

THE UNIVERSITY OF ZAMBIA

SCHOOL OF NATURAL SCIENCES

DEPARTMENT OF COMPUTER STUDIES



SPR

C/SCIENCE

VUS

2013

C.1

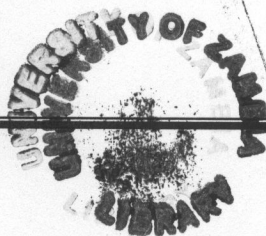
DEVELOPMENT OF A TIME TABLE CREATION SYSTEM

BY

VUSIMUZI JERE ALISTER (29001692)

SUPERVISOR: DAVID M ZULU

CST 4000 FINAL YEAR PROJECT



Time Table Creation System

BY VUSIMUZI JERE ALISTER

✱

September 2013

NAME: VUSIMUZI JERE ALISTER

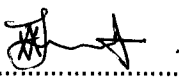
SUPERVISOR: DAVID M ZULU

DATE OF SUBMISSION: 4th September 2013

DECLARATION

I, **Vusimuji Jere Alister**, declare that this dissertation is my own original work and has not been presented and will not be presented to any other university for a similar or any other degree award.

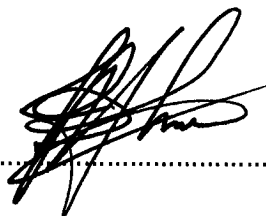
Name: **Vusimuji Jere Alister**

Signature: 

Date: 18th September 2013

SUPERVISOR

Name: **Mr. D.Zulu**

Signature: 

Date: 18/09/2013

00256323

ACKNOWLEDGMENTS

First of all, I would like to give thanks to the Almighty God who enabled me to work hard and produce this work. I thank God with all my heart for He has been on my side. Glory be to the Almighty God. I would like also to express my sincere gratitude to my supervisor, Mr. David Zulu for his support and valuable advice.

DEDICATION

I dedicate this work to my family and friends that have been with me during my tough times.

ABSTRACT

A college timetable is a temporal arrangement of a set of lectures and classrooms in which all given constraints be it hard or soft are satisfied. The Time Tabling Problem is a well-known combinatorial search problem that consists of scheduling events over a period of time. The events being scheduled usually require resources such as rooms and people and are usually subject to certain constraints.

The problem often arises in real-life situations, solving these instances of the problem from scratch is way beyond humans. Practically all available time-tabling research and software products concern themselves strictly with finding good feasible solutions. Trying to find an optimal solution is too hard of a problem, such attempts are simply not justified by practical considerations - searching for non-optimal solutions often yields worthy results and incurs relatively low costs.

The variant of the problem this project deals with is coming up with a time-table for courses being taught at a university. Creating such timetables manually is complex and time-consuming process as there has to be several if not all the universities lecturers in order for them to come to an agreement of which rooms or times they are to take. This is very impractical as a result administrators take on this role thereby leading to several errors and people are forced to abide to a time table they are not comfortable with. By automating this process with computer assisted timetable generator can save a lot of precious time of administrators who are involved in creating and managing course timetables. Since every college has its own timetabling problem, the commercially available software packages may not suit the need of every college.

This project introduces a practical timetabling algorithm capable of Taking care of both strong and weak constraints effectively, used in an automated exam timetabling system.

Table of Contents

DECLARATION..... ii

ACKNOWLEDGMENTS..... iii

DEDICATION..... iv

ABSTRACT..... v

CHAPTER ONE – INTRODUCTION TO THE RESEARCH 1

1.1 INTRODUCTION..... 1

1.2 MOTIVATION AND SIGNIFICANCE OF THE THESIS 2

1.3 SCOPE..... 3

1.4 PROBLEM STATEMENT..... 3

1.5 OBJECTIVES AND AIMS..... 3

1.5.1 AIM 3

1.5.2 OBJECTIVES 3

1.6 ORGANISATION OF THE THESIS 4

1.7 SUMMARY..... 5

CHAPTER TWO - BACKGROUND THEORY AND RELATED WORKS 6

2.1 INTRODUCTION..... 6

2.2 SOLUTION GENERATION ALGORITHMS..... 6

2.2.1 LINEAR PROGRAMMING/INTEGER PROGRAMMING 7

2.2.2 EVOLUTIONARY AND GENETIC ALGORITHMS..... 8

2.2.3 SIMULATED ANNEALING..... 10

2.2.4 TABU SEARCH 10

2.2.5 CONSTRAINT BASED REASONING 12

2.2.6 REPRESENTATION..... 15

2.3 RELATED WORKS 16

2.3.1 BIRBAS 16

2.3.2 M. Mitchell, “An introduction to Genetic Algorithms” 16

2.3.3 DANCUS (EVOLUTIONARY ALGORITHM)..... 17

2.3.4 Ahmed Al-Tehami, Ahmed Al-Salehi, AmmarZahary, "APPLICATION OF GENETIC ALGORITHMS IN THE PROBLEM SOLVING OF COURSE TIMETABLES,..... 18

2.3.5 Related Work Summary 19

2.4 SUMMARY..... 19

CHAPTER THREE - METHODOLOGY 21

3.1	INTRODUCTION	21
3.2	SYSTEMS REQUIREMENTS	21
3.3	SYSTEM DESIGN	23
3.3.1	GUI PACKAGE.....	24
3.3.1.1	CourseGUI	24
3.3.1.2	RoomGUI	24
3.3.1.3	PasswordDialog.....	24
3.3.1.4	MultiCellRenderer	24
3.3.1.5	DbInput	24
3.3.1.6	DbInput2	24
3.3.1.7	TabbedPane.....	25
3.3.1.8	UnzaImage2	25
3.3.2	DATA PACKAGE.....	25
3.3.2.1	Algorithm	25
3.3.2.2	ActivityEntry.....	25
3.3.2.3	Random.....	25
3.3.2.4	CourseData	25
3.3.2.5	RoomData	25
3.3.2.6	ActivitySet.....	26
3.3.2.7	TimeSet.....	26
3.4	SYSTEM IMPLEMENTATION	26
3.5	SUMMARY	30
CHAPTER FOUR – SYSTEM TESTING		31
4.1	INTRODUCTION.....	31
4.2	SYSTEM TESTING.....	31
4.2.1	APPLICATION TESTING	31
4.2.2	PORTABILITY TESTING	41
4.3	RESULTS.....	42
4.4	SUMMARY.....	44
CHAPTER FIVE – DISCUSSION AND CONCLUSION.....		45
5.1	INTRODUCTION.....	45
5.2	DISCUSSION	46
5.3	CONCLUSION.....	47

5.4 FUTURE WORKS47

5.5 SUMMARY.....47

REFERENCES.....52

List of Figures

Figure 1: Generation Phases	22
Figure 2: System USE CASE	23
Figure 3: System Design.....	23
Figure 4: Password Input Screen	26
Figure 5: TabbedPane	27
Figure 6: Algorithm class	27
Figure 7 MultilineCellRenderer class.....	28
Figure 8: Generated Time Table.....	29
Figure 9: html timetable view	30
Figure 10: security interface	31
Figure 11: Invalid entry	32
Figure 12: Front screen.....	32
Figure 13: Front Input screen	33
Figure 14: Room input	34
Figure 15: Database input1	35
Figure 16: DatabaseInput2	36
Figure 17: Course duplicate.....	37
Figure 18: Room duplicate	38
Figure 19: Edit view	38
Figure 20: Course presence	39
Figure 21: Room presence	40
Figure 22: Course database table.....	40
Figure 23: Room database table.....	41
Figure 24: PORTABLE PACKAGE	42

Figure 25: Time table42

Figure 26: HTML TABLE43

Appendices

Appendix I: Main Algorithm Class.....48

CHAPTER ONE – INTRODUCTION TO THE RESEARCH

1.1 INTRODUCTION

This thesis examines the general and well established research into the timetable scheduling problem. Even though most college administrative work has been computerized, the lecture timetable scheduling is still mostly done manually due to its inherent difficulties. The manual lecture-timetable scheduling demands considerable time and efforts. The lecture-timetable scheduling is a constraint satisfaction problem in which we find a solution that satisfies the given set of constraints.

The Time Tabling Problem is a well-known combinatorial search problem that consists of scheduling events over a period of time. [1] The first electronic timetable information systems were established in the late eighties of the last century. Current systems are for example HAFAS, which is used by many European railway companies, or EFA, which is mainly used for local traffic limited to smaller regions in Europe.

The events being scheduled usually require resources such as rooms and people. Even though most college administrative work has been computerized, the lecture timetable scheduling is still mostly done manually due to its inherent difficulties. The manual lecture-timetable scheduling demands considerable time and efforts. The college lecture-timetabling problem asks us to find some time slots and classrooms which satisfy the constraints imposed on offered courses, lecturers, classrooms and so on. For such a timetable such as a university type there are several constraints, as a result the problem and the computation time for timetabling tends to grow exponentially as the number of variables increase.

There have been a number of approaches made in the past decades to the problem of constructing timetables for colleges and schools. Timetabling problems may be solved by different methods inherited from operations research such as graph coloring and mathematical

programming, from local search procedures such as tabu search and simulated annealing, from genetic algorithms or from backtracking-based constraint satisfaction manipulation.

In this project, timetabling problem is formulated as a constraint satisfaction problem and a practical timetabling algorithm which is capable of taking care of both strong and weak constraints and finding variables' instantiation is to be used, which is based on the forward search method. There exist many different timetabling problems such as:

- University Timetabling
- Examination Timetabling
- School Timetabling
- Sports Timetabling
- Employee Timetabling

Furthermore, there exist many problem solving methods, which usually use the concepts of standard optimization algorithms such as Backtracking, Evolutionary Algorithms or Constraint Logic Programming.

In recent years two main approaches seem to have been successful.

- The first approach is based on Local Search Procedures
- The second approach is based on Constraint Programming (CP)

1.2 MOTIVATION AND SIGNIFICANCE OF THE THESIS

The Time Tabling Problem is a well-known combinatorial search problem that consists of scheduling events over a period of time. The events being scheduled usually require resources such as rooms and people and are usually subject to certain constraints (such as: a person cannot attend more than one event at the same time).

The problem often arises in real-life situations, solving these instances of the problem from scratch is way beyond humans [3](at HUJI no software is used to come up with time-tables, each year's time-table is a slight variation of the preceding year's one).

Practically all available time-tabling research and software products concern themselves strictly with finding good feasible solutions. Trying to find an optimal solution is too hard of a problem, such attempts are simply not justified by practical considerations - searching for non-optimal solutions often yields worthy results and incurs relatively low costs.

The variant of the problem this project deals with is coming up with time-tables for courses being taught at a university.

1.3 SCOPE

Exam Timetable Generation System generates timetable for each exam venue and teacher (invigilator), in keeping with the availability calendar of teachers, availability and capacity of physical resources (such as classrooms, laboratories and computer room).

Best of all, this Timetable Generation System tremendously improves resource utilization and optimization.

1.4 PROBLEM STATEMENT

The current system of manually creating a time table has proven ineffective as there have been several errors in universities. Errors mainly occur due to wrongly assigning resources to a class about to have an exam.

1.5 OBJECTIVES AND AIMS

1.5.1 AIM

The main aim of this project is to develop a program that creates a timetable that puts all constraints into consideration and ensures that none of the hard constraints are violated due to the fact that violation of any hard constraint defeats the purpose of using the program.

1.5.2 OBJECTIVES

The purpose of the project is to minimize the errors that occur on a daily basis involving creation of timetables as this has proven to be a very difficult task when many constraints are

put into consideration. On a large scale such as a University, errors in a timetable are a common factor therefore greatly reducing the errors is of great importance. The following are objectives of the project

- Obtain all necessary information concerning the resources of the university such as exam center venues and department names
- Obtain various knowledge concerning the properties of the resources such as room capacity as this is going to be of great importance.
- Research on the various constraints that may be needed to prevent clashes between different class exams.
- Ensure that the system is correctly storing the required data as it is inputted in order for processing.
- Ensure that the algorithm is correctly implemented and working on the data that is typed in so as to provide output.
- Ensure the system has a meaningful user interface.
- Ensure the system has a means of providing a meaningful output in a consecutive manner.

1.6 ORGANISATION OF THE THESIS

The next chapter presents an overview of the timetable generation literature. This overview is rationalized and classified by solution algorithm. It begins with a presentation of the major solution generation algorithms that have persisted in the literature. This required the treatment to be condensed to a succinct coverage of the salient features of each algorithm. The subtle nuances that differentiate the implementations of the algorithms from each other were covered in the extensive literature that follows. This broad literature survey is carried out in order to illustrate the evolution and trend of timetable problems from the 1950's through to date. The objective of the survey is to ascertain the way the solution to timetable problems is heading and then to provide a system that can endeavor to assist in this process. In order to carry out this task it is necessary to summarize the results of the literature survey and they are presented in Chapter 2. Following this a rationalization and classification of the literature is carried out in

order to highlight the trends in timetabling. This chapter then concludes with a concise analysis classification of the extent to which each of the solution algorithms has been used in the literature. This is followed in chapter three by the discussion of the development and implementation of an information system generic to supporting all aspects of the timetable problem. The chapter is concluded with a presentation of the results generated and a discussion on the effectiveness of the system. Chapter four presents the first benchmark test.

