University of Sheffield

A Timetabling System for a small primary school

Mariam Jibrin Usman

Supervisor: Professor Geog Struth

A report submitted in fulfilment of requirements for the degree of MSC in Advanced Computer Science in the Department of Computer Science

September 13 2017

Declaration

All sentences or passages quoted in this report from other people’s work have been specifically acknowledged by clear cross-referencing to author, works and pages. Any Illustrations that are not the work of authors of this project have been used with explicit permission to the originator and are specifically acknowledged. I understand that failure to do this project amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name:

____

Signature:

____

Date:

____

Contents

Table of Figures

1 Chapter 1: Introduction

Scheduling is an important aspect of research, and over the past few years, a lot of research has been focused on this field. [1] The primary school timetabling problem consists of scheduling a set of teachers to a given class at a particular timeslot for a specific subject. All assignments must be done in such a way that they do not violate the constraints imposed on them. An optimization solution to the timetabling problem is one that satisfies all hard constraints and minimizes the weighted sum of penalties associated with soft constraints. [2]. Hard constraints are those which must be satisfied to produce a feasible timetable, while the desirable but not essential constraints are called soft. The soft constraint are those that must not be satisfied, but if they are, they increase and promote the quality of the timetable. [3]

In the early years, constructing a timetable was done manually, this was a tedious and time consuming task, which often requires more than a day to complete. Why might this be so complicated one might ask. This is because of the large amount of data, the diversification of teaching methods and ever increasing requirements in the academic curriculum. In addition to this, the dynamic nature of constraints continually changes according to the country and organization. In some cases, the typical manual timetable may generate unsatisfactory solutions [4], for instance, a teacher might be scheduled to teach in two classes simultaneously. Because of this, considerable attention has been devoted to automatic timetabling [4]. This problem is is known as NP hard, because even with the use of a computer, it requires a lot of computational effort before a result is generated.

1.1 Aims and Objectives

As previously stated, previous research has been carried out in this area, we propose to use an already existing method to solve the timetabling problem. Our dissertation shall be based on creating an automated timetable that uses the Genetic Algorithm to solve the optimization problem. Different approaches to timetabling include constraint logic programing, graph heuristics and integer linear programing. But it has been discovered that most success is generated from using the meta heuristic methods, which includes simulated annealing, evolutionary algorithms and tabu search.
Past research works include simulated annealing by Abraham D [5], and Tarawneh el,Al. [6], genetic algorithm by Rusjil, R and Pillay [7] , Coloroni et. Al [8]. Finally we have the Tabu search approach used by Daniel Costa [1] Salman Hoshmand et.al [9] We shall discuss in further detail the academic works that employed each of these approaches. We shall also provide a critical review, with relative evidence as to why we chose our proposed algorithm. This algorithm would generate a best or near optimal solution to the problem. The aim would be achieved with the following objectives in mind:

- Initially define all hard and soft constraints that must be satisfied before a feasible timetable can be produced.
- Provide an academic curriculum which shall be manually populated by the timetabler.
- Define a representation for the timetabling problem in term of the proposed algorithm
- Manually Code the algorithm using our desired programming language
- Generate a feasible solution to the primary school timetabling problem.
- Provide solutions that help the system create a best optimal solution.

1.2 Organization of report

This section describes the purpose of the dissertation. Further across the report, we provide six chapters, which we shall briefly describe. The chapter 2 provides a critical literature review about previous meta-heuristic approaches used in solving the timetabling problem. In chapter 3, we shall provide the system requirements, ethical and professional rules, finally we state the testing we shall perform on the system. Chapter 4 shall describe the system design, which shall include suitable UML diagrams, ERD and SDLC. In relation to chapter 5, we proceed to transform the timetabling problem into an optimization problem. We shall describe in full detail how we applied the genetic algorithm in our small primary school to regenerate feasible results. Chapter 6, we discuss the results generated and further research works which could be applied to the system. Finally, we conclude by providing an evaluation of the entire dissertation work. Additional information shall be provided in the appendices and reference.

2 Chapter 2: Literature Review

In this literature review, area of concentration is directed to solving the timetabling problem in a primary school. Timetabling problem as NP hard can be justified through adequate research

The initial perspective of many users of a timetable would be a simple large puzzle. But the puzzle as many may call it fail to see that this an extremely difficult task in scheduling. This would require a lot of time for the basic human being to solve, but for the computer, it takes a matter of seconds. Why is this so difficult?. The difficulty lies in the constructing a feasible timetable satisfies all constraints. Because of this, numerous research and algorithms have been proposed to solving this problem.

According to Michel Gendreau et.al, this NP hard problem can be solved by using heuristic methods, such as the local search(LS) improvement technique. LS is an iterative search technique that progressively improves an initial feasible solution by applying a series of local moves or modifications. A known limitation of the LS method is local optimum, as the search for solutions begins, it encounters local optimum with respect to the transformations [10]. The world of optimization suddenly changed when the release of a new heuristic approach called Simulated Annealing(SA) suddenly came into existence. Proposed by Kirkpatrick et.al [11], the simulated annealing heuristic approach was successfully able to generate an optimal solution from a combinational problem. Immergence of this algorithm, proved to researchers that there were other ways to solve an optimization problem [10]. Therefore, this lead to creation of new analogies based on natural phenomena such as the Tabu search and genetic algorithm, now collectively known as meta-heuristic methods [12].

In chapter 1, we proposed to use an already existing scientific algorithm, which we shall prove to solve the timetabling problem in chapter 5. This chapter shall show justification as to why we choose to use the Genetic Algorithm(GA), as opposed to other scientific algorithm. We shall give an academic literature review on the various meta-heuristic algorithms which have been used for timetabling. We shall not only give a summary of past researched works and their algorithms, but we shall provide critical analysis of their works.

Explanation of the proposed algorithm, used in their research works shall be provided, along with their generated results. An honest critical analysis regarding the academic research works shall be provided from our point of view. Finally, to end this chapter, we shall justify with reasons provided from this, why we chose our preferred scientific algorithm to solve timetabling.

2.1 Simulated Annealing (SA)

The first scientific algorithm to be discussed, is the Simulated Annealing (SA). The Simulated Annealing algorithm, is a Monte Carlo technique which can be used to solve problematic optimization problem such as the timetable.

According to Steve, R, White,
“Simulated Annealing is a powerful technique for finding near optimal solutions to NP-complete combinatorial optimization problems”.

Chibant et.Al [13] described annealing as a metallurgical process of a solid being placed in a crystallized state, due to its temperature being reduced after it has been subjected to high temperature. As temperature decreases, there shall be a corresponding decrease in atomic energy till it reaches its lowest energy. In the context of optimization, the SA seeks to emulate the annealing process. The concept of SA can be described as cooling of hot vibrating atoms in random displacement. As the algorithm is applied, the inter-particle of the atoms are forced to bond to one another due to cooling. As atoms are cooled, they enter a frozen state, indicating no movement. If the mass is cooled quickly, the chance of obtaining a low cost solution is reduced, compared it if it being slowly cooled or annealed.
For instance, a situation whereby input values are allowed a great range of variation, therefore the assumption is, the SA is at a very high temperature. The temperature shall begin to fall as the algorithm proceeds, therefore imposing a restriction on the degree of inputs which are allowed to vary. The presence of the algorithm which lead to changes in input values generates a better solution, which eventually results to an optimum set of input values as the temperature is close to zero. The results generated shall produce a realistic and precise simulation results. The advantage of this algorithm is its ability to generate an optimum solution, and to avoid being trapped in local minimum. It avoids being trapped by accepting both good and bad solutions with a given probability. [13]

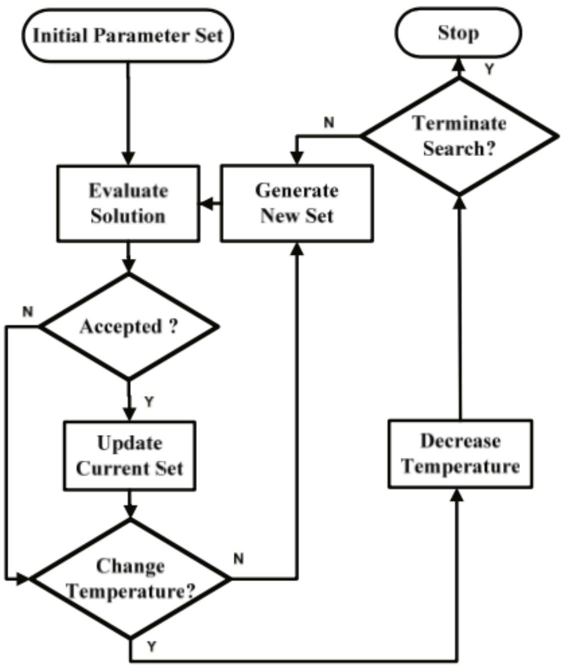


Figure 1: Simulated Annealing Flow Chart *[13]*

As illustrated in the Figure 1, a flow chart is used to represent the SA algorithm. As previously stated one of its main advantages is to escape from the local optimum based on the acceptance rule of a candidate solution. For instance, given a current and old solution f_{new} and f_{old}, each with an objective function. The values of both objective functions are compared, if that of f_{old} is greater than that of f_{new}, therefore the current solution would be accepted. This is because, it supports minimization. An alternative to the current solution being chosen could also be if Boltzmann distribution value is greater than a uniform random number in [0,1].

$$\boxed{e - \frac{f_{new}-f_{old}}{T}}$$

Equation 1: Boltzmann Distribution

As T the parameter for temperature control.

2.1.1 Initial Population

At the start of iteration, the initial parameters are created as a result of randomization. These population is randomly dispersed across a given set of boundaries. The initial parameter in SA do not require initial solutions, the closer the initial estimate is from the global optimum, the faster the optimization process.

2.1.2 Initial Temperature

The initial temperature is the parameter that controls the acceptance rule defined by the developer. Temperature T is a large number that enables the algorithm to make small moves which are enough to move off a local minimum but not reaching a global minimum.

2.1.3 Perturbation mechanism

The perturbation mechanism, creates changes in the current solution by exploring its neighbourhood, therefore, generating new solutions from the current solution. In [13], they defined a solution s as a vector $(y_1 \ldots, \ldots y_n)$ which represents a point in the search space. A perturbation from the current solution s generates a vector $\sigma = (\sigma_1 \ldots, .. \sigma_n)$ of a standard deviation. The solution from the perturbation creates a neighbour solution (Equation 2) where $N(0, \sigma_i)$ is a random Gaussian number with zero mean and σ_i standard deviation.

$$\boxed{x_{i+1} = x_i + N(0, \sigma_i)}$$

Equation 2: Neighbor Solution

2.1.4 Cooling Schedule

The SA algorithm makes use of the geometric rule for temperature variation, for its cooling schedule which is:

$$\boxed{T_{i+1} = sT_i}$$

Equation 3: Geometric rule for temperature variation *[13]*

In the research by Ru, Chibante(2010), they presented that temperature of s within the range of [0.8, 0.99] proved to have good results. In a review proposed by (Fouskakis & Draper, 2002), they proposed that a parameter which takes a number of iterations should be defined. The value of the parameter may be constant or may be a function of the temperature which is related to the size of the neighbourhood or search space.

2.1.5 Termination Criterion

The iteration shall terminate once these conditions are met, a maximum number of iteration has been reached, a minimum temperature value, a minimum value of objective function and a minimum value of acceptance rate.

2.1.6 Applying Simulated Annealing to Optimization Timetabling Process

In this section, we shall introduce two academic research works that applied the simulated annealing algorithm. The research work proposed by Abramson, D. [14] shall be the first, while work by Tarawneh el.al [6] shall be the second.

In the research work by Abramson, D. [14], a prototype application was created to solve timetabling problems for an Australian High School. In their system, they considered an element, which is a combination of a teacher, subject, class and room. The task of their application was to schedule the elements such that the teacher, class or room does not appear more than once per period. They replaced the elements with atoms and system energy with the timetable cost. For the first stage of the algorithm, the initial population was generated randomly by assigning the elements to individual periods. Also, an initial cost was generated, which was used as a reflection of the timetable quality. As for the initial temperature, they used a method proposed by [15] for its calculation. This was to allow the initial temperature to be high enough to perform swaps proportional to the number of elements which would concurrently raise the probability of the initial cost to occur. As initial temperature is present, two limits are introduced, which are the "maxswaps" and "maxsuccessswaps". These limits chosen to be proportional to the number of elements present. The maxsuccessswaps is to limit the number of successful swap, while the maxswaps limits the number of successful and unsuccessful swaps. As movement of swap occur from one period to another, this results in change in the cost. Therefore, at each iteration, a period is chosen, an element is moved from its current period and placed in a different period which is randomly selected. As iterations proceeds, a reduction in timetable cost eventually becomes zero, reaching the termination stage. At the end of this run, a feasible timetable will be created. An alternative to reaching to terminating a run was if the cost had not changed after a certain amount of iterations.

