a stand-alone program or as a program that runs on a stand-alone computer. Although it is possible to have both a client and SQL Server running on the same computer, it is not very useful. SQL Server is meant to be part of a network (local, wide or the Internet) and to serve more than one user.

SQL Server can store structured information in a variety of formats, and it enables the user to manipulate these information. For example, the user can instantly search through millions of records and view the results of the search in many different formats; combine different data into one set; transform some formats into others and set security rules to be enforced by SQL Server.

SQL Server is very flexible, and has to be told exactly what to do by the user [4].

2.9 Java Database Connectivity (JDBC):

JDBC is a Java Database Connectivity API (Application Programming Interface) that ensures access to virtually any tabular data source from a Java application. In addition to providing connectivity to a wide range of SQL databases, JDBC allows access other tabular data sources such as spreadsheets or flat files [1].

JDBC defines a low-level API designed to support basic SQL functionality independently of any specific SQL implementation. This means the focus is on executing raw SQL statements and retrieving their results. JDBC is an international standard for programming access to SQL databases, which is also the basis for Microsoft's ODBC interface [2].

2.10 Open Database Connectivity

Open Database Connectivity (ODBC) is an application programmer's interface (API) that exposes underlying database cap-abilities.

An abstract layer (called an ODBC Driver) allows applications to access virtually any relational database management system without paying much attention to the differences between them (as long as you have an appropriate ODBC driver) [4]

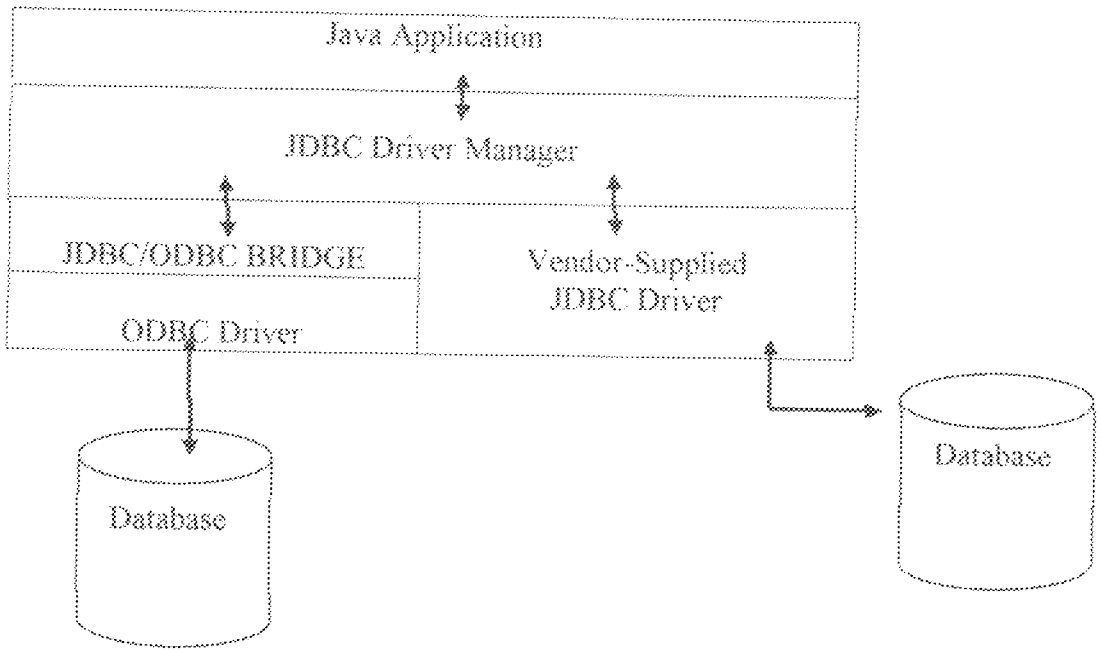


Figure 2.5: JDBC-to-database communication path [1]

2.11 Database Access Models

The JDBC API supports both two-tier and three-tier models for database access. In other words, JDBC can either be used directly from an application or as part of a middle-tier server application.

Two-Tier Model

In the two-tier model, a Java application interacts directly with the database. Functionality is divided into these two layers:

- **Application layer** which includes the JDBC driver, business logic, and graphical user interface.
- **Database layer** which includes the RDBMS

The interface to the database is handled by a JDBC driver appropriate to the particular database management system being accessed. The JDBC driver passes SQL statements to the database and returns the results of those statements to the application [2].

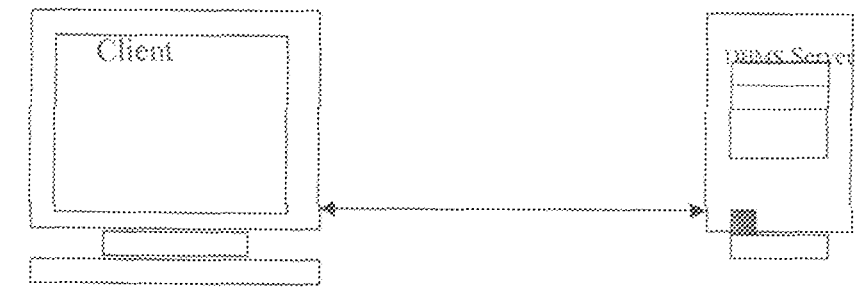


Figure 2.6: Two-tier client/server configuration [2]

A client/server configuration is a special case of the two -tier model, where the database is located on another machine, referred to as the *server*. The application runs on the client machine, which is connected to the server over a network [2]

2.12 SQL Conformance:

Although SQL is the standard language for accessing relational databases, different RDBMS systems (e.g. Oracle and Microsoft SQL Server) support a large number of different dialects of SQL. These differences range from such minor details as whether a SQL statement needs a closing semicolon to major variations such as the absence of support for stored procedures or some types of joins in some database systems.

Another major difference is that many database management systems offer a lot of advanced functionality that SQL standards do not cover. These advanced features may be implemented in ways that are not consistent across different database systems. A very important design requirement of the JDBC API is that it must support SQL as it is rather than as the standards define it [2].

2.13 JDBC-ODBC bridge plus ODBC driver:

The JDBC-ODBC bridge product provides JDBC access via ODBC drivers. ODBC (Open Database Connectivity) predates JDBC and is widely used to connect to databases in a non-Java environment. ODBC is probably the most widely available programming interface for accessing relational databases.

The main advantages of the JDBC-ODBC bridge are as follows:

- It offers the ability to connect to almost all databases on almost all platforms.
- It may be the only way to gain access to some low-end desktop databases and applications [2]

2.14 Java Database Connectivity Process:

2.14.1 *Connection:*

A *Connection* reference (from the package *java.sql*) is created. An object that implements interface **Connection** manages the connection between the Java program and the database. **Connection** objects enable programs to create SQL statements that manipulate databases and to perform *transaction processing*.

The *DriverManager* class (package **java.sql**) attempts to connect to the database specified by its URL argument. The URL helps the program locate the database (possibly on a network or in the local file system of the computer). For Example, in the URL **jdbc:odbc:school**, jdbc specifies the *protocol* for communication, **odbc** the *subprotocol* for communication and **school**, the name of the database.

2.14.2 **Statement:**

The program uses the **Statement** object to submit SQL statements to the database. The **Statement** object's ***executeQuery*** method executes a query that selects all the required information from the specified tables.

2.14.3 **ResultSet:**

This contains the result of the query. The ResultSet methods enable the program to manipulate the query results.

2.14.4 **ResultSetMetaData**

The metadata describes the **ResultSet**'s contents. Programs can use metadata pro-grammatically to obtain information about the **ResultSet**'s column names and types.

2.14.5 **Prepared Statements:**

The java **PreparedStatement** interface enables an application programmer to create SQL statements that are maintained in a compiled form that enables the statements to execute more efficiently than **Statement** objects. **PreparedStatement** objects also are more flexible than **Statement** objects, because they can specify parameters.

2.14.6 **Transaction Processing:**

Many database applications require guarantees that a series of database insertions, updates and deletions executes properly before the applications continue processing the next data-base operation.

Transaction processing enables a program that interacts with a database to treat a database operation (or set of operations) as a single operation. Such an operation also is known as an *atomic operation* or a *transaction*. At the end of a transaction, a decision can be made either to *commit the transaction* or *rollback*

the transaction. Committing the transaction finalizes the database operation(s); all insertions, updates and deletions performed as part of the transaction cannot be reversed without performing a new database operation. Rolling back the transaction leaves the database in its state prior to the database operation. This is useful when a portion of a transaction fails to complete properly [3].

Chapter Three

Software Development and Design

3.1 The Back-End Database:

The administration software employs the use of Microsoft SQL Server 2000 as the back-end database. The name of the database is **ManagementDB** and it consists of two files.

(i) **ManagementDB.mdf**, which is the primary data file that contains the start-up information for the database and is used to store data. This file is stored, by default, in the "C:\Program Files\Microsoft SQL Server\MSSQL\Data" directory on the computer system to be used.

(ii) **ManagementDB_log.ldf**, which is the transaction log file that holds the log information used to recover the database. This file is also stored by default in the "C:\Program Files\Microsoft SQL Server\MSSQL\Data" directory on the computer system to be used.

3.2 Structure of the ManagementDB Database:

There are seven tables present in the ManagementDB database, namely:

(i) admin

(ii) PersonalInfo

(iii) CourseRegFirst_Semester

(iv) CourseRegSecond_Semester

(v) FIRST_SEMESTER

(vi) SECOND_SEMESTER

(vii) Courses

With each having dedicated fields (columns), defined with specific data types, that can hold only specific kinds of data relating to the students.

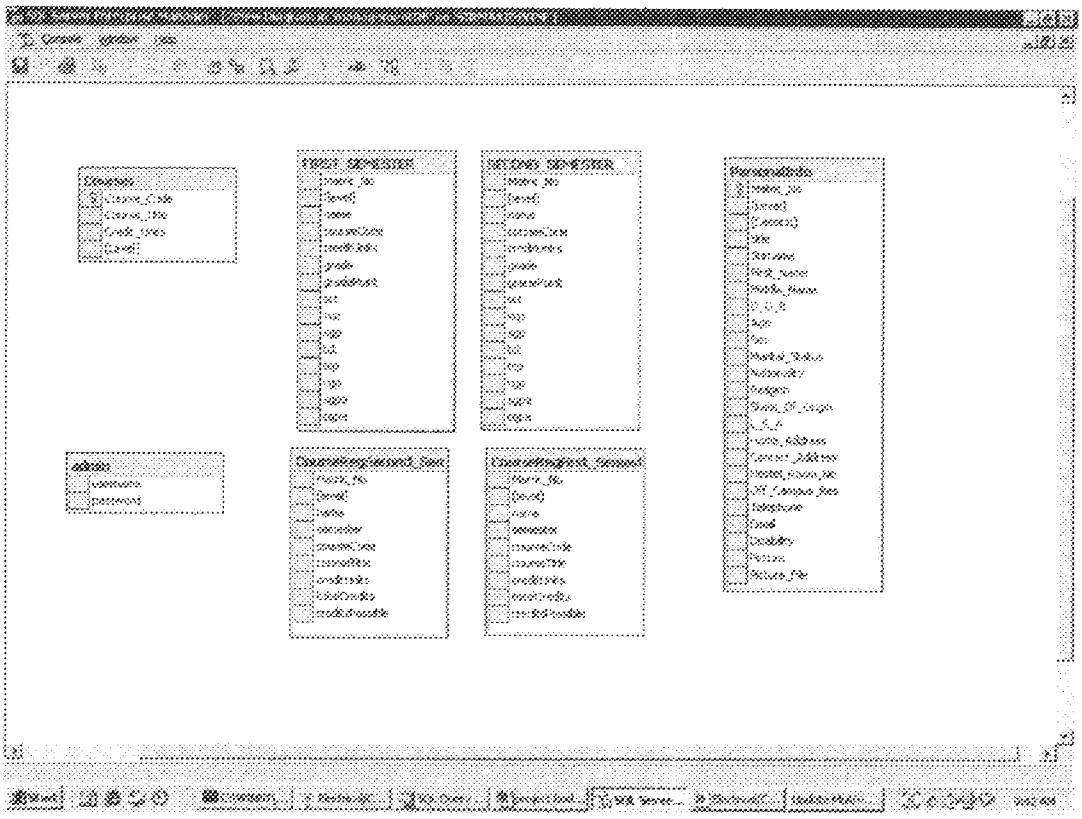


Figure 3.1: The ManagementDB diagram showing all the tables

A list of the tables and a brief description of their function is given below and more detailed information on the structures of the tables is given in Appendix 1.