1.7 SUMMARY

The primary objective in preparing timetable is obtaining conflict free for each activity sharing the same resources. Most of the method faces difficulties in generating a feasible timetable which faces long processing period without complying with several conditional constraints.

CHAPTER TWO - BACKGROUND THEORY AND RELATED WORKS

2.1 INTRODUCTION

This chapter provides an analysis of the automated timetable literature broadly organized by algorithmic technique. It begins with a presentation of the major timetable solution generation algorithms that have persisted in the literature. A detailed examination of the academic literature is provided within the context of these fundamental solution generation algorithms. An analysis of the literature, grouped by the solution generation technique used, is then presented. The results show some interesting trends in the progressive cumulative research that has evolved over the last 5 decades.

The chapter concludes with a summary of the major findings from the analysis. It is demonstrated that the literature is currently converging on the use of constraint based solution algorithms and implementations. It is also noted that the next most commonly reported implementation involves the use of hybrid algorithms. Possible avenues to further this analysis are then presented. A small, though not insignificant, number of more recent publications is noted to not contribute directly to the solution generation debate. These papers report on development of Timetable Information Systems rather than extending generation algorithms and solution techniques.

2.2 SOLUTION GENERATION ALGORITHMS

There are currently many different solution generation algorithms in existence. Some are so well known that they are the subject of standard textbook material. This section presents major algorithms that have appeared in the literature on generating timetabling solutions. For the sake of brevity only the fundamental algorithms themselves are presented, not the many hybrid approaches adopted.

[2]The fundamental solution generation techniques developed in the literature are generally acknowledged to be the following:

1. Integer Programming/Linear Programming
2. Constraint Satisfaction Programming
3. Genetic and Evolutionary Algorithms
4. Simulated Annealing
5. Tabu Search

A brief description of these fundamental techniques is now presented. Each of these descriptions is structured to begin with a general formulation of the technique.

2.2.1 LINEAR PROGRAMMING/INTEGER PROGRAMMING

The Linear and Integer Programming techniques, the first applied to timetabling were developed from the broader area of mathematical programming. Mathematical programming is applicable to the class of problems characterized by a large number of variables that intersect within boundaries imposed by a set of restraining conditions.[13]The word "programming" means planning in this context and is related to the type of application. [14] This scheme of programming was developed during World War II in connection with finding optimal strategies for conducting the war effort and used afterwards in the fields of industry, commerce and government services.

[7]Linear Programming (LP) is that subset of mathematical programming concerned with the efficient allocation of limited resources to known activities with the objective of meeting a desired goal such as maximizing profits or minimizing costs. Integer Programming (IP) deals with the solution of mathematical programming problems in which some or all of the variables can assume non-negative integer values only. Although LP methods are very valuable in formulating and solving problems related to the efficient use of limited resources they are not restricted to only these. Linear programming problems are generally acknowledged to be efficiently solved by just three methods, namely the graphical method, the simplex method, and the transportation method.

[15]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.

Q represents the number of subjects and r curricula in which each curricula is defined to a group of students.

$$\text{Find } y_{ik} \quad (i = 1..q; k = 1..p)$$

Such that

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

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

$$\sum_{i \in K_l} y_{ik} \leq 1 \quad (l = 1..r; k = 1..p) \quad (3)$$

$$y_{ik} \in \{0,1\} \quad (i = 1..q; k = 1..p) \quad (4)$$

2.2.2 EVOLUTIONARY AND GENETIC ALGORITHMS

[8]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. GA's 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. [10] 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. A GA differs from other search techniques in the following ways:

- GA's optimizes 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.

- GA has 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. Each schema denotes a hyper plane.
- GA's is 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. This means GAs use probabilistic transition rules, not deterministic rules.
- GA's operates 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.
- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.

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. GAs demonstrates 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. 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. Some applications where GA's have been used successfully are analogue circuits, fuzzy logic, and neural networks. These cover a wide range of practical applications such as job shop scheduling, training neural nets, image feature extraction, and image feature recognition.

```

generate an initial population,  $G(\text{at time}=0)$ 
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.

```

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. [5]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.

The process starts by creating a random initial solution. The main procedure consists of a loop that generates, at random and for each iteration.

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. [4]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.

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.

[6]The initial design and development in TS started in the late 1960's and early 1970's.[12] 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 conservation and scores of other problems .

[7]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. 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. [26]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.

described below:

Initialisation

```

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
initialise 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 that
    A(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*

```

2.2.5 CONSTRAINT BASED REASONING

[13] Constraint based reasoning has proven to be a productive research method for researchers in the past especially the areas of artificial intelligence, operational research, and logic programming. 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*. 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. A solution specifies an assignment of a value to each variable that does not violate any of the constraints. 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.

[14]A constraint-scheduling problem can be partially solved using search algorithms that search for best and feasible solutions with forward checking and constraint propagation. [23]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. [19] 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 R between nodes i and j . 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 and labels that cannot satisfy constraints are removed.

A simple timetabling can be described as having a set of v events E and a set of s time slots T where an assignment is defined as the ordered pair a, b such that $a \in E$ and $b \in T$ with the interpretation that that "event a occurs in timeslot b .

2.2.5.1 CONSTRAINTS

[16]In timetabling we are also given constraints and objectives that we must meet. Some of these constraints/objectives render most of the solutions from the search space infeasible. Various constraints will be discussed below but these are not the only ones and some constraints are problem specific.

2.2.5.2 EDGE CONSTRAINTS

Edge constraints are constraints where two events can't be scheduled for the same time slot. For example, it's the same student being scheduled to take a course at the same time. It's impossible for this to occur since the student can't be at two places at the same time. The word "collision(s)" is a violation of one or more edge constraints.

2.2.5.3 ORDERING CONSTRAINTS

Ordering constraints are constraints where there needs to be some type of ordering .For example, if a student is taking a science class that has a corresponding lab class then there might be some ordering requirement that the student's class has to occur before the lab. In a high school setting, a student might have two math courses and one is a support for the other one. So an ordering constraint might be that the main math class has to come prior to the math support class.

2.2.5.4 EVENT-SPREAD CONSTRAINTS

Sometimes there is a requirement for certain events to be spread out in some fashion. A classic example is exam timetabling where you want to schedule exams such that the student does not have more than two exams in a particular day.[24] Another example might be that you want professors in a university to only teach no more than three days so that they can have the other two days free for research purposes. In a high school setting one might decide that science and math courses be spread out and there should be at least one course in between them.

2.2.5.5 CAPACITY CONSTRAINTS

Capacity constraints deal with a room not exceeding its capacity or a teacher being limited to teach only so many students due to contract arrangements. In high school timetabling, specific courses regardless of room might not exceed a given number. For example, math courses might be limited to 30 spaces while art courses or computer courses might be limited to say 15 spaces due to the resources available for that particular course.

2.2.5.6 HARD/SOFT CONSTRAINTS

[17]Hard constraints are strictly enforced and cannot be violated. An example of a hard constraint is: no person can be allocated at the same time in more than one place. Soft constraints are constraints that we wish to have but are not absolutely necessary. An example of a soft constraint is: A teacher teaches all of their courses in succession. That is, they don't teach some courses in the morning and then some in the afternoon with a significant break in between.

2.2.6 REPRESENTATION

[18]Representation is an important part of a genetic algorithm. Representation allows us to represent a real world timetable and map it into a genotype that will be used as the basis to undergo crossover and mutation. There are various methods to represent a timetabling problem and two are discussed below.

2.2.6.1 SIMPLE ORDERING

Simple ordering orders the events in a genotype with the corresponding time slot that it should belong to. For example if we have 10 events that we need to place in time slots and the resulting genotype is $G\ 3721774658$ then we have event 1 in timeslot 3, event 2 in time slot 7, event 3 in time slot 2 and so on. [22]Simple ordering can be coded by using a one dimensional array where each cell contains a student's event and its corresponding location in the timetable as long as the timetable is also ordered sequentially.

2.2.6.2 MATRIX REPRESENTATION

Matrix representation can be used as an encoding scheme. A matrix representation of size $m.n$ was used to solve a school timetabling problem, where m is the number of classes and n is the number of time periods per class in a week. Special operators have to be defined for crossover and mutation otherwise it wouldn't work because they are designed for a one dimensional array representation.

2.2.6.3 FITNESS

In order for the genetic algorithm to know whether our genotypes are improving in fitness value we have to define a fitness function that will give us information about a particular timetable. [20]This fitness function has to take into consideration the various constraints that we have to meet and as better timetables are produced the better fitness values we get. Various fitness functions have been defined for timetabling and they are either a function that we are minimizing or maximizing. General guidelines for the behavior of a fitness function are defined with the assumption that we are trying to maximize our fitness function.

2.2.6.4 MUTATION

Mutating a genotype by using “inside information” is referred to as smart mutation or directed mutation. Smart mutation guides the evolutionary algorithm to more promising solutions by using information about the specific problem to aid the evolutionary cycle.

[19]A random mutation operator is also used to compare how the above mutation operators compare to this one. The specifics of using these operators in our implementation are discussed in the experimental results in chapter

2.3 RELATED WORKS

2.3.1 BIRBAS

Birbas, et al; (2009), presented a novel two-stage approach for the school timetabling problem.[28]The first stage consisted of a shift Assignment problem, where work shifts were assigned to teachers based on their preferences, the school policy regarding core courses and the needs of the curriculum. The second stage consisted of the actual timetabling problem resulting in an optimal timetable. At both stages, integer programming models were used for solving the corresponding problems. The approach was applied to the Hellenic secondary educational system to achieve high quality timetables which obeyed all functional and practical hard constraints and to a very satisfactory level, and a number of quality soft constraints. More specifically, they were conflict free and complete, they handled simultaneous, collaborative and parallel teaching requirements. In their research, the presented work had assumed unlimited availability of rooms and tutors requiring to be optimally used, which is not the case in Tanzania where teachers and rooms are very limited.

2.3.2 M. Mitchell, “An introduction to Genetic Algorithms”

[11]Authors proposed conduct an approach on the establishment of a timetable interactively (Interactive Timetable) for use in schools and universities. The interactive timetable enables the user to impact and adjustment in the table at the time of its composition (Real-time Interaction). Authors believe that the ability to see the process of configuring of the table in the Real-time and the possibility of amendment it would be better suited to the process of

configuring and modifying the table automatically(full automatic timetabling), which limit the amendment process to the table in case of dissatisfaction from the cell of the table. He establishes the interactive algorithm is that creates this schedule with the possibility of the province and edition by the user, whether at the time of creating the table(real-time) or after you configure this table, taking into consideration not to reconfigure the table (re-scheduling)after a change to the task of tasks, whether deleting or adding or modifying the task, but to continue and follow-up later in the make-table (sub-result), may lead to non-completion, but without the conflicts and this is the good thing about it.

Advantages are it can be summarized as high speed access to the best solution, the use of Java language, which is characterized as a Multi-platform, and do not use a database which is an advantage because it speeds up access to the results. On the other hand, there are some shortcomings are as the timetables edited by a user in the Real-time may lead to non-completion and therefore do not reach the optimal solution.

2.3.3 DANCIOUS (EVOLUTIONARY ALGORITHM)

Dancious (2003), in their research managed to show that concrete timetabling problems can be effectively addressed by an evolutionary algorithm which uses a direct matrix representation and well-selected genetic operators.[29] The biased genetic operators had been proposed as an alternative to the classic matrix operators. The operator efficiency metrics measure the effectiveness of the genetic operators from two points of view. The presence quotient showed how much each operator had contributed to creating the best individual, while the improvement rate measured how much each operator had contributed towards improving or repairing the individuals in the population. The experiments performed had shown that these two metrics are consistent and can be reliably used to compare the relative efficiency of the genetic operators defined for an evolution program. In their published work, they mentioned the case where simultaneous teaching of subjects had to take place. They pointed out that in case simultaneous teaching of the subjects was considered, the number of subjects held simultaneously must not exceed the number of available rooms. In the case of Tanzania, most schools are facing a problem of shortage of rooms, which means if the schools have many

subjects to be taught simultaneously, it is difficult to get rooms to accommodate all the subjects. This means that the proposed approach can only work for high schools with enough rooms.

2.3.4 Ahmed Al-Tehami, Ahmed Al-Salehi, AmmarZahary, "APPLICATION OF GENETIC ALGORITHMS IN THE PROBLEM SOLVING OF COURSE TIMETABLES,

[18]Authors dealt with a solution to the problem of course timetables of Department of Computer Science at the University of Science and Technology (UST), using genetic algorithms. Authors adopted a set of criteria or constraints to find the closest schedules to the ideal solution. However, the solution is limited to only one section of four levels, as any lecture will be only of two hours and the exclusion of lectures laboratories. Authors represented the algorithm using two-dimensional matrix where each row in the matrix represents the table of one room and all the rows in the matrix represent the entire chromosome. The value of the *fitness* function was calculated by the adoption of the following equation:

$$\text{Optimal value of the appropriate} = \text{total number of lectures} + \sum(\text{length of the chromosome}-1) + ((\text{number of rooms}-1) * \text{Number of possible lectures per day} * \text{Number of school days}).$$

This matrix has been utilized to make the Mutation and Selection in order to improve the chromosomes, increasing the appropriate values of the function from generation to generation until reaching the desired solution.

