TIME TABLE GENERATION USING HEURISTIC APPROACH

(CASE STUDY: DEPARTMENT OF COMPUTER SCIENCE, FEDERAL UNIVERSITY, OYE- EKITI)

BY

AKINTUNDE SAHEED OLUGBENGA
(CSC/11/0268)

BEING A PROJECT REPORT SUBMITTED TO

THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE, FEDERAL UNIVERSITY OYE EKITI, EKITI STATE, NIGERIA

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF BACHELOR OF SCIENCE (B.Sc) DEGREE IN COMPUTER SCIENCE.

OCTOBER, 2015.

CERTIFICATION

This is to certify that this Project Report titled **"TIME TABLE GENERATION USING HEURISTIC APPROACH"** was carried out by **AKINTUNDE SAHEED OLUGBENGA** with the Matriculation Number **CSC/11/O268** in partial fulfilment for the award of Bachelor of Science Degree in Computer Science, Federal University Oye Ekiti, Ekiti State, Nigeria.

[illegible]

Mr Adewole L. B

Supervisor

20-10-2015

DATE

[illegible]

Dr Oluwadare. S.A

Ag. Head of Department

20/10/2015

DATE

DEDICATION

I dedicate this work to the Almighty God for His infinite mercies over me; His grace and His faithfulness. I also dedicate this project to the entire family of Akintunde's for their love, care and support both financially and otherwise.

ACKNOWLEDGEMENTS

This work is a synergistic product of many minds and I feel a deep sense of gratitude to my parents, Mr. and Mrs. Akintunde for their encouragement and for being ever supportive. My sincere thanks goes to my supervisor Mr. Adewole Lawrence Bunmi for his thorough assistance with this work. I also acknowledge the Computer Science Students (set 2011/2012) for their active verbal participation and suggestions towards the evolvement of this project work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The manual system of time table preparation in higher institutions with large number of courses is very time consuming and usually ends up with various classes clashing either at same lecture venue or with same lecturer having more than one class at a time. These are human errors which are very difficult to prevent in the process of timetabling. Reliance on previous year or semester's timetable structure often reduces the stress and difficulty in timetable generation but the modification process is still a tedious job and it as well reduce the flexibility of the timetable. In order to overcome all these challenges and the shortcoming of the manual system, we propose to design an automated timetabling system. The system will take various inputs like details of students, courses and lecture rooms and lecturers available, depending upon these inputs it will generate a possible time table, making optimal utilization of all resources in a way that will best suit any of constraints or faculty rules. We implement the proposed design and also made useful recommendations on the deployment procedure.

CHAPTER ONE

1.1 INTRODUCTION

Information and Communication Technology has found its application in different areas. These include: education, health, communication, government, etc. This is due to its ability to process large volume of information within a limited time, high speed of execution compared to human brain, ability to work without biasness, etc.

Some systems that have the capability to mimic human brain have evolved over the year. Some of these are used as management information system while some serve as decision support system. Some system are even capable of inferring based on some set of rule supplied (expert systems)

There are also some class of problem that are referred to as Constraint Satisfaction Problem in which the computer tries to solve some problem without violating some set of constraint. This class of problem usually aims to proffer optimal solution where applicable to the problem being solved.

Timetable construction is the allocation, subject to constraints, of given resources to objects being placed in space-time in such a way as to satisfy or nearly satisfy a desirable set of possible objectives (Wren, 1996). Academic timetabling is a multi-dimensional and highly constrained problem. Generating academic timetables manually often involves numerous rounds of changes before they can be satisfactory. Usually such a process takes several days, and often the quality of the timetables is compromised due to pressure to release the timetables on time. Hence, automatic generation of timetables seems to be a better approach to manual approach. But this approach is not without problems. In fact, most timetabling problems are NP-complete and most researchers are interested in investigating efficient algorithms for solving the problem.

Operation Research methods are the major algorithms used over the last decades for solving timetabling related problems. These techniques are mathematical in nature. The most common ones are Linear Programming (LP) and Integer Programming (IP). In current decade, the contribution of artificial intelligence has provided the timetabling community with promising modern heuristics such as: Genetic Algorithms, Simulated Annealing, and Tabu Search (Schaerf, 1999)

1.2 PURPOSE OF STUDY

In this project work, we propose to carefully identify various constraints that are needed to be put into consideration in the design of lecture timetable for the entire college. Wefurther intend to analyze these constraints in order to ascertain if there exists a feasible solution that satisfies all the highlighted constrains. We further aim to produce, as an output, a well structured timetable that favours all the constraint optimally.

1.3 STATEMENT OF THE PROBLEM

The design of lecture timetable is in no doubt a tedious task. The complexity in the design process centers on the ability of the designer or the timetable committee to satisfy all available constraint. The design process often witness lots of designs and cancellation before reaching a compromise on a desired design. The timetable design processtakes days and sometimes weeks for the completion. In addition to this, the accepted design might not be optimal that is, there could be other design that will be much better than the designed work. Furthermore, adjusting timetable after the implementation of the design is always a great tasking job. In lieu of these, it is necessary to design and implement a new application that can automate the timetable design process in order to eliminate the complexity associated with the manual approach.

1.4 AIM AND OBJECTIVES

This project work aims at designing and implementing a computer-based timetable generator which can be implemented at various works of life to generate a series of time-based event that are also subjected to other factors. However, at the end of this project work, the following objectives would have been accomplished

a. To design a computer based timetable generator that is capable of accepting as input the courses to be offered, the available lecturers, list of courses to be taught by each lecturer, minimum number of hours for a particular course per week, maximum number of occurrence of course per week, courses with fixed time, break time, lecture venues, etc.

b. To evaluate the feasibility of satisfying all the constraints before attempting the solution

c. To generate a well structured timetable within a reasonable period of time.

1.5 RESEARCH METHODOLOGY

Information used in the development of this project work is primarily obtained via personal interview with member of the timetable generation committee for the college and the consultation of professionals in the field of problem analysis and system development.

Other sources of information include:

i. Internet
ii. E-Books
iii. Text Books and
iv. Review of some existing Timetable generator.

Visual Basic 6.0 will be used for designing the front end of the application while Microsoft Access Database will be used for the back end. The application main output (time table) will be generated using excel spreadsheet while DataReport will be used for generating other useful reports.

1.6 SCOPE AND LIMITATION

This project aim to develop a computer based timetable generator. The design is targeted at servicing a college at a time. That is, it cannot handle the entire institution at a time. However, it can be installed at each college and the output from each college can then be merged together to service the entire institution.

The architecture is also developed to be open enough such that it can be adapted to generate other types of timetable other than lecture timetable for academic institution. Such application include fixtures for football matches, Special week long Events, etc.

The application is also design to be flexible enough to accommodate various changes that may occur is lecture time. That is, lecture time could range from 1 hour period to 2 hours period as against some reviewed designs that are rigid.

1.7 DEFINITION OF TERMS

GAs: Genetic Algorithms

TS: Tabu Search

Scheduling: It can be defined as the process of deciding how to commit resources between varieties of possible tasks.

Expert System:An expert system is a software system that attempts to reproduce the performance of one or more human experts, most commonly in a specific problem domain, and is a traditional application and/or subfield of artificial intelligence.

CSP: Constraint Satisfaction Programming

1.8 CONTRIBUTIONS TO KNOWLEDGE

The main contribution of this project work to knowledge is the application of rule based approach to scheduling problem. In addition to this, we are able to establish that the process of timetabling among other processes can be completely computerized thereby removing the stress that the timetable committee need to pass through and also ensure timeliness of report.

1.9 ORGANIZATION OF PROJECT WORK

This project work is organized into chapters. The project comprises of five chapters, references and appendix. The first chapter contains the introductory notes as well as the statement of problem for the problem at hand. It also entails a well detailed aims and objectives. The second chapter reviewed related literature in order to have a better understanding of the extent to which researches has been carried out in the field of timetabling. The third chapter is dedicated to the design of the proposed system. It entails the details of the input design, interface design, database design, logic design and output design. The fourth chapter entails the implementation details for the designed architecture. The last chapter summarizes the entire work done and also presented useful recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

Timetable generation, which plays important role in education, is a special version of the optimization problems found in real life situations (Zoltan Petres, 2000). It is also a part of the large field of scheduling problems. Scheduling problems are essentially problems that deal with the effective distribution of resources, usually because of the availability of limited resources (Bajeh and Abolarinwa, 2011).

Wren (1996)described timetabling as: “the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives”.

The timetabling problem has always been solved by leveraging human resource in educational institutions. Duringthe process, numerous aspects have to be taken into consideration.

Timetable design for an average institution often require days or even week of work for an experienced person and the result is often not satisfactory; it usually does not meet all the requirements and also susceptible to biasness . Also, when the preconditions (set of constraints) change, the whole work becomes unusable, and has to be restarted from scratch. The problem, as almost all optimization problems, is computationally NP-hard. Therefore, only the important conditions can be considered during the manual arrangement process, but it is still extremely complex to find the optimal solution of this reduced problem. Thus, good timetable generator software would take into consideration not only the essential conditions necessary for a usable timetable, but also other important didactic and organizational requirements would be very useful. This became reality by the tremendous growth of computing capacity.

This chapter review different literatures on timetabling problem solving approaches. We examined some of the commonly used algorithms. We also reviewed the various categorization of constraints (soft and hard) as well compare the proposed system with the existing manual system in order to emphasize the importance of the proposed system to the academic institution.

2.2 SURVEY OF TIMETABLE GENERATION ALGORITHMS

Various algorithms have been developed for solving scheduling problems over the past decades. From the survey carried out on scheduling algorithms, we classify the existing algorithms into the following categories (Kuldeep, 2001)

i. Integer Programming or Linear Programming

ii. Genetic and Evolutionary Algorithms

iii. Simulated Annealing

iv. Tabu Search

v. Constraint Satisfaction Programming

2.2.1 LINEAR PROGRAMMING/INTEGER PROGRAMMING

This approach was developed from the area of mathematical programming. It is the first approach used in solving timetabling problem. Mathematicalprogramming is applicable to the class of problems characterized by a large number ofvariables that intersect within boundaries imposed by a set of restraining conditions(Thompson, 1967).

Linear Programming (LP) is that subset of mathematical programming concerned withthe efficient allocation of limited resources to known activities with the objective ofmeeting a desired goal such as maximizing profits or minimizing costs (Feiring, 1986).

Integer Programming (IP) deals with the solution of mathematical programmingproblems in which some or all of the variables can assume non-negative integer value sonly.

The construction of a linear programming model involves three successive problem solving steps. The first step identifies the unknown or independent decision variables.

Step two requires the identification of the constraints and the formulation of these constraints as linear equations. Finally, in step three, the objective function is identified and written as a linear function of the decision variables.

The formulation of LP/IP in the context of the general timetabling problem is represented as follows (Schaerf, 1999):

There are q subjects, represented as $S_1 \ldots S_q$ and each subject, S_i , consists of n_i lectures.

Furthermore, let there be r curricula $K_1 \ldots K_r$ in which each curricula is defined to a group of subjects that have common students. This means that subjects in K_i must all be scheduled at different times. If the number of lecture time periods is p, and M_k is the maximum number of lectures that can be scheduled at lecture time period k (i.e. The number of rooms available at time period k), then the formulation of this LP/IP timetable problem is as follow:

Find $y_{ik} \quad (i = 1 \ldots q, k = 1 \ldots p)$

Such that

$$\sum_{k=1}^{p} y_{ik} = k_i \qquad (i = 1 \ldots q) \qquad (1)$$

$$\sum_{i=1}^{q} y_{ik} \leq M_k \qquad (k = 1 \ldots p) \qquad (2)$$

$$\sum_{i \epsilon k_i}^{q} y_{ik} \leq 1 \qquad (i = 1 \ldots r, k = 1 \ldots p) \qquad (3)$$

$$y_{ik} \epsilon \{0,1\} \qquad (i = 1 \ldots q, k = 1 \ldots p) \qquad (4)$$

where we interpret $y_{ik} = 1$ if a lecturer of a course S_i is scheduled at lecture time period k, and $y_{ik} = 0$ otherwise.

Constraint set (1) imposes that each subject is composed of the correct number of lectures. Constraint set (2) enforces that at each time there are not more lectures than rooms. Constraint set (3) prevents conflicting lectures to be scheduled at the same period. Constraint set (4) contains the binary requirements that transform the problem from the LP domain into the LP/IP domain.

2.2.2 EVOLUTIONARY AND GENETIC ALGORITHMS

Evolutionary Algorithms (EAs) are a class of direct, probabilistic search and optimization algorithms gleaned from the model of organic evolution. A Genetic Algorithm (GA) is a type of EA and is regarded as being the most widely known EA in recent times (Back, 1995). GAs are a class of stochastic search algorithms based on biological evolution whose search strategy mimics natural selection by using an automated version of the survival of the fittest" analogy (for example see Sharma and Chandra, 1999; Sanchez, Shibata and Zadeh, 1997). It is a problem solving and optimization method that uses genetics as its model problem and applies the rules of reproduction, general crossover and mutation to pseudo-organizations so those organizations can pass beneficial and survival-enhancing traits to the new (next) generation (Chambers, 1995).