Their proposed method was tested on 10 existing timetable, all from different high schools. The first five test datasets, generated realistic timetables, with a timetable cost at 0, which we regard as successful. But the dataset used for the scheduling process can only be used for a small school. The next 6 – 9 datasets had more complex data, that were equivalent to a typical high school. They all generated a timetable cost of 0 except for dataset 9, which generated a final cost of 3. Therefore, it did not reach a feasible timetable, but rather its cost did not change after several iterations. Finally, as for the 10th dataset, it reached a cost of zero, the execution time was 14(fourteen) hours, which is extremely high. We discovered that this was because of high amount (complex) of [16]elements in the school.

In the second research work proposed by Tarawneh el.al [6], they developed an application which used a hybrid simulated annealing with solutions memory (SAM) to solve course timetabling problems in Universities. Their work was divided into two phases, which involved construction of the initial population and applying simulated annealing algorithm. In the first phase, they used a greedy heuristic method, to construct the initial solution which satisfies all hard constraints imposed. In the first stage, they also employed a steepest decent technique with the greedy heuristic in order to search for a feasible solution. This was useful in situations where the greedy algorithm was not able to generate a feasible solution. While in the second phase, they used the SAM to minimize the soft constraint violations. They designed a probability acceptance criterion to enable the algorithm accept new solutions that are worse, equal to or less than the current solution. The algorithm starts by generating n neighbors from neighborhood structures, a selection phase is invoked which retrieves the 2 best solutions. We define the first solution as sol_1 and the next as sol_2. sol_1 is accepted if the value from the objective function is less than the current solution, while sol_2 is kept in the memory M_1. The memory was defined to save all solutions. As search proceeds, if there are no improvements to generate a new solution, one solution is randomly selected from M_1, to result to a new solution. The new solution is subjected to a shaking procedure, which randomly swaps the highest penalty lectures with other lectures that satisfy the constraints to generate sol_m . If the solution is still not improved, a penalty guided perturbation is employed to improve the best solution. The perturbation operator was used to solve local optimum by randomly selecting highly penalized lectures to new positions or swapping two different lectures. In their research work, they used the cooling schedule proposed by [17] was used to calculate the cooling rate of the algorithm.

The application was subjected to test on 21 datasets from the second international timetabling competition in 2007. The results generated from their work proved to be successful as it was able to generate feasible timetable solutions based on curriculum based course timetabling used in the timetabling competition. We cannot judge the processing time used to generate a solution in this work because, they were not indicated.

2.2 Tabu Search:

Proposed by Fed Glover [12], the Tabu Search approach is focused on following local search whenever it encounters local optima by allowing on improvement moves. Mohamed Tounsi defined the approach as an extension of the local search procedure that focuses on finding a minimum from a feasible set of solutions. The Tabu Search approach was developed to satisfy the short comings of the local search method. It satisfies this short coming by using a tabu list, in order words memories. As local search is executed, it encounters non-improvement moves (local optimum), the tabu search keeps history of all solutions visited by the local search. All visited solutions are then kept in a tabu list. It stores all visited solutions to avoid the local search from cycling back to already visited solutions while searching for new solutions. Associated with the TS are two basic elements, which include the search space and neighborhood space. The search space is a list of all possible solutions visited by the LS. While the neighborhood space is derived by applying a local transformation S in each solution in a search space $N(S)$. An advantage of TS over LS is, during the search process, it keeps replacing previous neighbourhood solutions $N(S)$. with new ones $N^*(S)$. The most distinctive element in the TS are the tabu list "memories", which are achieved by declaring moves that reverse the erect of recent moves. These negative or disallowed moves are referred to as tabu, for instance, if we are given a problem where we must visit a set of cities only once. We visit C_1 then C_2, we could declare a tabu as moving back from C_2 to C_1 to move to the third city. Without the presence of tabu's, the best move, could be taken away from a local optimum, therefor making a non-improving move. If this happens, the next search shall fall back into the local optimum by taking the best move available at that time [12] . Aggressive moves are made by the tabu list which seeks to make the best possible decisions to satisfy the tabu. As effective as how the tabu list is, there is also a limitation which prevents the search from cycling. This is the fixed length, every tabu list is given a fixed length which affects its process. If the solution in the list is greater than the size of the list, this causes cycling because, no more space to accept solution [10]. The use of tabu are so powerful that they prohibit attractive moves even where there are no more cycles. This results to infinite loop of searching for solutions, described by Micheal Gendreau and Jean-Yves Potvin as an "overall stagnation of the searching process". Therefore, to resolve this, an aspiration criterion is used to revoke the effects of tabus. For instance, the simplest aspiration criteria used in tabu search method is to allow a move even if its tabu. If the results generated is greater than that of the best solution, then terminate [10].

2.2.1 Notations:

S the current solution
S^* the best solution
f^* the value of S^*
$N(S)$ as the neighbourhood solution of S.
$N_i(S)$ as the admissible subset of $N(S)$.

2.2.2 Initialization

Set $S \leftarrow S_0$, $f^* \leftarrow f(S_0)$, $S^* \leftarrow S_0$, $T \leftarrow \emptyset$.

Search

```
While termination criterion not satisfied do
    select $S$ in $argmin_{S' \in \tilde{N}(S)}[f(S')]$;
    if $f(S) < f^*$, then set $f^* \leftarrow f(S)$, $S^* \leftarrow S$;
    record tabu for the current move in $T$ (delete oldest entry if necessary).
```

2.2.3 Application of Tabu Search on Timetabling

A new approach proposed by Daniel Costa [18] a solution to solve timetabling problems called COSTA which stands for "Computing an Operational Schedule with a Tabu algorithm". In their research work, Costa was created as a computerized algorithm for solving the course timetabling problems. We shall describe how the tabu search approach was applied in Costa, by describing the initial population, neighbourhood, and aspiration criteria. To facilitate the generation of an initial timetable, available lectures are assigned to periods, teachers and classes. The process of assignment starts by selecting lectures with the smallest number of available periods. Since this is also a constrained based satisfaction problem, they defined six types of constraints that should be satisfied. These include, no overlaps authorized, no teacher overlaps, no class overlaps, no teacher and class overlaps, no room overlaps and no overlap restrictions. In their research work, a feasible timetable is one which assigns an available lecture to a feasible period, such that all constraints were satisfied. Just as in the basic Tabu Search algorithm, they defined a neighbourhood solution $N(T)$ as the set of feasible timetables from a timetable T. $N(T)$ was derived by changing the periods of exactly one lecture where two lectures are simultaneously scheduled in two class timetables. To prevent cycling, rather than making use of the single tabu list, they made use of two. We assume that they based their reasons on Michael Gendreau definition of on using multiple tabu list. He proposed that creating separate tabu list for each value is better and more effective than using a single tabu list. They denoted T_1 and T_2 as two separate tabu list, in T_1 we introduce l which is any lecture moved from period p_1 to p_2. While in T_2 they defined a pair (l, p_i), which is the lecture l at a specific position p_i. Therefore during T_1 steps of the tabu search the lecture l cannot be moved while during T_2 steps, lecture l cannot be replaced at period p_1. Also for the aspiration function, they used two, denoted as $A_1(z)$ and $A_2(z)$. Where $A_1(z)$ is active when the new solution is less than the best known solution and $A_2(z)$ is, when the new solution is equal to the best known solution. For aspiration function A1(*z)* and A2(*z*) would be updated whenever timetable *T* moves to the best timetable *T**. Finally, which brings us to the generation of neighbour solution.

As conflicting lectures, for instance l_1 will be conflicting if it is an isolated lecture, there is no other lecture before or after it. A solution to fix a conflicting lecture, would be to move it to another period even though it doesn't improve the objective function.
In order to improve the speed of local search in the search space, they made a distinction between the conflicting and non-conflicting lectures. As long as the best solution reached so far does not satisfy the overlap constraints, the moves of the lectures which do not respect the constraints were allowed. This method allowed them to reach regions of the search space faster, compared to searching all areas in the search space. Therefore, to reach a feasible timetable, every conflicting lecture is moved without restriction to overlap free solutions, while the search is taking place.

Through relevant data provided in the research work, the application COSTA was successfully able to generate a feasible course timetable solution for a high school in Porrentruy. The test data used were 780 lectures, which had to be scheduled in 5 days of 10 periods each, making a total of 50 periods in a week. Aside from the lectures, they used 32 classes, 32 subjects and 12 different types of special rooms. The first application of the timetable produced 20521 iterations and 41 minutes of CPU time while reaching the first overlap free timetable with 969 iterations. This results generated indeed prove COSTA success in solving the problem, but in our understanding the amount of computational time required to generate the optimal solution was rather long. Irrespective of this limitation, their application saved a considerable amount of manual labour spent in creating a feasible timetable.

In our second research paper proposed Salman Hoshmand et.al [9] was an automated timetabling system used to so solve high school timetabling problems. They also described this problem as a constraint satisfaction problem, where lessons of a set of courses would be scheduled into a weekly timetable in accordance with a given set of constraints [9] [19]. The application composed of three phases, which include scheduling special class session as the first and using a greedy heuristic method to generate a feasible timetable as the second. While the third involved using the Tabu search algorithm to improve the initial timetable in the second phase. We shall focus on analysing steps used in the second and third phase because, the first uses a simple graph based algorithm to create special sessions. These special sessions groups of lessons to be scheduled simultaneously, such as maths and English. In the second phase, the system uses a sequential greedy heuristic approach to create an initial feasible solution. Just as in COSTA by Daniel Costa the initial solutions of the timetables involved scheduling lessons l to an available period p without violating any constraints. The system counts the amount of unscheduled lessons which could be suitable for that period, then chooses the period with the smallest number. Finally, the third phase, which is the optimization phase uses the tabu search algorithm to find an optimal solution. We describe their solution in relation to the tabu search algorithm.

Their search space was a set of schedules that satisfied all constraints. Their aim was to minimize the objective function which is a set of unscheduled lessons in the search space by satisfying soft constraints. The system neighbourhood $N(T)$ is obtained by changing periods of lessons in timetable T, just as in COSTA. The steps involved in neighbourhood includes selecting a lecture l and assigning it to a new period p. Assign new lectures to new periods only when a previous schedule violates constraints. In order to find the neighbourhood, they devised two types of moves, which were out-in move and intra move. The out-in move was used to minimize the objective functions by moving a lesson l into period p. While the intra move solves the problems caused by the out-in move. As the out-in move proceeds, the amount of period p, reduces leading to a local minimum. The Intra move solves this problem by moving previously scheduled lectures to new periods. In the aspect of tabu list, unlike COSTA, which used multiple tabu list, their approach used a single tabu list. The tabu list consisted of a pair (l, p), which is the lecture and the period in which it was scheduled. They considered a move as tabu if a lesson moves back to its previous period instead of moving into a new period. The aspiration criteria involved dropping the tabu status when the best solution is found.

Compared to the previous approach, they used the tabu search algorithm based on a frequency based diversification to create the optimal solution. The diversification method was employed to minimize the risk of establishing local optima by revisiting previously unvisited regions in the search space. They employed a transition based long term memory to store the frequency of movements involving each lessons and periods. When diversification is active, the long term memory shall evaluate and prioritize new and previous moves from a best solution.

Our understanding of the research, their application proved to be successful as it solved the timetabling problem. We have to say that their approach to incorporate the long-time memory structures to guide diversification in tabu search was quite brilliant, as this resulted in creating a feasible timetable. Although this solved the problem, we also have to point out the time taken to generate a solution, was also high. Although they did not state it, but the amount of iterations required to generate a solution was 2664.

The two tabu search applications discussed in this subsection both proved to be successful in finding an optimal solution to the timetabling problems. This proved that the tabu search algorithm would be successful in creating a timetabling application such as ours.

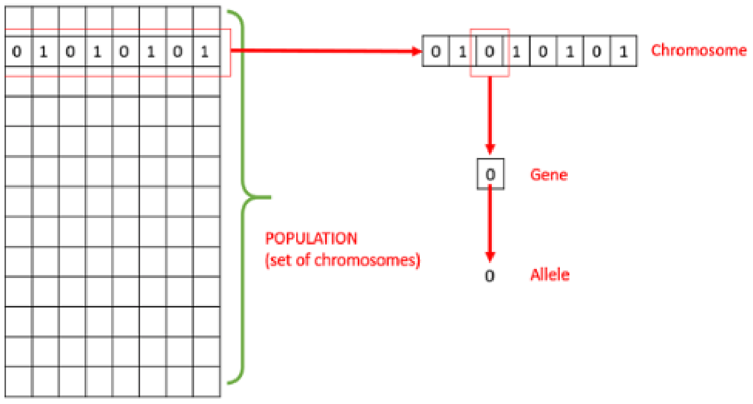
2.3 Genetic Algorithm

Created by John Holland in 1960 [20], the Genetic Algorithm represents a branch of evolutionary computation that mimics the theory of evolution. It is based on three principles, which include evaluation, selection by fitness and reproduction which all simulate biological processes. GA is a search based optimization technique based on natural selection, which it uses to solve optimization problems. The success rate of GA is relatively high as it has been previously used by researches in the field of machine learning to find optimal or near-optimal solutions to extremely difficult problems (complex search problems). To give a detailed explanation of the term “Genetic Algorithm”, we shall briefly explain some biological terms, which shall enable reader understanding. In biology, it is a known fact that all living organisms constitutes of billions of cells, and each cell consists of one or more chromosome. Each chromosome can be subdivided into a genes (traits), with alleles which are the values of the genes. Chromosomes undergo recombination (or crossover), where the parent exchange each pair of genes to produce a gamete (single chromosome). These offspring from the parents are subjected to mutation which are results from copying errors. Finally a fitness is calculated, which is the probability of which child chromosome shall live to reproduce, to continue the cycle.