Advantages are can be summarized as the use of genetic algorithm, and issuing tables and solves the problem of conflicts. Shortcomings are limiting the application of the software on only one section of the four levels only, and the lack of flexibility in times and hours of lectures has been identified only with two hours.

2.3.5 Related Work Summary

[22] Automated timetabling includes class and examination timetabling and has been one of the main optimization problems in GA applications. The concept uses hybrid approaches by which an evolutionary algorithm performs a search in the domain for candidate solutions to mutate. The initial timetable is supported by specific mutation operators to generate a final timetable. GA is capable of producing examination timetables that fulfill both the hard and soft constraints. [23] However, when there is no penalty in an algorithm with zero solution, it indicates that the solution does not exist. Representation language was introduced in [24] using UniLang. The input language algorithm provides a clear and simple natural data representation. The algorithm also considers the constraints and provides quality measures and solutions to different and related examination timetabling problems. The GA approach to solve timetabling problems is also investigated in [27] and a framework is proposed that gives greater flexibility on the declaration of specified constraints and evaluation function of the solutions. The study also provides tips on the use of GA for resolution.

A generic language that incorporates timetabling problems and its constraints is proposed in [25]. In the study, the researcher concluded that the generic language can be attempted to solve class-lecturer timetabling problems for small schools [23]. A more efficient searching algorithm is developed by [26] for one-dimensional Cellular Automata (CA) in which the GA gene is utilized to represent the rule table and the fitness function is used in building the CA structure. The study found that optimal mutation rate for fitness evolution was present and some typical patterns can be drawn by the genes if the genes were to evolve successfully. There are many more researchers that use different algorithmic approaches and provide ways to optimize solutions applied for the whole process or for part of the process.

2.4 SUMMARY

The above completes the brief overview of the major fundamental techniques in the literature. As indicated there is a trend towards the use of constraint based reasoning. This becomes obvious as one survey the literature presented in the next section. As will be shown in the final section this is the most utilized technique in hybrid approaches.

The first most striking fact that reveals itself when attempting to solve a time-tabling problem that is anything over trivial is the helplessness a naive genetic algorithm may possess. Without some fine-tuning and parameter tweaking the algorithm usually lets its convergent nature rule out its explorative one.

Even when fine-tuning is performed for a certain problem, it quickly becomes obvious that it is relevant only for a certain type of similar problems. The inescapable conclusion is that either we develop domain-specific genetic operators that will produce a sound genetic search for many types of problem configurations, or we remain domain-independent and design an adaptive genetic mechanism that tweaks its own parameters according to the state of the population (e.g. increasing mutation when variance in the population is low). A third option, and probably the best one, would be a compromise between the two.

CHAPTER THREE - METHODOLOGY

3.1 INTRODUCTION

This chapter proposes and develops an information system to support the production and management of timetables in a university setting. The work presented here attempts to integrate the work of implementation of a solution generation algorithm with the very real problem of administration and management of the resulting timetable solution. As noted in the previous chapter, there have been many reported attempts at developing timetable information systems. All of these systems, however, have begun with an analysis of the information needs of the university administration of timetables.

Following this, the development of the Timetable Information System is discussed. It is noted that the IS presented here begins with an analysis of the data entities required to support the generation and maintenance of a university timetable. The SDLC (Systems Development Life Cycle) was used in the construction of this system.

An initial study was conducted, at the department of computer studies at the University of Zambia, with administrators involved in the timetabling function to gauge the need and use of such a system. The results indicated a system that would provide efficient timetabling would be very useful as there have been great inconsistencies in the university exam timetables. As a result from the information gained from the Literature review and the results of the initial staff study a proposal for a prototype of the system was agreed upon. The next step in the SDLC required a design of the system. An analysis of the requirement was carried out from a general perspective. In doing this analysis entities required and their relationships to each other were identified and pictorial representation was made using Entity Relationship Diagram (ERD).

3.2 SYSTEMS REQUIREMENTS

The system developed mainly requires input of course details and room details to be specific below is the course details required.

- Course name/code.

- Invigilator attending the examination venue to oversee the students for the above stated course code.
- The number of students seating for the exam for the stated course name/code.

Information pertaining to the courses is to be repeatedly inputted in the system and after that the room details are inputted. Below are the details required for every room.

- Examination venue name.
- Examination venue seating capacity.

Like before room information is to be repetitively input into the system. The developed system is in the prototype phase therefore for testing purposes details are to be input manually but in future additional modules for reading course and room details from a file can be added as this can be easier for the user. Below is the phase the system goes through as it is running.

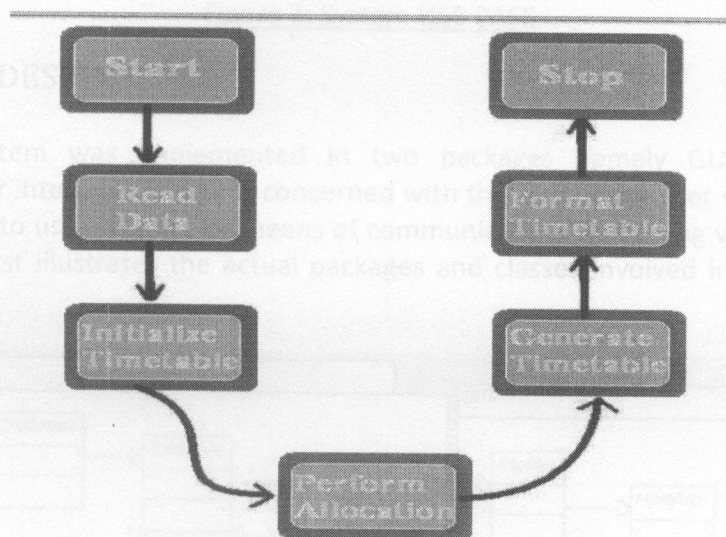


Figure 1: Generation Phases

The system starts, and then reads the data it needs to perform its function. It then initializes data structures used in allocation phase and also randomizes course data before the next stage. It then performs the allocation of the time table entities using the forward search algorithm. Generation of a timetable then occurs and formatting of the time table for

3.1 GUI PACKAGE

permanent storage can be performed to html. Depend on the amount of data input into the system it can produce more than one time table depending also on the number of rooms provided for allocation as examinations usually have two sessions. Therefore the fewer the number of venues provided with more courses the more the generated time tables. This is done to ensure the correct number of students will fit in each venue provided.

3.1.2 RoomGUI

ed to provide an interface for the user to capture room details in order to later on be putted into the

3.1.3

called first a

the system

ved which pr

3.3.1.4 MultiCellRender

3.3 SYSTEM DESIGN

Basically the system was implemented in two packages namely GUI and DATA. The GUI(Graphical user interface) is mainly concerned with the output the user sees and deals with providing an easy to use interface or means of communicating/interacting with the system the diagram below best illustrates the actual packages and classes involved in the design of the project.

ultimately called when data is to be inputted into the system by use of a database. The

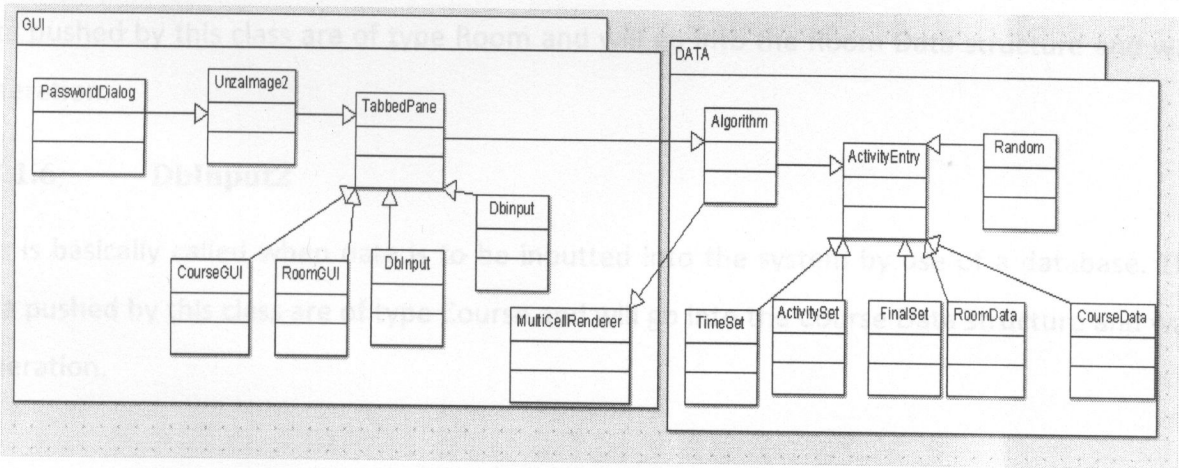


Figure 3: System Design

3.3.1 GUI PACKAGE

3.3.1.1 CourseGUI

Used to provide an interface for the user to capture course details in order to be inputted into the ActivityEntry class, which pushes this information to the CourseData class.

3.3.1.2 RoomGUI

Used to provide an interface for the user to capture room details in order to later on be inputted into the ActivityEntry class which pushes this information to the RoomData class.

3.3.1.3 PasswordDialog

Is called first and provides an interface that provides a password dialog used to provide security to the system as it is only for admin use. After input of correct password the main interface is called which provides the CourseGui on the screen.

3.3.1.4 MultiCellRenderer

Provides the final input that has the time table in Jtable form and the interface provides a button that provides permanent storage in html.

3.3.1.5 DbInput

This is basically called when data is to be inputted into the system by use of a database. The data pushed by this class are of type Room and will go into the Room Data structure and wait generation.

3.3.1.6 DbInput2

This is basically called when data is to be inputted into the system by use of a database. The data pushed by this class are of type Course and will go into the Course Data structure and wait generation.

3.3.1.7 TabbedPane

This is the main GUI class that holds everything together and helps in providing an easy means of accessing methods and functions

3.3.1.8 UnzaImage2

This is simply a cover page that is shown between the main GUI and the Password input dialog.

3.3.2 DATA PACKAGE

3.3.2.1 Algorithm

This is where the main algorithm is and does the computation with data provided by ActivityEntry class as it is connected to all the Data classes and is used as a means of pulling data from any part of the project.

3.3.2.2 ActivityEntry

Is the connector for all the data classes and is very important in that every function across classes is called by use of a method provided by ActivityEntry.

3.3.2.3 Random

Is used by the Algorithm class and is used to perform one of the initial tasks of randomizing data inputted by the user and then pushing it into a data structure provided by the ActivitySet class.

3.3.2.4 CourseData

Simply holds Course data and is connected to the ActivityEntry class to provide information to the other system parts.

3.3.2.5 RoomData

Simply holds Room data and is connected to the ActivityEntry class to provide information to the other system parts.

3.3.2.6 ActivitySet

Holds data after undergoing randomization to ensure proper allocation of time table values.

3.3.2.7 TimeSet

Provides time vector data structures and also connected to ActivityEntry for distribution of data to other project areas.

3.4 SYSTEM IMPLEMENTATION

The implementation of the system was done using the forward search algorithm with addition of a randomization aspect to ensure randomization of data being acted upon. Was the system is first fired up the image below is displayed and is used to input a password as a means of providing security to the system.

Input of course details follows after the input of the password. The course GUI detects duplicate entries and prevents this. It also flags errors if wrong values are entered such as null.

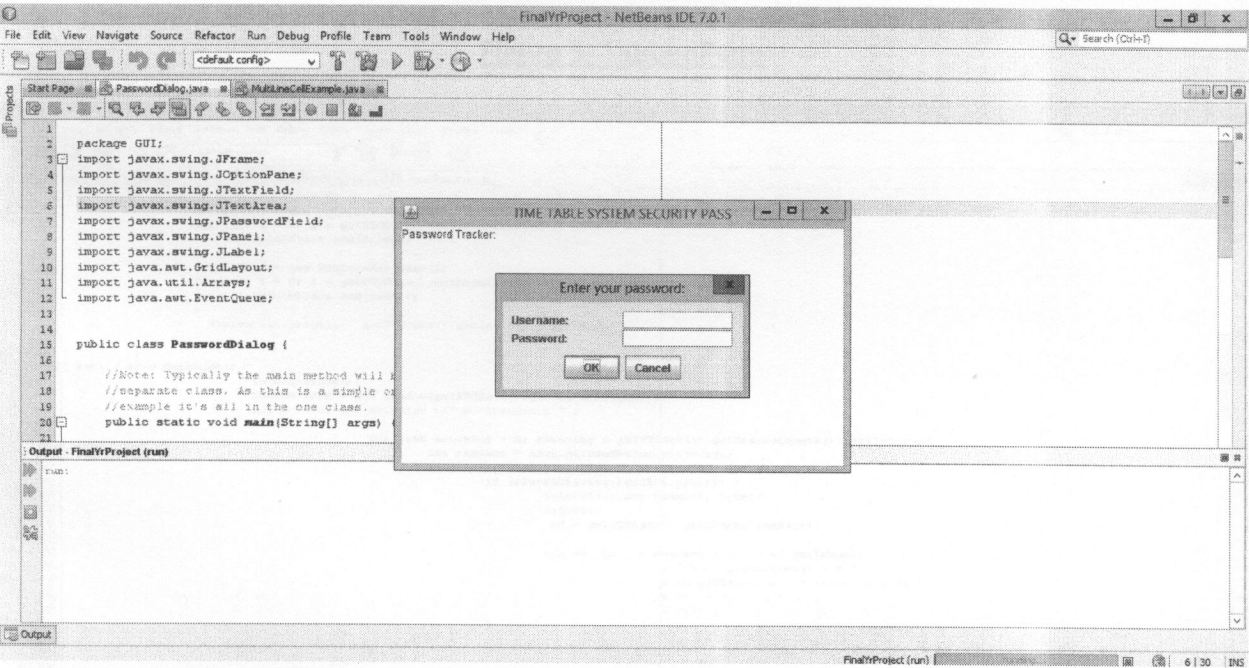


Figure 4: PasswordInputScreen

After correct input of password and input of the course and room details the main algorithm is run that calls the random class to perform randomization and then pushes the data into a data structure for allocation to occur. Below is a screenshot of the main algorithm class.