A GA differs from other search techniques in the following ways:

i. GAs optimize the trade-off between exploring new points in the search space and exploiting the information discovered thus far. This was proved using the K-armed bandit (an extension of the one-armed bandit) problem (Buckles and Petry, 1992).

ii. GAs have the property of implicit parallelism. Implicit parallelism means that the GA's effect is equivalent to an extensive search of hyper planes of the given space, without directly testing all hyper plane values (Goldberg, 1989). Each schema denotes a hyper plane.

iii. GAs are randomized algorithms, in that they use operators whose results are governed by probability. The results for such operations are based on the value of a random number (Buckles and Petry, 1992). This means GAs use probabilistic transition rules, not deterministic rules.

iv. GAs operate on several solutions simultaneously, gathering information from current search points to a direct subsequent search. Their ability to maintain multiple solutions concurrently makes them less susceptible to the convergence problem of local maxima and noise (Goldberg, 1989).

v. GAs work with a coding of the parameter set, not the parameters themselves (Goldberg, 1989) .

vi. GAs search from a population of points, not a single point (Goldberg, 1989).

vii. GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge (Buckles and Petry, 1992).

An abstract view of the problem solution phase using a GA can be illustrated as follows (Buckles and Petry, 1992):

```
generate an initial population, G(at time=0)
evaluate the fitness function f[G(0)]
$t = 0$
repeat
        $t = t + 1$
        Generate G($t$) using G($t-1$)
        Evaluate f[G($t$)]
    until either an acceptable solution is found or population convergence is achieved.
```

The representation of the timetable problem as a GA is very problem specific. Unlike LP/IP (where an almost "exact" mathematical formulation can be ascribed) the encoding scheme, which represents the template for the solution, must be determined.

Here, we use a recent innovation (Sharma and Chandra, 1999) in which integer index strings are used (instead of the traditional bit strings) in the representation. This is claimed to improve the computational performance due to the vast reduction in the length of the encoded chromosome. In this schema a sample examination timetable problem of, say, 17 subjects would be represented graphically as (Sharma and Chandra, 1999):

It is claimed that GAs have a greater capacity than any other search method in finding the largest number of possible solutions and depict the best possible results (Maxfield, 1997). GAs demonstrate real-world significance when applied to timetabling problems where a complex set of scheduling constraints and a various collection of individuals exist. It is generally argued that a timetabling GA develops the best possible schedule (Plain, 1995). However, GAs still fall short in the choice of control parameters, the exact roles of crossover and mutation, and the characterization of search landscapes for optimization and convergence characteristics (Srinivas and Patnaik, 1994). Some applications where GAs have been used successfully are analogue circuits, fuzzy logic, and neural networks (Maxfield, 1997). These cover a wide range of practical applications such as job shop scheduling, training neural nets, image feature extraction, and image feature recognition (Buckles and Petry, 1992).

2.2.3 SIMULATED ANNEALING

Simulated Annealing (SA) is a randomized local search optimization technique for finding solutions to optimization problems. The name is derived from the analogue to the chemical physics simulation of the cooling of a collection of a Boltzmann distribution of atoms. SA is highly resource intensive and one of its setbacks is its requirement of utilizing a large amount of computational time for obtaining a near-optimal solution. As such some attempts at speeding up annealing algorithms have been based on shared memory multiprocessor systems, and parallelization for certain problems on distributed memory multiprocessor systems (Abramson, 1991; Dikmann,Luling and Simon, 1993).

The process starts by creating a random initial solution. The main procedure consists of a loop that generates, at random and for each iteration, a neighbor (and thus classification as a local search technique) of the current solution. The definition of neighbor in SA is dependent upon the specific structure of each problem. The basic steps involved in using SA are as follow (Schaerf, 1999):

i. Let Δ be the difference in the objective function between the new solution and the current one.

ii. If $\Delta < 0$ the new solution is accepted and becomes the current one

iii. If $\Delta > 0$ the new solution is accepted with probability $e^{-\Delta/T}$ where T is a parameter called temperature

iv. The temperature T is initially set to an appropriate high value T_0. After a fixed number of iterations, the temperature is decreased by the cooling rate a, such that $Tn = a \times Tn\text{-}1$, where $0 \leq a \leq 1$.

v. The procedure terminates when the temperature value is close and no other solution increases the value of the objective function, i. e. the system is frozen. The solution obtained when the system is frozen is obviously a local minimum.

vi. The control knobs of the procedure are the cooling rate *a*, the number of iterations at each temperature, and the starting temperature T_0.

The application of the timetable problem using SA is a relatively straightforwardprocedure. The atoms are replaced by elements and the system energy is replaced bythe timetable costs. An initial allocation is made in which elements are placed in arandomly chosen period. The initial cost and the initial temperature are then computed.

The cost is used to reflect the quality of the timetable and the temperature is used tocontrol the probability of an increase in cost. At each iteration a period is chosen atrandom, called the "*from*"period, and an element randomly selected from that period.

Another period is chosen at random, called the "*to*"period. The change in cost is thencalculated from these two components (Abramson, 1991). An example of this processis illustrated using SA to decrease the cost of a school/university timetable solution is asfollows: (Abramson and Dang, 1993).

Compute an Initial temperature

While (cost <> 0) and timetable not frozen repeat

i. Repeat some constant number of times

ii. Choose a tuple Tn (class, room, period)

iii. Choose a new field value for Period called Period'

iv. Evaluate the cost of removing this tuple from Period

v. Evaluate the cost of inserting this tuple into Period'

vi. Compute change in cost

vii. If (change in cost 0) or (change in cost is acceptable at this temperature)

viii. then accept change and update cost

ix. compute new temperature

SA, which was first introduced by Kirkpatrik, Gelatt and Vecchi (1983) in the paper Optimization by Simulated Annealing" for solving hard combinatorial optimization problems, proved to be a good technique for many such applications (Dikmann, Luling,and Simon, 1993). Attention has been given to hard combinatorial problems like schedulling, the travelling salesman and the quadratic assignment problem (QAP)(Paulli, 1993). The advantage of SA for scheduling problems is its use as an optimization procedure for solving the school and universities timetabling problems.

2.2.4 TABU SEARCH

Tabu Search (TS) is a meta-heuristic technique that guides a local heuristic search procedure to explore the solution space beyond local optimality (Glover and Laguna, 1997). Formally, a meta-heuristic is defined as a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality in optimization problems (Glover and Laguna, 1997).

Meta-heuristic procedures have improved enormously due to advances in recent years, especially due to the developments of new implementations of TS that are more effective in solving difficult problems, than previously considered possible (Glover, 1997).

The initial design and development in TS started in the late 1960's and early 1970's.

Glover produced the present form of TS in his paper "Future Paths for Integer

Programming and Links to Artificial Intelligence" (Glover, 1986). Present research indicates that TS has become a well-established optimization approach that is rapidly spreading in various new fields (Glover, 1993). TS helps to solve problems in a wide area of fields such as resource planning, telecommunications, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis, environmental conversation and scores of other problems (Glover and Laguna, 1997).

The TS method was partly motivated by the observation that human behavior appears to operate with a random element that leads to inconsistent behavior given similar circumstances. The TS method operates in this way with the exception that new subjects are not chosen randomly. Instead the TS proceeds according to the supposition that there is no point in accepting a new (poor) solution unless it is to avoid a path already investigated. This ensures that new regions of a problem's solution space will be investigated with the goal of avoiding local minima and which should ultimately lead to the desired solution.

The TS begins by seeking a local minimum. In order to avoid retracing the steps used the method records recent moves in one or more Tabu lists. The original intent of the lists was not to prevent a previous move from being repeated, but rather to ensure it was not reversed. The Tabu lists are historical in nature and form the nucleus of the Tabu search memory. The role of the memory can change as the algorithm proceeds. At initialization the goal is to make a broad examination of the solution space, known as 'diversification', but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of 'intensification'. In many cases the differences between the various implementations of the Tabu method have to do with the size, variability, and adaptability of the Tabu memory to a particular problem domain.

Hertz (1991) provides a good algorithm for the general tabu search technique as described below:

```
Initialization
        s := initial solution in X
        nbiter := 0 {current iteration}
        bestiter := 0 {iteration when the best solution has been found}
        bestsol := s {best solution}
        T := 0
        initialize the aspiration function A
        while (f(s)>f*) and (nbiter-bestiter<nbmax) do
        nbiter := nbiter +1
        generate a set V* of solutions s in n(s) which are either not tabu or such thatA(f(s))>= f(s)
        choose a solution s* minimising f over V*
        update the aspiration function A and the tabu list T
        if f(s*)<f(bestsol) then
        bestsol := s*
        bestiter := nbiter
s := s*
```

An example of using TS to solve the timetable problem can be formulated as follows: (Glover, 1989; Glover and Laguna, 1997).

For each solution *s* this method requires the definition of a neighborhood *V(s)*, consisting of solution reachable in one step from (*s*). The basic step is to move from the current solution (*s*) to the best solution *s* V (s)*. A tabu list *(T)* is used to avoid cycling as it stores descriptions of the last move or solution, while finding *(s*)*. *T* is scanned to avoid so called tabu moves that could bring the search back to a previous iteration. The procedure stops after a maximum number of iterations or until the best solution is found.

In an optimization problem the search space *(S)* is minimized by the objective function *(f)*. A function *N* which depends on the structure of a specific problem is assigned to each feasible solution *(s)* which belongs to *S* in its neighborhood *[N (s) S]* each solution *s` N (S)* is called a neighbor of *S*.

TS is based on selected concepts that unite the fields of artificial intelligence and optimization. In addition to Simulated Annealing (SA) and Genetic Algorithms (GA),

TS was evaluated in the widely referenced report by the Committee on the next Decade of Operations Research (Condor, 1988) to be "extremely promising" for the future treatment of practical applications (Glover and Laguna, 1997).

2.2.5 CONSTRAINT SATISFACTION PROGRAMMING

Constraint based reasoning has proven to be a productive research method for researchers in the areas of artificial intelligence, operational research, and logic programming (Eugene and Mackworth, 1994). More recently a theoretical framework called Constraint Satisfaction Programming (CSP) has evolved and been defined (Tsang et. al., 1999). The process of *Constraint Satisfaction* involves finding values for problem variables subject to constraints on acceptable combinations of values. The satisfaction of all constraints can sometimes lead to conflicts. In these cases constraints have to be prioritized so that the more important constraints are satisfied first.

Furthermore, the satisfaction of some constraints could also give rise to conflicts with the stated objective. Therefore, in some cases it is impossible to solve the problem completely and thus a subset of the problem is solved which is called *partial constraint satisfaction* (Freuder and Wallace, 1994).

Constraint Satisfaction Problems (CSPs) involve a set of problem variables, a domain of potential values for each variable and a set of constraints specifying which combinations of values are acceptable (Lhomme, 1993). A solution specifies an assignment of a value to each variable that does not violate any of the constraints (see Mackworth, 1994; Freuder and Wallace, 1994). A large number of problems in Artificial Intelligence (AI) and other areas of computer science can be viewed as special cases of CSP. Some examples are machine vision, belief maintenance, scheduling, temporal reasoning, graph problems, floor plan, the planning of genetic experiments, the satisfiability problem, circuit design, machine design and manufacturing, and diagnostic reasoning (Kumar, 1992).

A constraint-scheduling problem can be partially solved using search algorithms that search for best and feasible solutions with forward checking and constraint propagation.

A solution to CSP involves assigning values from domains of all variables such that all constraints are satisfied. Constraint Logic Programming is the integration of two methods: logic programming and constraint solving. Logic programming has the capability of supporting declarative programming where formulation is in terms of true or false. Constraint solving reduces the search space by pruning all impossible values through constraint propagation. Constraint based reasoning is a reasoning process that uses an arc consistency technique to propagate constraints. The arc consistency algorithm is derived from the representation of constraints in a graph that can be associated with a constraint network where nodes correspond to variables in the constraint network and edges link nodes *i* and *j* if there is a relation *Rij* between nodes.

During assignment an algorithm (arc consistency algorithm) is used to check the consistency of labels for each couple of nodes that are related by a binary constraint andlabels that cannot satisfy constraints are removed (Deris, Omatu, Ohta and Samat,1997).

A solution to a timetabling problem using CSP is described as follows: (Deris et al. ,1997).

The problem depicts each course as offering several subjects per semester and each subject having a specified number of lessons per week. Each lesson can thus be defined as the contact hours between lecturers and students at a specific time and place (room).