Table 3.1: List of tables and brief description of function

Table	Brief Description of function
Admin	This table holds the user name and password of the authorized user of the software
PersonalInfo	This table holds information regarding personal details of the student, including his matriculation number, address, picture and other vital student personal data
CourseRegFirst_Semester	This table holds information regarding all courses registered for by the student in the first semester of an academic year
CourseRegSecond_Semester	This table holds information regarding all courses registered for by the student in the first semester of an academic year
FIRST_SEMESTER	This table stores students' academic records, including his grades and grade points for the first semester
SECOND_SEMESTER	This table stores students` academic records, including his grades and grade points for the first semester
Courses	This table holds all the courses available in the department from 100 level to 500 level

3.3 Front-end of the Software:

The front end of the administration software was designed in Java language, using the JDK 1.5 software. Departmental administrative tasks handled by the software are: Student Registration, involving entering the students' Personal Information and Course Registration, Students Academic Records compilation and generating Transcripts for the students.

3.4 Operation of the Software:

When the software is executed, a splash screen appears for a few seconds, after which a login dialog box appears. The authorised user must enter a valid username password (which must both exist in the ManagementDB database) to gain access to the main program.

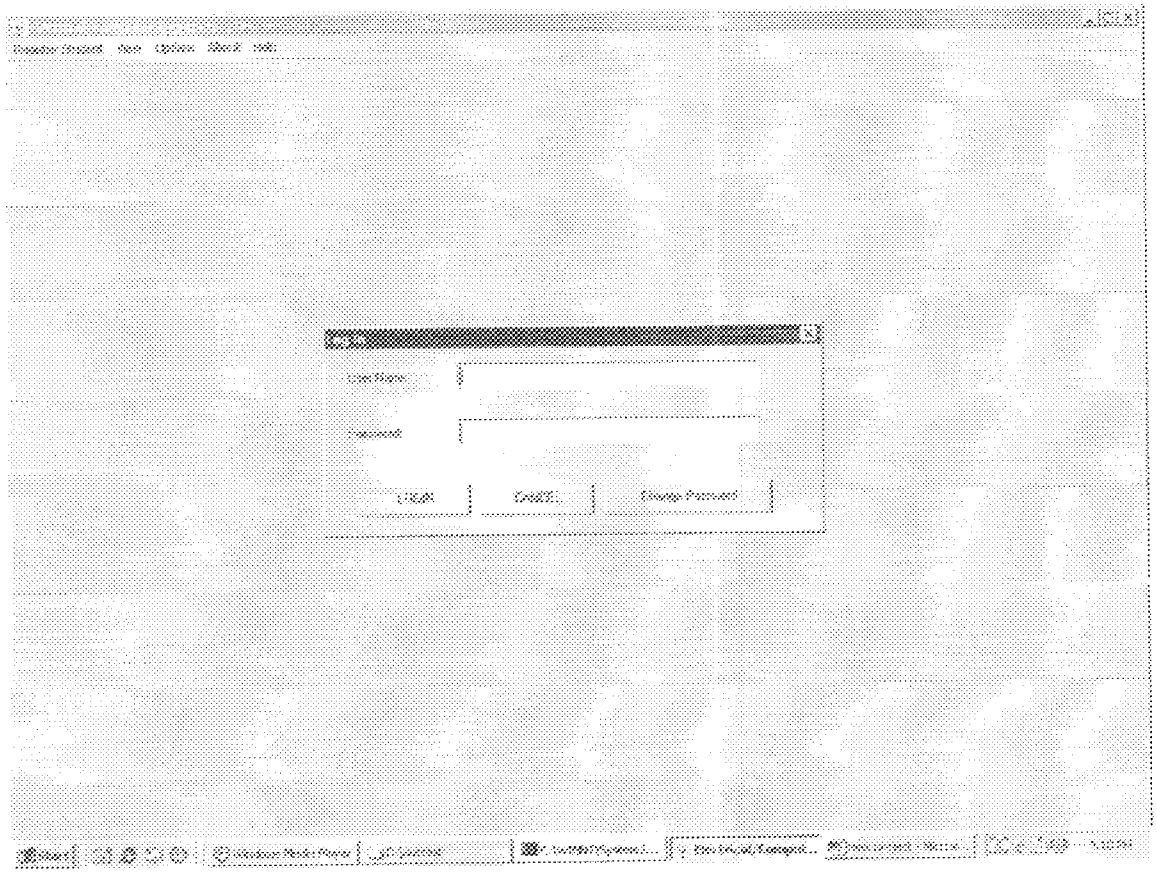


Figure 3.2: Login for the software

The user of the program then has the following options on the menu bar of the program:

(i) Register Student

(ii) View

(iii) Update

(iv) About

(v) Help

3.4.1 **Register Student:**

At the click of the Register student menu button, the user of the program has options to make a fresh student registration based on:

(i) Personal Information

(ii) Course Registration

(iii) Academic Records.

The software provides dedicated forms through which the application user can enter information for the students based on the above specified data requirements.

3.4.1.1 **Personal Information entry Form:**

Personal Information for students, including matriculation number, full name, permanent home address, picture and other important student details are entered into the form and at the click of the *insert* button the data is transferred into the PersonalInfo table in the ManagementDB database. This form must be the first to be filled for each student registered, as filling other forms requires that the student must have already have their personal information filled into the database through the form.

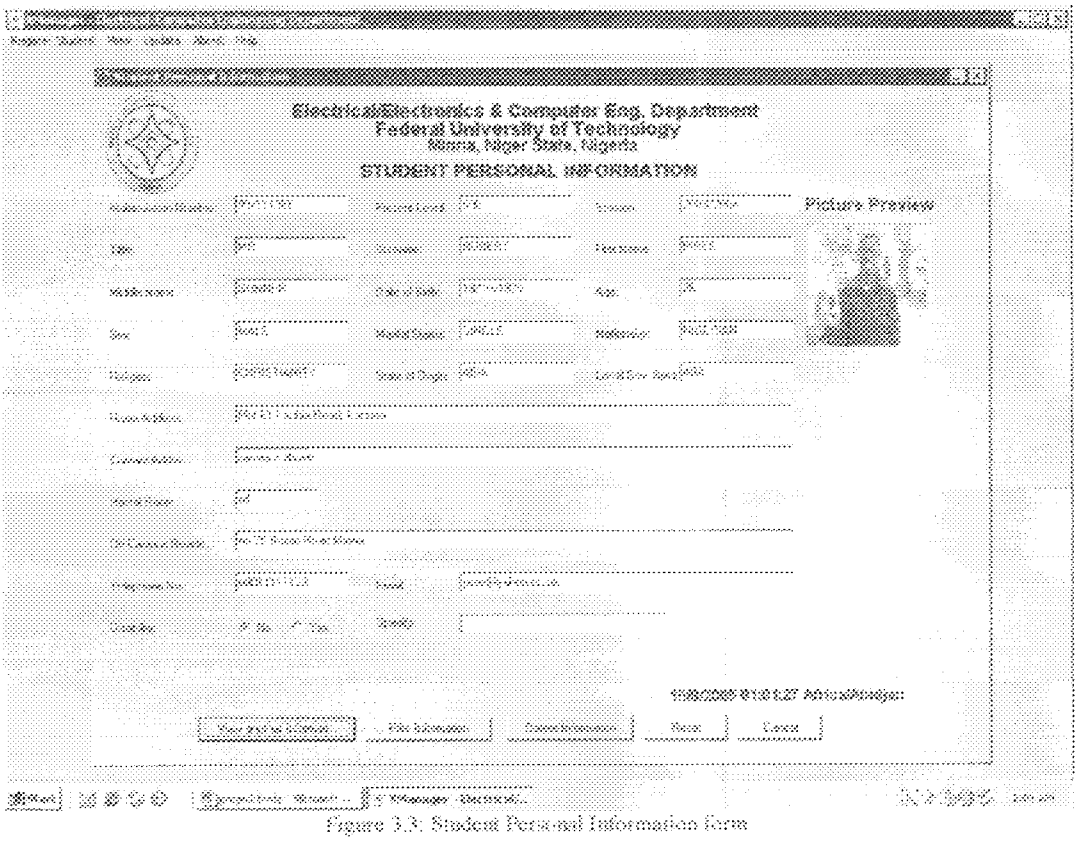


Figure 3.3: Student Personal Information form

3.4.1.2 **Course Registration entry Form:**

Course registration is done on a per level and per semester basis. Before the user makes a fresh course registration he must supply the matriculation number for the student he wants to register, which must already have been entered into the PersonalInfo table of the ManagementDB database. All available courses are listed on the **Available Departmental Courses** table of this form and the user has to highlight the required courses for registration one at a time, and at the click of the **AddCourse** button the courses are added to the **Semesterial Courses** table of the form. The students are not allowed to register courses totalling more than 24 credits units per semester and any attempt to do this will be met by an error message. At the click of the **Register Student** button, the information is transferred into the **CourseRegFirst_Semester** and **CourseRegSecond_Semester** tables in the **ManagementDB** database.

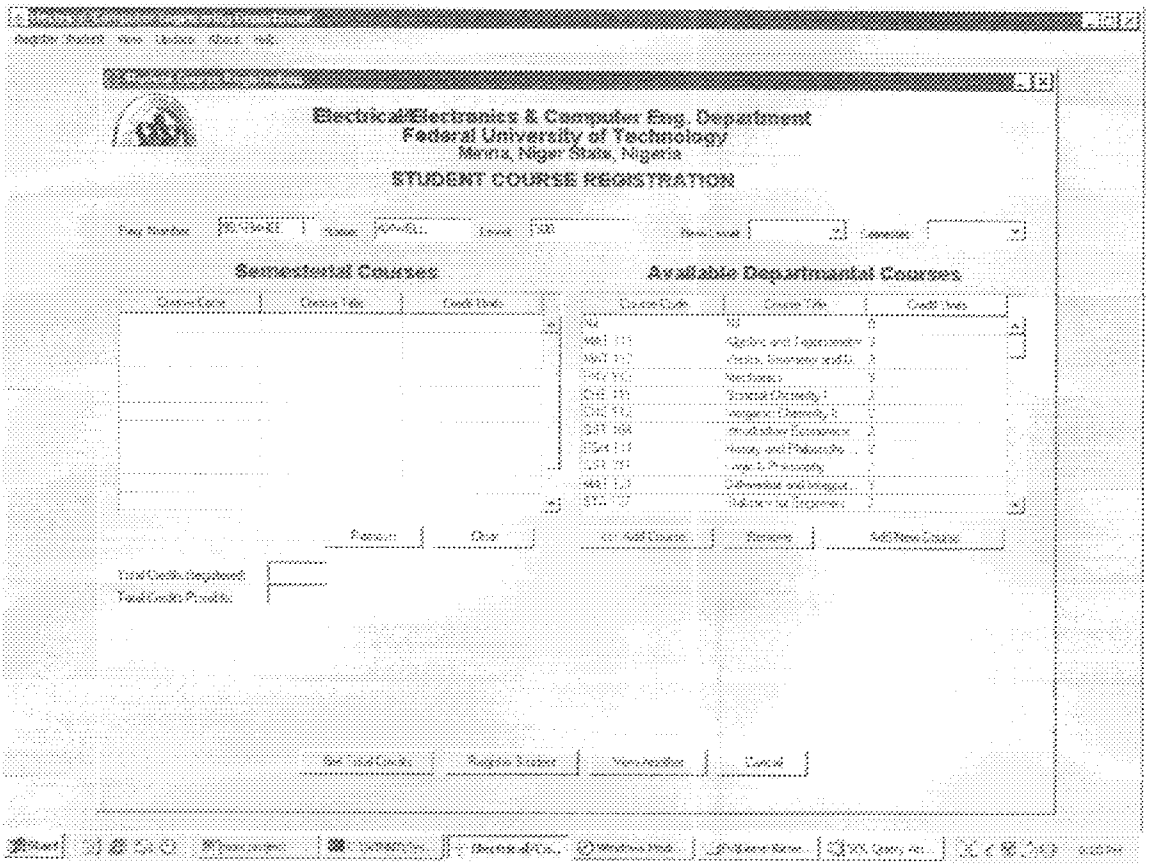


Figure 3.4: Student Course Registration form

3.4.1.3 **Academic Record entry Form:**

Here, the program prompts the user to enter an already existing matriculation number before the user can proceed to enter fresh academic records for the student. The user of the program specifies the level for which he wants to enter academic results for the student and by default, courses means for that level and semester appear at the appropriate tables, based on the courses that were registered for the student. At the click of the **Get GP** button, the grade points for each of the grades are displayed and on clicking the **Compute Result** and **Insert Record** buttons, which are inactive by default, results are computed and the records are transferred into the **FIRST_SEMESTER** and **SECOND_SEMSTER** tables respectively.