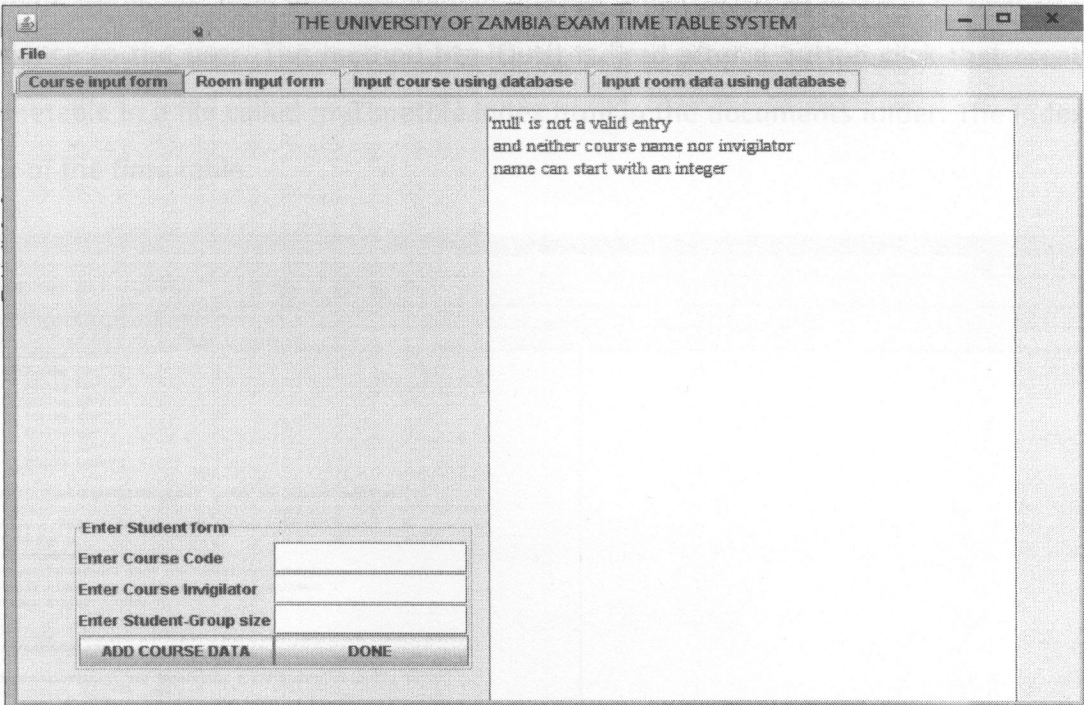


Figure 5: TabbedPane

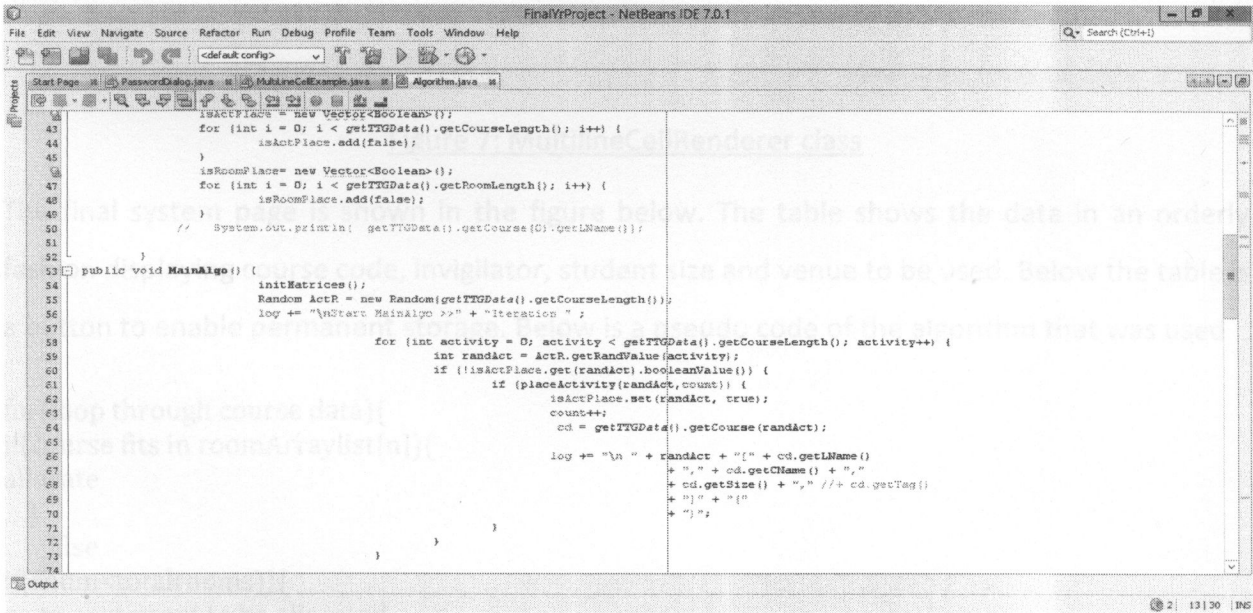


Figure 6: Algorithm class

Allocation is performed in the algorithm class by adding a data value to an array as a sign that the room is full. This is later interpreted by the MultiLineCellRenderer class which in turn is used to create the time table and also provides the method htmlOut() as a means of creating permanent storage to html.html permanent storage was chosen as it is the easiest to use for convenience to the user. The method htmlOut() is fired after a button click that creates the html timetable in a file called myTimetble.index.html in the documents folder. The index is the number of the time table.

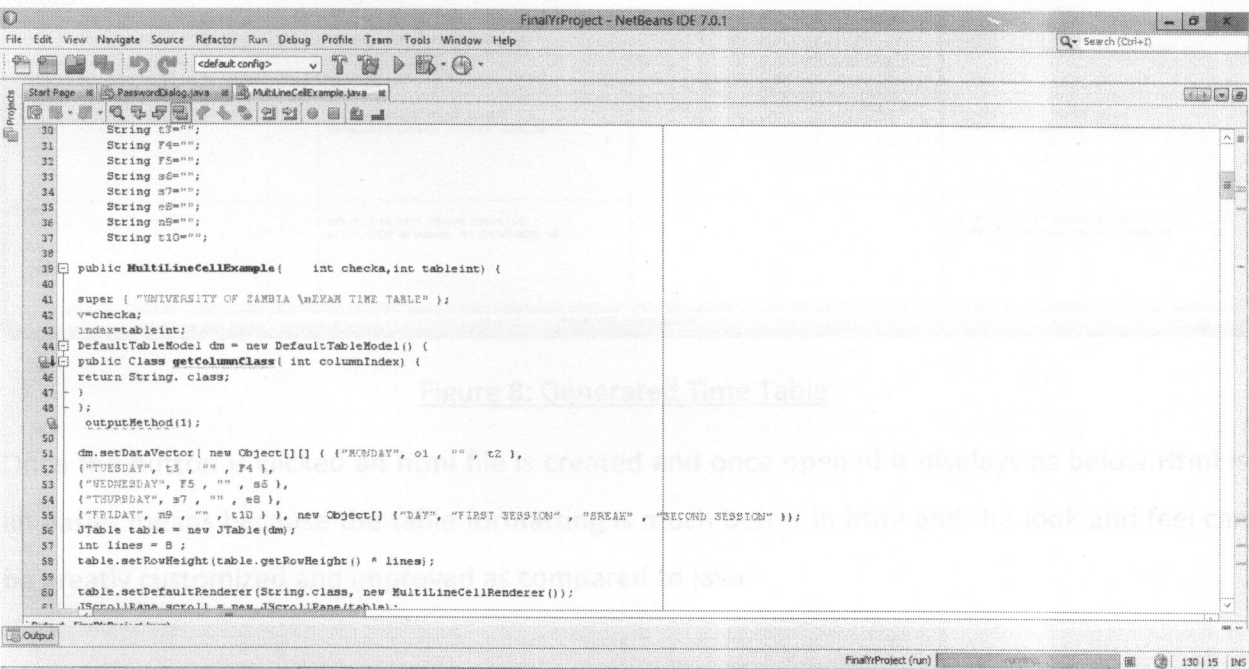


Figure 7: MultiLineCellRenderer class

The final system page is shown in the figure below. The table shows the data in an orderly fashion displaying course code, invigilator, student size and venue to be used. Below the table is a button to enable permanent storage. Below is a pseudo code of the algorithm that was used

```

for(loop through course data){
if(course fits in roomArrayList[n]){
allocate
}
Else
if(rnum<totalrooms)){
go to next room to be allocated
if(course fits in room[n+1]){
allocate

```

```

}
}
else if(room is last room){
initialize to first room
if(course fits in room[n]){allocate } }

```

UNIVERSITY OF ZAMBIA EXAM TIME TABLE			
DAY	FIRST SESSION	BREAK	SECOND SESSION
MONDAY	COURSE: cst1000 VENUE: new DH INVIGILATOR: Mr chishala STUDENT SIZE: 33 COURSE: cst0333 VENUE: siwela INVIGILATOR: Mrs khuma STUDENT SIZE: 22		COURSE: cst1333 VENUE: siwela INVIGILATOR: Mr thea STUDENT SIZE: 23 COURSE: cst2000 VENUE: Sports Hall INVIGILATOR: Mr Jere STUDENT SIZE: 30
TUESDAY	COURSE: cst3233 VENUE: Sports Hall INVIGILATOR: Mr chibuta STUDENT SIZE: 20 COURSE: cst2909 VENUE: new DH INVIGILATOR: Mr chiliando STUDENT SIZE: 17 COURSE: cst10000 VENUE: new DH INVIGILATOR: Mr nimrod STUDENT SIZE: 22		COURSE: cst2222 VENUE: new DH INVIGILATOR: Mr kabulubulu STUDENT SIZE: 26 COURSE: cst15556 VENUE: new DH INVIGILATOR: Mr swaga STUDENT SIZE: 21 COURSE: cst13443 VENUE: siwela INVIGILATOR: DMZ STUDENT SIZE: 22
WEDNESDAY	COURSE: cst14000 VENUE: siwela INVIGILATOR: Mr nyirenda STUDENT SIZE: 30 COURSE: cst13111 VENUE: Sports Hall INVIGILATOR: Mr Mulalo STUDENT SIZE: 19		COURSE: cst13000 VENUE: Sports Hall INVIGILATOR: Mr Kasanda STUDENT SIZE: 20 COURSE: cst12020 VENUE: Sports Hall INVIGILATOR: Mr kamanga STUDENT SIZE: 20
THURSDAY	COURSE: cst0030 VENUE: new DH INVIGILATOR: DR phis STUDENT SIZE: 20		COURSE: cst12022 VENUE: siwela INVIGILATOR: Mr chisanga STUDENT SIZE: 34
FRIDAY	COURSE: cst3211 VENUE: Sports Hall INVIGILATOR: Mr chibuye STUDENT SIZE: 18		COURSE: cst11111 VENUE: new DH INVIGILATOR: Mr victor STUDENT SIZE: 34
SAVE TIME TABLE IN HTML			

Figure 8: Generated Time Table

Once the button is clicked an html file is created and once opened it displays as below. Html is an easier means because the table formatting is much better in html and the look and feel can be greatly customized and improved as compared to java.