Each lesson lasts for a specific period of time. Thus the timetabling problem can be defined as an assignment of time t_j, $1<j<m$ and rooms r_k, $1<k< p$ to lessons $s(i)$, $1<i<n$ taught by a lecturer $L(S(i))$ such that all constraints $C(S(i))$ are satisfied. $L(S(i))$ *and*

$C(S(i))$ are lecturers and constraints of a lesson respectively, while m, p and n are infinity variables. Thus the general CSP for a timetabling problem is as follows:

A finite set of variables, $X_1 \ldots, X_n$

For each variable Xi a set of domains $D_1 \ldots, D_n$ containing possible values of X_i

A finite set of constraints, $C_1 \ldots. C_q$ representing relations between variables.

A solution to the CSP involves assigning values from domains of all variables such that all constraints are satisfied.

In terms of the CSP, a timetabling problem can be formulated by representing a timeslot and a room of a lesson as variables of the CSP, available timeslots and rooms as values of the CSP, whereas constraints are the various relationships between lessons.

Therefore the CSP model for timetabling problems can be formulated by deciding the variables, values and constraints.

CSP is a decision making tool that satisfies all constraints and its major advantages are as follows:

i. Constraint propagation reduces the search space so it takes less time to search thus minimizing backtracking;

ii. Memory requirement is smaller since the search space has been reduced;

iii. All available resources are represented in the form of constraints and hence user preferences and requirements can be easily satisfied.

Aside the aforementioned methods for solving scheduling problems, (Muhammad et.al., 2006) identified another approach to timetable generation which is the Immune-Based approach that uses the Negative Selection Algorithm. The Artificial Immune System approach is said to be more efficient compared to the classical heuristic scheduling algorithms such as SA, TS, and Gas (Hart and Ross, 1999).

2.3 CONSTRAINTS

It is very difficult to define how good a potential timetable is, but it is much easier to define when a timetable is usable. This is measured by the degree to which it satisfies the set of pre-defined constraints. These constraints can be broadly classified into two categories:

i. Hard Constraints
ii. Soft Constraints

2.3.1 HARD CONSTRAINTS

Hard Constraints are the set of constraints that must be taken into consideration strictly in the design process of the timetable. Examples of Hard Constraints are:

i. No lecturer or student should have different classes at the same time slot
ii. No two different classes can be allocated to the same venue

iii. Lecture must be fixed at lecturer's availability period (if specified)

iv. General Course must be fixed in-line with the University timetable

v. Allocated Lecture room is big enough to accommodate all attending students

vi. There should be required number of periods for each course

2.3.2 SOFT CONSTRAINTS

Soft constraints are those that are desirable but not absolutely essential. In real-world situations it is, of course, usually impossible to satisfy all soft constraints. Examples of soft constraints are:

i. Every staff should get at least one first hour

ii. A particular class may need to be scheduled in a particular time period.

iii. Lab Classes may not be in consecutive hours.

iv. Lecturers may prefer to have all their lectures in a number of days and to have a number of lecture-free days.

v. Senior Staff may not be scheduled for class between 7 and 9pm

2.4 SHORTCOMINGS OF THE EXISTING SYSTEM

The existing system of constructing a timetable at the department of Computer Science is currently the manual approach. It requires lot of user input in terms of time required and concentration in order to ensure that the constraints are not violated and if not possible, it is reduced to the minimum.

Also, a change in the initial conditions specified might render the entire timetable un-usable and also requires that the development procedure be re-started from the scratch.

In addition to this, timeliness of report is one of the problems often encountered by the timetable committee. They often deliver behind schedule.

Finally, the generated timetable is often not optimal. In order words, better allocation of time slot to courses can still be achieved without violating the constraints.

2.5 ADVANTAGES OF THE PROPOSED SYSTEM OVER EXISTING SYSTEM

The proposed system promises the following features:

i. Timeliness of report

ii. Ability to generate multiple timetables

iii. Ability to enforce hard constraint

iv. Ability to check the feasibility of the solution before the scheduling process

v. Optimality

CHAPTER THREE

SYSTEM DESIGN

3.1 INTRODUCTION

In the previous chapter, we reviewed some of the existing approaches/ algorithms for generating timetable. In addition to this, we review the various categories of constraints. In this chapter, we propose and design an information system to support the production and management of timetables in a university environment. The design entails the elicitation of the system requirements which was carried out by examining in details, sample of previous timetable generated by the college. From the system requirements, we were able to identify the basic input into the system which in-turns guided the design of the database. The structures of the databases were illustrated using tables. The logic design of the system was also carried out and is illustrated at the latter part of this chapter using flowcharts, Entity-Relationship Diagrams, Data Flow Diagrams and pseudo codes.

3.2 SYSTEM ARCHITECTURE

The high-level view of the proposed timetable generator is as depicted in figure 3.1 below. The figure depicts the system as a 3-stage system. These include the input stage, the processing stage and the output stage.

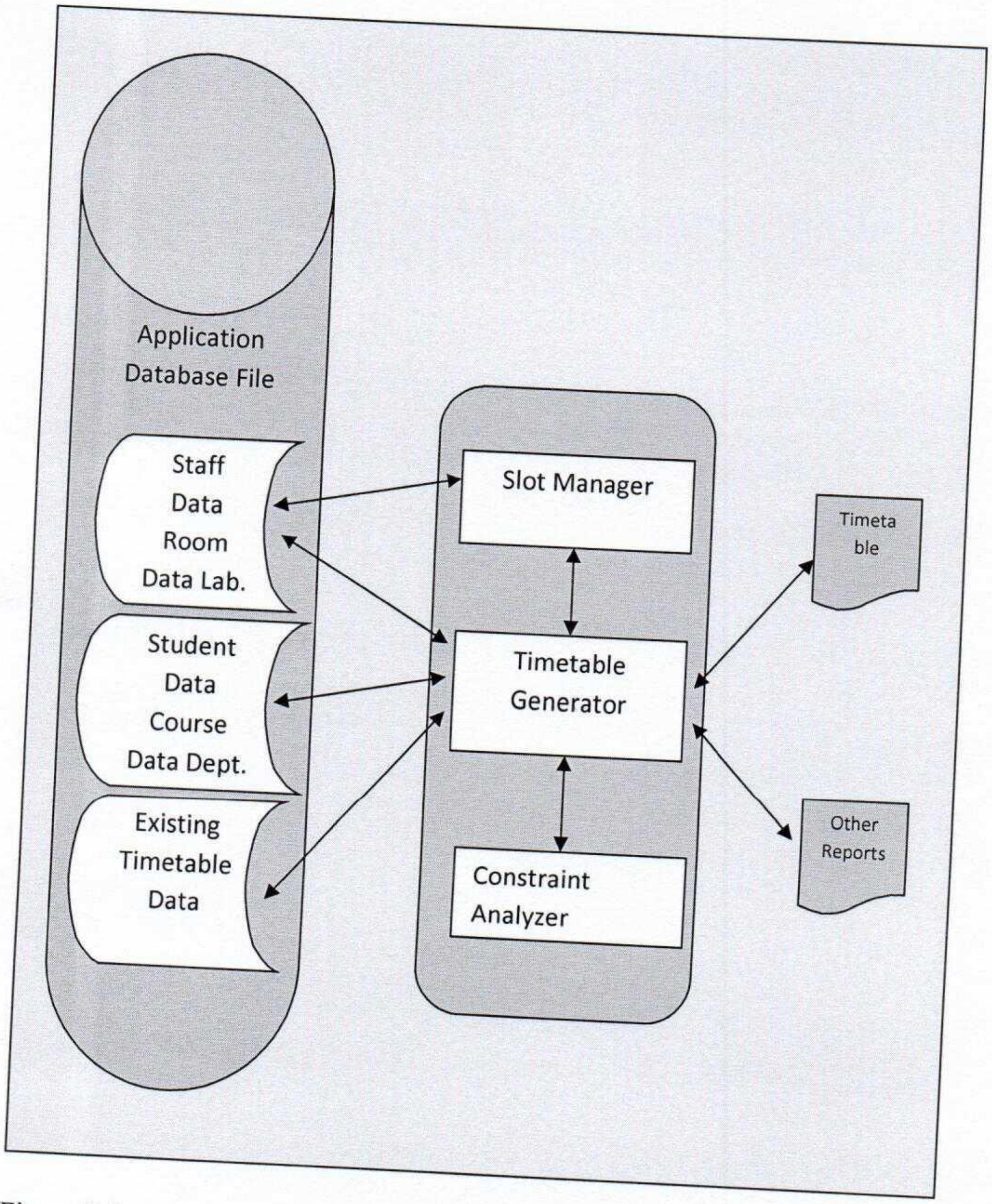


Figure 3.1: High level Architecture of the proposed Timetable Generator

From the figure above, the input, processing and output stages are clearly depicted. The input into the system comprises of data for each course, rooms, lecturers, departments, etc. The processing stage comprises of modules that first analyzes the constraints to determine if the solution is feasible or not. If the solution is not feasible, an error report will be generated containing details of the reasons that led to the system failure to produce output. Otherwise, the timetable generator works with the slot manager to allocate courses. The output stage comprises of modules that displays the generated timetable with special formatting depending on the user which include the Provost (college view), Head of Department (Departmental view), Staff/ Lecturer (Staff view) and Student (Student view).

The timetable generator module allocate courses using the model in figure 3.2 below

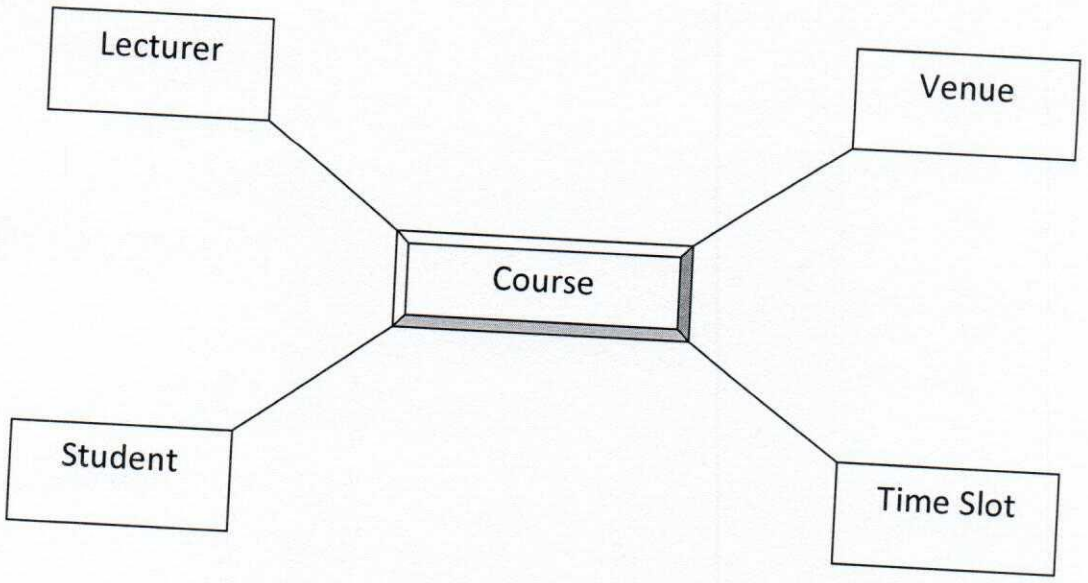


Figure 3.2: Course Structure for Timetable Generation Module

3.3 INPUT DESIGN

The input design is an important stage in system design. This is because the systems often work on Garbage in garbage out basis. In order words, whatever the user enters into the system will be manipulated by the system based on the processing instructions specified and then an output will be generated. Since the output is directly dependent on the input, it is very important that input into the system are correct and also presented in the right format. In view of this, different mechanisms such as easy to use, user-friendly interfaces have been put into consideration in the design process to aid easy and correct input of both data and constraints into the system.

3.4 INTERFACE DESIGN

Usability is one of the key issues in designing a system. No matter how sophisticated a system might be, if it is not usable by the intended users, then the system becomes useless. Hence, much effort is invested in the aspect of human-computer interface design when developing a new system. In this project work, various interfaces have been designed to allow easy interaction and operation of the application in order to actualize the purpose for which it has been developed. The screen shot for some theses interface will be explored under the implementation section in the next chapter.

3.5 DATABASE DESIGN

The time table generator application requires various inputs to aid it decision and allocation. Some of these inputs as illustrated in figure 3.1 are staff data, courses data, lecture room data, department data among others. Some of these data do not change with time hence, creation of database for each of these entities is important. In order to ensure proper tracking of data required to actualize the allocation of resources, the following database files were created.

i Staff Database File

ii College Database File

iii Department Database File

iv Class Database File

vi. Course Database File

vii. TimeSlot Database File

The structure and brief description of each of the database file are as follows.