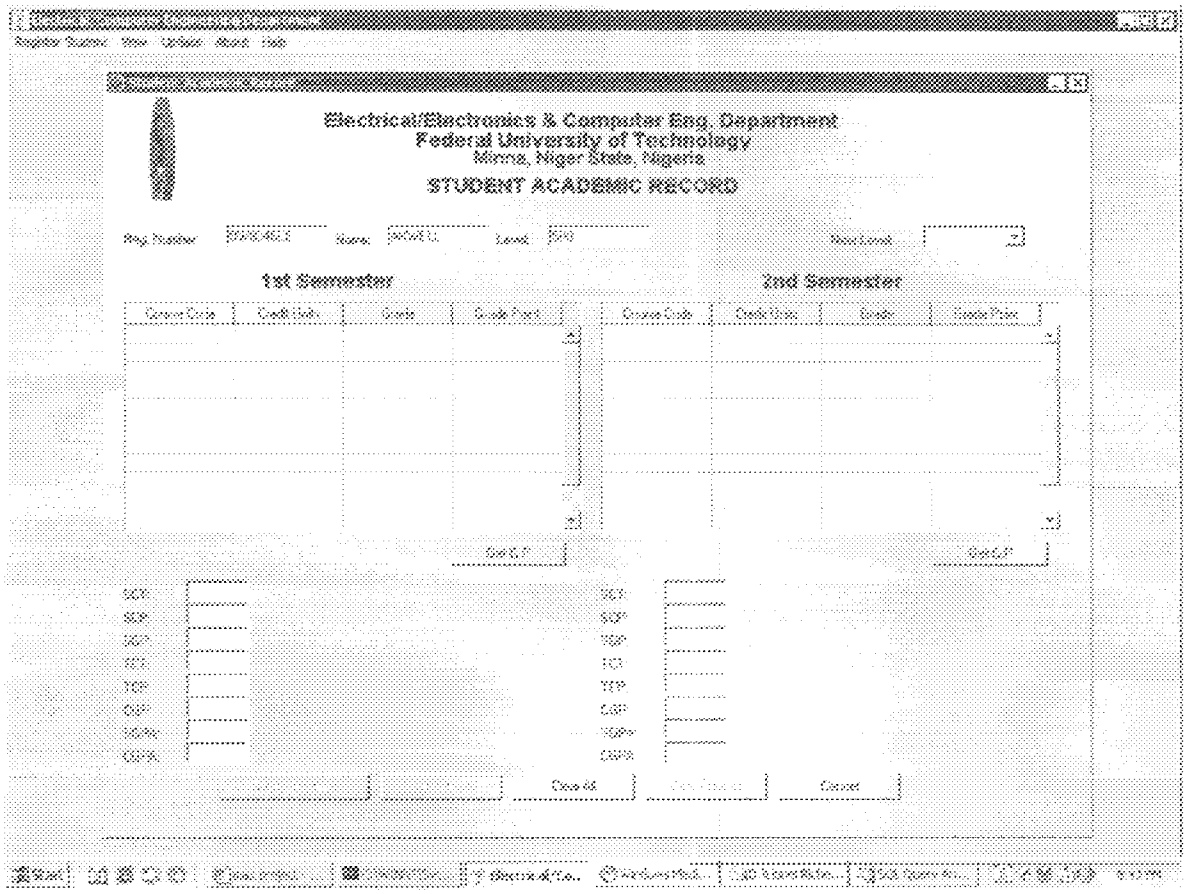


Figure 3.5: Student Academic Record Form

3.4.2 **View:**

At the click of the View menu button, the user of the program has options to view an existing student registration based on:

(i) Personal Information

(ii) Course Registration

(iii) Academic Records

(iv) Transcript

The program prompts the user to enter a matriculation number, which must already exist in the database before any view can be made.

3.4.2.1 **Personal Information view Form:**

The authorised user of the application can view already existing personal information of a student with this form and while in that view, can print out the information at the click of the *Print Information* button and can view personal information of another student by clicking the *view another information* button and supplying the matriculation number of the required student in the dialog box that appears.

3.4.2.2 **Course Registration view Form:**

The application prompts the user for the matriculation number of the required student and after the right number is supplied, the user is expected to select the particular level of course registration for the student, for which he wants to view. The user can view course registration for another student and can print out the course registration form by clicking the *view another* and *print record* buttons respectively.

3.4.2.3 **Academic Records view Form:**

The application prompts the user for the matriculation number of the required student and after the right number is supplied, the user is expected to select the particular level of academic records (including the cumulative grade points and academic status) for the student, for which he wants to view. The user can view academic records for another student and can print out the academic records form by clicking the *view another* and *print record* buttons respectively.

3.4.2.4 **Transcript**

Students transcripts, which is a very important part of the software can be generated based on the academic records of the student. When the transcript

view is run, a form appears where the user can fill additional student information required for the transcript after which the user enters the transcript proper. The user can print out the transcript for a level of the student that is required by clicking the appropriate buttons shown on the form. A Sample printout of a transcript for a student is shown in the appendix and below is a screen capture showing the 100 transcript for a student.

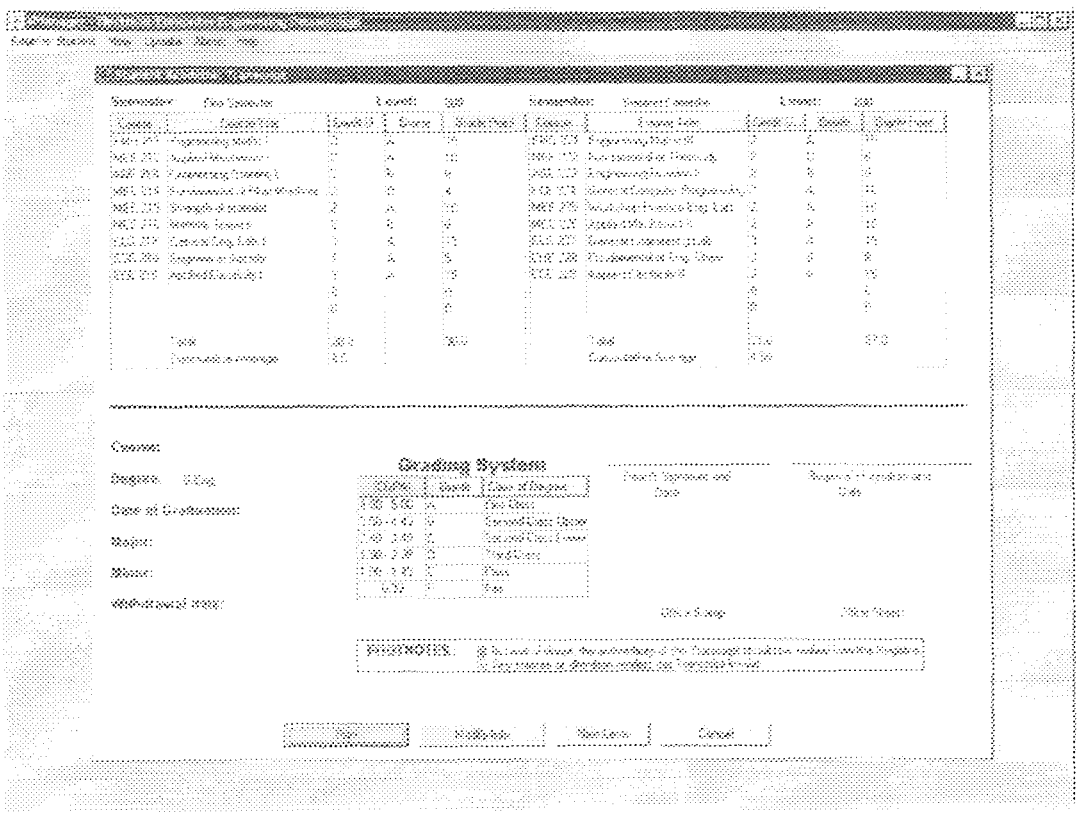


Figure 3.6: Sample Transcript for a student

3.4.3 **Update:**

At the click of the **Update** menu button, the user of the program has options to make changes to already existing student information based on:

(i) Personal Information

(ii) Course Registration

(v) Academic Records.

The program prompts the user to enter a matriculation number, which must already exist in the database before any update can be made.

3.4.4 About

The *about* menu has provision to check up the system information of the computer system on which the software and the SQL Server is installed. System information like the hardware requirements and software environment of the system can be viewed at the click of the *system info* button.

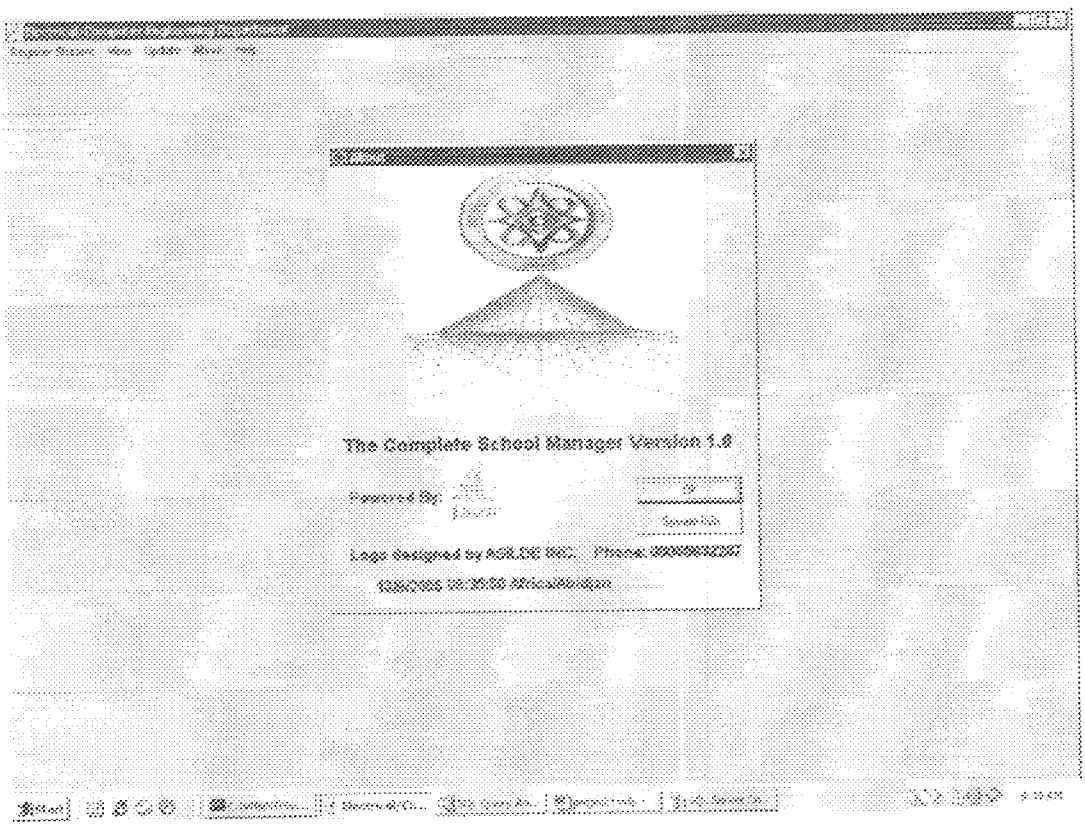


Figure 3.7: The *About* view

3.4.5 Help

Contains information about the software, guidance on how to navigate through the software, troubleshooting tips in the event of any problems

encountered and other information that greatly assist the user in utilising the software without difficulty.