DAY	FIRST SESSION	BREAK	SECOND SESSION
MONDAY	COURSE: cst 1000 VENUE: new DH INVIGILATOR: Mr chishala STUDENT SIZE: 33 COURSE: cst3333 VENUE: siwela INVIGILATOR: Mrs khuma STUDENT SIZE: 22		COURSE: cst3332 VENUE: siwela INVIGILATOR: Mr theo STUDENT SIZE: 23 COURSE: cst 2000 VENUE: Sports Hall INVIGILATOR: Mr Jere STUDENT SIZE: 30
TUESDAY	COURSE: cst 3233 VENUE: Sports Hall INVIGILATOR: Mr chibuta STUDENT SIZE: 20 COURSE: cst 2909 VENUE: new DH INVIGILATOR: Mr chilando STUDENT SIZE: 17 COURSE: cst 0000 VENUE: new DH INVIGILATOR: Mr simrod STUDENT SIZE: 22		COURSE: cst 2222 VENUE: new DH INVIGILATOR: Mr kabuhubulu STUDENT SIZE: 28 COURSE: cst 5656 VENUE: new DH INVIGILATOR: Mr swaps STUDENT SIZE: 21 COURSE: cst 3443 VENUE: siwela INVIGILATOR: DMZ STUDENT SIZE: 22
WEDNESDAY	COURSE: cst 4000 VENUE: siwela INVIGILATOR: Mr nyirenda STUDENT SIZE: 30 COURSE: cst 3111 VENUE: Sports Hall INVIGILATOR: Mr Mufalo STUDENT SIZE: 19		COURSE: cst 3000 VENUE: Sports Hall INVIGILATOR: Mr Kasanda STUDENT SIZE: 20 COURSE: cst 2020 VENUE: Sports Hall INVIGILATOR: Mr kamanga STUDENT SIZE: 20
THURSDAY	COURSE: cst3030 VENUE: new DH INVIGILATOR: DR phiri STUDENT SIZE: 20		COURSE: cst 2022 VENUE: siwela INVIGILATOR: Mr chisanga STUDENT SIZE: 34
FRIDAY	COURSE: cst 3211 VENUE: Sports Hall INVIGILATOR: Mr chibuye STUDENT SIZE: 18		COURSE: cst 1111 VENUE: new DH INVIGILATOR: Mr victor STUDENT SIZE: 34

Figure 9: html timetable view

3.5 SUMMARY

The stated objective to develop a university examination timetable information system that supported all aspects of other teaching related administrative functions has been achieved. However given the enormity of the task already undertaken there is still a significant amount of hack programming required to be completed before a production run of the system for widespread use could be achieved. Nonetheless the current version could be customized for in-house use in any one university in a relatively straight forward manner. The issue of conversion from current systems to this system, though, would require the careful planning and transition that goes with such information system change overs. On a positive final note there would appear to be almost no training issues to be addressed.

CHAPTER FOUR – SYSTEM TESTING

4.1 INTRODUCTION

This section of the thesis is mainly focused on the testing of the implementation and the results produced. This is mainly depicted by the screen shots of the program that were taken and also a view of the test data that was used to test the systems functionality.

4.2 SYSTEM TESTING

4.2.1 APPLICATION TESTING

As GUI is main entry point to any software application, the Timetabling System was created to make it easy to interact with Naive User as well as Sophisticated User and was designed in Java (Swingset).The screen shot below shows the entry point to the application which provides a password dialog.

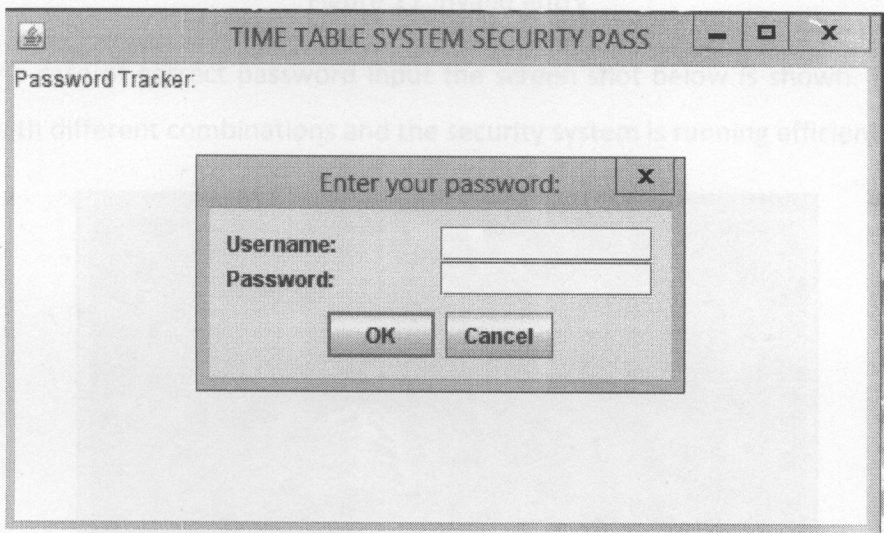


Figure 10: security interface

This provides a means of having the system only used by administrators and people that have the authoritative right to create the time table for the school. In the event that the wrong password is inputted the system does not grant entry. Below shows a screen shot of the system after input of a wrong password.

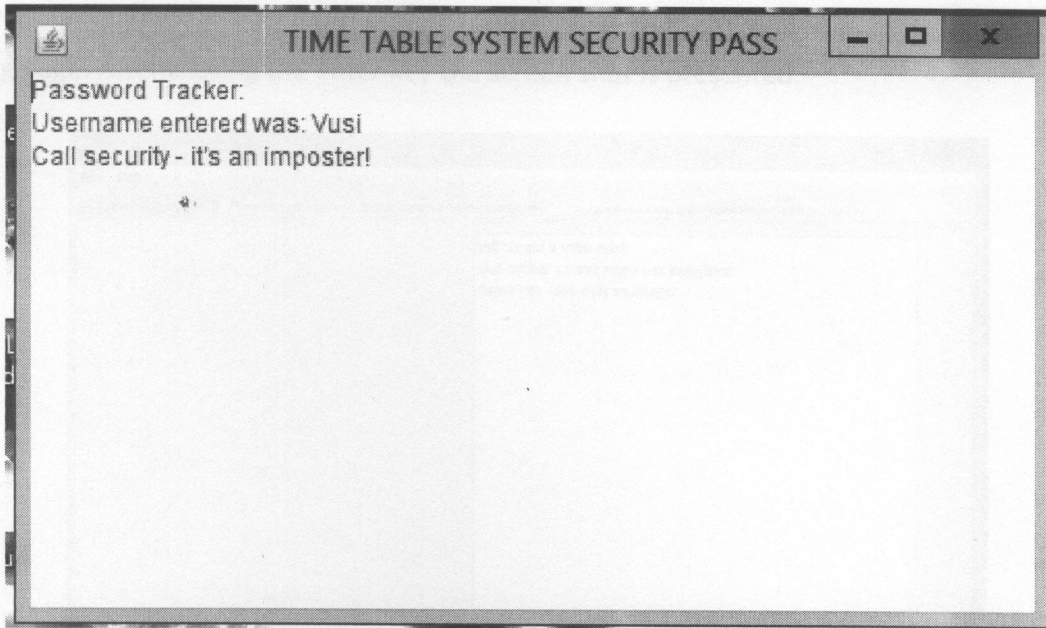


Figure 11:Invalid entry

Though in the case of correct password input the screen shot below is shown. This was tried and tested with different combinations and the security system is running efficiently.

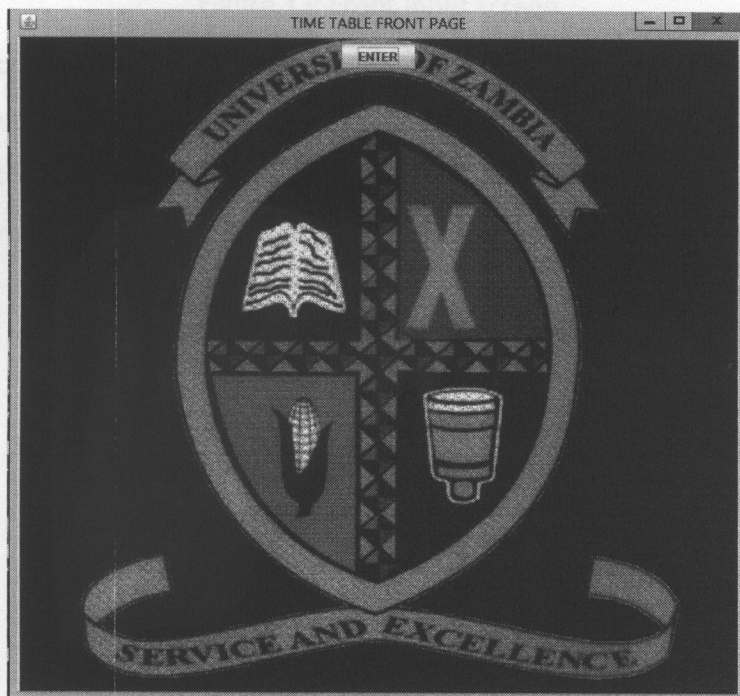


Figure 12: Front screen

The page provides the university logo and an enter key in order to get into the actual time tabling page. After clicking the enter key the screen shot is presented.

THE UNIVERSITY OF ZAMBIA EXAM TIME TABLE SYSTEM

File Edit

Course input form Room input form Input course using database Input room data using database and generate

'null' is not a valid entry
and neither course name nor invigilator
name can start with an integer

Enter Student form

Enter Course Code

Enter Course Invigilator

Enter Student-Group size

ADD COURSE DATA DONE

Figure 13: Front Input screen

The front input screen provides course data input facility and an editing option for data that might have been entered wrongly. This is all easily accessible by use of a tab on the top named edit.in the event that the user clicks on the room input form, the screen shot below shows the way the system provides room entry facility.

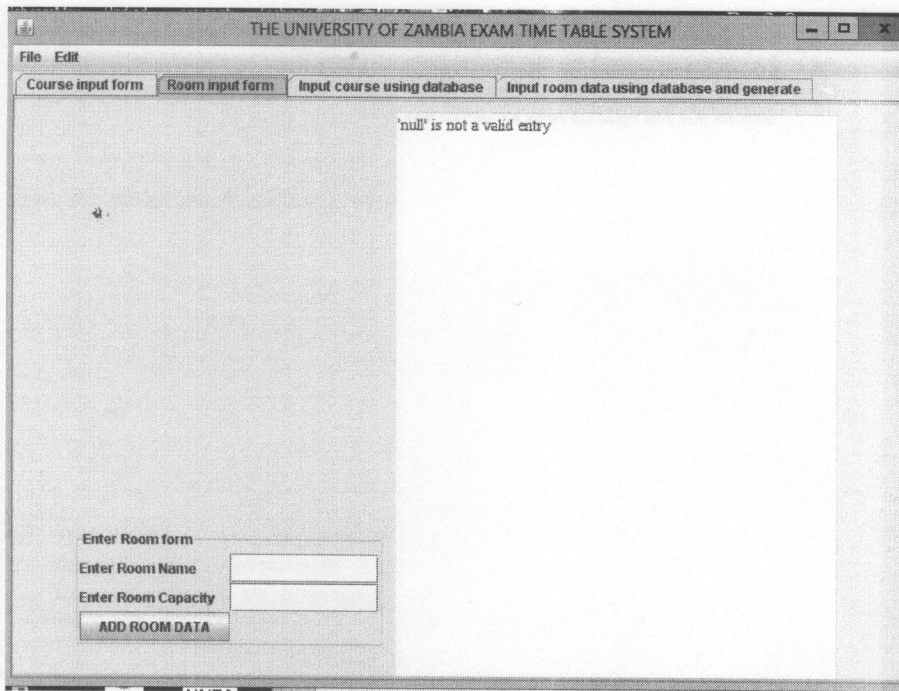


Figure 14:Room input

Data input is done in two main methods. By manual input of data one by one in the input form of the system or by having the system read the data from a table. The screen shot below depicts input by use of a data. An access database is used for input of several values so as to reduce on amount of time it would take to input data from manual input. The screen shot below shows the system when the user clicks on 'course input using database'. It provides a button that reads data from a course table in an access database.

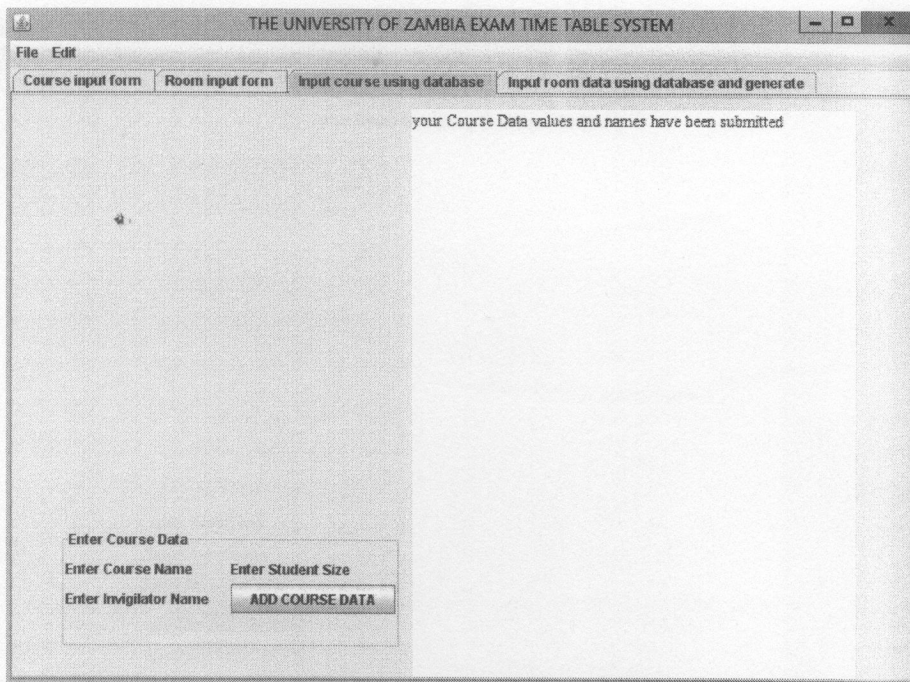


Figure 15: Database input1

This is similar for input of room data. by clicking on the 'Input room data by database' tab the user is presented with the system display depicted below. Just as the course input form it presents an interface that offers a button that if clicked, it reads the access database and recursively reads data from the room table. This data is inputted into the system and the algorithm is run.