Staff Database File: The Staff database file tracks record of all staff for each department. The structure of the staff database file is as shown in table 3.1 below

Table 3.1: Staff Database File

Field Name	Field Size	Data Type	Description
StaffID	6	Number	Staff Unique Identification Number
Staff Name	50	String	Full Name of Staff
Cadre	50	String	Switches between two values. Senior and Junior Staffs
Department	50	String	Department the staff belongs to

College Database File: The college database file contains information about each college in the University environment. The structure is as shown in table 3.2 below

Table 3.2: Structure of College Database File

Field Name	Field Size	Data Type	Description
CollegeID	6	Number	College Unique Identification Number
College Name	50	String	Name of College
Description	50	String	Information on how to locate the college within the University Environment
Provost	50	String	Name of Provost

Department Database File: The department database file contains record of all the departments within the Institution. The structure is as shown in table 3.3 below

Table 3.3: Structure of Department Database File

Field Name	Field Size	Data Type	Description
DeparmentID	6	Number	Department Unique Identification Number
Department Name	50	String	Name of Department
CollegeID	6	String	Name of College under which the department is
Description	50	String	Information on how to locate the departmental office within the College Environment
HOD	50	String	Name of Head of Derpartment

Class Database File: The class database contains record of each distinct class. Each record in the database represents a particular level in a specified department at a particular point in time. The structure is as shown in table 3.4 below

Table 3.4: Structure of Class Database File

Field Name	Field Size	Data Type	Description
ClassID	6	Number	Class Unique Identification Number
DepartmentID	6	Number	Unique Identifier for the department it belongs to
Class Size	4	Number	Contain information about the class population
Session	10	String	Current Session

Course Database File: The course database file contains information about all courses that are being offered within the University environment. These include the course title, course unit, course code, and course description. The structure is as depicted in table 3.5 below.

Table 3.5: Course Database File

Field Name	Field Size	Data Type	Description
Course Code	6	String	Course Unique Identifier
Course Title	50	String	Name of Course
Description	255	String	Description of Course. Course Outline
Course Unit	1	Number	Course Weight

TimeSlot Database File: The Time Slot Database File keeps record of lecture period and break for each day. The structure is as shown in table 3.6 below

Table 3.6: Time Slot Database File Structure

Field Name	Field Size	Data Type	Description
DayID	1	Number	Day Unique Identifier
DayName	10	String	Day Name
Start Time		Time	Lecture Start Time for the day
End Time		Time	Lecture End Time for the day
Break Start		Time	Break Start Time
Break End		Time	Break End Time

Aside the database files created above, some other database files were also created to keep temporary data during the process of timetable generation. These include:

i. Staff Constraints Database File
ii. Room Constraint Database File
iii. Course Constraint Database File
iv. Time table Database File

Staff Constraint Database File: This database file keeps track of the availability and non-availability of academic staff. Assuming the school usually have 10 periods per day, each and the school lecture day are Mondays to Fridays, then each staff will have 50 entries in the database file which will either be set to zero (0) or One (1).. the structure is as shown in table 3.7 below.

Table 3.7: Staff Constraint Database File Structure

Field Name	Field Size	Data Type	Description
StaffID	6	Number	Staff Unique Identifier
Day	10	String	Day Name
Period	5	String	Lecture Period for the day
Status	1	Number	Can be zero or one

Room Constraint Database File: This database file has a similar structure with the staff constraint database file. It keeps track of the availability of each lecture theatre at every lecture period for each day.

Course Constraint Database File: The Course constraint database file is records temporary allocation for each course during the scheduling process. Each record contains the course code, the room allocated, Time slot 1, time slot 2 and the lecturer's name. The time slot is encoded as day[starttime-endtime] e.g. 1[8-10], which is interpreted as Monday, 8am to 10am. The period can take value between 1 and 24 to specify the hour of the day.Its structure is as shown in table 3.8 below

Table 3.8: Structure of Course constraint Database File

Field Name	Field Size	Data Type	Description
Course Code	6	String	Course Unique Identifier
RoomID	6	Number	Unique Identifier from room allocated
TimeSlot 1	10	String	Lecture Period for the day
TimeSlot2	10	String	Stores second period for courses that requires more than two hours lecture/ practical
StaffID	6	Number	Unique identifier for lecturer-in-charge

Timetable Database File: The timetable database file store the final result of the schedule. The structure is as shown in table 3.9 below.

Table 3.9: Timetable Database Structure

Field Name	Field Size	Data Type	Description
ClassID	6	Number	This identifies the level and department
Course Code	6	String	Course Unique Identifier
RoomID	6	Number	Unique Identifier from room allocated
TimeSlot 1	10	String	Lecture Period for the day
TimeSlot2	10	String	Stores second period for courses that requires more than two hours lecture/ practical
StaffID	6	Number	Unique identifier for lecturer-in-charge

3.6 LOGIC DESIGN

The logical structure of a system depicts the actual way the system works. It shows interaction between different modules of the application and the flow of data in the program. The logical structure of this application will be delineated using the following Use Case Diagram, data flow diagram (DFD)as shown in figure 3.3 and figure 3.4 respectively.

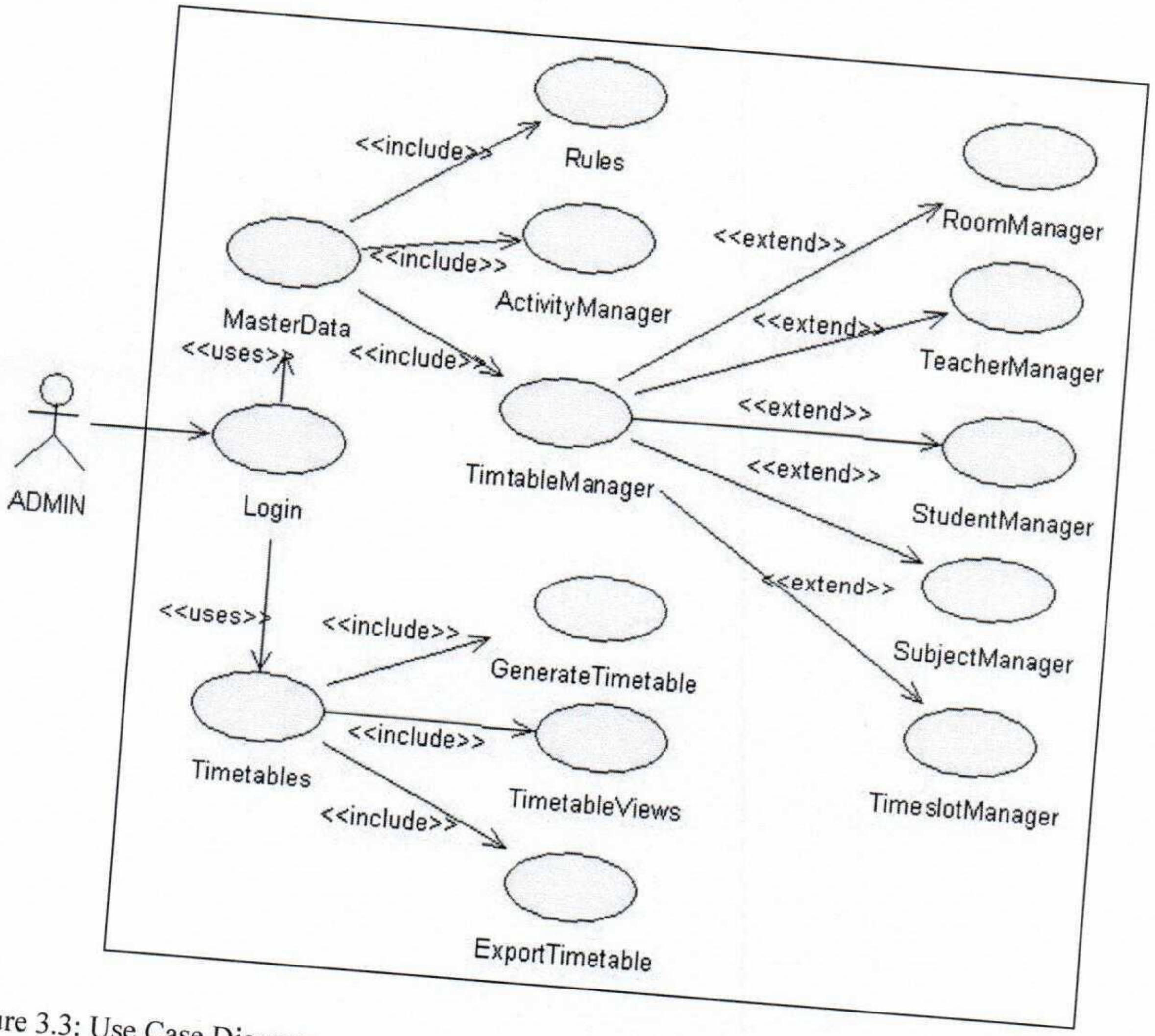


Figure 3.3: Use Case Diagram

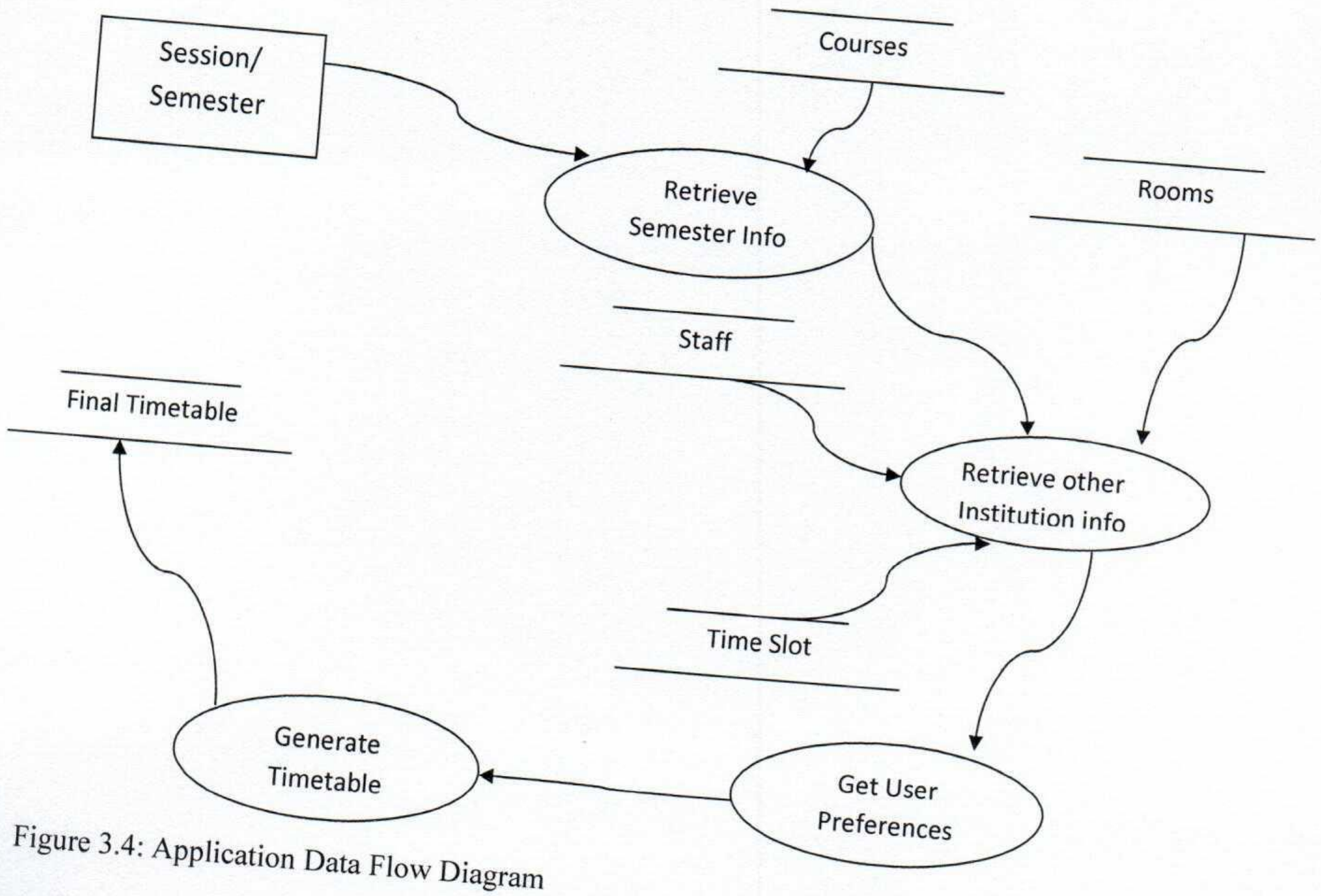


Figure 3.4: Application Data Flow Diagram

3.7 OUTPUT DESIGN

The output from any system is the major need of the user. As a matter of fact, any system that produces no output is useless. In the same vein, if the output generated by the system is not tailored to the need of the intended user, it is also useless. Hence, much effort needs to be invested on the generation and formatting of the output to suite the purpose for which the system has been developed for.