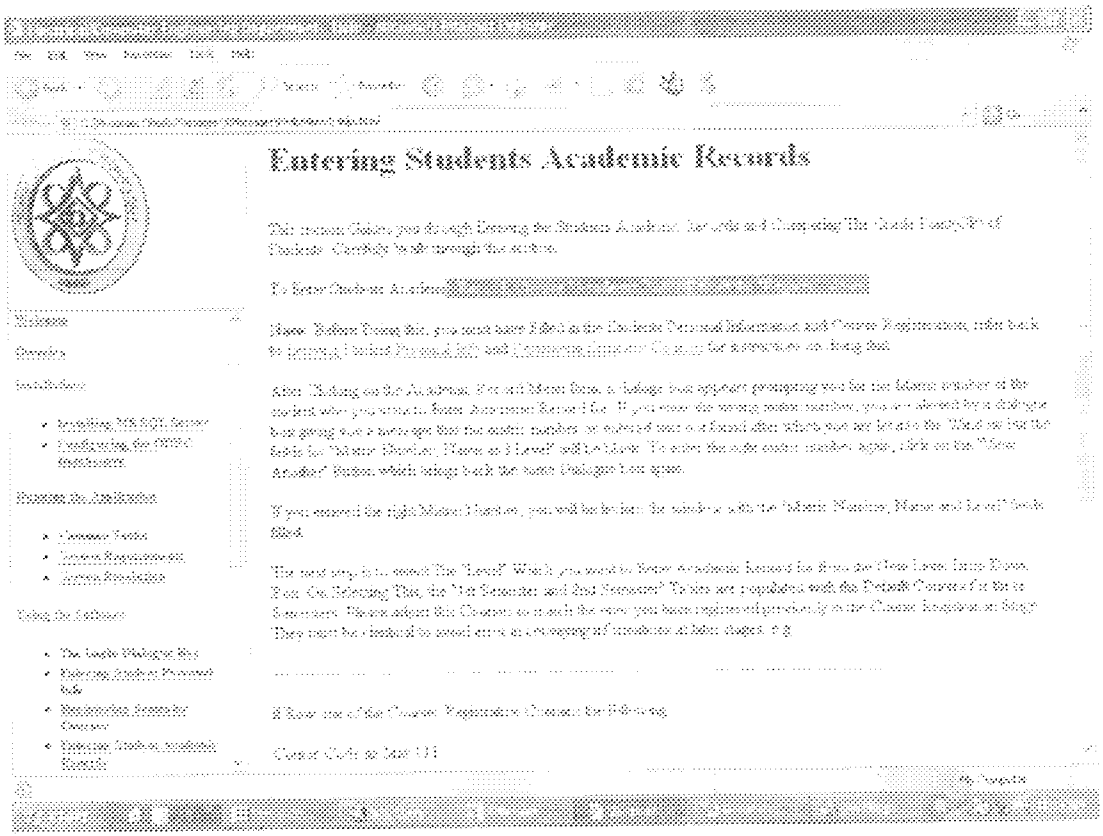


Figure 3.8: *Help* for the Software

3.8 Installer for the Program

Microsoft SQL Server 2000 must first be installed on the system it is required to install the XManager software, since it is the database program that stores the data entered using the software.

Also, every other versions Java present on the system must be uninstalled before the software is installed.

Once the XManager CD is slotted on the computer system and the contents of the CD drive displayed, the setup icon is double clicked and the installation screen comes up. Once the user chooses to install the software, the

first of the installation is extraction of the Java Runtime Environment (version 1.5).

After that, the SQL codes that create the databases and tables are run, using OSQL, which is command-line utility used to connect to SQL Server and execute Transact-SQL statements. The results of the executed commands will be displayed in the DOS console window. OSQL uses the ODBC (Open Database Connectivity) interface to connect to SQL Server.

After the computer system restarts, a link between the XManager software and SQL Server is created by navigating through the menus: Start->Programs->XManager and clicking on the ODBCConnect menu.

The program can now be accessed by navigating through Start->Programs->XManager and clicking on XManager. A default user name and password (as supplied on the XManager CD) is required to gain access to the XManager Software each time it starts up.

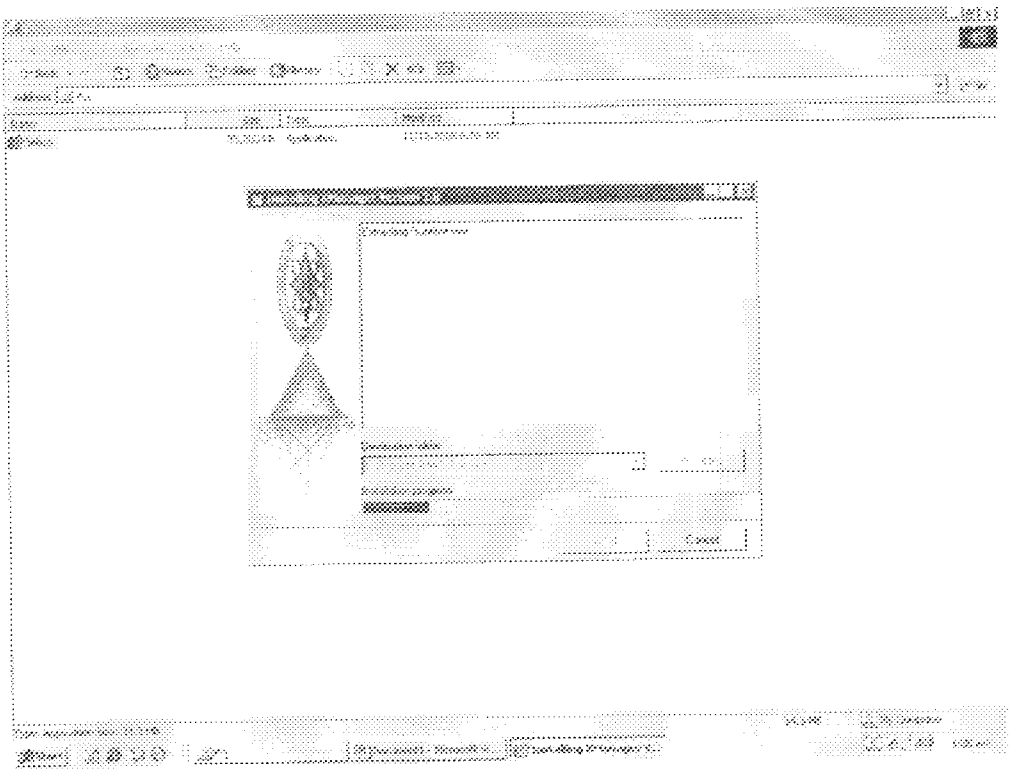


Figure 3.9. Screen capture for installation of the XManager Software

Chapter Four

Tests, Results and Discussion of Results

4.1 Tests:

The importance of software testing and it's implications with respect to software quality cannot be over emphasised. Testing a software involves checking whether it does what it is meant to do, if data entry can make the system 'crash', whether constraints work etc.

All new software systems must be tested thoroughly. It is realised that complete testing can never really be possible except on the simplest of programs. One can never be completely certain that all errors have been removed, but sufficient tests can be performed to give a reasonable measure of confidence in the software. Some testing procedures are analysed below:

4.1.1 Unit Testing:

Individual components of the software are tested to ensure that they operate correctly. Unit testing treats each component as a stand-alone entity which does not need other components during the testing process. In this project, every single menu, such as Register Student menu, View menu, Update menu, About menu and Help menu, has been checked individually.

4.1.2 Integration Testing:

Integration testing involves checking all the modules together. This is necessary, even after a successful unit testing, to avoid data loss across an interface as one module may have an inadvertent, adverse affect on another.

4.1.3 System Testing:

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer based system. Simply here the software as well as hardware will be integrated and also the testing verifies that all elements mesh properly and that overall system function/performance is achieved. In this project, some of the subsystems that are integrated to make up the entire system are the Java Package, SQL Server, Hard drive and memory, among others.

4.2 Case Studies:

Tests, using real life scenarios are made to ensure that the software conforms to acceptable standards. The validation of calculations used within the program forms an important part of the test plan.

4.2.1 Case 1:

It is required to compute the cumulative grade point for Denis Jiya Law, with matriculation number 99/9999EE, who is a graduating 500 level student. Computation of his result should begin from the first semester of his 100 level, right to his 500 level.

4.2.2 Case 2:

Personal information regarding a student in the Electrical/Computer Engineering Department with the matriculation number 99/1111EE is required by the School's Senate. A print out of this information is to be issued the Senate by the department.

4.2.3 Case 3:

A 300 level student, Saha Christiano Smith with matriculation number 99/0115EE is required to obtain his 200 level transcript which is an important document required when for applying for the Commonwealth Scholarship Exams for undergraduates. He is to obtain the transcript from the Department after his application for same has been approved

4.3 Results of Tests from the Case Studies:

4.3.1 Case 1 Results:

Table 4.1 below shows manual computations of Mr. Denis Jiya Law's results right from his first (100) level to his last (500) level. His result were also calculated using the software and both computations compared and a screen capture for his 500 level results calculated using the software is shown in Figure 4.1 below.

Table 4.1: Manual results computation for Mr. Denis Jiya Law

Course Code	Credit unit	Grade	Grade point	Course Code	Credit unit	Grade	Grade point
100 Level First Semester				100 Level Second Semester			
MAT 111	3	A	15	MAT 121	3	A	15
MAT 112	3	A	15	STA 127	3	A	15
PHY 113	3	A	15	CHE 121	3	A	15
CHE 111	3	A	15	PHY 126	3	A	15
CHE 112	2	B	8	PHY 123	2	B	8
GST 110	3	C	9	CHE 191	2	A	10
GST 104	2	A	10	GST 103	2	C	6
MIC 111	2	B	8	GST 121	3	A	15
GST 211	2	A	10	PHY 108	2	B	8
SCT: 23		TCP: 105		SCT: 23		TCP: 107	
SGP: 23		CGP: 105		SGP: 46		CGP: 212	
SGPA: 4.56		CGPA: 4.56		SGPA: 4.65		CGPA: 4.61	
200 Level First Semester				200 Level Second Semester			
EMG 211	3	A	15	EMG 221	3	A	15
MEE 212	2	A	10	MEE 222	2	C	6
AGE 213	2	B	8	MEE 223	2	B	8
MEE 214	2	D	4	ECE 224	2	A	10
MEE 215	2	A	10	MEE 225	2	A	10
MEE 216	2	B	8	MEE 226	2	A	10
ELG 217	3	A	15	ELG 227	3	A	15
ESG 218	1	A	5	CHE 228	2	B	8
ECE 219	3	A	15	ECE 229	3	A	15
SCT: 20		TCP: 90		SCT: 21		TCP: 97	
SGP: 66		CGP: 302		SGP: 87		CGP: 399	
SGPA: 4.5		CGPA: 4.58		SGPA: 4.62		CGPA: 4.59	
300 Level First Semester				300 Level Second Semester			
EMG 311	3	A	15	EMG 321	3	A	15
MEE 312	2	B	8	ECE 321	2	B	8
ECE 311	2	A	10	ECE 322	2	B	8
ECE 312	3	A	15	ECE 323	3	A	15
ECE 313	2	B	8	ECE 324	2	A	10
ECE 314	2	A	10	ECE 325	2	C	6
ECE 315	3	A	15	ECE 326	2	C	6
ECE 316	3	A	15	ECE 327	3	A	15
				ECE 328	2	A	10
SCT: 20		TCP: 86		SCT: 21		TCP: 93	
SGP: 187		CGP: 495		SGP: 138		CGP: 588	
SGPA: 4.8		CGPA: 4.63		SGPA: 4.43		CGPA: 4.92	
400 Level First Semester				400 Level Second Semester (NOT)			
AGE 412	1	D	2				
ECE 411	3	A	15				
ECE 412	3	B	12				
ECE 413	3	A	15				
ECE 414	3	B	12				
ECE 415	3	A	15				
ECE 416	3	A	15				
SCT: 19		TCP: 86					
SGP: 147		CGP: 674					
SGPA: 4.53		CGPA: 4.58					
500 Level First Semester				500 Level Second Semester			
MEE 515	3	A	15	ECE 521	2	B	8
ECE 511	2	A	10	ECE 522	3	A	15
ECE 512	2	B	8	ECE 523	3	A	15
ECE 513	2	A	10	ECE 524	2	A	10
ECE 514	2	A	10	ECE 525	6	A	30
ECE 526	2	B	8	ECE 529	2	A	10
ECE 517	2	A	10				
ECE 527	2	C	6				
SCT: 17		TCP: 77		SCT: 18		TCP: 86	
SGP: 264		CGP: 781		SGP: 162		CGP: 859	
SGPA: 4.83		CGPA: 4.58		SGPA: 4.69		CGPA: 4.61	