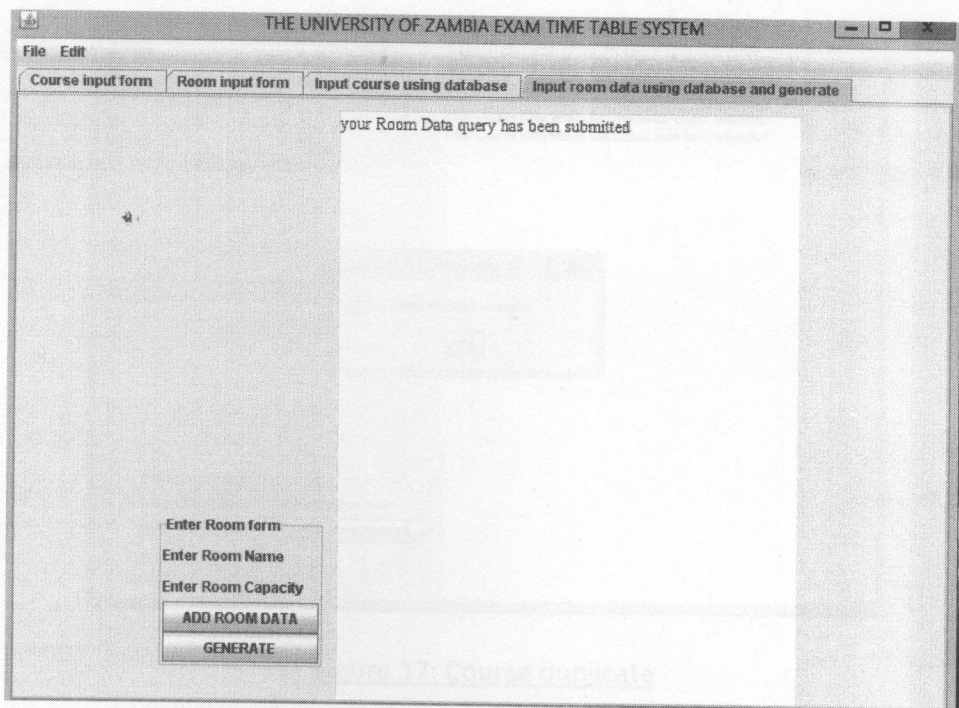


Figure 16:DatabaseInput2

The system was also designed to detect duplicate entries so as to prevent inconsistencies when the time table is generated. This leads to a lot of errors if a course is scheduled more than once. This was tested by inputting the same data from the database. A dialog is flashed showing that a duplicate was entered and the inputted value is not saved in the algorithm. Below shows a screen shot showing the course duplicate detection working by flashing that the input a duplicate and also showing the course that was entered twice.

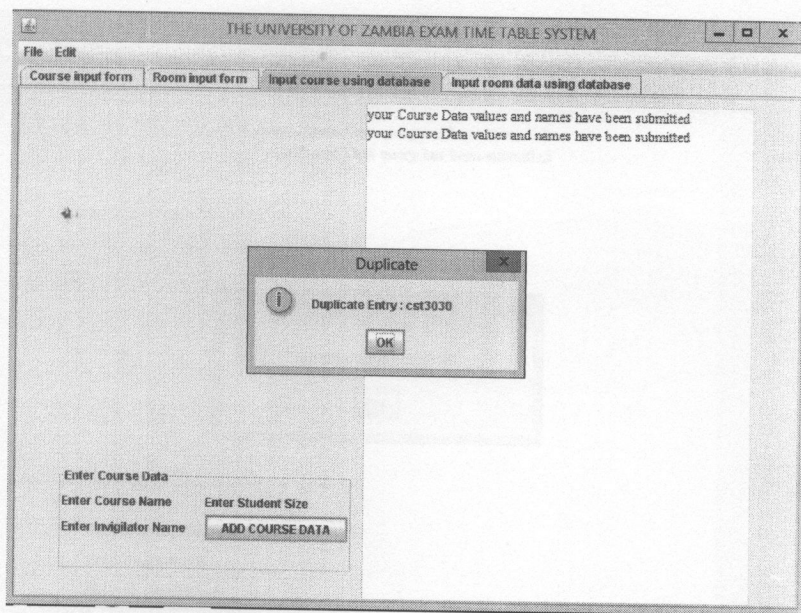


Figure 17: Course duplicate

Similarly for room entry, duplicates may cause chaotic errors which may result in greater errors because courses can be scheduled in a duplicate entry which may result in a group of students not having anywhere to write an exam. Duplicates are also detected and this was tested. A dialog shows the duplicated room name and the inputted room is not entered into the system.

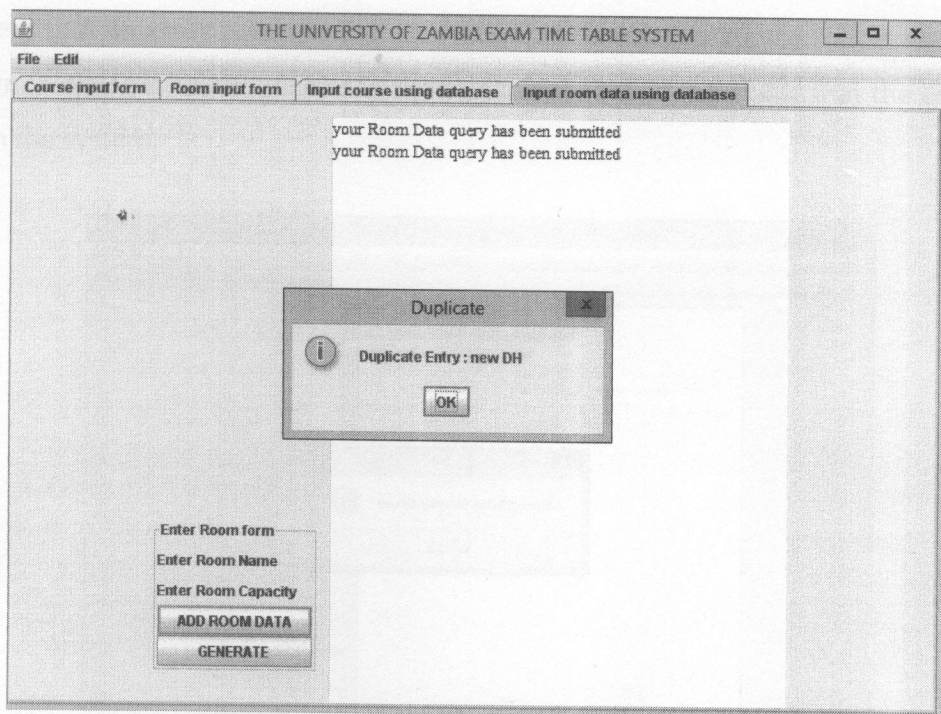


Figure 18: Room duplicate

On the edit tab the view below is presented that offers editing of room details and course details.

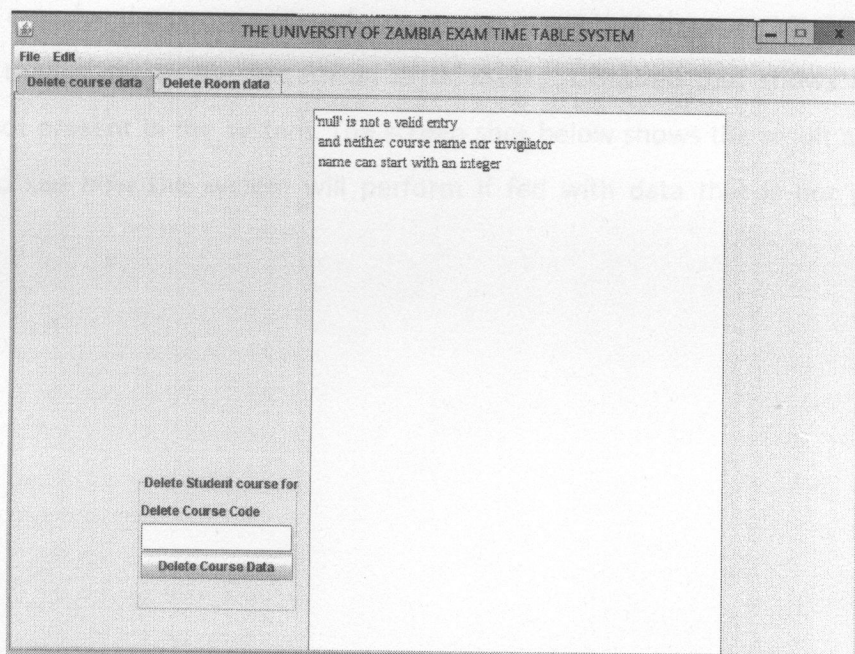


Figure 19: Edit view

In the event that an error occurs and a user attempts to delete a course that is not present in the system .A dialog is shown such as in the screen shot below specifying that the course is not present in the system.

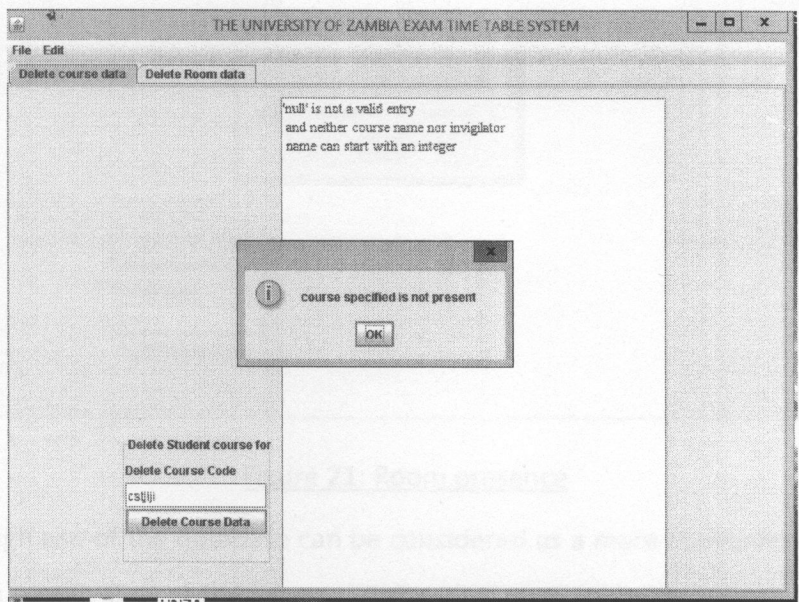


Figure 20: Course presence

It's a similar case for the room editing form. In the event that the user attempts to delete a room name that is not in the system an error dialog is flashed that shows that the room specified is not present in the system. The screen shot below shows the result after a test was performed to see how the system will perform if fed with data that is not present in the algorithm.

Figure 22: Course details table

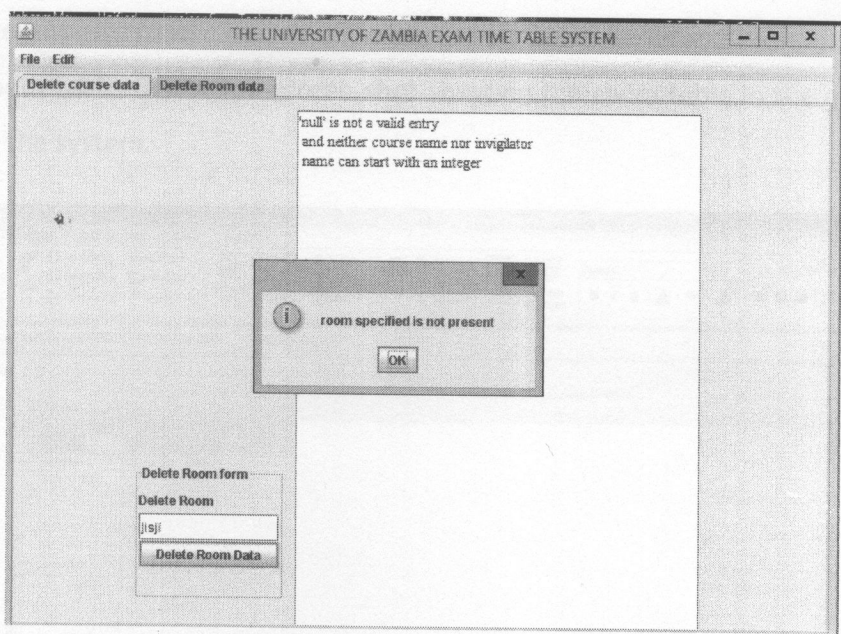


Figure 21: Room presence

The input through use of the database can be considered as a more convenient way of input in bulk. Below is a screen shot of the access database's course table showing the various courses that have been typed in and are ready to be read and processed by the system. The data shown in the table below is also the test data that was used in generating sample time tables to ensure usability of the system

The screenshot shows the Microsoft Access interface with the 'Course' table selected. The table contains the following data:

CourseID	Invigilator	Students	Click to Add
cst1000	Mr nimrod	22	
cst1100	Mr chishala	33	
cst1111	Mr victor	34	
cst2000	Mr jero	30	
cst2020	Mr kamanga	20	
cst2022	Mr chisanga	34	
cst2222	Mr kabulubulu	28	
cst2909	Mr chikanda	17	
cst3000	Mr Kasanda	20	
cst3111	Mr Mufalo	35	
cst3211	Mr chibuya	38	
cst3233	Mr chibuta	20	
cst3443	DMZ	22	
cst4000	Mr nyirenda	30	
cst5056	Mr savaps	21	
cst5080	DR phiri	20	
cst3332	Mr theo	23	
cst3333	Mrs khuma	22	

Figure 22: Course database table

Similarly the rooms also have a means of bulk entry of room names and room attributes. This is by use of a room table. Below is a screen shot showing the room table in the access database being used by the system.