This Genetic Algorithm allows one to set a level of randomization and level of control, making it a very effective and efficient algorithm compared to the basic randomized search and exhaustive search algorithm(Jenna Carr, 2014) (Golsberg, D. E, 1989) and other traditional methods. Further alternatives involved in selecting this algorithm is , it provides a list of possibly good solution which improves over time. It is useful when the search space is relatively large with a variety of parameters involved, such as the timetable. Above all, the most important reason is, it has high success rate of finding a solution to the problem.

2.3.1 Basic Terminology: Elements of Genetic Algorithm

To further explain Genetic Algorithm, we shall define the basic terminologies common to all genetic algorithms which shall be used throughout the report.



A population of Chromosomes

At the inital stage of GA, a population of chromosomes is created. The GA performs a randomised selection of chromosomes to serve as the first population. Locus of chromosomes are usually represented in 0 and 1, therefore two possible alleles in chromosomes can be represented in 0 and 1. These chromosomes represents a candidate solution to the problem that the genetic algorithm is trying to solve. After a new generation is created, it successfully replaces the old population(generation) with the new population. As previously stated, the GA requires the fitness function of each chromosome to reproduce.

2.3.1.1 The Fitness Function:

This is regarded as the function the algorithm is trying to optimize.

2.3.1.2 GA Operator

The GA operator used by all genetic algorithm include, selection, crossover and mutation.

2.3.1.2.1 Selection Operator:

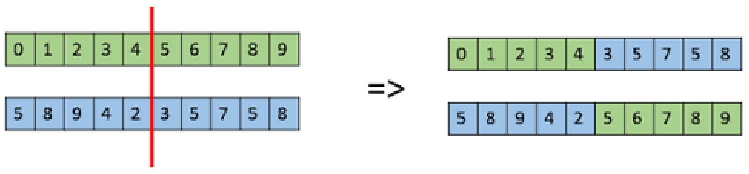
The selector operator uses the user defined probability distribution to perform the function of identifying or selecting chromosomes that are fit for reproduction. For instance, the sum of probability of a chromosome is equal to 1, therefore the probability of a complete set, added all up is equal to 1. We define a function in eqn(1) where the probability is proportional to the fitness divided by the sum of fitness over i.

$$p\ i=fiifi$$

It acts on the basis that, the fitter the chromosome, the more times it is likely to be selected to reproduce.

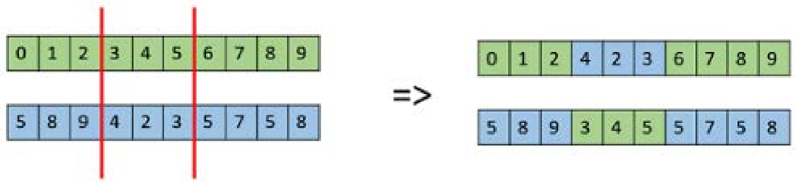
2.3.1.2.2 Crossover Operator:

The crossover operator is analogous to biological crossing over and recombination of chromosomes in cell meiosis. It selects two fit chromosomes in the population, creates an offspring by swapping their chromosomes. There are four types of crossover, which involves, one point crossover, multi point crossover, uniform crossover, uniform crossover and whole arithmetic recombination.
One point crossover: This involves swapping the tails of two locus parent chromosomes to produce a new offspring.

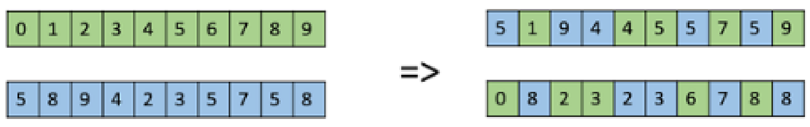


2.3.1.2.3 Multi Point Crossover :

In this crossover, offspring are generated by swapping alternating segments. It is a generalised form of the one-point crossover.



Uniform Crossover: This crossover, involves “randomization”, compared to the other crossovers which split (divide) the chromosomes. This is performed based on random decision to decide if a chromosome from a parent shall be included in the offspring.

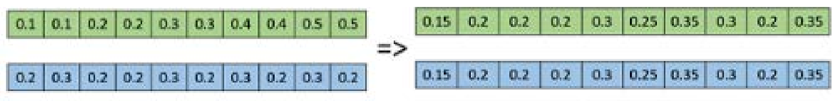


2.3.1.2.4 Whole Arithmetic Recombination

This crossover takes the weighted average of two parent locus chromosomes by using this formula.
Child 1=a.x + (1-a).y

Child 2 = a.x + (1-a).y

If a = 0.5, then both the children would be identical as shown

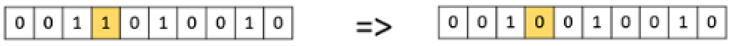


2.3.1.3 Mutation Operator:

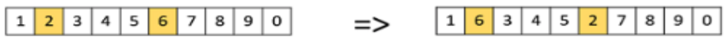
As previously stated genetical mutation in living organisms is caused by cloning errors. In the GA, it randomly flips one or more genes in the chromosome. There are three types of GA mutation, which include the bit flip mutation, random resetting, swap mutation, scramble mutation and inverse mutation.

2.3.1.3.1 Bit flip mutation:

This is very straightforward as it selects one or more bits from locus parent chromosomes and swap them.



Swap Mutation: In this mutation, two genes from one locus parent chromosomes are randomly chosen and swapped

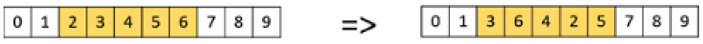


2.3.1.3.2 Random Mutation:

This is an extension of the bit swap mutation, but the only difference is, gene value form chosen chromosomes are randomly selected, and swapped with a randomly chosen gene.

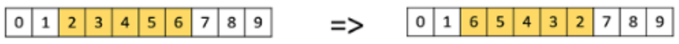
2.3.1.3.3 Scramble Mutation:

This is used with permutation representation where there is a specific ordering. Selected from a locus chromosome, a subset of gene are selected and randomly scrambled.



2.3.1.3.4 Inversion Mutation:

In this mutation a subset of genes from a locus chromosome position are inversely flipped.



After all operators have been applied, the next generation created, must be equal to the initial population. The GA is run in multiple iterations therefore repeating the cycle until the fitness value of the “best-so-far” chromosome stabilizes and does not change for many generations. This means the algorithm has finally reached a solution.

2.3.2 Applying Genetic Algorithm to Optimization Timetabling Problem

In this section, we shall provide two academic research works, which applied the Genetic Algorithm to their proposed works. We shall consider a two-phased genetic algorithm by [21] which proposed that, using the two GA algorithms, is better suited to solve an optimization problem. While the second proposed by Colorni et.Al [22] ,this algorithm was used to create an application that solved real world instance of the timetabling problem.

In the first academic research paper by [21], they proposed a two-phased genetic algorithm, which produced a feasible timetable that satisfied all constraints. They assumed that by using two genetic algorithms, they were able to produce not just a feasible timetable, but a very effective timetable. The GA begins with an initial population of individual elements represented as a timetable matrix. The timetable matrix represents the periods(timeslots) as rows and the columns as a specific class to be taught. The teacher teaching the class is stored at the intersection of each row and column. For the first phase of the timetable generation(population) random assignment of tuples to periods are made. A tuple is a combination of the class, teacher and venue (C1, T1, V1) or class-teacher (C1,T1) assignment. While creating the initial population, a low-level heuristic such as graph colouring is used to assign the tuples according to the level of difficulty in the schedule. Therefore, assignments are done in such a way that they do not violate the hard constraints. As the search is performed if there are two feasible periods available, the tuple would be allocated to the minimal penalty period. If there are no available periods, then an assignment is made to a randomly selected slot. A second phase, which was evaluation and selection. In evaluation stage, the fitness measure of each individual(tuple) is calculated. Evaluation performs 2 phases, where by phase 1 measures the fitness according to the amount of hard constraints violated. While phase 2 is used to measure violations of the soft constraint as a secondary measure for situations where at least two individuals have the same fitness. After evaluation, the selection stage is invoked. Which selects individuals with the best fitness form the population as parents. The third stage is for regeneration, here, one or more mutation operators are applied to chosen parent from the selection phase 2. After the mutation operator is applied, an offspring with a fitness equal or close to that of the parent is created. The iteration runs till a feasible timetable is generated. Unlike other research works, the crossover operator was never involved. They supported what was proposed in [23], by stating that, the crossover operator causes violations of problem requirements.

Their approach was tested on four different types of timetable sets with a total of 13 problems. These problem set were based on real primary and high school’s data. We agree that different combinations such as construction heuristics, selection methods and genetic operators is necessary as they proposed. Through results generated, it is evident that the use of the two-phased genetic algorithm is required to produce a feasible timetable of good quality [21]. It not only solved timetabling problems for a primary school, but also a high school, which have different constraints and requirements. We consider this work as brilliant, as it has solved its main purpose.

The second research paper by Colorni et.Al. [22] also used a GA approach to solve the timetabling problem. The main goal of their research was to understand the limitation caused by the GA and its potentialities in addressing high constrained problems. That is optimization problems that shifts a solution from being feasible to infeasible due to minimal change. Their work represented the problem as a matrix R (an n.m matrix of $r_{ij} \in A$) where A is set of jobs for each teacher. Within the matrix, the ith row represents the teacher, while the column represents the timeslots. Each element r_{ij} representing the gene, with its allelic value of ***A*** to a specific teacher corresponding to the row containing the gene. To be a feasible timetable, the matrix must satisfy all timetabling constraints, therefore, they introduced the genetic operators, a filtering algorithm and an objective function. The constraints were split into two kinds, which were the row and column constraint. The row constraints were handled by the genetic operators, which restricted the change of teacher hours after it had been set at the initial phase. The row constraints were regarded as the “hard”, while that of the column were “soft”. The soft constraints are infeasibilities due to superimpositions, they were managed by means of a combination of fitness function and genetic repair
At each generation the minimum and maximum objective function of individual in the population is calculated. Their approach was focused on minimizing the objective functions and maximizing the fitness function. In order to the objective function, their application made use of three structures, where level 1 is the feasibility condition that handles conflict such as superimpositions of teachers. Level 2 was the management condition which involved three topologies which were didactic, organizational and teacher’s requirements. Finally the level three involved preferences or conditions imposed by teachers with respect to requirements in level 2. After the initial feasible population is created by assigning each teacher to a a fixed position without exceeding teaching hours, we calculate the fitness and objective function. After the genetic operators are applied, the first stage is evaluation, by selecting the best fit population from two individual timetable. The crossover is applied which recombines the two best fit timetables to create an offspring with better fitness values. Finally, it applies the mutation operator, which takes the genes in offspring generation and swaps with another gene in the same offspring. Finally, before termination the system applies the filter algorithm which eliminates infeasible solutions. The filter algorithm captures all infeasible solutions generated by the genetic operators, returns feasible solutions by eliminating them.

The results generated from their approach also proved to be successful, as it was able to solve the high school timetabling problem.

2.4 Evaluation of Simulation Annealing, Tabu search and Genetic Algorithm

During the evaluation of the research work proposed by Colorni et.Al [22], we stated it was "successful". On the contrary, this was more than successful, this was excellently brilliant. The success of their system led to our decision in choosing GA to solve the small primary school timetable. In their work, they compared their approach with SA and TS, which produced better results. This led to our investigation more on genetic algorithm, where we discovered lots of research support GA. Compared to SA and TS the GA cannot be easily fooled by local optima, it gives the flexibility to users, and it always generates a solution. Where as in SA if the temperature is cooled too quickly, it generates infeasible solutions, and TS falls into local optimization.

2.5 Summary

In this chapter, we provided a brief description of meta heuristic methods used to solve optimization problems. We described simulated annealing, tabu search and genetic algorithm. After providing descriptions of what the algorithm entails, we provided two academic works, and described how these works used the algorithms to solve timetabling problems. We gave a review on each work listed in the literature, finally we compared all three algorithms with respect to research works. With evidence provided in the literature, we made decision as to why we selected the Genetic Algorithm.

3 Chapter 3: Requirement Analysis

This chapter contains detailed explanation of the system analysis and requirements. In this section, we provided the functional and non-function requirements, algorithmic functionalities, legal and ethical related issues and finally we stated the testing and evaluation carried on the system.