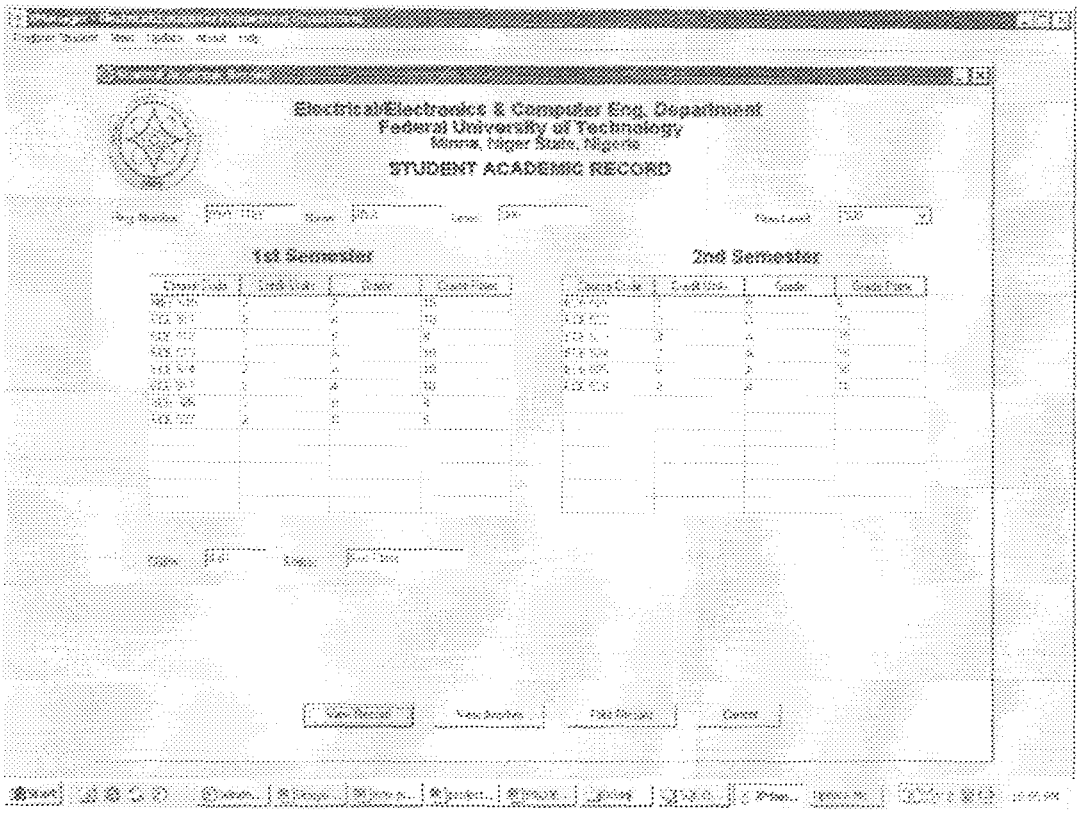


Figure 4.1: Screen Capture showing 300 level result for Mr. Denis Jiya Law

4.3.2 Case 2 Results:

A print out of the personal information for the student can be gotten using the XManager software by clicking on the **view** menu and then **Personal Information.** The student's matriculation number, 99/11UEE, is then entered in the prompt that shows up, after which the student's personal information are displayed. At the click of the **Print Information** button the student's personal information is printed. A print out of the personal information is show in the appendix.

4.3.3 **Case 3 Results:**

Transcript for the student can easily be gotten using the XManager software by clicking on view and then Transcript. The student's matriculation number, 99/0115EE is then entered in the prompt that shows, after which the student's transcript is displayed. The transcript can then be printed by clicking on the **Print** button. A print out of his transcript is shown in the appendix.

4.4 Discussion of Results

The academic results gotten using XManager software tallied with the one gotten through manual computation

Generating students' transcript using the XManager software is a fast and relatively easy process which is gotten at few clicks of the button as compared to having to manually type information concerning the student's results on a transcript paper, using a typewriter.

Print outs from the XManager software regarding the student's personal information was gotten in a very short time and only the matriculation number of the required student had to be specified.

Chapter Five

Conclusion and Recommendation

5.1 Conclusion

Important administrative tasks in the Electrical/Computer Engineering Department will be automated by utilising the XManager software. The software can be used to successfully enter and store students' personal information, carry out course registration, compute results, generate transcripts and print out data for students based on the above listed tasks. These tasks are accomplished in very short time compared with carrying them out manually. There is also a considerable reduction in stress involved in carrying out the tasks with the software and fewer errors are possible.

5.2 Recommendation

It is recommended that the Electrical/Computer Engineering Department is automated with the XManager software to ease departmental administrative tasks. Below are the recommended hardware and software for the automation process:

5.2.1 Types of Software:

- Microsoft SQL Server 2000 Software
- The XManager Software

5.2.2 Types of Hardware

- Pentium III computer systems with processing speed of at least 1 GHz, RAM of at least 256MB and a hard disk space of about 30GB, dedicated to storing and handling student information.
- Printers
- Scanners/digital cameras

REFERENCES:

[1] Cay S. Horstmann, Gary Cornell, Core Java 2, Volume 2: Advanced Features, *Prentice Hall PTR*, December 13, 2001, pp. 322.

[2] John O'Donahue, Java Programming Database Bible, *John Wiley and Sons*, 2002, pp. 6-7, 113-115, 118.

[3] H. M. Dietel, P. J. Deitel, S. E. Santry, Advanced Java 2 Platform :How to Program, *Prentice Hall, New Jersey*, 2001, pp 446, 453-454, 467-468, 480, 482.

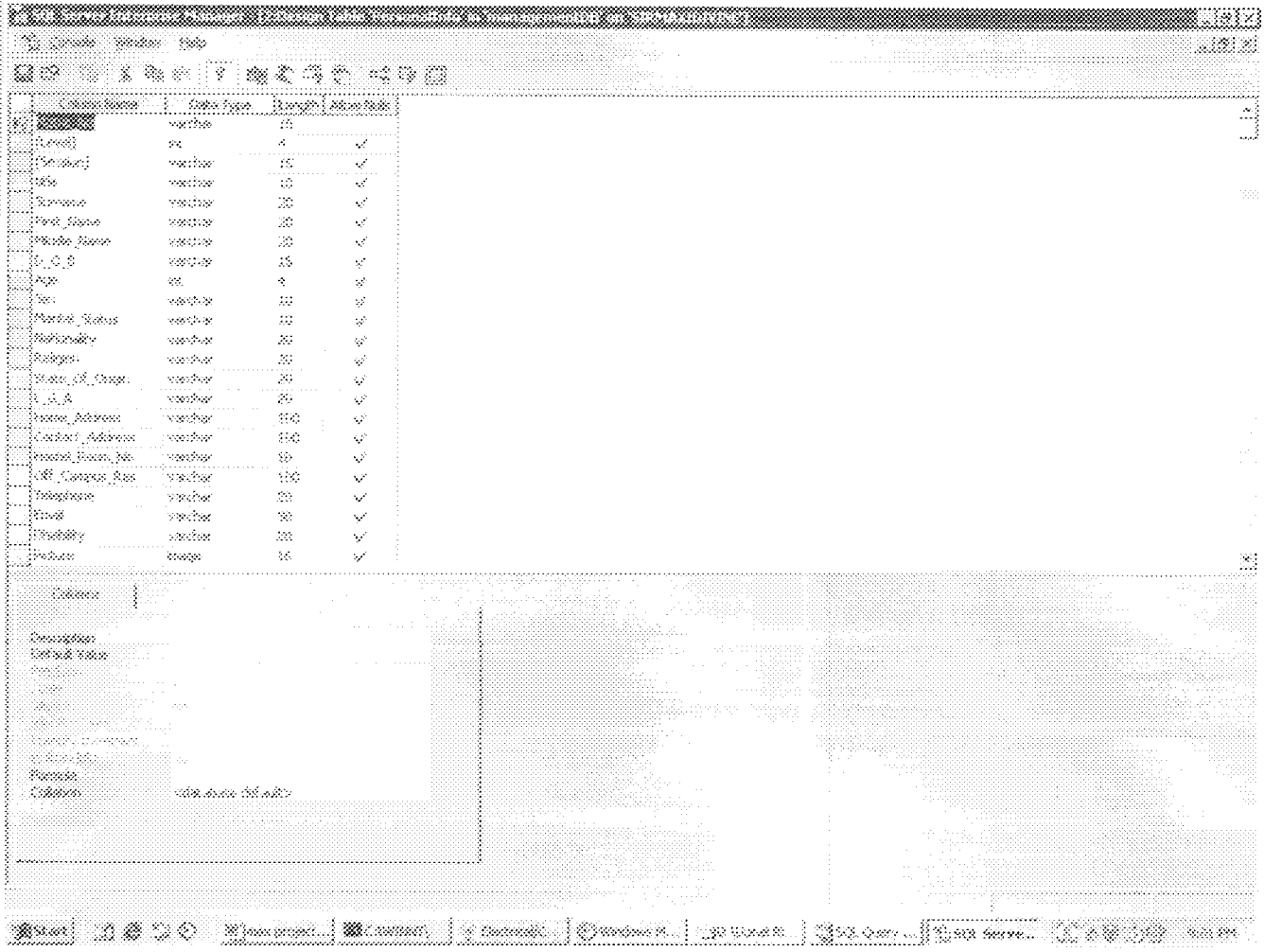
[4] Alex Kriegel, Microsoft SQL Server 2000 Weekend Crash Course, Hungry Minds Inc., New York, 2001, pp 5-6, 44,

[5] Fernando G. Guerrero, Carlos Eduardo Rojas, Microsoft SQL Server 2000 Programming by Example, *Que Corporation*, 2001, pp 1.

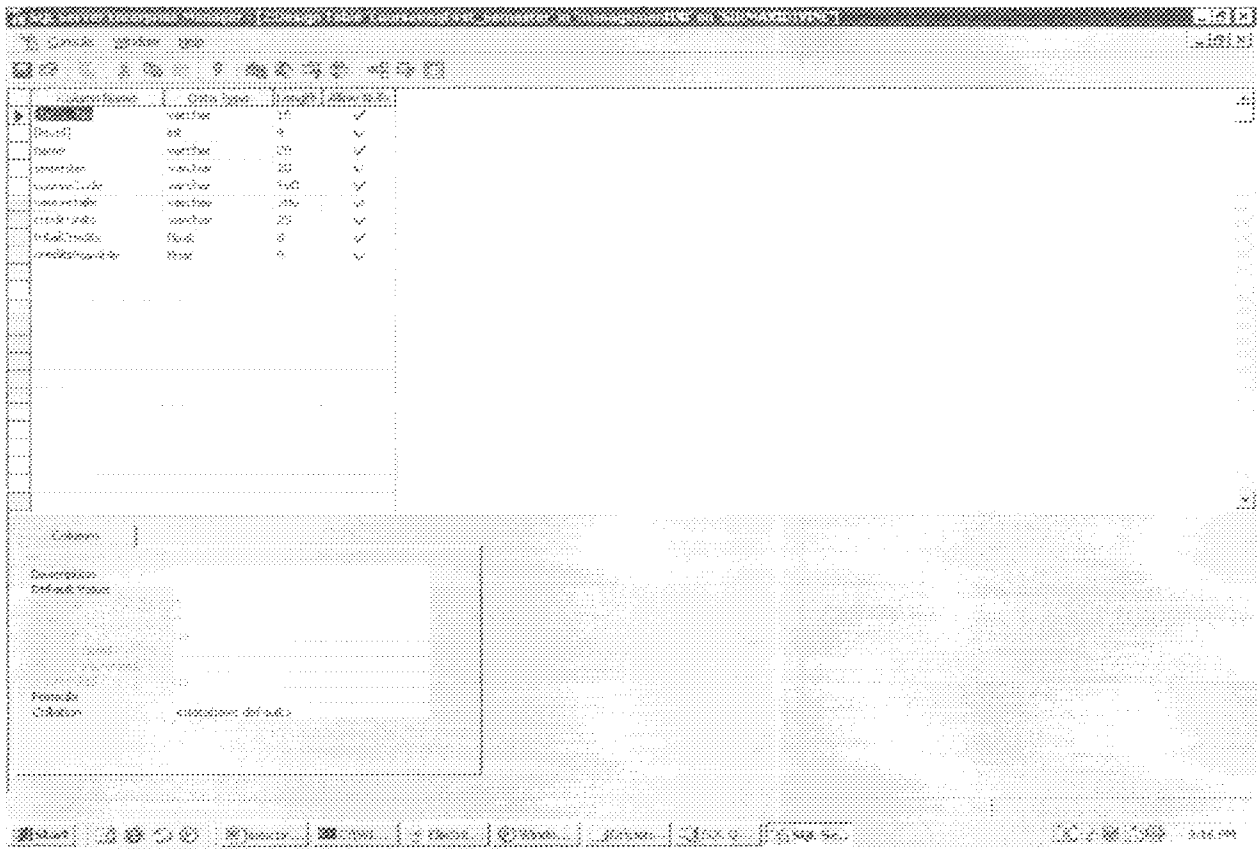
[6] Sun Microsystems website http://java.sun.com/docs/books/tutorial/getStarted/index.html