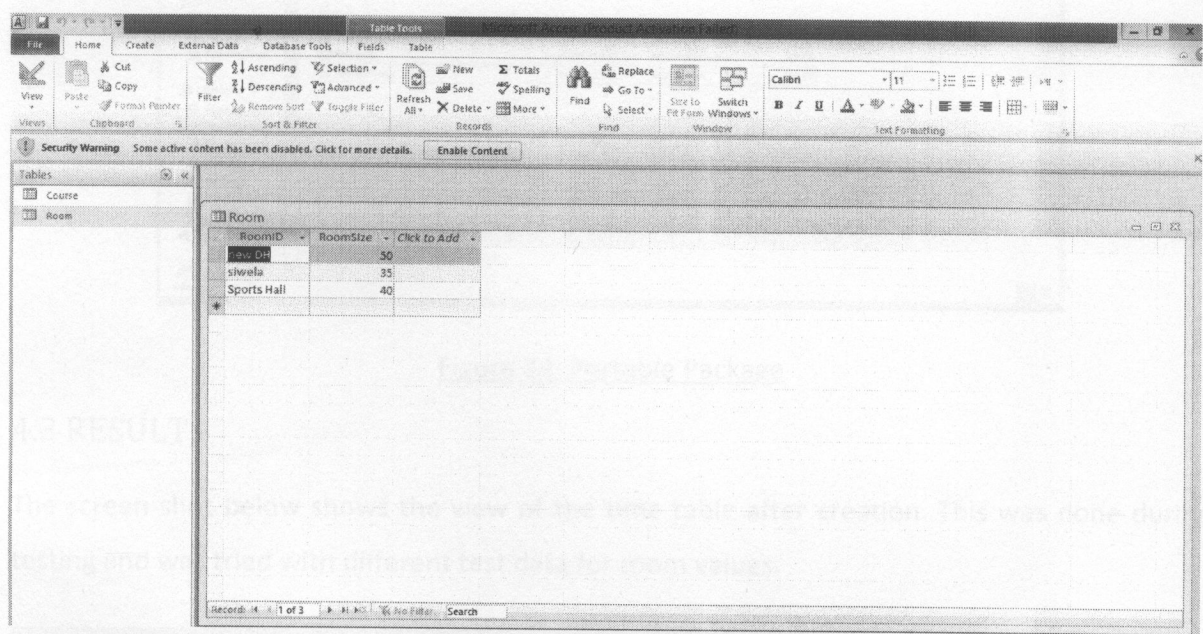


Figure 23: Room database table

After database input the user clicks the generate button in order to generate the time table. In the event that there are not enough rooms for scheduling that particular week the system shows which rooms couldn't be scheduled.

4.2.2 PORTABILITY TESTING

The stand along application was packaged and tested on a different machine. The software package includes the application program and a database file that is to be used. Below is a screen shot of the package containing the application and database.

Figure 25: Time table

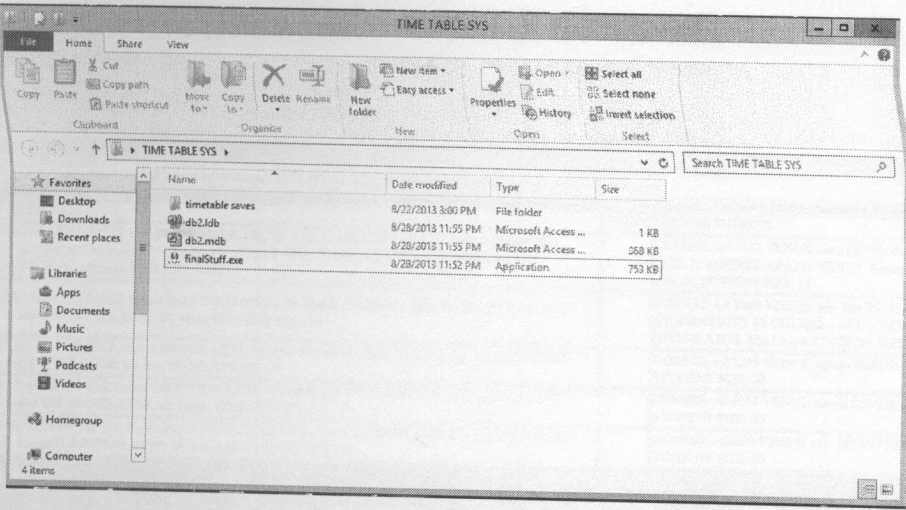


Figure 24: Portable Package

3 RESULTS

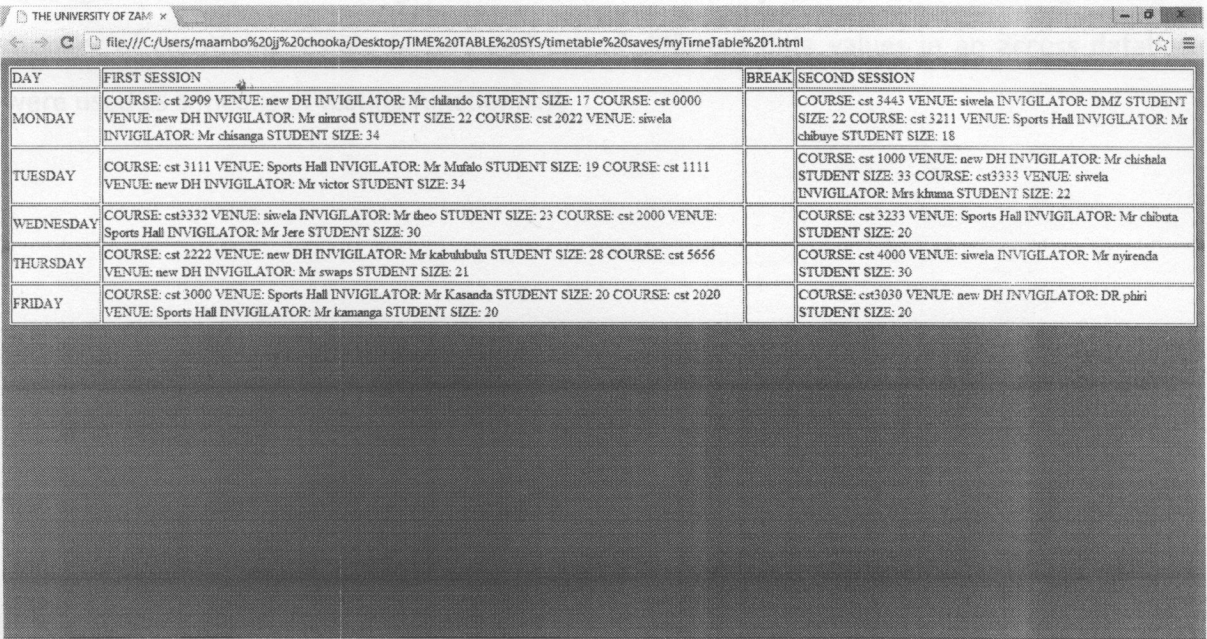
The screen shot below shows the view of the time table after creation. This was done during testing and was tried with different test data for room values.

UNIVERSITY OF ZAMBIA EXAM TIME TABLE			
DAY	FIRST SESSION	BREAK	SECOND SESSION
MONDAY	COURSE: cst 2009 VENUE: new DH INVIGILATOR: Mr chitando STUDENT SIZE: 17 COURSE: cst 0050 VENUE: new DH INVIGILATOR: Mr mlimrod STUDENT SIZE: 22 COURSE: cst 2072 VENUE: siwela INVIGILATOR: Mr chisanga STUDENT SIZE: 34		COURSE: cst 344? VENUE: siwela INVIGILATOR: DMZ STUDENT SIZE: 22 COURSE: cst 3211 VENUE: Sports Hall INVIGILATOR: Mr chibuye STUDENT SIZE: 18
TUESDAY	COURSE: cst 2111 VENUE: Sports Hall INVIGILATOR: Mr Mufalo STUDENT SIZE: 19 COURSE: cst 1111 VENUE: new DH INVIGILATOR: Mr visitor STUDENT SIZE: 34		COURSE: cst 1000 VENUE: new DH INVIGILATOR: Mr chishala STUDENT SIZE: 33 COURSE: cst 933? VENUE: siwela INVIGILATOR: Mrs Nhuma STUDENT SIZE: 22
WEDNESDAY	COURSE: cst 2332 VENUE: siwela INVIGILATOR: Mr thea STUDENT SIZE: 23 COURSE: cst 2000 VENUE: Sports Hall INVIGILATOR: Mr Jere STUDENT SIZE: 30		COURSE: cst 2233 VENUE: Sports Hall INVIGILATOR: Mr chibuta STUDENT SIZE: 20
THURSDAY	COURSE: cst 2222 VENUE: new DH INVIGILATOR: Mr kabulubulu STUDENT SIZE: 20 COURSE: cst 5555 VENUE: new DH INVIGILATOR: Mr swaps STUDENT SIZE: 21		COURSE: cst 4000 VENUE: siwela INVIGILATOR: Mr mlimrod STUDENT SIZE: 30
FRIDAY	COURSE: cst 3000 VENUE: Sports Hall INVIGILATOR: Mr Kasanda STUDENT SIZE: 20 COURSE: cst 2020 VENUE: Sports Hall INVIGILATOR: Mr kamanga STUDENT SIZE: 20		COURSE: cst 3030 VENUE: new DH INVIGILATOR: DR phiri STUDENT SIZE: 20
SAVE TIME TABLE IN HTML			

Figure 25: Time table

In order for better viewing the user can click the 'SAVE TIME TABLE IN HTML' button and the table is exported to html. This is for storage and better viewing of the time table.

Below is a view of the Time table in html.



DAY	FIRST SESSION	BREAK	SECOND SESSION
MONDAY	COURSE: cst 2909 VENUE: new DH INVIGILATOR: Mr chlando STUDENT SIZE: 17 COURSE: cst 0000 VENUE: new DH INVIGILATOR: Mr nimrod STUDENT SIZE: 22 COURSE: cst 2022 VENUE: siwela INVIGILATOR: Mr chisanga STUDENT SIZE: 34		COURSE: cst 3443 VENUE: siwela INVIGILATOR: DMZ STUDENT SIZE: 22 COURSE: cst 3211 VENUE: Sports Hall INVIGILATOR: Mr chibuye STUDENT SIZE: 18
TUESDAY	COURSE: cst 3111 VENUE: Sports Hall INVIGILATOR: Mr Mufalo STUDENT SIZE: 19 COURSE: cst 1111 VENUE: new DH INVIGILATOR: Mr victor STUDENT SIZE: 34		COURSE: cst 1000 VENUE: new DH INVIGILATOR: Mr chishala STUDENT SIZE: 33 COURSE: cst3333 VENUE: siwela INVIGILATOR: Mrs khuma STUDENT SIZE: 22
WEDNESDAY	COURSE: cst3332 VENUE: siwela INVIGILATOR: Mr theo STUDENT SIZE: 23 COURSE: cst 2000 VENUE: Sports Hall INVIGILATOR: Mr Jere STUDENT SIZE: 30		COURSE: cst 3233 VENUE: Sports Hall INVIGILATOR: Mr chibuta STUDENT SIZE: 20
THURSDAY	COURSE: cst 2222 VENUE: new DH INVIGILATOR: Mr kalubukulu STUDENT SIZE: 28 COURSE: cst 5656 VENUE: new DH INVIGILATOR: Mr swaps STUDENT SIZE: 21		COURSE: cst 4000 VENUE: siwela INVIGILATOR: Mr nyirenda STUDENT SIZE: 30
FRIDAY	COURSE: cst 3000 VENUE: Sports Hall INVIGILATOR: Mr Kasanda STUDENT SIZE: 20 COURSE: cst 2020 VENUE: Sports Hall INVIGILATOR: Mr kamanga STUDENT SIZE: 20		COURSE: cst3030 VENUE: new DH INVIGILATOR: DR phiri STUDENT SIZE: 20

Figure 26:html table

4.4 SUMMARY

Testing and functionality testing was carried out successfully. The system was taking through a number of constraint testing factors and use of different data values in an access database were used as a means of testing functionality

CHAPTER FIVE – DISCUSSION AND CONCLUSION

5.1 INTRODUCTION

This section of the thesis basically walks through certain problems that were faced during the system development phases and how they were overcome and over-all outcome of the project.

Several factors came into play and contributed to the difficulties that were faced. Time being one of the main limiting factors the project was not implemented with all objectives implemented.

On a positive note the system is functional and will require improving the algorithm to deal with an additional constraint that was overlooked.

5.2 DISCUSSION

Due to immense work and implementation of many methods to process data three major challenges was encountered. That is having a substantial means of capturing data in large amounts and preventing the user from having input information one by one as this would be a very difficult means of getting the job done. This means of data capture is very unrealistic as in the real world a time table would have over a thousand courses to be inputted and entering of each would defeat the purpose of using a program to carry out computation.