3.1 Problem Description:

In this section of the report, we shall explain the Primary School timetabling problem (PSTP). The (PSTP) can be regarded as simplistic taking into consideration its task of assigning subjects to classes. Despite the simplicity of the model, it turns out to be an accumulation of very complex models, once broken down, which we would take into consideration as we proceed.
First of all, it should be well noted that in the educational sector, there are three basic categories of timetabling problems, which include the school timetabling, course timetabling and examination timetabling. Our area of concentration is on the school timetabling problem. We shall introduce the general terminologies in timetabling problems, which perform the task of creating a weekly course timetable for students and teachers. To create a feasible course timetable, the timetabler, has the task of providing adequate and balanced workload for each class. This aspect is called the Balanced Academic Curriculum Problem. Aside from this aspect of timetabling, there is another which also has to be considered, which is the Class Selection Problem (CSP). In every primary school a set of predefined subjects for each class has to be taken into consideration. The task of the CSP is it assigns specific subjects to a particular group of students. Given adequate understanding concerning these sections we would like to further explain in this section, the school timetable

3.2 School Timetabling Problem

We define the following elements to the problem.
A set of teachers T = {ti,........tn}
A set of Classes C = {ci,.........cn}
A set of timeslots P = {pi,........pn}
A set of teaching hours H = {hi,.......hn}
A set of subjects S = {si,..........sn}
Given the premises, this problem can be described as a list of teacher's T teaching a subject S in a specific class C at a specific period P not exceeding their weekly teaching hours H. While identifying the problem, we based our research on the typical Nigerian Primary

School. Aside from this being our medium of investigation, our work in this paper can also be applied to other possible different instances of timetabling problem [8]. For an instance a feasible timetable(STP) the results generated would regarded as a consequence of Konig's theorem. But in situations where constraints are added to the model, automatically results in an NP Complete feasible timetable. These constraints are divided into hard and soft constraints, which we shall explain in the next section.

3.3 Requirements:

3.3.1 Functional System Requirements

Functional requirements are the requirements that the system must perform. In most literature, in reference to timetabling, these requirements are divided into hard and soft constraints. While developing, we decided to focus on the most important constraint which is the Hard constraint. Because a violation of just one of the hard constraints listed shall result in an infeasible timetable.

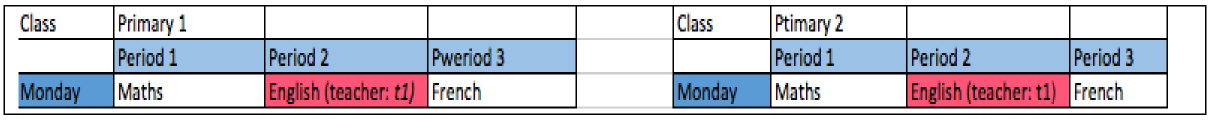
3.3.1.1 Primary School: Hard Constraints

To produce an optimal timetable, it is mandatory for the system to satisfy all constraints listed below. These include:

3.3.1.1.1 Clash Free Timetable:

This subsection represents the types of clashes which shall be avoided by application. They are two very important clashes that occur in most timetable, which includes the teacher and subject overlap.

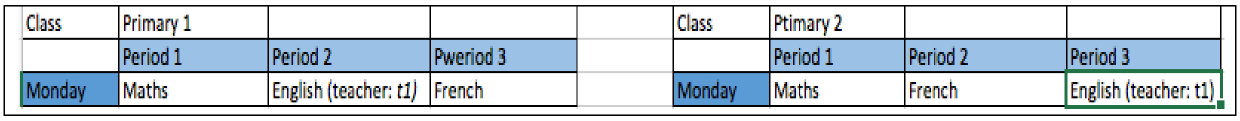
3.3.1.1.1.1 Teacher Clash



Class	Primary 1		
	Period 1	Period 2	Pweriod 3
Monday	Maths	English (teacher: *t1)*	French

Class	Ptimary 2		
	Period 1	Period 2	Period 3
Monday	Maths	English (teacher: t1)	French

Figure 2: Teacher Clash



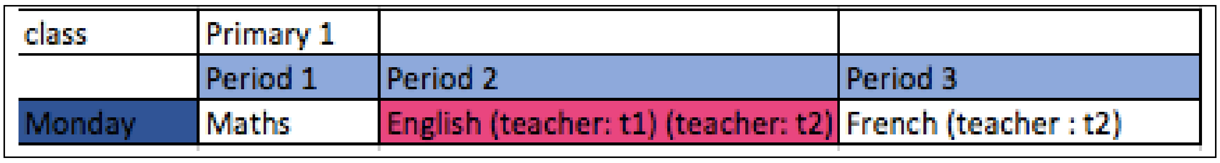
Class	Primary 1		
	Period 1	Period 2	Pweriod 3
Monday	Maths	English (teacher: *t1)*	French

Class	Ptimary 2		
	Period 1	Period 2	Period 3
Monday	Maths	French	English (teacher: t1)

Figure 3: Teacher Scheduled Correctly

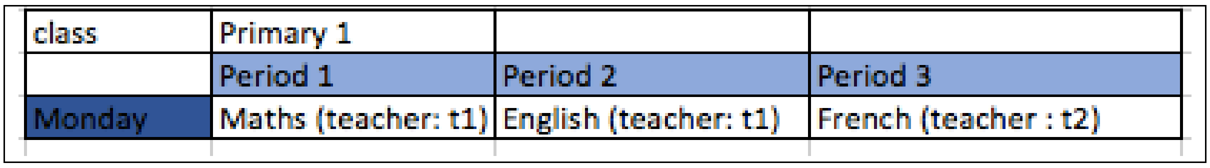
A teacher is not allowed to teach two different classes at simultaneous times. For instance, teacher t_1 cannot teach class c_1 as primary 1 and c_2 as primary 2 at the same time. This causes a conflict in the timetable as the teacher t_1 cannot be in two different classes at the same time teaching different subjects. As indicated in Figure 3, this assignment is feasible as the same teacher t_1 can teach in c_1 and teach in c_2 at p_3 as the third period.

3.3.1.1.1.2 Teacher Overlap



class	Primary 1		
	Period 1	Period 2	Period 3
Monday	Maths	English (teacher: t1) (teacher: t2)	French (teacher : t2)

Figure 4: Teacher overlap

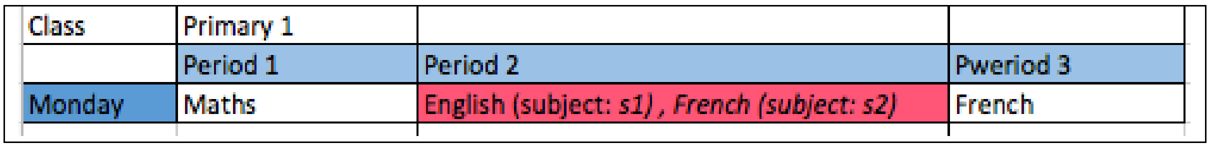


class	Primary 1		
	Period 1	Period 2	Period 3
Monday	Maths (teacher: t1)	English (teacher: t1)	French (teacher : t2)

Figure 5: Teacher Scheduled Correctly

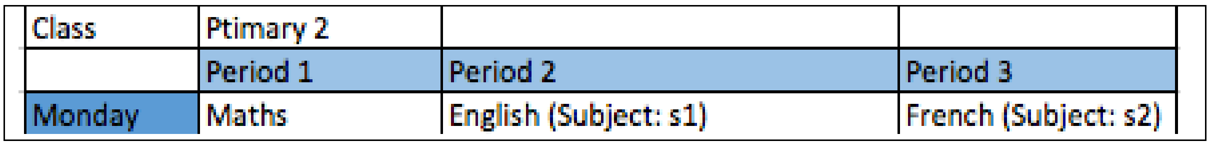
Only one teacher can teach a class at a time. In every primary school, there must be at most one teacher teaching a subject. Figure 4 illustrates two English teachers t_1 and t_2 assigned to the same class at the same period. Where as Figure 5 shows a balanced spread of teachers across periods without overlap. Therefore Figure 5 is the best solution to our timetabling problem.

3.3.1.1.1.3 Subject Overlap(Clash):



Class	Primary 1		
	Period 1	Period 2	Pweriod 3
Monday	Maths	English (subject: *s1)* , *French (subject: s2)*	French

Figure 6: Subjects Clash

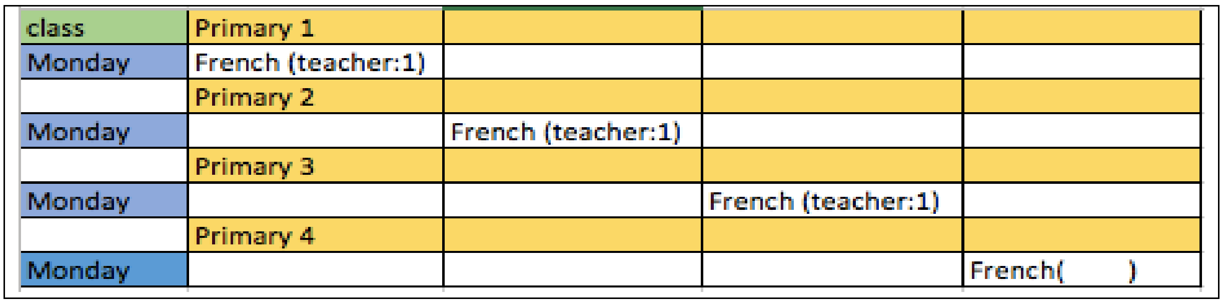


Class	Ptimary 2		
	Period 1	Period 2	Period 3
Monday	Maths	English (Subject: s1)	French (Subject: s2)

Figure 7: Subjects Scheduled Correctly

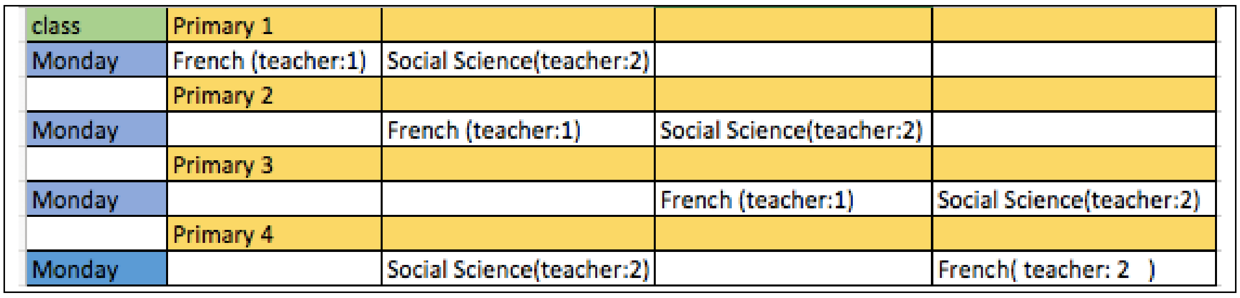
At most one subject can be taught in a class at a specific time. A class c_1 cannot take two subjects s_1 and s_2 at the same time p_1 .For instance, c_1 cannot take French s_1 and English s_2 at the same time p_2 . Figure 7 represent a feasible timetable, as the subjects are allocated across different periods $p_2$and p_3.

3.3.1.1.2 Teaching Hours



class	Primary 1			
Monday	French (teacher:1)			
	Primary 2			
Monday		French (teacher:1)		
	Primary 3			
Monday			French (teacher:1)	
	Primary 4			
Monday				French( )

Figure 8: Teacher Hours



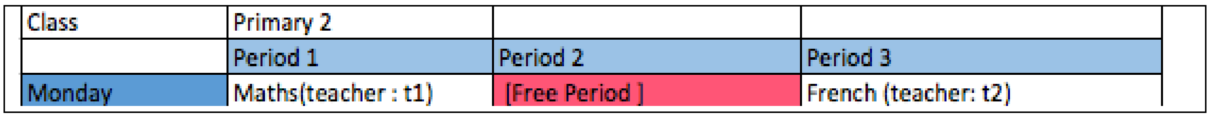
class	Primary 1			
Monday	French (teacher:1)	Social Science(teacher:2)		
	Primary 2			
Monday		French (teacher:1)	Social Science(teacher:2)	
	Primary 3			
Monday			French (teacher:1)	Social Science(teacher:2)
	Primary 4			
Monday		Social Science(teacher:2)		French(teacher: 2)

Figure 9: Teacher Hours Scheduled Correctly

The timetable assignment should not exceed working hours for specific teacher. In Figure 8, we have a French teacher t_1 with 3 working hours, assigned to 4 classes . The system does not assign t_1 to the last class, because there are no more hours left.Figure 9, is a feasible timetable, which assigns teacher t_1 in the French slot of primary 4.