The main purpose of this project work is to automate the generation of timetable and make the resultant timetable available to different stakeholders within the University environment.

For example the college view of the timetable is expected to be as shown in table 3.10 below

Day/Time	8-9	9-10					
Monday	CSC301(Rm9) BIO101(Rm2) MCB201(Lab4)						
Tuesday							
Wednesday							
Thursday							
Friday							

```
For Each Department
    For Each Level
        For Each Period
            Get Course Code, Room Allocation
        Next Period
    Next Level
Next Department
```

In addition to that we can also generate timetable report for each department from the same database. The structure is similar to the college view except that; only the courses that are related to the department (general courses, departmental courses, borrowed courses and service courses) are displayed on the timetable.

Another report that can be generated is the level view or student view. It shows the periods at which a particular level, say 100 Level Computer Science, has lecture for the entire week.

In addition, a lecturer can also view his schedule of lecture for the entire week

Sample reports for each category of view are presented in the output section of the next chapter.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 INTRODUCTION

System implementation is the conversion of steps of system design to actual instructions and codes indicated to perform the tasks indicated in the system specification.System implementation concerns the period from the acceptance of the system design to its satisfactory operation supported by different categories of users that can operate the system with ease. It has to do with setting developed system functioning mode and mounting the procedure of converting or changing from the existing system of operation into newly developed system.

This phase refers to the creation of the new system from the designs established in the preceding chapter. It involves the following steps;

i. Programs development

ii. System testing

In this chapter, the requirement for the efficient implementation of the proposed system will be spell out. The programs will be listed and the user documentation will be listed out..

4.2 HARDWARE AND SOFTWARE REQUIREMENT

4.2.1 HARDWARE REQUIREMENT

The hardware requirements are the set of component of the system which can be seen or felt. For efficient and effective functioning of the system, the following hardware specifications are recommended for the Workstation:

i. A VGA/EGA Colour Monitor

ii. A LaserJet or DeskJet printer

iii. An Uninterruptible Power Supply (UPS) unit

iv. A stabilizer of about 1KVA

v. External storage device (Flash, Diskette, CD-R etc.)

vi. A Central Processing Unit with the following configuration

   i. A Intel Pentium® Dual Core Processor with a speed not less than 2.0GHz each

   ii. 40GB Hard disk size

   iii. 4GHz RAM (Random Access Memory) size

vii. A standby Generating Set,

4.2.2 SOFTWARE REQUIREMENT

The Software Requirement comprises of programs and data that make the hardware to carry out the specified task. The main software needed for this application is the Windows Operating System (e.g. Windows XP, NT, ME, Vista, 7or 8). This is because the application is window-based. Other required software such as Microsoft Access (used for back end design) will be bundled alongside with the application.

4. 3 SYSTEM TESTING

This is another step in system development. It is always necessary to test a program to ascertain whether or not it is able to produce correct result and that it conforms to the content of the requirement and specification document. It involves the testing running of the integrated program, with programs being run in their intended sequence using sample data. The sample data are selected so as to test various functions and sub-routines of the programs. The results of the testing are as shown in the figures below. The program listing is included in the appendix.

4.4 CHOICE OF PROGRAMMING LANGUAGE

There are various languages used in instructing the computer to carryout specified tasks. Most standalone applications are implemented using high level languages such as Visual BASIC 6.0, FORTRAN, QBASIC, C++, etc. Of al these tools, Microsoft Visual Basic 6.0 is selected for the application implementation.

The following are the justification for the selection of VB6.0 among others for the implementation:

i It is design specifically for Microsoft Windows Operating System which is the dominating operating system.

ii It support object oriented programming.

iii Visual basic 6.0 programming language provides forms, command buttons and other controls that you can easily drag and drop, thus making programming very easy.

iv It requires less line of code for the implementation when compared with to other programming languages.

v Microsoft Access which is use for the database implementation fully integrate with Visual Basic 6.0.

vi The language is rich with windows in-built-functions and components which can be incorporated into the program.

vii. It is easy to learn

4.5 INSTALLATION DESCRIPTION

The program for this project, which is Automated Timetable Generator is stored on a CD-ROM, and can be installed following these steps:

i. Insert the CD-ROM into the CD-ROM drive of the computer, then access the CD-ROM from my computer by double clicking the drive icon if it does not come up automatically.

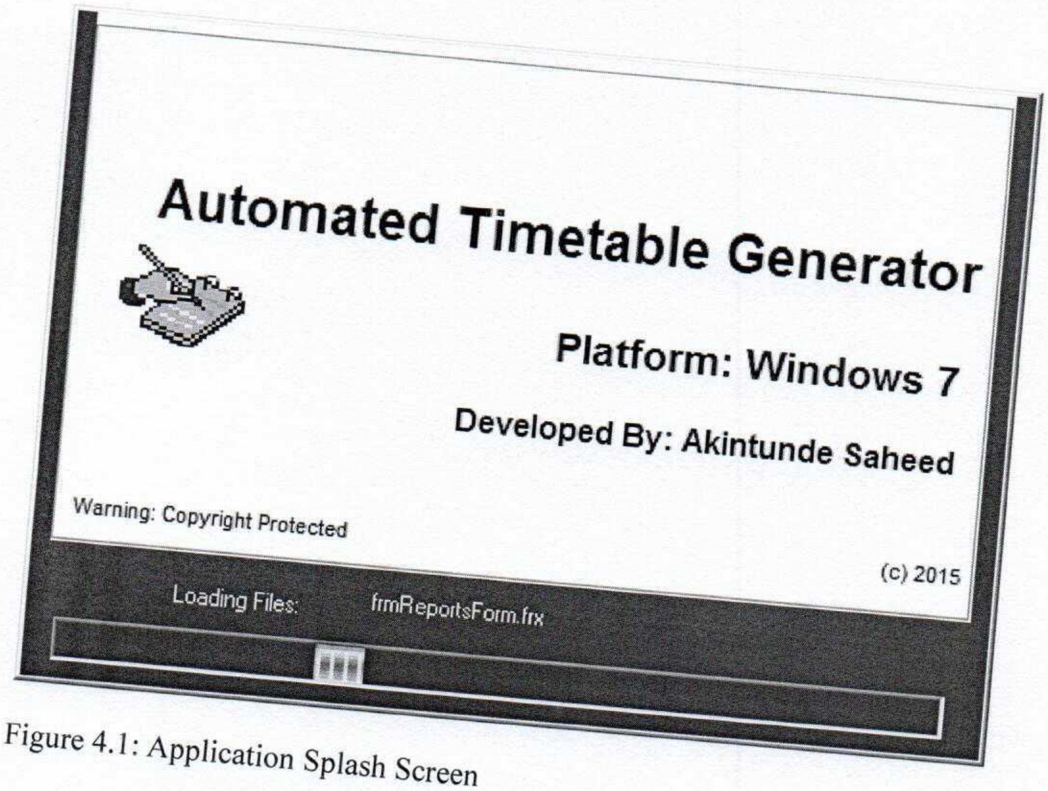
ii. Double-click the setup file located on the CD (…wait for some seconds)

v Follow the on-screen instructions which has been added to the application in order to aid the easy installation of the application.

vi After the installation, click on start menu, select all programs and then select Timetable Generator.

4.6 PROGRAM ANALYSIS

The program starts and presents the user with a splash screen as shown in figure 4.1 below. The splash screen which can also be referred to as welcome screen contains a little introduction and welcome note. The splash screen can be discarded by clicking on the form, pressing a key on the keyboard or automatically after 5 seconds. The login window is then presented afterwards asking the user to enter username and password. The login window is as shown in figure 4.2 below.



Automated Timetable Generator

Platform: Windows 7

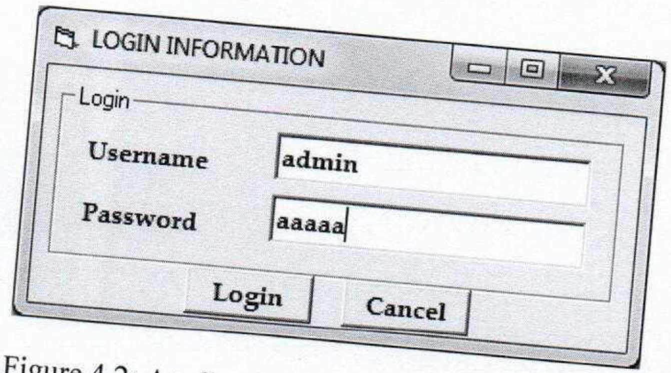
Developed By: Akintunde Saheed

Warning: Copyright Protected

(c) 2015

Loading Files: frmReportsForm.frx

Figure 4.1: Application Splash Screen



LOGIN INFORMATION

Login

Username: admin

Password: aaaaa

Figure 4.2: Application Login Screen

If the user specifies a valid username and password, the administrator panel will be displayed as shown in figure 4.3. The administrator panel allows the system administrator to setup the system by supplying the necessary data that the system will operate on as well as generate the timetable and other useful reports.

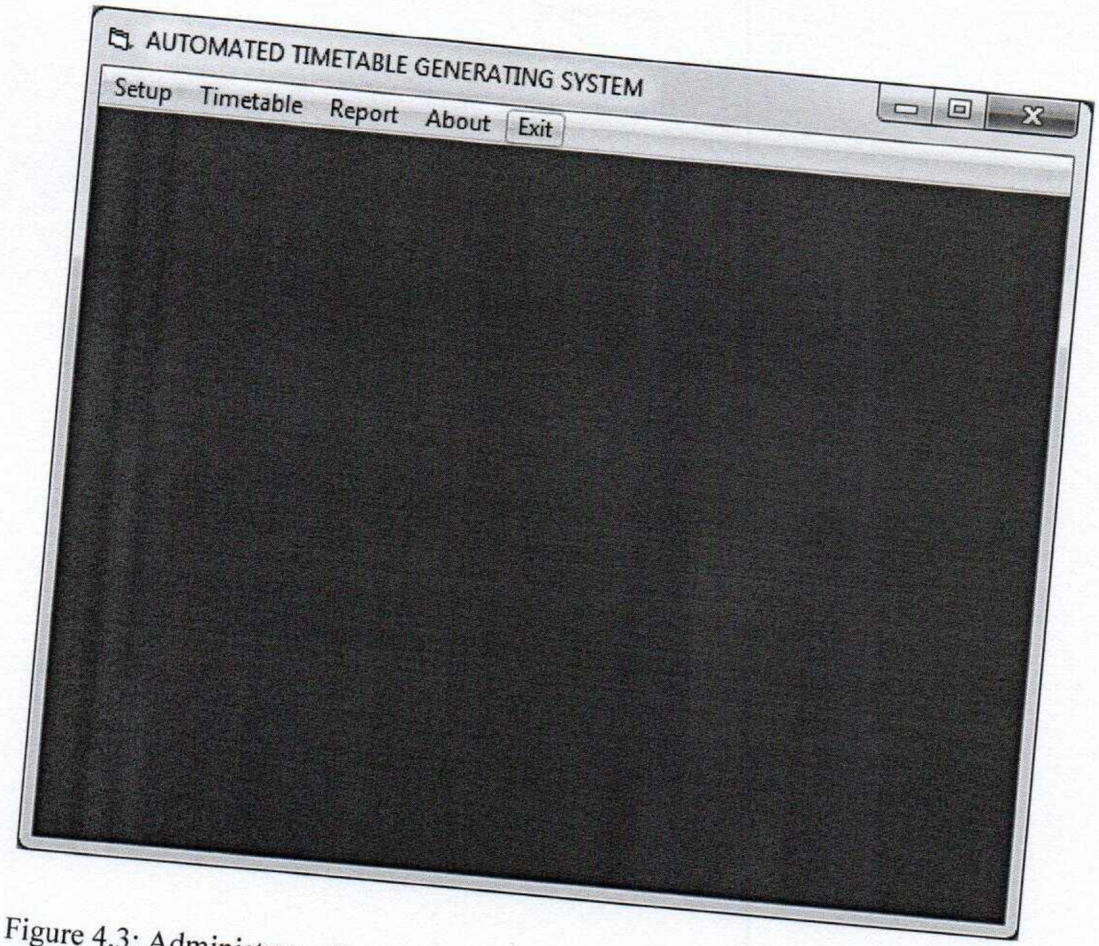


Figure 4.3: Administrator Panel

The administrator is presented with five (5) menu items as depicted in figure 4.3 above. The Setup, Timetable, Report About and Exit.

EXIT

The exit menu item is used to terminate the application. When clicked, it displays a message requesting the administrator to confirm his intention to close the application after which the application will be closed if the administrator selects yes.