This problem was realized while carrying out tests on the system and it was noticed that entering only 20 courses and 5 rooms to schedule proved to be a tiring thing to do as this needed to be done repeatedly and made the testing process very straining.

The problem was looked at and corrected by implementation of a module that reads courses and room data from an access database thereby having input in bulk. This can prove very useful as a user can simply have someone type the courses and student sizes in advance. Such information is usually held by the administrative authority and can be entered once thus having to schedule the time table as many times as the user may please but not having to reenter courses and room details.

The other major challenge faced was having to deal with java's table mechanism that is not suitable for holding large amounts of data in a table. JTable is not best suited for this purpose and was noticed that if many courses and room details were scheduled it would result in scheduled information being pushed down to a point in the dialog box that wasn't reached and thus prevented the user from seeing the rest of the time table.

This was solved by providing a permanent storage mechanism that enables one to save there produced time table in html which has a table method that is more suited for holding of large amounts of information as the size of the table grows or shrinks according to how data is present in the table.

The third and final challenge encountered was having to build a system that should be able to cope with duplicate entries and prevent scheduling information that may have been duplicated by mistake. By use of the access database this was solved by use of primary keys for the room and course data thereby making sure the user does not make a mistake of which if the user did, an error message would be flagged.

And in the case of the system and direct entry was also programmed to deal with duplicate entry by having the system flash the duplicated entity and not allow its passage to the systems data structure.

5.3 CONCLUSION

The project was a success. In the end all aims and objectives were achieved and production of a fully runnable working system was achieved.

5.4 FUTURE WORKS

The project would be made perfect by improving the algorithm so that it can be able to deal with other constraints that were not implemented such as having to deal with linking of courses taken by students and ensuring they are not scheduled on the same day.

Additional improvements on the actual algorithm could greatly improve the system thus having better results by scheduling rooms in order to have all rooms' filled to full capacity and have all courses scheduled.

Lastly having lecturers place and input soft constraints would be a great advantage in that both efficiency and flexibility would be achieved.

5.5 SUMMARY

Generally the time tabling constraint based problem has been a big factor in mathematical algorithms and was tackled from many points of views resulting in an implementation that works and is operable.

APPENDICES

APPENDIX I: MAIN ALGORITHM CLASS

```
package DATA;
import GUI.*;
import java.util.ArrayList;
import java.util.Vector;
import javax.swing.*;
import java.awt.EventQueue;

public class Algorithm {

    private int week;
    private int day;
    private int timeSlot;
    private int dataIndex;
    private int count = 0;
    private int rnum = 0;
    private String log ;
        private int csiz = 0;
    protected RoomGUI ttgs = null;
    boolean ACTIVITY_SUCCESS = false;
        boolean GENERATE_SUCCESS = false;
        private CourseData cd;
    private RoomData rd;

        Vector<Boolean> isActPlace;
        Vector<Boolean> isRoomPlace;

    public Algorithm() {

    }

    public static ActivityEntry getTTGData() {
        return RoomGUI.getTTGSDData();
    }

    public void initMatrices() throws NullPointerException {
        System.out.println("Init is Start");
        GENERATE_SUCCESS = false;

        isActPlace = new Vector<Boolean>();
        for (int i = 0; i < getTTGData().getCourseLength(); i++) {
            isActPlace.add(false);
        }

        isRoomPlace = new Vector<Boolean>();
        for (int i = 0; i < getTTGData().getRoomLength(); i++) {
            isRoomPlace.add(false);
        }
    }
}
```

```

    }

    }

    public void MainAlgo() {
        initMatrices();
        Random ActR = new Random(getTTGData().getCourseLength());
        log += "\nStartMainAlgo>>" + "Iteration ";

        for (int activity = 0; activity
<getTTGData().getCourseLength(); activity++) {
            intrandAct = ActR.getRandValue(activity);
            if (!isActPlace.get(randAct).booleanValue()) {
                if (placeActivity(randAct,count)) {
                    isActPlace.set(randAct, true);
                    count++;
                }
            }
            cd = getTTGData().getCourse(randAct);

            log += "\n " + randAct + "[" + cd.getLName()
+ "," + cd.getCName()
+ "," + cd.getSize() + ","

        }
    }

    struct();
    checker();
    createTable();
}

private boolean placeActivity(int actnum, int count) {
    CourseDataaa = getTTGData().getCourse(actnum);
    getTTGData().addActivity(count, new ActivitySet(aa.getCName(), aa.getLName(), aa.getSize()));

    return true;
}

private void struct(){
    //room size comparison module
    for(int h=0;h<getTTGData().getActivityLength();h++){
        csiz= csiz+getTTGData().getActivity(h).getSize();
        if(csiz<=getTTGData().getRoom(rnum).getCapacity()){
            getTTGData().getRoom(rnum).addActivity(dataIndex, new
            FinalSet(getTTGData().getActivity(h).getCName(),getTTGData().getActivity(h).getLName(),getTTG
            Data().getRoom(rnum).getName(),getTTGData().getActivity(h).getSize()));
            dataIndex=dataIndex+1;
            System.out.println("ADDING FIRST");
        }
    }

    else if(rnum<getTTGData().getRoomLength()-1&& !(rnum==getTTGData().getRoomLength())){
        addNull(rnum);
    }
}

```

```

csiz=0;
dataIndex=0;
csiz=csiz+getTTGData().getActivity(h).getSize();
rnum=rnum+1;
if(csiz<=getTTGData().getRoom(rnum).getCapacity()){
System.out.println("ADDING SECOND");
getTTGData().getRoom(rnum).addActivity(dataIndex, new
FinalSet(getTTGData().getActivity(h).getCName(),getTTGData().getActivity(h).getLName(),getTTG
Data().getRoom(rnum).getName(),getTTGData().getActivity(h).getSize()));
}
else if(true){
JOptionPane.showMessageDialog(null, "course could not be allocated due to unfit room-sizes : " +
getTTGData().getActivity(h).getCName(),
"Allocation error", JOptionPane.INFORMATION_MESSAGE);
}
}
else if(rnum==getTTGData().getRoomLength()-1){
addNull(rnum);
rnum=0;
csiz=0;

csiz=csiz+getTTGData().getActivity(h).getSize();
if(csiz<=getTTGData().getRoom(rnum).getCapacity()){
getTTGData().getRoom(0).addActivity(dataIndex, new
FinalSet(getTTGData().getActivity(h).getCName(),getTTGData().getActivity(h).getLName(),getTTG
Data().getRoom(rnum).getName(),getTTGData().getActivity(h).getSize()));
System.out.println("THIRD ADD");
}
else if(true){
JOptionPane.showMessageDialog(null, "course could not be allocated due to unfit room-sizes : " +
getTTGData().getActivity(h).getCName(),
"Allocation error", JOptionPane.INFORMATION_MESSAGE);
}
}
}
}
private void addNull(int u){
getTTGData().getRoom(u).addActivity(getTTGData().getRoom(u).getGroupListSize(), new
FinalSet("null","null","null",0));
System.out.print("adding null to dataset "+u+" and move to next schedulling algorithm\n");
}
private void checker(){

for(int u=0;u<getTTGData().getRoom(0).getGroupListSize();u++){
System.out.println(getTTGData().getRoom(0).getGroupList().get(u).getCName());

```

```

    }}

private void createTable(){
    Runnable r = new Runnable() {

        @Override
        public void run() {
            new MultiLineCellTableGen(0,1);
        }
    };

    EventQueue.invokeLater(r);
}

}

```

REFERENCES

1. O. T. Arogundade, A. T. Akinwale, and O. M. Aweda, "A Genetic Algorithm Approach for a Real-World University Examination Timetabling Problem, *International Journal of Computer Applications*, Vol. 12, no. 5 (2010), pp. 1-4.
2. S. Yang and S. N. Jat, "Genetic Algorithms and Local Search Strategies for University Course Timetabling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 41, no. 1 (2011), pp. 93-106.
3. S. Petrovic, "Towards the Benchmarks for Scheduling," in Proc. *The International Conference on Automated Planning and Scheduling, ICAPS07*, Rhode Island, USA (2007).
4. V. Sapru, K. Reddy, and B. Sivaselvan, "Time Table Scheduling Using Genetic Algorithms Employing Guided Mutation," in Proc. *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*(2010), pp. 1-4.
5. W.S. AlSharafat and M.S. AlSharafat, "Adaptive Steady Genetic Algorithm for Scheduling University Exams," in Proc. *International Conference on Networking and Information Technology (ICNIT)* (2010), pp. 70-74.
6. R. Raghavjee and N. Pillay, "Using Genetic Algorithms to Solve the South African School Timetabling Problem," in Proc. *Second World Congress on Nature and Biologically Inspired (NaBIC)* (2010), pp. 286-292.
7. S. Abdullah, H. Turabieh, B. McCollum, and E. K. Burke, "An Investigation of Genetic Algorithm and Sequential Local Search Approach for Curriculum-based Course Timetabling Problems," in Proc. *Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009)*, Dublin, Ireland (2009), pp. 727-731.
8. S. Abdullah and H. Turabieh, "Generating University Course Timetable Using Genetic Algorithms and Local Search," in Proc. *3rd International Conference on Convergence and Hybrid Information Technology* (2008), pp. 254-260.
9. G.N. Beligiannis, C.N. Moschopoulos, G.P. Kaperonis, and S.D. Likothanassis. Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Re- search*, (2008) pp. 35
10. P. Hansen, N. Mladenovi_c, and J.A. Moreno P_erez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*,(2010),pp. 175.

11. M. Mitchell , "An introduction to Genetic Algorithms", The MIT Press (Massachusetts Institute of Technology), (2008).
12. N. Pillay. An overview of school timetabling research. Practice and Theory of Automated Timetabling (PATAT 2010), August 11-13, Queen's University, Belfast, UK, 2010.
13. R. Raghavjee and N. Pillay. An application of genetic algorithms to the school timetabling problem. In Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology, pages 193{199. ACM, 2008.
14. H.G. Santos, L.S. Ochi, and M.J.F. Souza. A tabu search heuristic with e_cient diversi_cation strategies for the class/teacher timetabling problem. Journal of Experimental Algorithmics (JEA) 2008.
15. M.J.F. Souza, LS Ochi, and N. Maculan. A grasp-tabu search algorithm for solving school timetabling problems. Metaheuristics: Computer Decision-Making. Kluwer Academic Publishers, Boston, pp. 659 (2003).
16. K. Yuri, K. Polina, and P. Mikhail. Formulation space search approach for the teacher/class timetabling problem. Yugoslav Journal of Operations Research,pp.18, 2008.
17. D. Zhang, Y. Liu, R. M'Hallah, and S.C.H. Leung. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. European Journal of Operational Research,(2010).
18. Ahmed Al-Tehami, Ahmed Al-Salehi, Ammar Zahary, "Application of genetic algorithms in the problem solving of course timetables, Case Study: timetables of Faculty of Science and Engineering, University of Science and Technology, Yemen", Proceedings of International The International Arab Conference on Information Technology (ACIT), ACIT 2009, 14-16 December (2009).
19. M. Khedr, A-K. Abbadi, A. Al-Omari, M. Daud, kyphotic, "Using genetic algorithm to solve the problem of scheduling the school", Journal of Science and Technology (JST), Issue No. 9, (2005).
20. T. Moller and R. Partak, "Interactive Timetable", Department of Theoretical Computer Science, Charles University, (2001).
21. A.V. Moura and R.A. Scara_cci. A grasp strategy for a more constrained school timetabling problem. International Journal of Operational Research, (2010) pp.152.

22. Bufe, M., Fischer, T., Gubbels, H., Hacker, C., Haspirch, O., Scheibel, C., Weicker, K., Wenig, and M.: Automated Solution of a Highly Constrained School Timetabling Problem – Preliminary Results. In: Boers, E.J.W. et al. (Eds.): Proc. EvoWorkshops 2001. Springer-Verlag, Berlin Heidelberg New York. 431–440 (2001)
23. Moreira, J.J.: A System for Automatic Construction of Exam Timetable Using Genetic Algorithms. *Tékhné*, vol. 6, pp. 319-336(2008)
24. Reis, L., Oliveira, E.: A language for specifying complete timetabling problems. *Practice and Theory of Automated Timetabling III*, pp. 322-341(2001)
25. Gröbner, M., Wilke, P.: A General View on Timetabling Problems. 4th International Conference on the Practice and Theory of Automated Timetabling - PATAT'2002. (2002)
26. Kajisha, H., Saito, T.: Synthesis of self-replication cellular automata using genetic algorithms. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, 2000*, pp. 173-177 vol.5 (2000)
27. Dammak, A., Elloumi, A., Kamoun, and H.: Lecture timetabling at a Tunisian university. *International Journal of Operational Research*, vol. 4, pp. 323-345(2009)
28. Birbas, T., Daskalaki, S and Housos, E (2006), “School Timetabling Quality Students Teacher Schedules”, *Journal of Scheduling* 12(2): 177 – 197.
29. Dancius, D (2003), “Evolution Timetabling Using Biased Genetic operators”, *Journal of Computing and Information Technology – CIT* 11, 2003, 3, pp. 193 – 199.