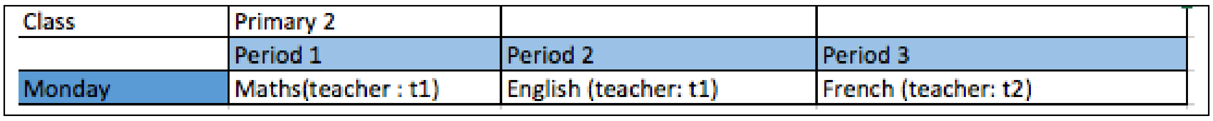
3.3.2 Primary School: Soft Constraints

3.3.2.1.1 Free Periods(timeslots):



Class	Primary 2		
	Period 1	Period 2	Period 3
Monday	Maths(teacher : t1)	[Free Period]	French (teacher: t2)

Figure 10: Free Period

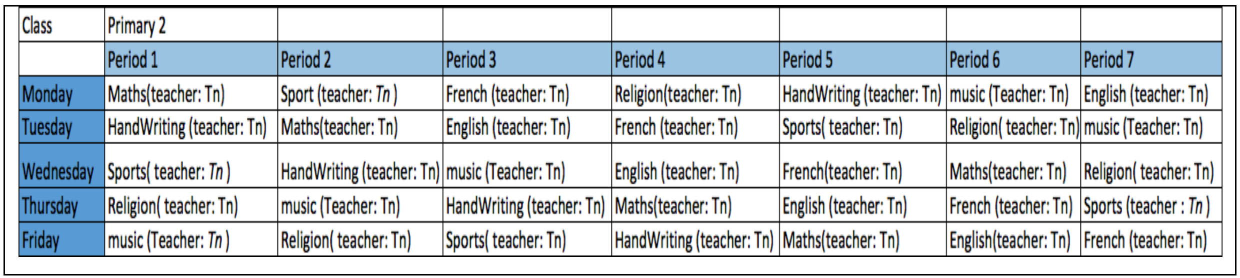


Class	Primary 2		
	Period 1	Period 2	Period 3
Monday	Maths(teacher : t1)	English (teacher: t1)	French (teacher: t2)

Figure 11: Scheduled Correctly

Balanced or spreading subjects out over the periods. In here the BACP in the system should be correctly balanced by the timetabler. In our timetable, we have 35 slots for each class, therefore, the timetabler has to balance the hours of each subject in each class such that they are equal to 35.Figure 10 is an illustration of what happened when the curriculum is not evenly balanced. A solution is displayed in **Error! Reference source not found.**, which evenly distributes all subjects to different timeslots for that specific class.

3.3.2.1.2 Class Meetings:



Class	Primary 2						
	Period 1	Period 2	Period 3	Period 4	Period 5	Period 6	Period 7
Monday	Maths(teacher: Tn)	Sport (teacher: *Tn*)	French (teacher: Tn)	Religion(teacher: Tn)	HandWriting (teacher: Tn)	music (Teacher: Tn)	English (teacher: Tn)
Tuesday	HandWriting (teacher: Tn)	Maths(teacher: Tn)	English (teacher: Tn)	French (teacher: Tn)	Sports(teacher: Tn)	Religion(teacher: Tn)	music (Teacher: Tn)
Wednesday	Sports(teacher: *Tn*)	HandWriting (teacher: Tn)	music (Teacher: Tn)	English (teacher: Tn)	French(teacher: Tn)	Maths(teacher: Tn)	Religion(teacher: Tn)
Thursday	Religion(teacher: Tn)	music (Teacher: Tn)	HandWriting (teacher: Tn)	Maths(teacher: Tn)	English (teacher: Tn)	French (teacher: Tn)	Sports (teacher : *Tn*)
Friday	music (Teacher: *Tn*)	Religion(teacher: Tn)	Sports(teacher: Tn)	HandWriting (teacher: Tn)	Maths(teacher: Tn)	English(teacher: Tn)	French (teacher: Tn)

Figure 12: Class Meeting assignment

This constraint is concerned with the spread of class-teacher-subject allocation throughout the week. For instance, our approach, there are 7 periods (timeslots) for each day in a week. A week is composed of 7 days with 5 teaching days and 2 resting days. Therefore, for each week, we generate 35 timeslots, we assign each subject hour for each subject being taught for a specific class. The sum of all subject hours should be proportional to total number of timeslots per week.

3.3.3 User Requirements

3.3.3.1 Administrative Functionalities

In our system, the administrator is regarded as the “timetabler”. The timetabler is one who performs the low-level operations in the system. The admin interacts with the system by performing operations such as:

a. Insert, update and delete user records in the database. The users of the system shall be the timetabler and the teachers. The teacher provides user information, which is timetabler places in the system record. The timetabler shall also have authority to change pre-existing user record, and perform deletion if necessary.
b. The timetabler shall be responsible for inserting data into the balanced curriculum. This is very important as it is required for the development of best optimal solution
c. After populating data in the balanced curriculum, the timetabler shall also be responsible for generating the optimal timetable.
d. The timetabler shall import pervious optimal timetable, which shall be used by the genetic algorithm to produce a new feasible timetable, with any change made in the curriculum.

3.3.3.2 Teacher Functionality

In our system, the teachers do not have as many functionalities as the timetabler. There are just two functionalities for the teachers, which include:
a. The system shall store all teacher information.

b. The system shall display the school timetable for every registered teacher.

3.3.4 Non-Functional Requirements

3.3.4.1 Usability:

The system was designed to be relatively easy for all users. It shall make use of html5, css and bootstrap to create a beautiful user interface. For instance, to activate the algorithm used to create the timetable, the timetabler has to push a button.

3.3.4.2 Security:

The system performs user validation at the login interface. It checks the system to recognize the teachers and the timetabler. Within the system backend, the system was hard coded in such a way that it prevents sql injection from malicious intruders.

3.3.4.3 Performance and speed:

We consider the performance and speed of the system as to be of high priority. Because of the nature of the algorithm, it data inserted into is high, it becomes more complex for the system to find a solution. This cannot be regarded as a limitation of system, as we initially stated in chapter 1, this is regarded as NP hard.

3.3.5 Algorithm functionalities

We developed a system that solves the timetabling problem using the Genetic Algorithm(GA). The system is divided into two phases, which include the academic curriculum and genetic Algorithm. The system was developed to be an interactive system as it requires the number of hours per subject from the user, which it uses to generate the timetable. The second phase is applied after provision of hours is the timetable. First we provide the actual timetable allocation space ‘timetable slots’. We create class group which is a three-dimensional array of the class-teacher-subjects. We create the population of the gene that is equal to the timeslots. We assign allele (values) for each gene which is each individual class-teacher-subject group. We assign the array of genes as chromosomes and calculate the fitness function. We find a best combination from the fitness function which is the number of possibilities of selecting a teacher, a class and a period. A crossover is applied which combines the fitness of two parent chromosomes to create an offspring of best match. Finally, mutation, which searches for the infeasible timetable, performs , makes swaps to change their position of chromosomes in the timetable. The process is repeated till the system is able to find the best or close to optimal solution for the problem.

3.4 Requirement Analysis

3.4.1.1 Use Case Diagram

A use case diagram is employed to describe the system behavior. In this diagram, we represent the system functionalities and how they are used by the users of the system Figure 13 represents a high level description of the system functionalities.

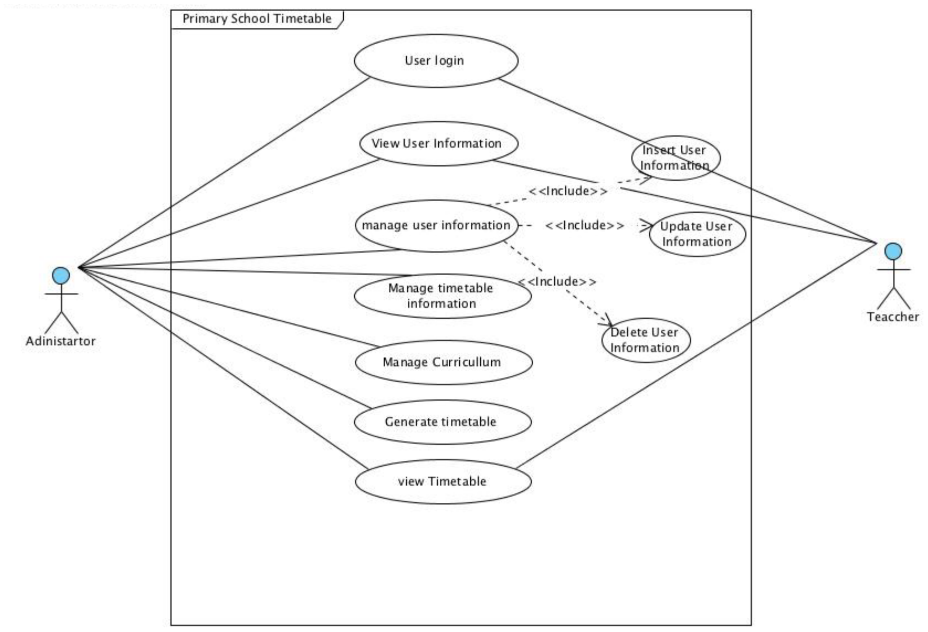


Figure 13: Use Case Diagram

Actor	**Action:**	**Description**
All user	User logs into the system	The user is authenticated after login details have been provided. After authentication, each user is directed to their specific page.
Administrator	View user details	The system displays all user specific information.
Administrator	Manage user Information	After authentication, the system displays all user records, which timetabler must right to modify.
Administrator	Manage timetable information	The admin can manage all information regarding the timetable system. In this use case, the administrator performs actions such as assignments of teachers to a variety of subjects.

Administrator	Manage Curriculum	The administrator interacts with the system by providing a certain number of subject hours for each class
Administrator	Generate Timetable	After providing information for the curriculum, the administrator generates the timetable. This timetable is generated using the genetic algorithm.
All users	View timetable	The timetable is made available to all authorized users after its creation.

3.5 Ethical, Professional and Legal Issues

In a manner that conforms with the “code of good practice” by the British Computer Society (BSC), no ethical nor professional rules were broken during the implementation of this project. All researched works used in this projects were correctly referenced to avoid plagiarism. No sensitive topics such as race, color, disability or age was discussed in our work. We followed the standard rules and regulations provided by the University of Sheffield, we kept all research work current and up to date. We also made sure that all there were no false information used, most of the referenced works were based on true academic research.

3.6 Project Testing and Evaluation

To evaluate the system performance and compliance with our specific domain, system testing was performed as stated in the section 3.1. We decided to generate test data, that was not based on a real primary school. All system testing listed was performed before final submission to the University of Sheffield’s department of Computer Science.

3.6.1.1 Black-box testing:

At every development phase, this was continuously applied to make sure all system user functionalities were all working correctly. We testing the login, user pages, navigation, timetable and curriculum audit.

3.6.1.2 White-box testing:

White box testing was applied to make sure the algorithm used, solved the problem correctly. We investigated the internal logic and structure of the code, to make sure that it was easy for any developer to understand

3.6.1.3 Requirement Testing:

Test was done to ensure that the system met and satisfied all domain requirements. More information about this subsection is fully described in chapter 6.

3.7 Summary

In this chapter, we defined the system functionalities, which include functional requirements, nonfunctional requirements and algorithmic functionalities. The functional requirements include the system requirements which must satisfy all hard constraint and user requirements. These functionalities shall be performed by the system. Nonfunctional requirements which include usability, security, performance and speed were described with respect to our system. We ended the system requirements by explaining how the genetic algorithm performs its operation in reaching a feasible timetable. For the requirement analysis, we provided a use case diagram, with detailed use case description that described system functionalities. In the aspect of ethical, professional and legal issues, we gave a description regarding specific BSC code of practice we adhered to. Finally, to end this chapter, we listed three evaluation of system performance, which included the black-box, white box and requirement testing.

4 Chapter 4: Design

In this section of the report we shall describe with appropriate justification the system design of the project. We shall provide the software development lifecycle, ERD diagram and associated UML diagrms.

4.1 Software Development Life Cycle

The software development life cycle (SDLC) can be described as a framework used to define tasks performed at each step of the software development process. We applied the earliest SDLC approach which was the simple waterfall model, for the system development. The waterfall model is regarded as a linear-sequential life cycle model that complete the previous stage before moving to the next stage. The approach enabled us generate a system that conforms to as system requirements. The water fall model involves six stages, which include requirement analysis, system design, implementation, testing, deployment and maintenance. As described in chapter 3, we obtained all system requirements, conforms with the requirement analysis phase. All system requirements obtained, were obtained and successfully met. The system design we made use of UML diagrams to discuss the structure of the software, which we would provide more detail in this chapter. Based on system requirements, which stated that we create a system that produces a feasible timetable, we performed implantation at this phase. Integration and testing was carried after system implementation, as we tested each unit of the system, especially the algorithm which generated successful results. The fifth stage, which is deployment, was not carried out, as we could not deploy the system in a real primary school setting. For the last stage, if the system is eventually deployed, we shall make further improvements to the version of our current system.