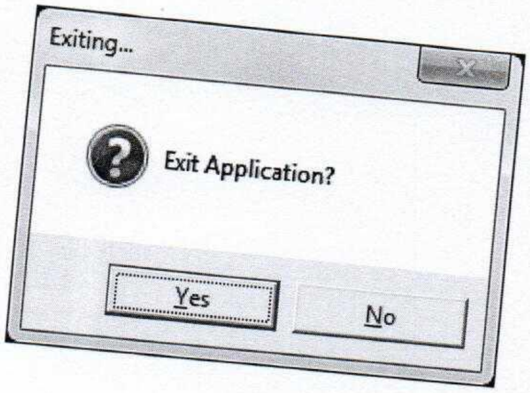
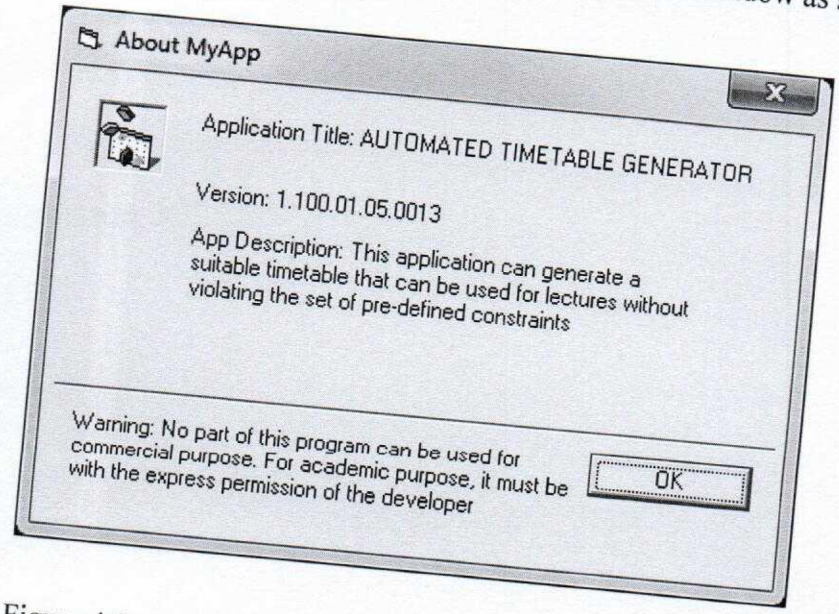


Figure 4.4: Exit confirmation dialog

ABOUT

The about menu item when clicked load the about window as shown in figure 4.5



Application Title: AUTOMATED TIMETABLE GENERATOR

Version: 1.100.01.05.0013

App Description: This application can generate a suitable timetable that can be used for lectures without violating the set of pre-defined constraints

Warning: No part of this program can be used for commercial purpose. For academic purpose, it must be with the express permission of the developer

Figure 4.5: About Window.

SETUP

The setup menu item contains some sub-menu items such as College, Department, Class, Courses, Staff, Lecture Room and Lecture Period manager. These sub-menu items give the administrator links to the form for managing each of the modules they correspond to. In order to register the colleges in the university, the administrator will select the College sub-menu item. A new window will be displayed as shown in figure 4.6 below

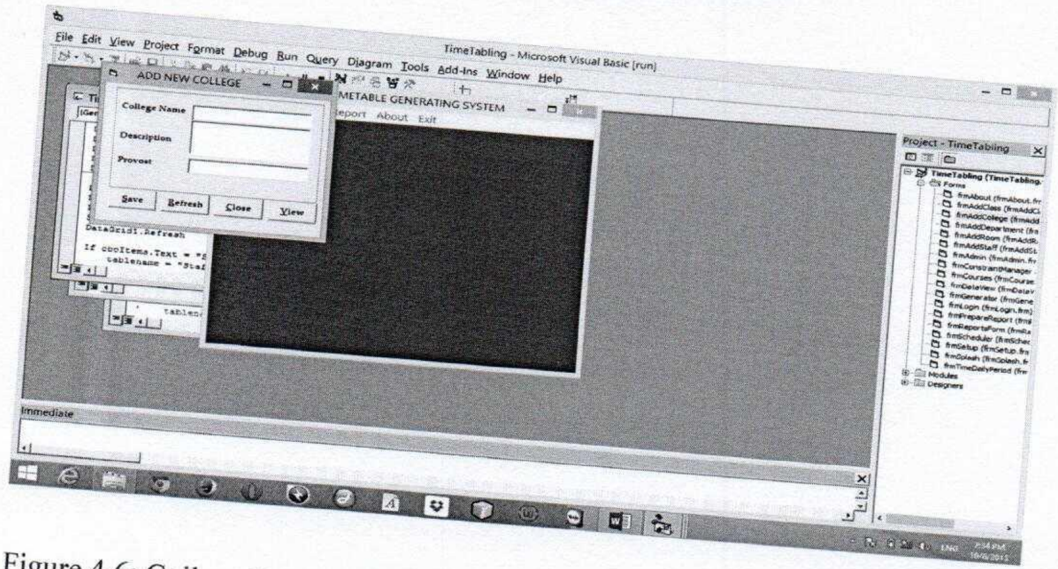
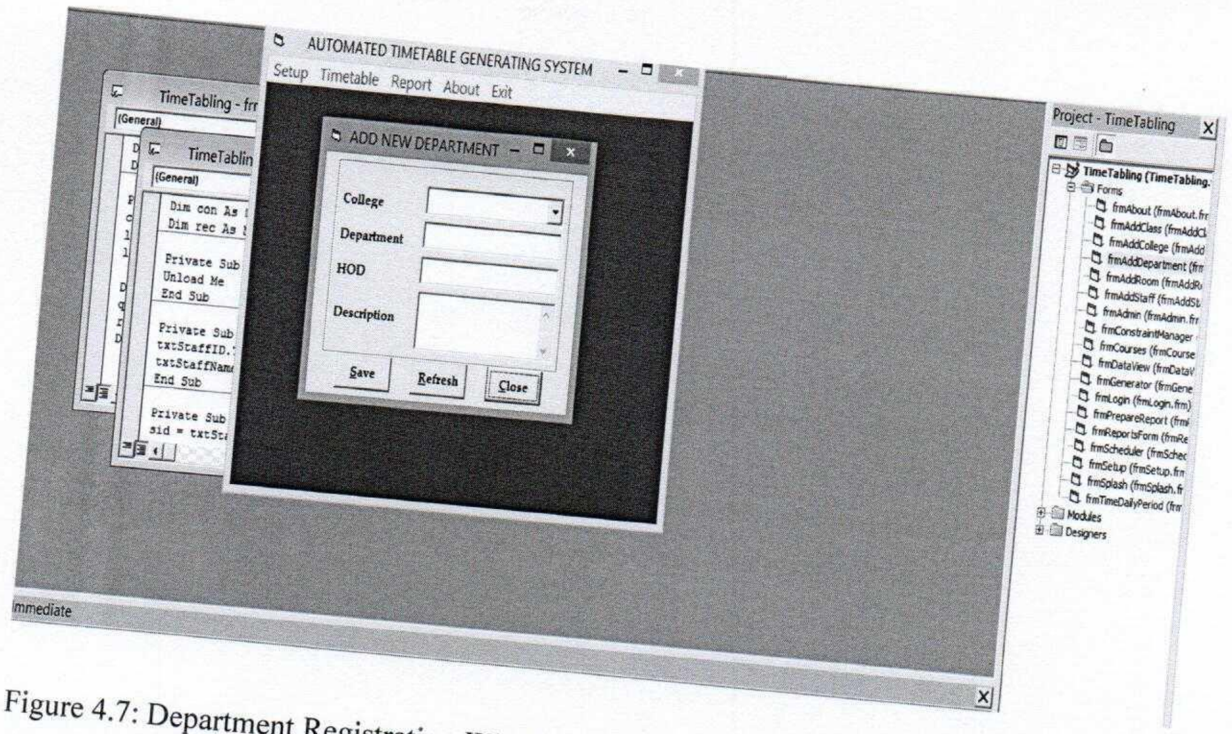


Figure 4.6: College Registration Window.

Each college in the school will be registered one after the other using the same window.

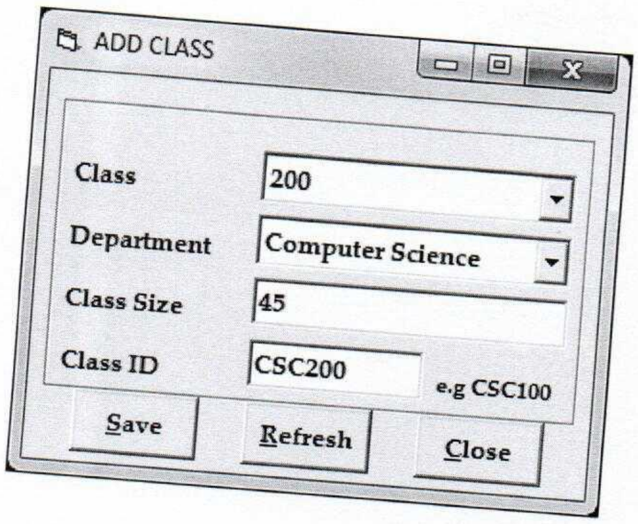
After the registration of college, the administrator needs to populate the database with the list of departments available. In order to achieve this, the administrator selects the Department sub-menu item from the Setup Menu list. The department registration window is as shown in figure 4.7 below.



```
Dim con As
Dim rec As

Private Sub
Unload Me
End Sub

Private Sub
txtStaffID.
txtStaffNam
End Sub

Private Sub
sid = txtSt
```

Figure 4.7: Department Registration Window.

Once the form is loaded, the college drop-down menu will automatically be populated with the list of all registered college. The administrator will then specify the departments in each college alongside other required information one after the other. After the completion of this stage, the administrator proceeds to add class to each department. These classes represent each level. This information will be updated each session to reflect changes in class size. The class registration window is as shown in figure 4.8.



ADD CLASS

Class: 200

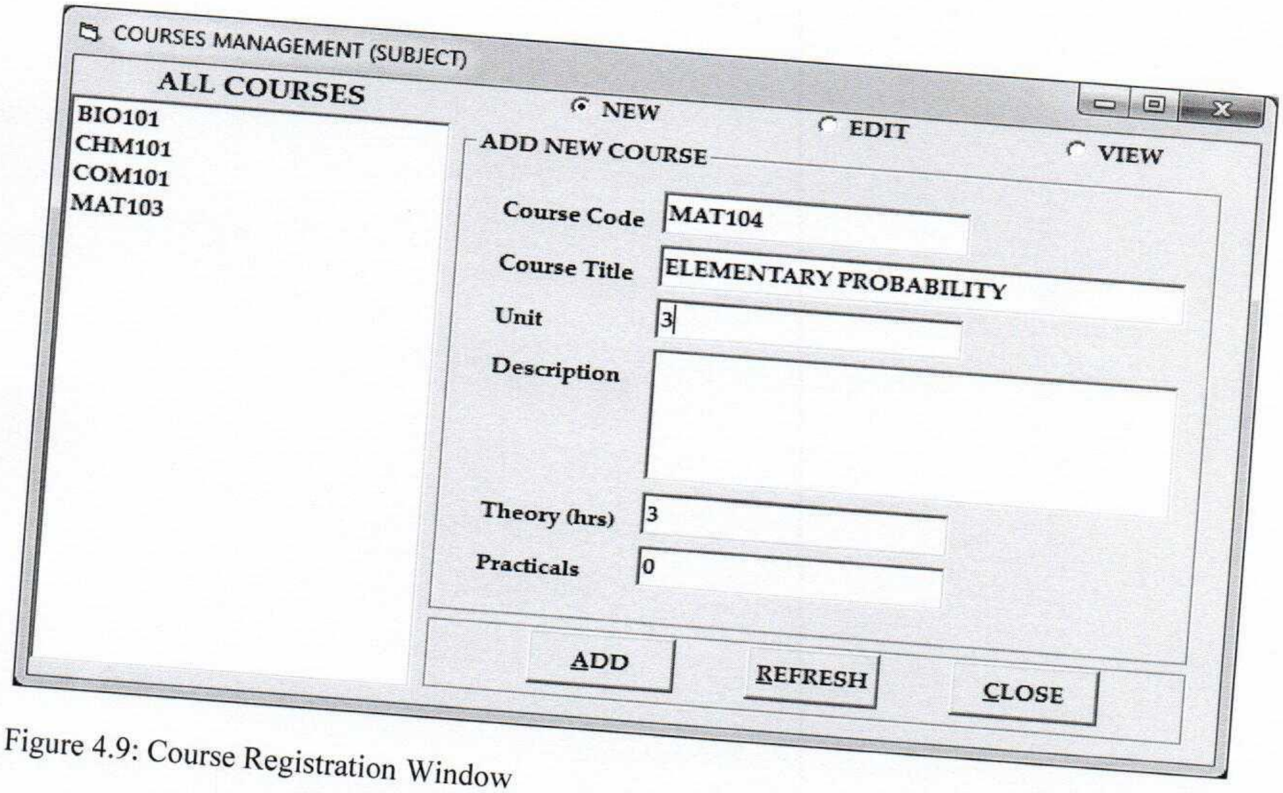
Department: Computer Science

Class Size: 45

Class ID: CSC200 e.g CSC100

Figure 4.8: Class Registration Window

The course registration is the next step after class creation. The course/ subject registration window is as shown in figure 4.9 below