APPENDIX

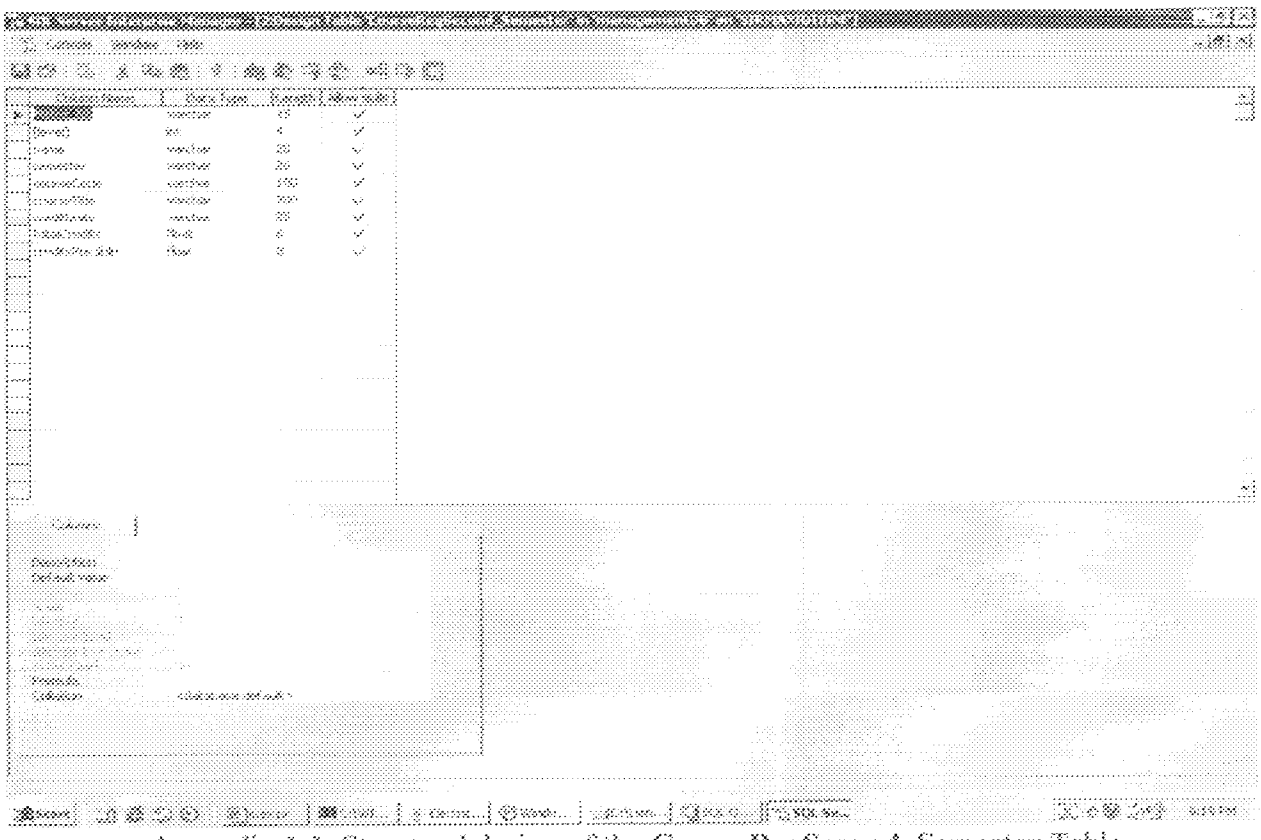
APPENDIX I
SAMPLE STRUCTURES OF THE TABLES IN THE ManagementDB DATABASE



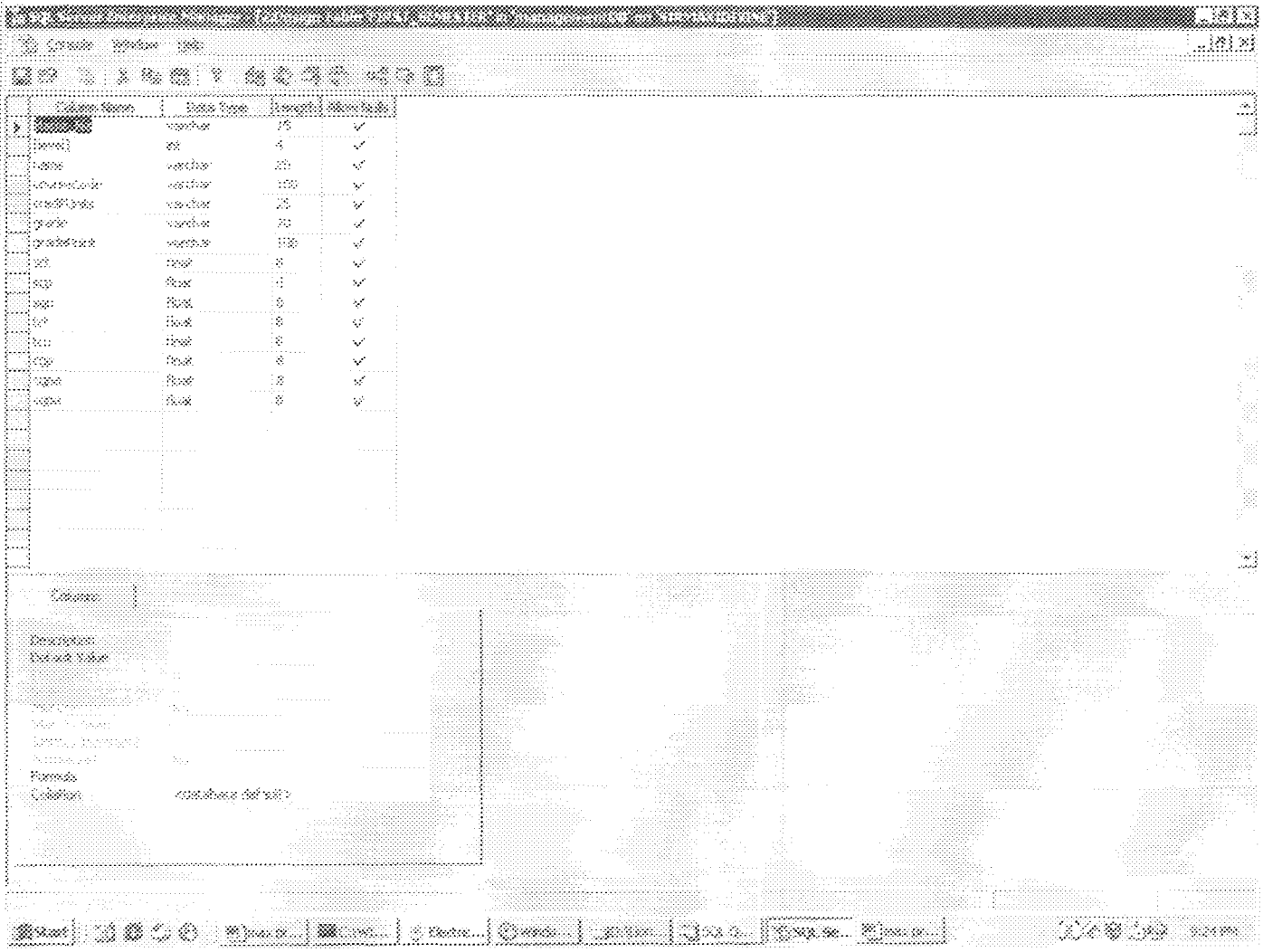
Appendix 1.1: Structural Design of the PersonalInfo Table



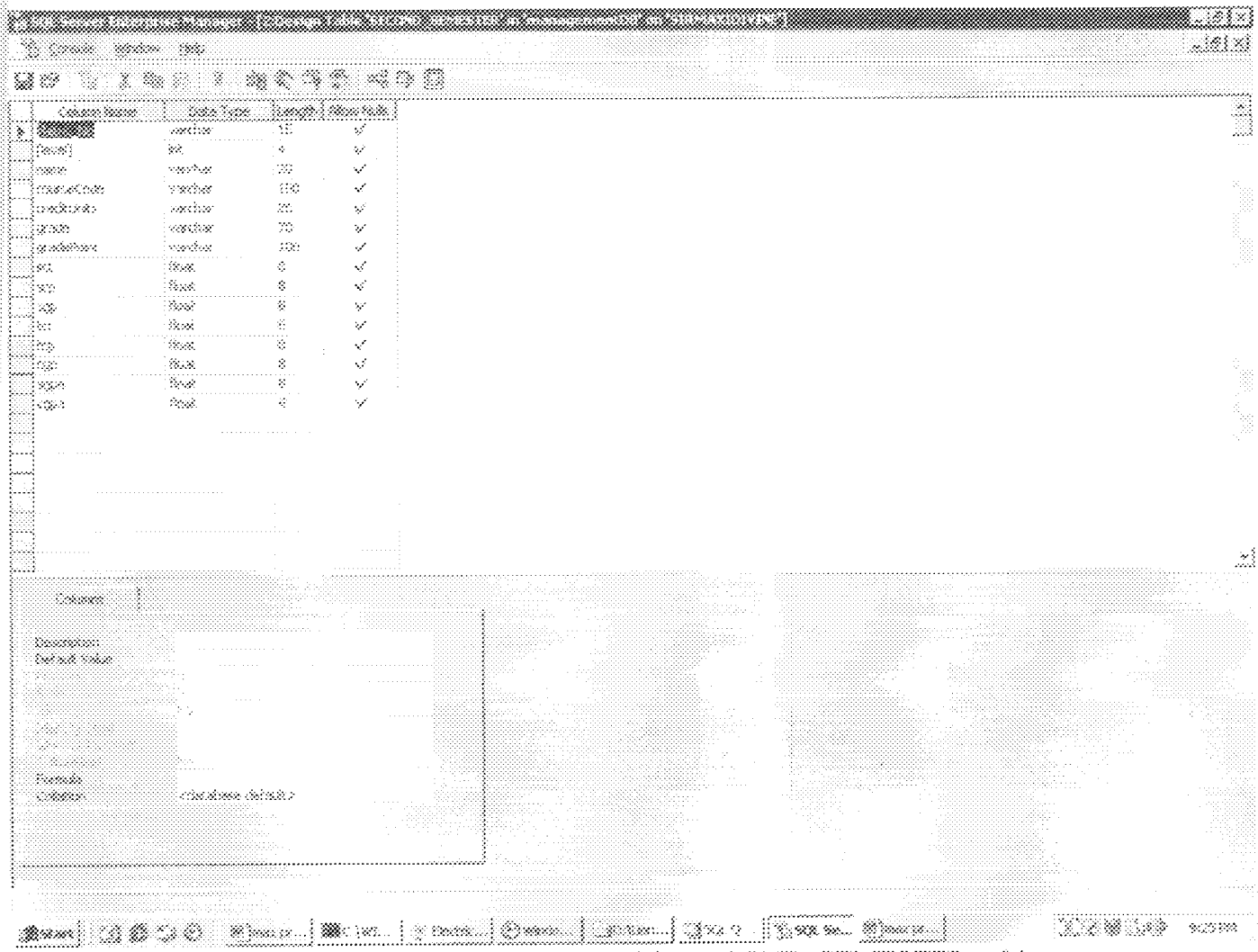
Appendix 1.2: Structural design of the **CourseRegFirst_Semester** Table



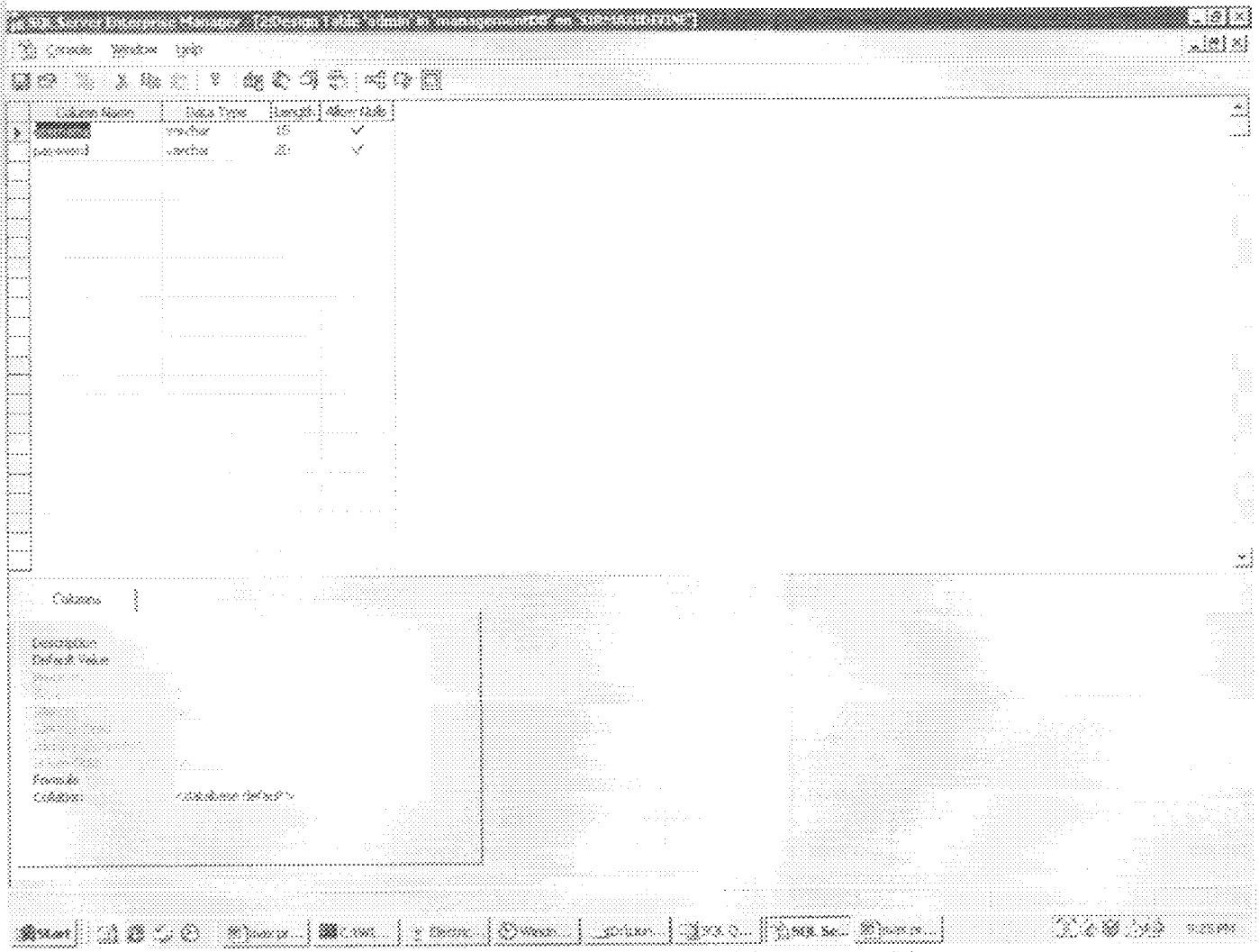
Appendix 1.3: Structural design of the **CourseRegSecond_Semester** Table