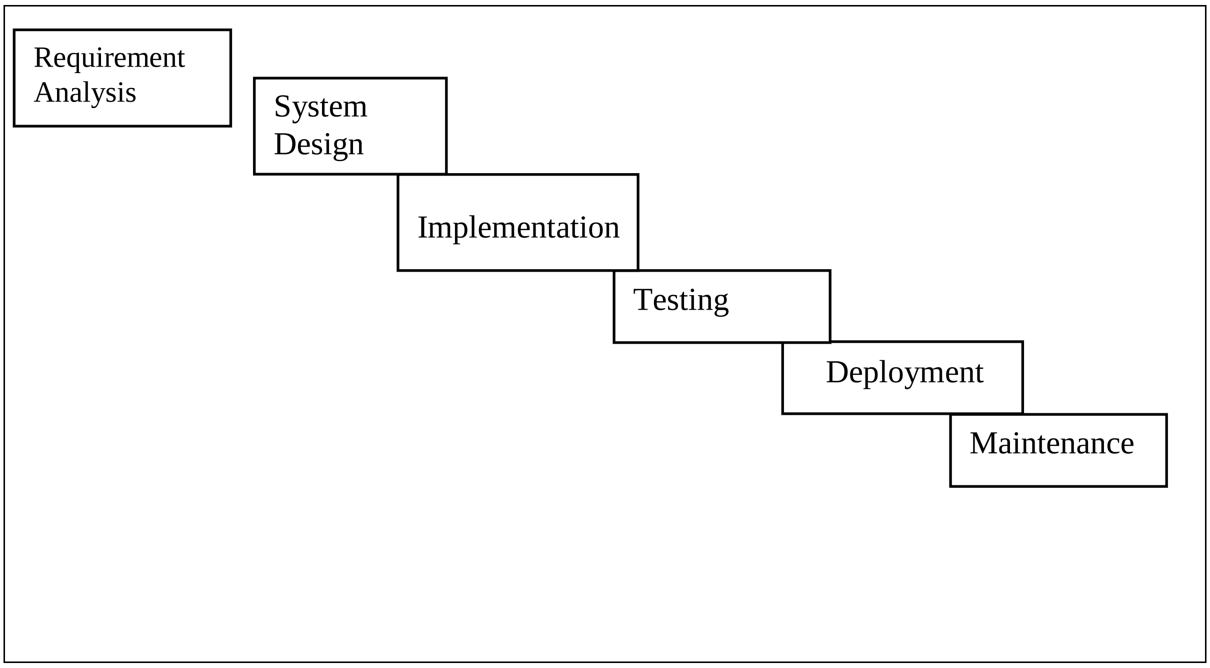


Figure 14: Water Fall Model

4.2 ERD Diagram

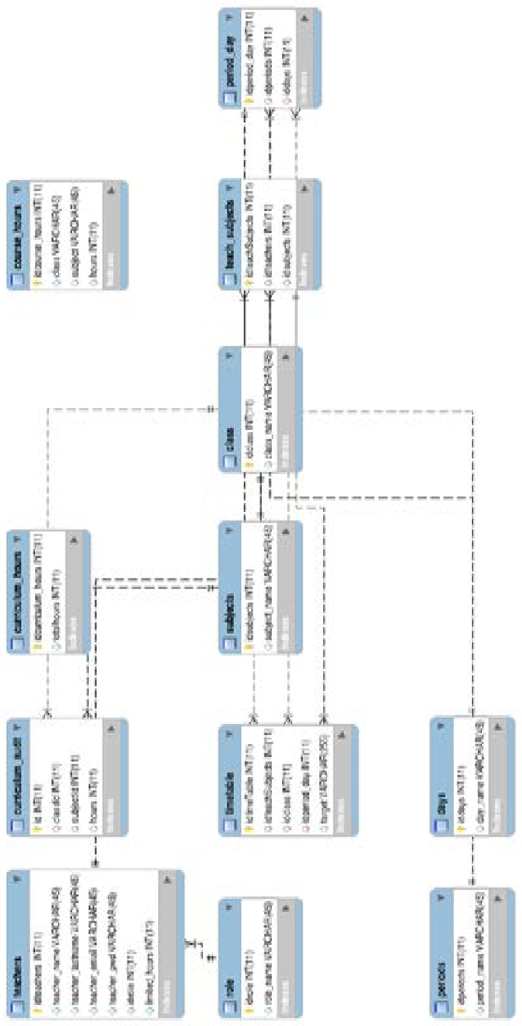


Figure 15: Primary School Entity Relationship Diagram

4.2.1 Database Design

Since our proposed application is a based-on web development, it is impericable to use a database to store all records. In this section of the report, we shall describe a normalized database structure that was used in the production of the timetable. We shall provide an image of each database table and additional information for the better understanding on how they are related to one another. Along with this section, we shall provide and Entity Relationship Diagram (ERD), which would shall represent the relationships of each database table.

4.2.2 Class Diagram

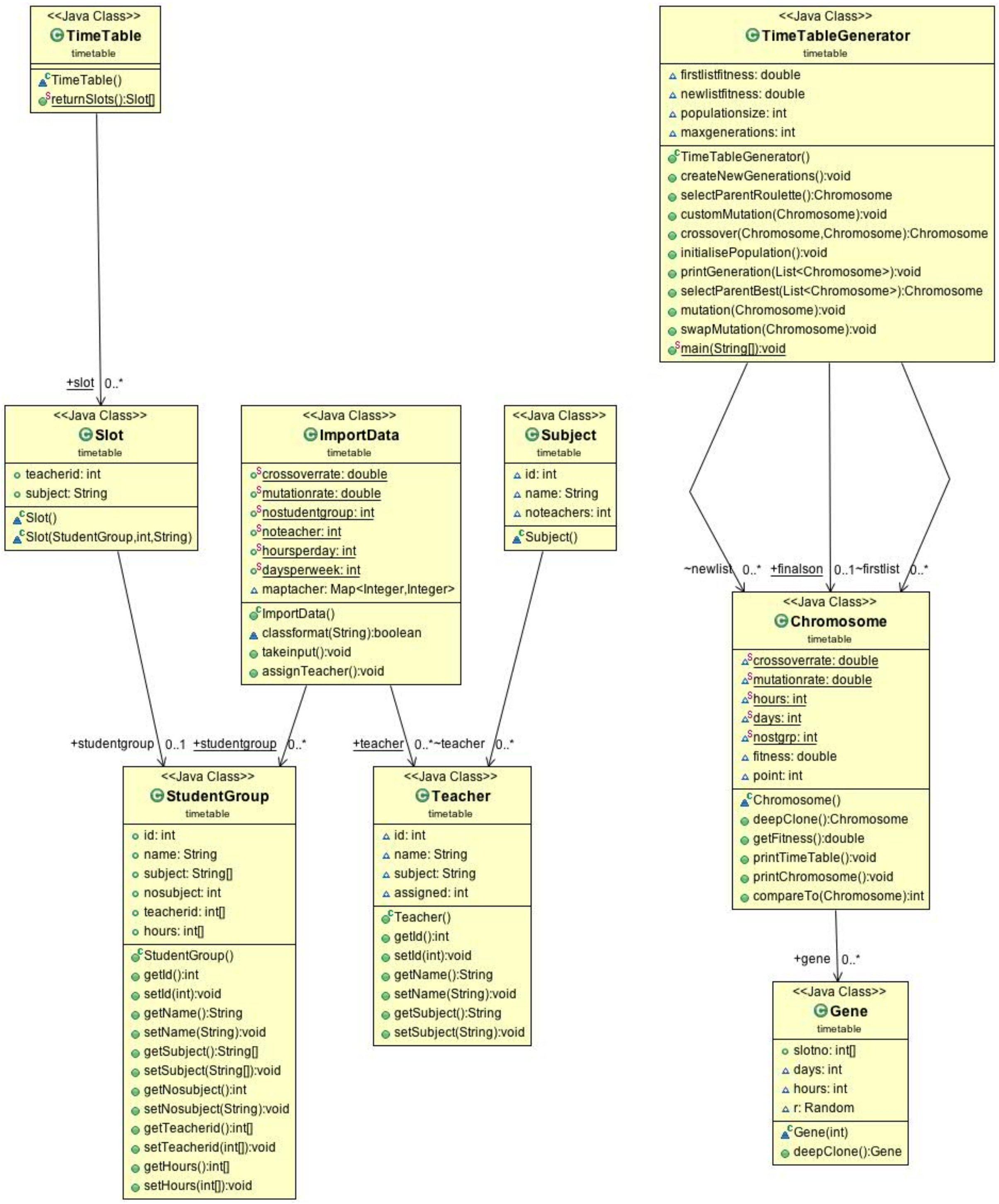


Figure 16: Class Diagram

4.3 Algorithm Design and Mathematical Modeling

To improve reader understanding, we provided a class diagram, which would be very useful to explain the systems algorithmic parameters. In this section, we provide the basic notation, chromosome representation, constraints, genetic operators and fitness function.

4.3.1.1 Notation

We define the following elements to the problem.
A set of teachers T = {ti,........tn}
A set of Classes $C = \{ c_i \dots \dots c_n \}$
A set of timeslots $P = \{ p_i \dots \dots p_n \}$
A set of teaching hours $H = \{ h_i \dots \dots h_n \}$
A set of subjects $S = \{ s_i \dots \dots s_n \}$
A set of groups $G = \{ g_i \dots \dots g_n \}$
A slot SLOT= [S][T]
A Gene = [Slot][Slot][Slot]……
A Chromosome = [Gene][Gene][Gene]
A population = [Chromosome]
Given the premises, this problem can be described as a list of teacher's T teaching a subject S in a specific class C at a specific period P not exceeding their weekly teaching hours H. Each assignment is regarded as a timetable slot, which is the characteristic value of the gene. A chromosome which is an array of 35 genes for one class.

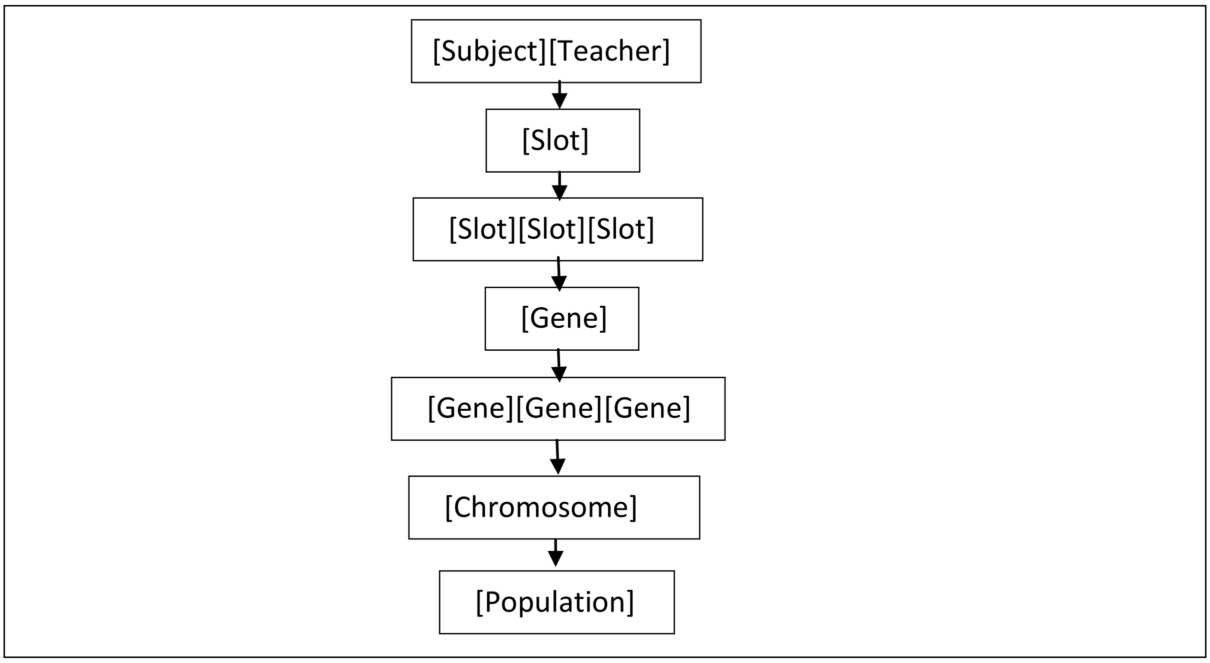


Figure 17: Population

4.3.1.2 Genetic Representation

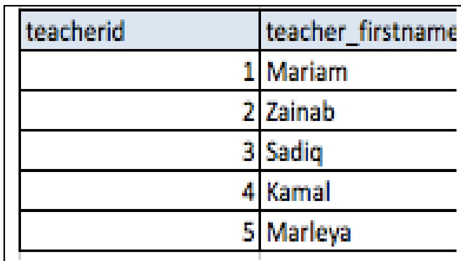
This section defines actual gene values represented in computer space. For the system to process the data, the problem must be represented in such a way that the computer could understand and generate a solution. Genetic representation shall be in random permutation encoding, of each gene value

1	4	2	3	7	6	8	5	9	10

Figure 18: Genetic Representation

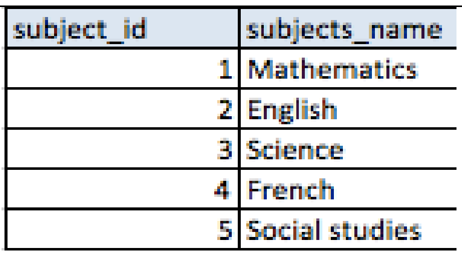
4.3.1.3 Phenotypic Representation

This sub-section shall represent the population in the actual real world, therefore in the user interface. For real world representation, its shall be a two-dimensional matrix of time and days representing the teachers and subjects being taught. We access the genetic data by unique identifiers(id), then represent values of each as gene allele (value) in real world. To describe the phenotypic representation, we shall represent database tables, which we used to retrieve the data.