COURSES MANAGEMENT (SUBJECT)

ALL COURSES

BIO101
CHM101
COM101
MAT103

[✓ NEW] EDIT VIEW

ADD NEW COURSE

Course Code: MAT104

Course Title: ELEMENTARY PROBABILITY

Unit: 3

Description:

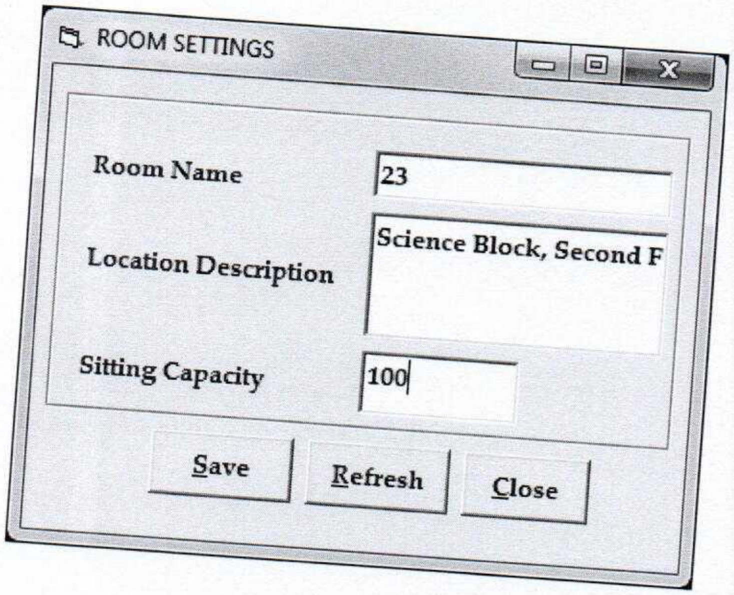
Theory (hrs): 3

Practicals: 0

Figure 4.9: Course Registration Window

All courses, general courses inclusive, are added to the courses database using this window.

The last on the list the is the room settings. The administrator registers all the available lecture theatre and their capacities. The Lecture Room registration window is as depicted in figure 4.10 below.


ROOM SETTINGS

Room Name: 23

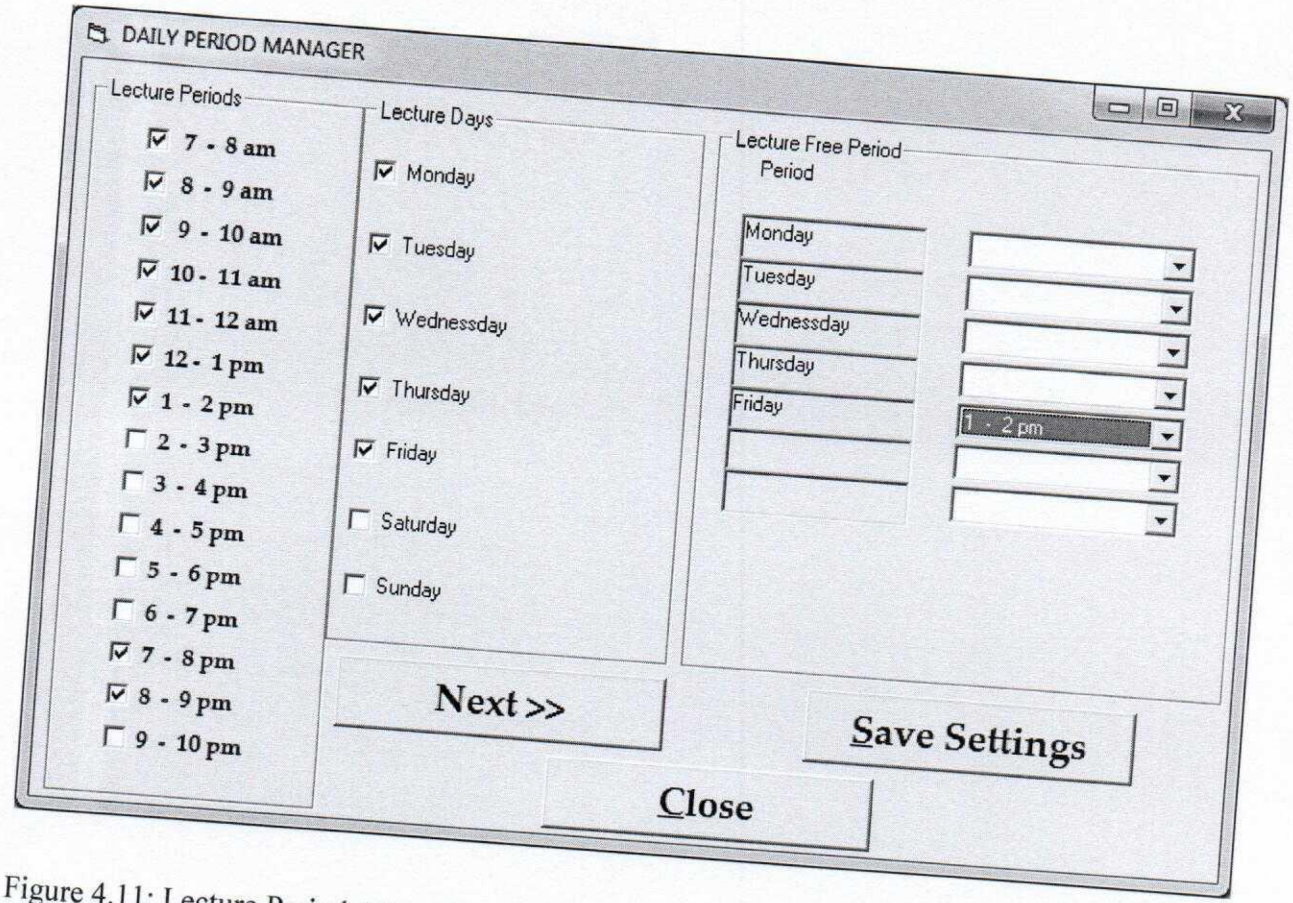
Location Description: Science Block, Second F

Sitting Capacity: 100

Figure 4.10: Lecture theatre registration window

The lecture period specification is the next step in setting up the system. The administrator specifies the lecture days such as Monday, Tuesday, etc and also the lecture hours for each day. Break periods such as Jumat break on Fridays are also specified on this screen. However, it should be noted that general break periods should not be included in the lecture period. Other break period could be probably weekly school programme etc.

Figure 4.11 below shows the lecture period allocation window


DAILY PERIOD MANAGER

Lecture Periods
- [✓] 7 - 8 am
- [✓] 8 - 9 am
- [✓] 9 - 10 am
- [✓] 10 - 11 am
- [✓] 11 - 12 am
- [✓] 12 - 1 pm
- [✓] 1 - 2 pm
- [ ] 2 - 3 pm
- [ ] 3 - 4 pm
- [ ] 4 - 5 pm
- [ ] 5 - 6 pm
- [ ] 6 - 7 pm
- [✓] 7 - 8 pm
- [✓] 8 - 9 pm
- [ ] 9 - 10 pm

Lecture Days
- [✓] Monday
- [✓] Tuesday
- [✓] Wednessday
- [✓] Thursday
- [✓] Friday
- [ ] Saturday
- [ ] Sunday

Lecture Free Period

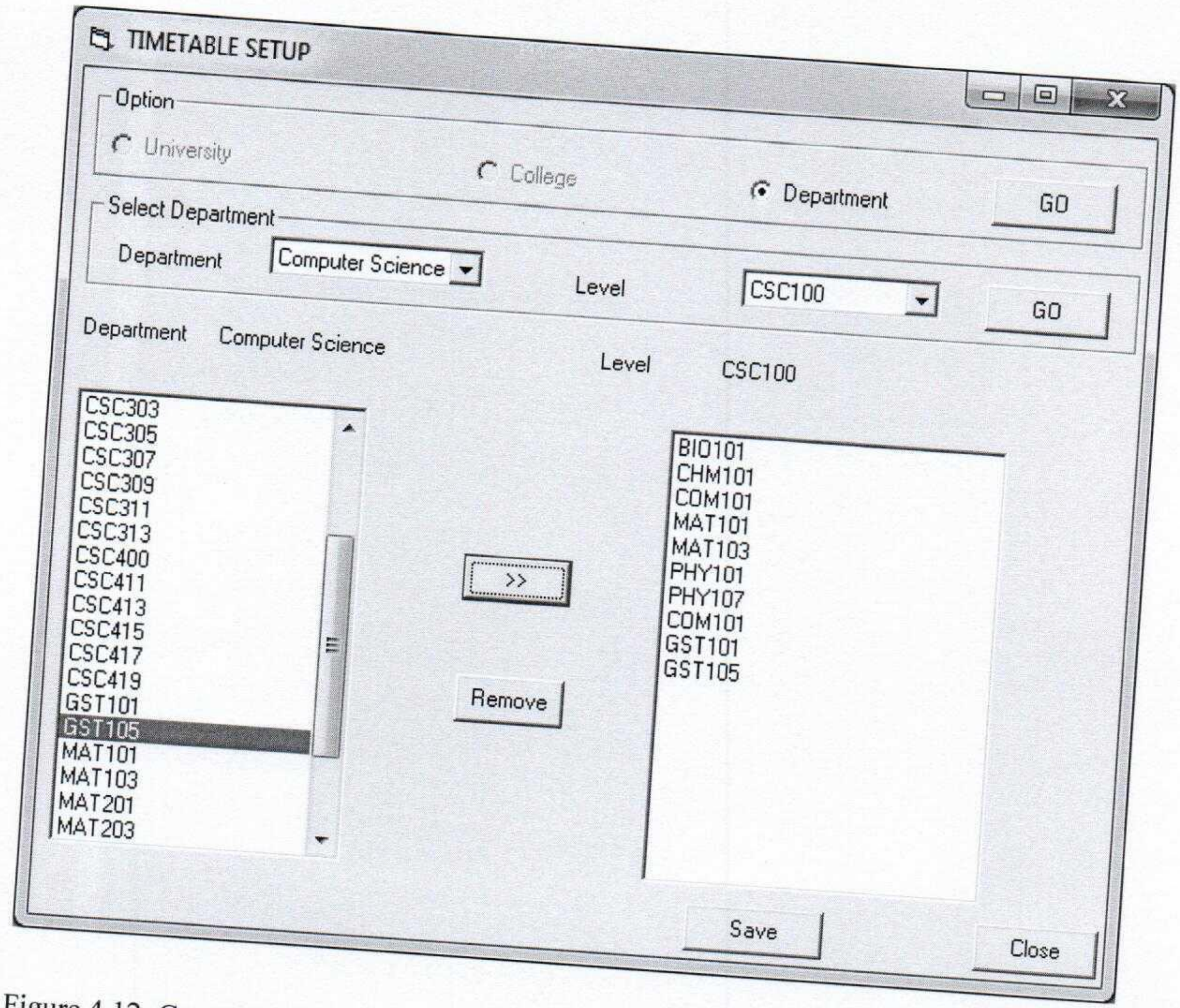
Period	
Monday	
Tuesday	
Wednessday	
Thursday	
Friday	1 - 2 pm

Figure 4.11: Lecture Period Allocation Window

The administrator selects the lecture period and lecture day before clicking the Next Button. After clicking the Next Button, the lecture free period can then be specified.