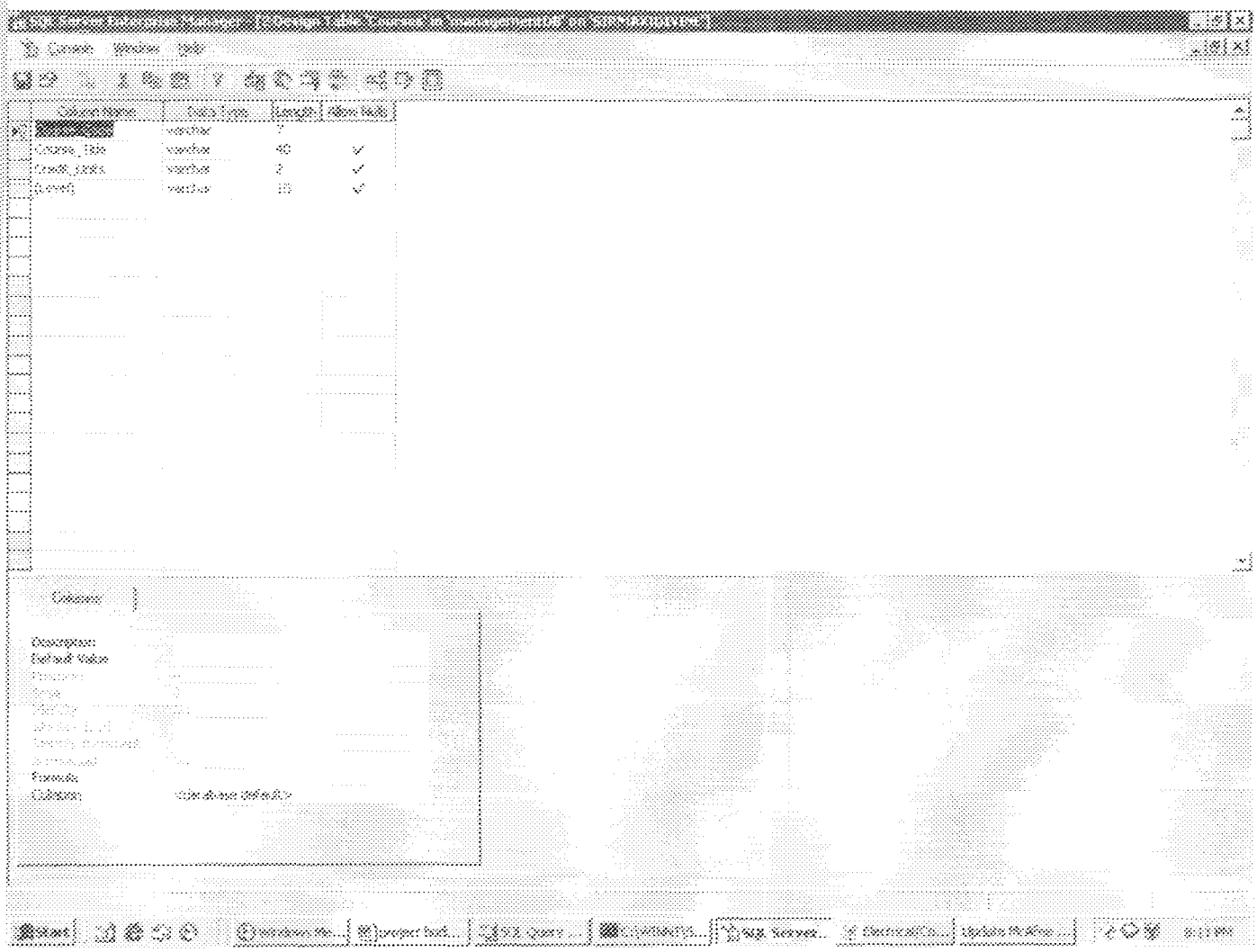
Appendix 1.4. Structural design of the **FIRST_SEMESTER** table



Appendix 1.9 : Structural design of the SECOND_SEMESTER table



Appendix 1.6: Structural design of the **admin** table



Appendix 1.7: Structural design of the **Courses** table

APPENDIX 2

SAMPLE CODES FOR THE MODULES USED IN THE FRONT-END SOFTWARE

The Student Personal Information Registration Form:

Saved As: PersonalInformation.java

Code:

```
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.BorderFactory.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.sql.*;
import java.util.*;
import javax.infobus.*;
import java.io.*;
import java.applet.*;
import java.awt.Graphics.*;

//Creating the logo
 class Logo1 extends Frame implements Icon
 {
 int width;
 int height;
 Image image;
 Toolkit toolkit;
 public Logo1(int w, int h)
 {
  width = w;
  height = h;
  toolkit = getToolkit();
  image = toolkit.getImage("LOGO2.gif");
 }

 public int getIconWidth()
 {
  return width;
 }
 public int getIconHeight()
 {
  return height;
 }

 public void paintIcon(Component c, Graphics g, int x, int y)
 {
  //Image im = image.getScaledInstance(getIconWidth(), getIconHeight(), Image.SCALE_DEFAULT);
  g.drawImage(image, 0, 0, getIconWidth(), getIconHeight(), this);
```

```
  }
}


class PersonalPicture1 extends Frame implements Icon
{
int width;
int height;
String pictureName;
Image image;
Toolkit toolkit;
public PersonalPicture1(int w, int h, String picName)
{
 width = w;
 height = h;
 pictureName = picName;
 toolkit = getToolkit();
 image = toolkit.getImage(pictureName);
 }

 public int getIconWidth()
 {
  return width;
 }
 public int getIconHeight()
 {
  return height;
 }

 public void paintIcon(Component c, Graphics g, int x, int y)
 {
  //Image im = image.getScaledInstance(getIconWidth(), getIconHeight(), Image.SCALE_DEFAULT);
  g.drawImage(image, 0, 0, getIconWidth(), getIconHeight(), this);
 }
}

public class PersonalInformation extends JPanel implements Runnable
{
 JLabel school;
 JLabel place;
 JLabel topic;
 Icon logo;
 Icon picture;
 Icon imco;
 JLabel logoLabel;
 JLabel pictureLabel;

 JLabel matricNo;
 JLabel level;
 JLabel session;
 JLabel title;
 JLabel surname;
 JLabel firstName;
 JLabel middleName;
 JLabel DOB;
```

```
JLabel age;
JLabel sex;
JLabel maritalStatus;
JLabel nationality;
JLabel religion;
JLabel SOR;
JLabel LGA;
JLabel homeAddy;
JLabel contactAddy;
JLabel HRN;
JLabel OCR;
JLabel telNo;
JLabel email;
JLabel disability;
JLabel no;
JLabel yes;
JLabel specify;
JLabel pictureFile;
JLabel date;
JLabel time;
JLabel placecont;

JRadioButton noButton;
JRadioButton yesButton;

JTextField matricNoField;
JTextField levelField;
JTextField sessionField;
JTextField titleField;
JTextField surnameField;
JTextField firstNameField;
JTextField middleNameField;
JTextField DOBField;
JTextField ageField;
JTextField sexField;
JTextField maritalStatusField;
JTextField nationalityField;
JTextField religionField;
JTextField SORField;
JTextField LGAField;
JTextField homeAddyField;
JTextField contactAddyField;
JTextField HRNField;
JTextField OCRField;
JTextField telNoField;
JTextField emailField;
//JTextField disabilityField;
JTextField specifyField;
JTextField pictureFileField;

TimeDisplay display;

GregorianCalendar gc;

JButton browse;
```

```
JButton insert;
JButton reset;
JButton cancel;

Thread logot;

public void run()
{
 while(true)
 {
 logoLabel repaint();
 pictureLabel.repaint();
 try
 {
 Thread.sleep(100);
 }
 catch(Exception e){}
 }
}


//Constructor

public PersonalInformation(final JInternalFrame own, final Frame owner)
{
 logot = new Thread(this);


 this.setLayout(null);
 Font f = new Font("Times New Romans", Font.BOLD, 18);
 Font f2 = new Font("Times New Romans", Font.BOLD, 17);
 Font f3 = new Font("Times New Romans", Font.BOLD, 15);
 Font f4 = new Font("Times New Romans", Font.BOLD, 16);


 logo = new Logo1(100, 100);
 logoLabel = new JLabel(logo);

 school = new JLabel("Electrical/Electronics & Computer Eng. Department");
 school.setFont(f);
 //school.setColor(Color.purple);
 place = new JLabel("Federal University of Technology");
 place.setFont(f);
 //place.setColor(Color.red);
 placecont = new JLabel("Minna, Niger State, Nigeria");
 placecont.setFont(f4);
//placecont.setColor(Color.yellow);
 topic = new JLabel("STUDENT PERSONAL INFORMATION");
 topic.setFont(f);
 //topic.setColor(Color.black);

 browse = new JButton("Choose Picure");
 pictureFileField = new JTextField(10);
 pictureLabel = new JLabel();
```

```
//Border b = createBevelBorder(BevelBorder.LOWERED);
pictureLabel.setBorder(BorderFactory.createBevelBorder(BevelBorder.LOWERED));
pictureLabel.setMaximumSize(new Dimension(120, 120));

JLabel lab = new JLabel("Picture Preview");
lab.setFont(f2);
lab.setBounds(690, 102, 150, 19);
this.add(lab);

pictureLabel.setBounds(690, 130, 120, 120);
this.add(pictureLabel);

browse = new JButton("browse");
browse.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent eve)
{

FileDialog f = new FileDialog(owner, "Choose a Picture", FileDialog.LOAD);
f.setDirectory(".");
f.show();

if(f.getFile() == null)return;
String Directory = f.getDirectory();
String file = Directory+f.getFile();

pictureFileField.setText(file);
imco = new PersonalPicture1(120, 120, file);
pictureLabel.setIcon(null);
pictureLabel.setIcon(imco);
//pictureLabel.setBounds(690, 102, 120, 120);
//add(pictureLabel);
}
});

matricNo = new JLabel("Matriculation Number:");
level = new JLabel("Present Level:");
session = new JLabel("Session:");
title = new JLabel("Title:");
surname = new JLabel("Surname:");
firstName = new JLabel("First Name:");
middleName = new JLabel("Middle Name:");
DOB = new JLabel("Date of Birth:");
age = new JLabel("Age:");
sex = new JLabel("Sex:");
maritalStatus = new JLabel("Marital Status:");
nationality = new JLabel("Nationality:");
religion = new JLabel("Religion:");
SOR = new JLabel("State of Origin:");
LGA = new JLabel("Local Gov. Area:");
homeAddy = new JLabel("Home Address:");
contactAddy = new JLabel("Contact Address:");
HRN = new JLabel("Hostel Room No.:");
OCR = new JLabel("Off Campus Residence:");
telNo = new JLabel("Telephone No.:");
```

```
email = new JLabel("Email:");
disability = new JLabel("Disability:");
pictureFile = new JLabel("Picture File:");
/*no = new JLabel("No");
yes = new JLabel("Yes");*/

ButtonGroup bg = new ButtonGroup();

noButton = new JRadioButton("No", false);
yesButton = new JRadioButton("Yes", false);
bg.add(noButton);
bg.add(yesButton);
specify = new JLabel("Specify:");

matricNoField = new JTextField(10);
levelField = new JTextField(10);
sessionField = new JTextField(10);
titleField = new JTextField(10);
surnameField = new JTextField(10);
firstNameField = new JTextField(10);
middleNameField = new JTextField(10);
DOBField = new JTextField(10);
ageField = new JTextField(10);
sexField = new JTextField(10);
maritalStatusField = new JTextField(10);
nationalityField = new JTextField(10);
religionField = new JTextField(10);
SORField = new JTextField(10);
LGAField = new JTextField(10);
homeAddyField = new JTextField(20);
contactAddyField = new JTextField(20);
HRNField = new JTextField(10);
OCRField = new JTextField(20);
telNoField = new JTextField(10);
emailField = new JTextField(10);
specifyField = new JTextField(10);
TimeZone tz = TimeZone.getTimeZone("Africa/Lagos");

Calendar cal = new GregorianCalendar();
java.util.Date d = new java.util.Date();
cal.setTime(d);

String timeSubmitted =
cal.get(Calendar.DATE)+"/"+cal.get(Calendar.MONTH)+"/"+cal.get(Calendar.YEAR);

date = new JLabel(timeSubmitted);
date.setFont(f3);

Thread generator = new Thread(new TimeGenerator());
display = new TimeDisplay();
display.setFont(f3);
generator.start();

//time = new JLabel(display);
```

```
cancel = new JButton("Cancel");

cancel.addActionListener(new ActionListener()
{
 public void actionPerformed(ActionEvent canc)
 {
 own.dispose();
 }
});

insert = new JButton("Insert");

insert.addActionListener(new ActionListener()
{
 public void actionPerformed(ActionEvent ins)
 {
 File pic = new File(pictureFileField.getText());

 try
 {
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

 Connection dbcon = DriverManager.getConnection("jdbc:odbc:MyDataSource");

 PreparedStatement pinsert = dbcon.prepareStatement("insert into PersonalInfo(Matric_No, Level,
 Session, title, Surname, First_Name, Middle_Name, D_O_B, Age, Sex, Marital_Status, Nationality,
 Religion, State_Of_Origin, L_G_A, Home_Address, Contact_Address, Hostel_Room_No,
 Off_Campus_Res, Telephone, Email, Disability, Picture, Picture_File)
 values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

 pinsert.setString(1, matricNoField.getText());
 pinsert.setInt(2, Integer.parseInt(levelField.getText()));
 pinsert.setString(3, sessionField.getText());
 pinsert.setString(4, titleField.getText());
 pinsert.setString(5, surnameField.getText());
 pinsert.setString(6, firstNameField.getText());
 pinsert.setString(7, middleNameField.getText());
 pinsert.setString(8, DOBField.getText());
 pinsert.setInt(9, Integer.parseInt(ageField.getText()));
 pinsert.setString(10, sexField.getText());
 pinsert.setString(11, maritalStatusField.getText());
 pinsert.setString(12, nationalityField.getText());
 pinsert.setString(13, religionField.getText());
 pinsert.setString(14, SORField.getText());
 pinsert.setString(15, LGAField.getText());
 pinsert.setString(16, homeAddyField.getText());
 pinsert.setString(17, contactAddyField.getText());
 pinsert.setString(18, HRNField.getText());
 pinsert.setString(19, OCRField.getText());
 pinsert.setString(20, telNoField.getText());
 pinsert.setString(21, emailField.getText());
 String disable = "";
 if(noButton.isSelected())
 {
  disable = new String("Non");
```

```
 }
 else
 {
  if(yesButton.isSelected())
  {
   disable = specifyField.getText();
  }
 }
 pinsert.setString(22, disable);
 pinsert.setBinaryStream(23, new FileInputStream(pictureFileField.getText()), (int)pic.length());
 pinsert.setString(24, pictureFileField.getText());

 pinsert.executeUpdate();

 pinsert.close();

 JOptionPane.showMessageDialog(null, "Successfully inserted into Database", "Success",
 JOptionPane.INFORMATION_MESSAGE);
 }
 catch(Exception e)
 {
       JOptionPane.showMessageDialog(null, e.toString(), "Error Inserting to Table",
       JOptionPane.ERROR_MESSAGE);
  //e.printStackTrace();
 }
}
});

reset = new JButton("Reset");

reset.addActionListener(new ActionListener()
{
 public void actionPerformed(ActionEvent re)
 {
 matricNoField.setText("");
 levelField.setText("");
 sessionField.setText("");
 titleField.setText("");
 surnameField.setText("");
 firstNameField.setText("");
 middleNameField.setText("");
 DOBField.setText("");
 ageField.setText("");
 sexField.setText("");
 maritalStatusField.setText("");
 nationalityField.setText("");
 religionField.setText("");
 SORField.setText("");
 LGAField.setText("");
 homeAddyField.setText("");
 contactAddyField.setText("");
 HRNField.setText("");
 OCRField.setText("");
 telNoField.setText("");
 emailField.setText("");
```

```
  specifyField.setText("");
  pictureFileField.setText("");
  pictureLabel.setIcon(null);
 }
});

logoLabel.setBounds(2, 2, 100, 100);
this.add(logoLabel);
school.setBounds(190, 12, 500, 20);
this.add(school);
place.setBounds(270, 29, 320, 20);
this.add(place);
placecont.setBounds(320, 45, 270, 19);
this.add(placecont);
topic.setBounds(255, 69, 350, 20);
this.add(topic);
matricNo.setBounds(15, 100, 105, 30);
this.add(matricNo);
matricNoField.setBounds(135, 102, 80, 20);
this.add(matricNoField);
level.setBounds(270, 100, 80, 30);
this.add(level);
levelField.setBounds(350, 102, 80, 20);
this.add(levelField);
session.setBounds(480, 100, 80, 30);
this.add(session);
sessionField.setBounds(560, 102, 80, 20);
this.add(sessionField);
title.setBounds(15, 140, 80, 30);
this.add(title);
titleField.setBounds(135, 142, 80, 20);
this.add(titleField);
surname.setBounds(270, 140, 80, 30);
this.add(surname);
surnameField.setBounds(350, 142, 80, 20);
this.add(surnameField);
firstName.setBounds(480, 140, 80, 30);
this.add(firstName);
firstNameField.setBounds(560, 142, 80, 20);
this.add(firstNameField);
middleName.setBounds(15, 180, 80, 30);
this.add(middleName);
middleNameField.setBounds(135, 182, 80, 20);
this.add(middleNameField);
DOB.setBounds(270, 180, 80, 30);
this.add(DOB);
DOBField.setBounds(350, 182, 80, 20);
this.add(DOBField);
age.setBounds(480, 180, 80, 30);
this.add(age);
ageField.setBounds(560, 182, 80, 20);
this.add(ageField);
sex.setBounds(15, 220, 80, 30);
this.add(sex);
sexField.setBounds(135, 222, 80, 20);
```

```
this.add(sexField);
maritalStatus.setBounds(270, 220, 80, 30);
this.add(maritalStatus);
maritalStatusField.setBounds(350, 222, 80, 20);
this.add(maritalStatusField);
nationality.setBounds(480, 220, 80, 30);
this.add(nationality);
nationalityField.setBounds(560, 222, 80, 20);
this.add(nationalityField);
religion.setBounds(15, 260, 80, 30);
this.add(religion);
religionField.setBounds(135, 262, 80, 20);
this.add(religionField);
SOR.setBounds(270, 260, 80, 30);
this.add(SOR);
SORField.setBounds(350, 262, 80, 20);
this.add(SORField);
LGA.setBounds(480, 260, 80, 30);
this.add(LGA);
LGAField.setBounds(560, 262, 80, 20);
this.add(LGAField);
homeAddy.setBounds(15, 300, 105, 30);
this.add(homeAddy);
homeAddyField.setBounds(135, 302, 505, 20);
this.add(homeAddyField);
contactAddy.setBounds(15, 340, 105, 30);
this.add(contactAddy);
contactAddyField.setBounds(135, 342, 505, 20);
this.add(contactAddyField);
HRN.setBounds(15, 380, 105, 30);
this.add(HRN);
HRNField.setBounds(135, 382, 80, 20);
this.add(HRNField);
OCR.setBounds(15, 420, 110, 30);
this.add(OCR);
OCRField.setBounds(135, 422, 505, 20);
this.add(OCRField);
telNo.setBounds(15, 460, 100, 30);
this.add(telNo);
telNoField.setBounds(135, 462, 80, 20);
this.add(telNoField);
email.setBounds(270, 460, 80, 30);
this.add(email);
emailField.setBounds(330, 462, 290, 20);
this.add(emailField);
disability.setBounds(15, 500, 100, 30);
this.add(disability);
noButton.setBounds(135, 500, 50, 30);
this.add(noButton);
yesButton.setBounds(185, 500, 80, 30);
this.add(yesButton);
specify.setBounds(270, 500, 80, 20);
this.add(specify);
specifyField.setBounds(350, 502, 80, 20);
this.add(specifyField);
```

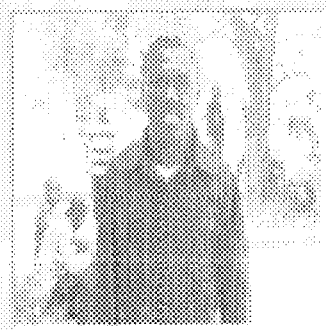
```
pictureFile.setBounds(15, 540, 100, 30);
this.add(pictureFile);
pictureFileField.setBounds(135, 542, 100, 20);
this.add(pictureFileField);
browse.setBounds(240, 542, 80, 20);
this.add(browse);
date.setBounds(550, 565, 100, 30);
this.add(date);
display.setBounds(615, 560, 160, 30);
this.add(display);
insert.setBounds(250, 600, 80, 23);
this.add(insert);
reset.setBounds(350, 600, 80, 23);
this.add(reset);
cancel.setBounds(450, 600, 80, 23);
this.add(cancel);

logo1.start();

}
}
```

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

STUDENT PERSONAL INFORMATION

Matriculation Number:	99/1111EE	Present Level:	500	Session:	2003/2004
Title:	MR.	Surname:	DENIS	First Name:	JIYA
Middle Name:	LAW	Date of Birth:	19/10/1978	Age:	27
Sex:	MALE	Marital Status:	MARRIED	Nationality:	NIGERIAN
Religion:	CHRISTIANITY	State of Origin:	NIGER	Local Gov. Area:	BOSSO
Home Address:	PLOT 23, KINGSTON STREET, BOSSO ESTATE, MINNA, NIGER STATE				
Contact Addre...	SAME AS ABOVE				
Hostel Room ...	NIL				
Off Campus Reside...	PLOT 23, KINGSTON STREET, BOSSO ESTATE, MINNA, NIGER STATE				
Telephone No.:	08011111123	Email:	jiyalaw@yahoo.co.uk		
Disability:	(•) No ( ) Yes	Specify:			

Picture Preview



Semester: First Semester **Level:** 200

Course ...	Course Title	Credit U...	Grade	Grade Point
EMG 211	Engineering Maths I	3	A	15
MEE 212	Applied Mechanics I	2	A	10
AGE 213	Engineering Drawing I	2	B	8
MEE 214	Fundamental of Fluid Mechani...	2	D	4
MEE 215	Strength of Material	2	A	10
MEE 216	Material Science	2	B	8
ELG 217	General Eng. Lab. I	3	A	15
ESG 218	Engineer in Society	1	A	5
ECE 219	Applied Electricity I	3	A	15
		0		0
		0		0
	Total	20.0		90.0
	Cummulative Average	4.5		

Semester: Second Semester **Level:** 200

Course ...	Course Title	Credit U...	Grade	Grade Point
EMG 221	Engineering Maths III	3	A	15
MEE 222	Fundamental of Thermody	2	C	6
AGE 223	Engineering Drawing II	2	B	8
ECE 224	General Computer Programming	2	A	10
MEE 225	Workshop Practice Eng. Lab.	2	A	10
MEE 226	Applied Mechanics II	2	A	10
ELG 227	General Engineering Lab.	3	A	15
CHE 228	Fundamental of Eng. Chem.	2	B	8
ECE 229	Applied Electricity II	3	A	15
		0		0
		0		0
	Total	21.0		97.0
	Cummulative Average	4.59		

Course:

Degree: B.Eng.

Date of Graduation:

Major:

Minor:

Withdrawal date:

Grading System

(CGPA)	Grade	Class of Degree
4.50 - 5.00	A	First Class
3.50 - 4.49	B	Second Class Upper
2.40 - 3.49	C	Second Class Lower
1.50 - 2.39	D	Third Class
1.00 - 1.49	E	Pass
0.99	F	Fail

Dean's Signature and Date

Registrar's Signature and Date

Office Stamp

Office Stamp