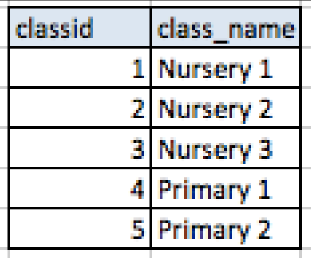
teacherid	teacher_firstname
1	Mariam
2	Zainab
3	Sadiq
4	Kamal
5	Marleya

Figure 19: Teacher Table



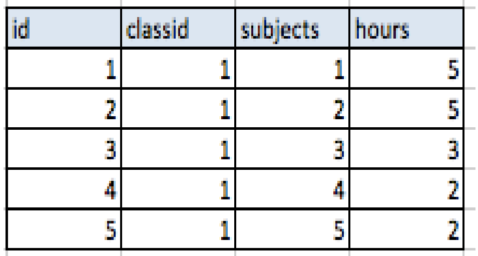
subject_id	subjects_name
1	Mathematics
2	English
3	Science
4	French
5	Social studies

Figure 20: Subjects Table



classid	class_name
1	Nursery 1
2	Nursery 2
3	Nursery 3
4	Primary 1
5	Primary 2

Figure 21: Class Table



id	classid	subjects	hours
1	1	1	5
2	1	2	5
3	1	3	3
4	1	4	2
5	1	5	2

Figure 22: Academic Curriculum Table

Between all four database tables considered in this section, we shall define relationships between them, for finding a solution.

4.3.1.3.1 Teacher Table:

This table consists of all recorded information for each teacher. As the teacher provides information for the timetable during registration, the timetable inserts each specific data into this database table. Each teacher has a unique id, which is the primary key, first name, list name, email, password and roleid. Although we did not represent all columns used, we considered to display only the most important. To view the full database table, refer to the ERD diagram.

4.3.1.3.2 Subject Table:

This table consists of all records of subjects being taught in a primary school according to the academic curriculum. Each subject has a unique identifier, which we shall use to ensure normalization. It is represented in Figure 20.

4.3.1.3.3 Class Table:

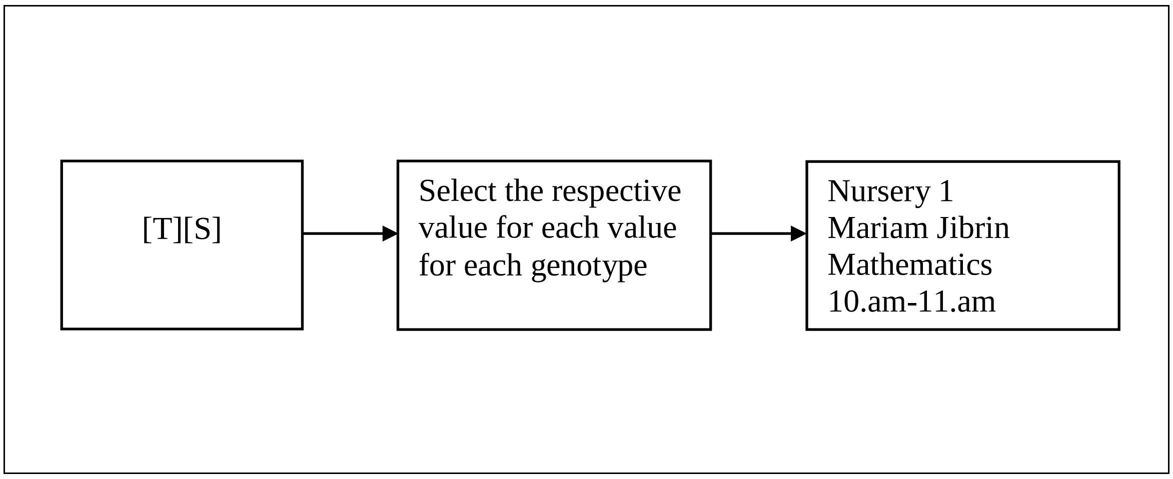
This table consists of all classes in the primary school, as illustrated in Figure 21.

4.3.1.3.4 Academic Curriculum Table:

This table represents the subjects and corresponding hours for each individual class that conforms with the national curriculum. This table represents a relationship between the subject and class table. It I represented in

4.3.1.4 Genotype and Phenotype Transformation

In this section, we shall explain the transformation process from the genotypic to phenotypic representation. Given the tuple T, C, P, H S, G, we represent the relationship and connections in the database. The population shall be an array of chromosomes in a matrix, which is the timetable. The matrix, which is the teacher and the subject [T][S] for each class [C] at time [P] Therefor the transformation involves "decoding", that is retrieving unique identifiers (id) representing them as permutation encoding in genotypic space. Thereafter converting "encoding" it to their phenotypic value as the real-world solution. The process of decoding is repeating till an optimal solution is reached.


[T][S]
Select the respective value for each value for each genotype
Nursery 1
Mariam Jibrin
Mathematics
10.am-11.am
Figure 23: Transformation Process from Genotype to Phenotype.

4.4 UML Deployment Diagram

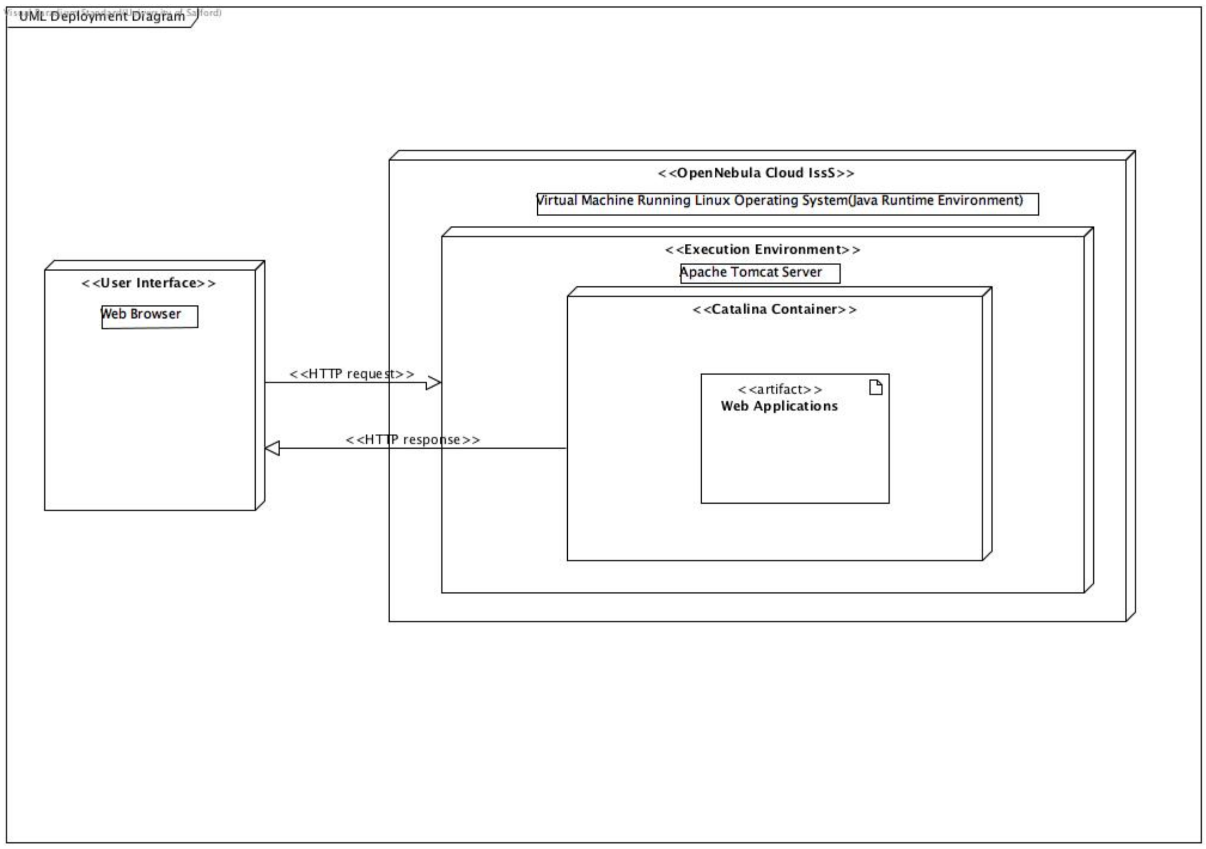


Figure 24: UML Package Diagram

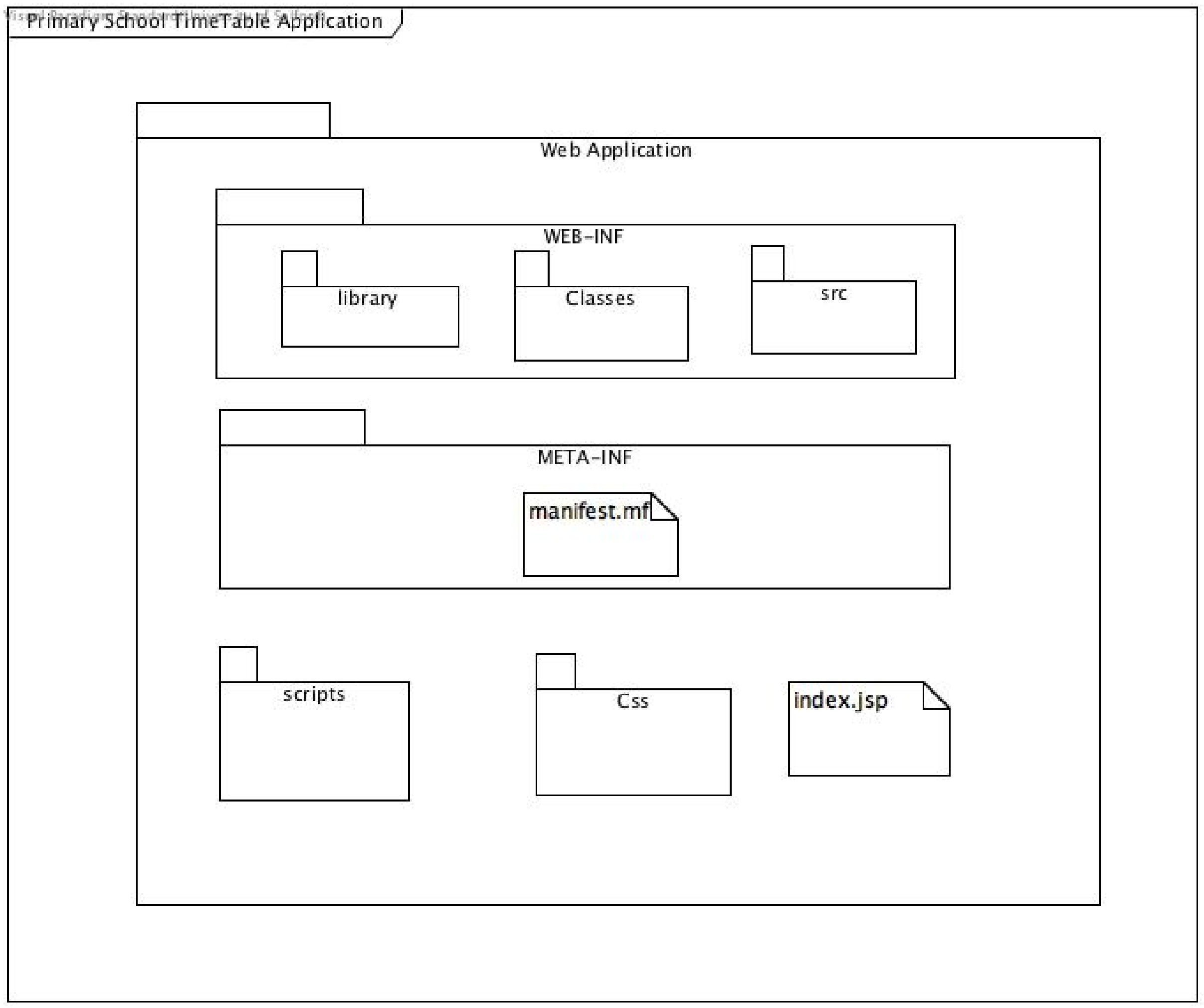


Figure 25: UML Deployment Diagram

4.5 Technology(stack)

In this subsection, we shall discuss the technology stack used in the implementation of this project. For the purpose of user interaction with the system, we define our front-end technology stack with JavaScript, html5, W3-css, java server pages and bootstrap. As for the system's back-end, we proposed to use apache Tomcat server, java, java servlet and MySQL workbench.

4.5.1 Front End Technology Stack.

We initially considered using hypertext markup language(HTML5) for rendering contents on the World Wide Web. We also employed using JavaScript because it enabled us use special features such as json, ajax and jQuery. Combined with html elements, it is easy to operate and reduces internet transportation. To create a beautiful user interface that improves the systems ease of use, we combined personalized W3-css and bootstrap. Finally, we used the Java server pages (jsp) in combination with java servlets, they were used to handle the business logic and support complex tasks. All these stacks listed above were used to develop a rapid and simplified web-based timetabling application.

4.5.2 Back End Technology Stack

For our proposed back end technology stack, we used java, MySQL and apache tomcat to perform all business logic for the application. Our choice for using java as a programming language is due to its simplicity and ease of use for the developers of the application. Aside from this, our main reason, was because its platform-independent. We needed to develop an application that could be executed on many computer systems. For the database, we used MySQL workbench as it is suitable in working with large amounts of data, and performs complex queries. Finally, we deployed our system on apache tomcat server 8.5 as a container for our application. Compared to XML, this is more effective and convenient.