TIMETABLE

The Lecture period settings en the first phase of the system setup. The second phase is the timetable generation. The first step in generating the timetable is to select the set of courses to be offered by each class. That is, adding up courses that each class such as 100 Level Computer Science, 200 Computer Science, etc will offer for the particular semester. The course selection window is as shown below



TIMETABLE SETUP

Option

( ) University ( ) College [✓ Department] GO

Select Department

Department: Computer Science Level: CSC100 GO

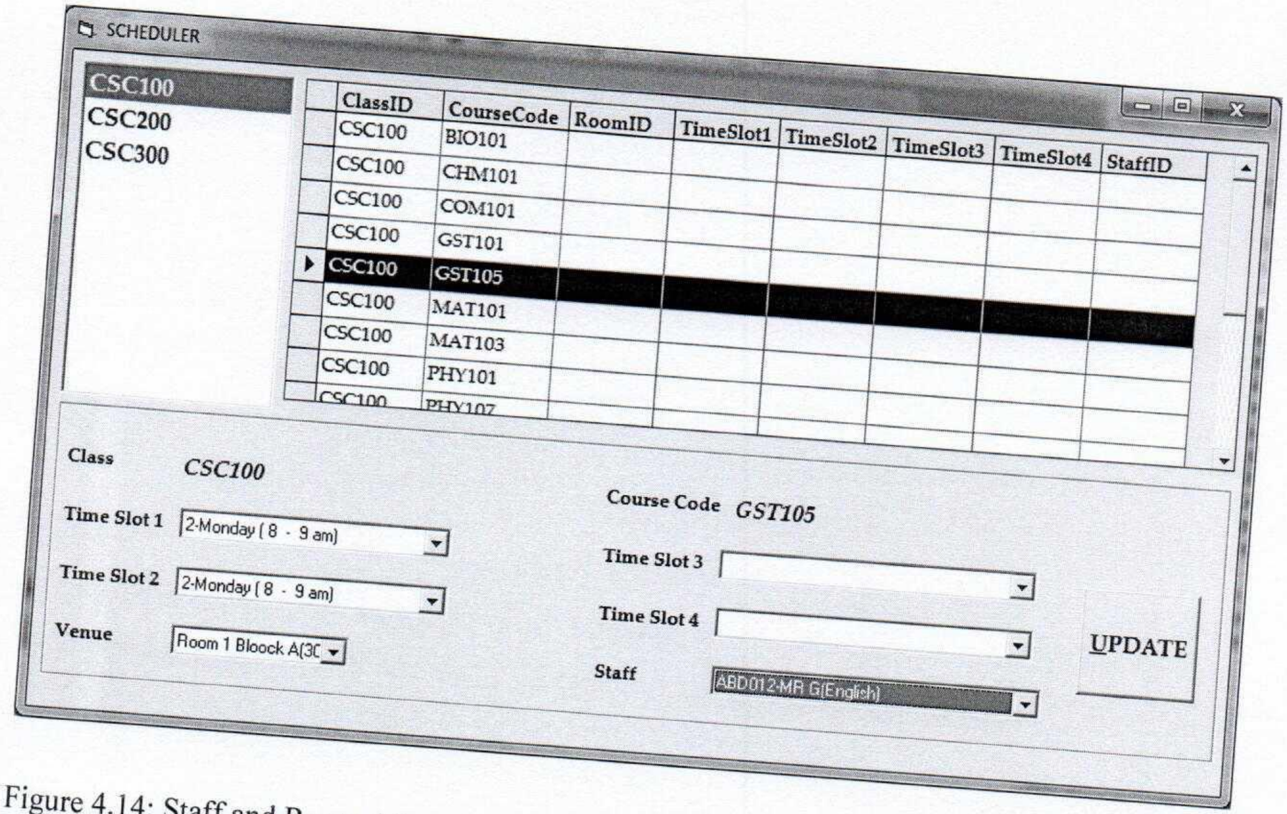
Department Computer Science Level CSC100

CSC303
CSC305
CSC307
CSC309
CSC311
CSC313
CSC400
CSC411
CSC413
CSC415
CSC417
CSC419
GST101
GST105
MAT101
MAT103
MAT201
MAT203

BIO101
CHM101
COM101
MAT101
MAT103
PHY101
PHY107
COM101
GST101
GST105

Figure 4.12: Course Selection Window

As shown in the figure above, the courses for each class will be selected. The next step is to allocate to each course, academic staff that will handle it and the venue where it will take place. The staff and room allocation window is as shown in figure 4.14 below



CSC100
CSC200
CSC300

ClassID	CourseCode	RoomID	TimeSlot1	TimeSlot2	TimeSlot3	TimeSlot4	StaffID
CSC100	BIO101						
CSC100	CHM101						
CSC100	COM101						
CSC100	GST101						
CSC100	GST105						
CSC100	MAT101						
CSC100	MAT103						
CSC100	PHY101						
CSC100	PHY107						

Class *CSC100*

Course Code *GST105*

Time Slot 1: 2-Monday (8 - 9 am)

Time Slot 2: 2-Monday (8 - 9 am)

Time Slot 3:

Time Slot 4:

Venue: Room 1 Bloock A(3C

Staff: ABD012-MR G(English)

Figure 4.14: Staff and Room Allocation Window

It should also be noted that lectures with already fixed period such as General Courses or Special Consideration can be specified manually on this screen while other unallocated courses will be left for the system to allocate.

The next step after staff and room allocation is the constraint specification. Constraint specification allows the administrator to specify the availability, tentative or non-availability of a particular resource. Other specific constraint can also be set on or off.

The timetable generation is the last step in the second phase.

A status message is being display at the completion of the allocation process.

4.7 OUTPUT OF THE SYSTEM

There are various outputs that can be generated from the system. These include List of Staff, Faculty, Department, Rooms and Courses. In addition to this, we can generate different views of the generated timetable such as College View, Departmental View, Student View, Staff View and Room Allocation.

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

The quest to make life easier and processing faster has led to computerization of various processes. In addition, Computer Technology has transformed so many sectors especially the education sector in no small measure. In an effort to foster technology driven education, we have designed and developed a timetable generation system that can generate timetable within a reasonable frame of time based on a set of given constraints. In addition to this, the solution can generate multiple arrangement of courses from which users are left with the task of choosing their choice from the set of generated solutions since this problem has more than one feasible solution. Although, the test data used were for a particular department, the software can easily be extended by populating information about other departments within the school and then adding the newly added departments to the generator list.

5.2 CONCLUSION

The developed solution will not only save time and stress that are involved in timetable generation but will also give users access to customize the report for each stakeholder such as Provost, Head of Department, Staff, Student, etc.

5.3 RECOMMENDATIONS

Due to the fact that the software was designed based on the evaluation of the various requirements and the study and analysis of existing system, and also that the system has been tested with live data, I will recommend that the system should be adopted in the department for timetable generation.

Also, I will like to recommend that further research should be encouraged on this area in order to extend the application to accommodate examination timetable generation.

Finally, I will like to recommend that entry level student should be encourage to identify problems within their environment and start working on such project right from their first year in school; as this will promote the design and development of quality systems and also remove the time constraint faced by many final year students at the implementation stage of their project work.

REFERENCES

Abramson, D. 1991. Constructing School Timetables Using SimulatedAnnealing: Sequential and Parallel Algorithms Management-Science

Abramson, D. and Dang, H. 1993. School Timetables: A Case Study inSimulated Annealing," Applied Simulated Annealing, Lecture Notes inEconomics and Mathematics Systems, Springer-Verlag, Ed. V. Vidal.

Alvarez-Valdes R., Crespo E., and Tamarit J. M., 2002: "Design And Implementation of A Course Scheduling System UsingTabu Search", *European Journal of Operational Research,*137: 512–523.

Back, T. 1995. Evolutionary Algorithms in theory and practice New-York:Oxford University Press.

Bajeh A.O and Abolarinwa K.O, 2011: Optimization: A comparative study of Genetic and Tabu Search Algorithms. International Journal of Computer Applications 31 (5): 43-48.

Buckles B. P. and Petry F. E. , 1992. Genetic Algorithms. IEEE ComputerSociety Press.

Chambers L. 1995: Practical Handbook of Genetic Algorithms: Applications

Condor, 1988: Operations Research: The Next Decade, Operations ResearchCommittee On the Next Decade of Operations Research.

Deris, S. B. , Omatu, S. , Ohta, H. , and Samat, P. , 1997. Universitytimetabling by constraint-based reasoning: a case study. Journal of theOperational Research Society.

Dikmann R. , Luling R. and Simon J. , 1993. Problem independent distributedsimulated annealing and its applications. Technical Report -- Paderborn Centerfor Parallel Computing.

Egwali A.O. and Imouokhome F.A., 2012: "Synthesis-Algo: An inclusive timetable generating algorithm", *African Journal of Computing and ICT,* 5 (4): 92-100.

Eugene. F and Mackworth A. , 1994. The complexity of some polynomialsearch algorithms for constraint satisfaction problems. Artificial Intelligence

Feiring B.R., 1986: Linear Programming 60: An introduction Publication

Freuder, E. C. and Wallace R. J. , 1994. Partial Constraint Satisfaction. Artificial Intelligence.

Glover F. , 1986. Future Paths for Integer Programming and Links toArtificial Intelligence. Computers and Operations Research.

Glover F. 1993. Future path for integer programming and links to artificialintelligence. Computer & Ops. Res.

Glover F. and Laguna M. 1997. Tabu Search. Kluwer Academic Publishers,Boston, MA.

Glover, F. , 1989. Tabu search- Part I. ORSA Journal on computing.

Glover, F. , 1997. A Template for Scatter Search and Path Relinking. LectureNotes in Computer Science.

Goldberg, 1989. Genetic Algorithms in Search, Optimization and MachineLearning. Addison-Wesley, Hart E. and Ross P, 1999: An Immune System Approach to Scheduling in Changing Environments, Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO '99), Florida, July 13-17 1999, pp. 1559-1566.

Hertz A , 1991. Tabu search for large scale timetabling problems. European Journal of Operational Research.

Khaled Mahar, 2006: "Automatic generation of University timetables: An evolutionary approach" *IADIS International Conference on Applied Computing* 570-574.

Kirkpatrik S. , Gelatt C. D. and Vecchi M. P. , 1983. Optimization bysimulated annealing. Science.

Kuldeep S.S., 2001: Automating Class Schedule Generation in the Context of a University Timetabling Information System. Ph.D. (Unpublished) Thesis, Nathan Campus, Griffith University.

Kumar V. , 1992. Algorithms for Constraint Satisfaction Problems: ASurvey. AI Magazine.

Lhomme O. 1993. Consistency techniques for numeric CSPs. 13th Intl. JointConference on Artificial Intelligence

Mackworth A. , 1994. The Dynamics of Intelligence: Constraint-SatisfyingHybrid Systems For Perceptual Agents.

Maxfield, C. , 1997. Genetic algorithms: programs that boggle the mind.EDN.

Muhammad Rozi Malim, Ahamad Tajudin Khader, Adli Mustafa, 2006: An Immune-Based Approach to University Course Timetabling: Negative Selection Algorithm

Paulli, J. , 1993. , Information utilization in simulated annealing and tabusearch. COAL Bulletin no.

Plain 1995

Sanchez E. , Shibata T. Zadeh L. Eds. , 1997. Genetic Algorithms and FuzzyLogic Systems. Soft Computing Perspectives. World Scientific.

Sandeep S. Rawat and Lakshmi Rajamani, 2005: "A timetable prediction for technical educational system using genetic algorithm" *Journal of Theoretical and Applied Information Technolog.* 13 (1) 59-64.

Schaerf A. , 1999. Local Search Techniques for Large High School

Sharma D and Chandra N. , 1999. An evolutionary approach to constraintbasedtimetabling. Suva, Fiji : Department of Mathematics and ComputingScience, SPAS, USP

Srinivas, M. and Patnik, L. M. 1994. Genetic Algorithms: A Survey.Dordrecht: Kluwer Academic Publishers.

Thompson E.P., 1967: Time, Work-Discipline and Industrial Capitalism. Past and Present.

Timetabling Problems. IEEE Transactions on Systems, Man And Cybernetics.

Tsang, E. P. K. Mills P. Williams R. Ford J. & Borrett, J. 1999. A computer-aided constraint programming system. The First International Conference onThe Practical Application of Constraint Technologies and Logic Programming(PACLP), London.

Wren, A. (1996): Scheduling, Timetabling and Rostering – a special relationship? In Lecture Notes in Computer Science: Practice and Theory of Automated Timetabling, E. Burke and P. Ross, editors. Springer Berlin, Germany, 1153: 46-75.

Zoltan Petres, Sandor Gyori, and Annamaria R.V., 2000: Genetic Algorithms in Timetabling: A new Approach.

APPENDIX

```
Dim con As New ADODB.Connection

Private Sub cmdClose_Click()

Unload Me

End Sub

Private Sub cmdRefresh_Click()

txtCollegeName.Text = ""

txtDescription.Text = ""

txtProvost.Text = ""

End Sub

Private Sub cmdSave_Click()

On Error GoTo SaveError

cn = txtCollegeName.Text

dsc = txtDescription.Text

pn = txtProvost.Text

If (cn = "") Then

    MsgBox "College Name must be specified", vbInformation, "Entry Validator"
    txtCollegeName.SetFocus

    Exit Sub

End If

qry = "INSERT INTO College VALUES('" & cn & "','" & dsc & "','" & pn & "')"

con.Open getConnection

con.Execute qry

MsgBox "Record Saved", vbInformation, "Database"

Call cmdRefresh_Click

Exit Sub

SaveError:

MsgBox Err.Description & crlf & "Unable to save record", vbCritical, "Error 1101"

End Sub
```