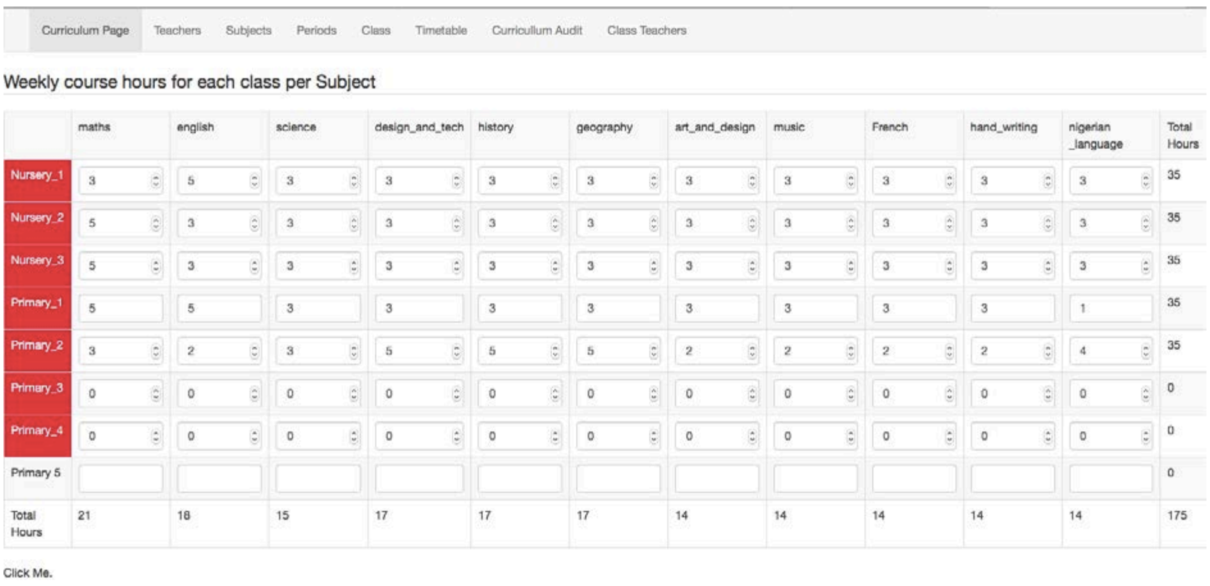
4.5.3 Summary

In this chapter, we provided detailed explanation with regards to the system design. We started by giving a brief description of the software development lifecycle being used, the algorithm and mathematical design as well as the technology stack. With different section of the report, we provided suitable diagrams to provide more adequate understanding. In the algorithmic and mathematical design, we listed and explained the mathematical notations, genetic and phenotypic representation, as well as its transformation process. We also provided a UML deployment and package diagram which we used in relation to the technology stack.

5 Chapter 5: Implementation and Testing

In this section, we shall describe the implementation process in our system. The main features of our application includes, the balanced curriculum and timetable generation. We shall describe how the balanced curriculum works with respect to an actual school curriculum. We shall also provide code snippets that satisfy the constraints in chapter 3.31. Finally, we shall describe how we incorporated the genetic algorithm in our system

5.1 Balanced Academic Curriculum



Weekly course hours for each class per Subject

	maths	english	science	design_and_tech	history	geography	art_and_design	music	French	hand_writing	nigerian _language	Total Hours
Nursery_1	3	5	3	3	3	3	3	3	3	3	3	35
Nursery_2	5	3	3	3	3	3	3	3	3	3	3	35
Nursery_3	5	3	3	3	3	3	3	3	3	3	3	35
Primary_1	5	5	3	3	3	3	3	3	3	3	1	35
Primary_2	3	2	3	5	5	5	2	2	2	2	4	35
Primary_3	0	0	0	0	0	0	0	0	0	0	0	0
Primary_4	0	0	0	0	0	0	0	0	0	0	0	0
Primary 5												0
Total Hours	21	18	15	17	17	17	14	14	14	14	14	175

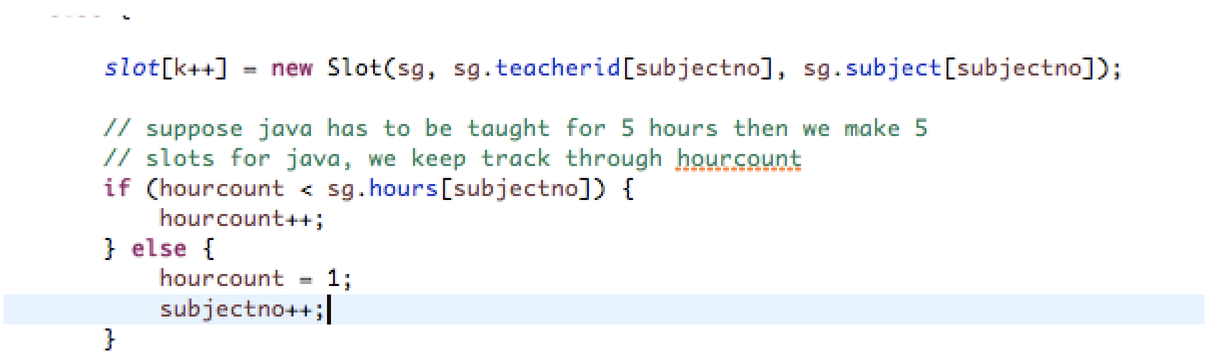
Figure 26: Academic Curriculum

Amongst the three educational sectors which include primary, secondary and university, they all require a fundamental aspect called the “curriculum”. A curriculum is a set of courses for a specified selection of students. According to the national curriculum of England(www.gov.uk), it states that:

“Each state funded school must offer a curriculum which is balanced and broadly based”.

The term “balanced” and “broadly based” is our main focus of consideration. The task of the Academic curriculum is the assignments of subjects to teaching periods such that the prerequisites, and subject course load must be balanced. The academic Curriculum page is populated by the timetabler; he/she inserts the amount of hours with which the subject is to be taught. In our application, we defined a set of seven period and five days, therefor the timetabler, has to evenly distribute 35 hours for all subjects in a class, unless free periods may be present.

5.2 Constraints

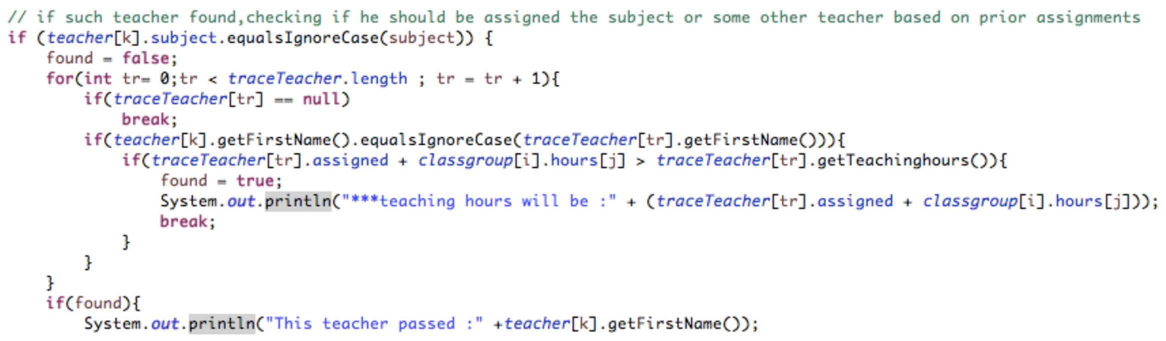


```
slot[k++] = new Slot(sg, sg.teacherid[subjectno], sg.subject[subjectno]);

// suppose java has to be taught for 5 hours then we make 5
// slots for java, we keep track through hourcount
if (hourcount < sg.hours[subjectno]) {
    hourcount++;
} else {
    hourcount = 1;
    subjectno++;
}
```

Figure 27: Subject Clash and Free periods

5.2.1 Subject Clash and Free periods

As described in section 3.3.1.1.1.3, which describes the subject clashes that may occur in a timetable. We satisfied this by retrieving subject-hours data from curriculum data table, count the number of hours for each subject. Then assign for each subject group the subjects , count and assign the amount of hours for that subject in that subject group



```
// if such teacher found,checking if he should be assigned the subject or some other teacher based on prior assignments
if (teacher[k].subject.equalsIgnoreCase(subject)) {
    found = false;
    for(int tr= 0;tr < traceTeacher.length ; tr = tr + 1){
        if(traceTeacher[tr] == null)
            break;
        if(teacher[k].getFirstName().equalsIgnoreCase(traceTeacher[tr].getFirstName())){
            if(traceTeacher[tr].assigned + classgroup[i].hours[j] > traceTeacher[tr].getTeachinghours()){
                found = true;
                System.out.println("***teaching hours will be :" + (traceTeacher[tr].assigned + classgroup[i].hours[j]));
                break;
            }
        }
    }
    if(found){
        System.out.println("This teacher passed :" +teacher[k].getFirstName());
```

Figure 28: Subject Teacher assignment

5.2.2 Subject Teacher assignment

Before assignment of teachers to specific classes, the system checks amongst the list of registered teachers, the actual subjects they were emploed to teach. If it searches and finds a candidate for that subject, it assigns a specific timeslot for that teacher in a specific class group. As assignment takes place, a very important aspect of consideration is the teachers working hours. The system keeps assigning teachers and subjects taught to a specific class without exceeding the teachers working hours.

5.3 Genetic Algorithm

1) Generate Random Initial Population
2) Calculate the fitness Value
3) Perform Genetic operators
  - **a.** Use Roulette Wheel Selection method, to gain the two best mother and father chromosome.
  - **b.** Use Single Point Crossover operator on the two parents to generate new offspring
  - **c.** Perform Swap mutation on the offspring
  - **d.** Finally evaluate the fitness of the offspring

4) Run till the best set of chromosomes are generated.

Figure 29: Genetic Algorithm

As stated in section 3.4, which describes the algorithmic requirements. In this section, we describe in the detail how the genetic algorithm was used in combination to generate an optimal solution to the timetabling problem. Just as in Figure 29, we listed out the genetic algorithm, we further explain every little detail with relation to our application.

5.3.1 Initial Population

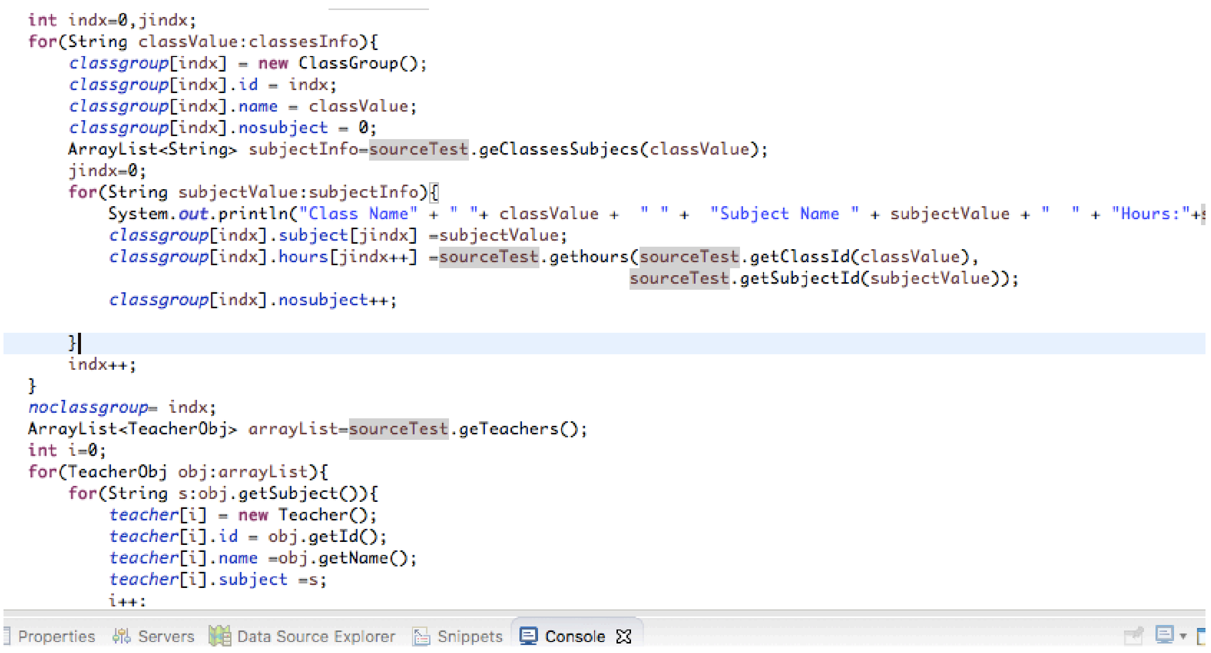
In our proposed method, we break down the initial population into three distinct stages, which include the slots, genes ad chromosome. Each of these stages, has to be satisfied before the next stage is invoked. In this subsection, we shall explain theses stages as the first instance needed to create initiate the algorithm.

5.3.1.1 Slots:

As described in the section 4.3.1.1of the genetic algorithm, each chromosome is an array of genes, with each of its gene having a unique value called the allele. In our approach, the allele shall be called the slots, as they represent the basic unit of a single character gene.

5.3.1.2 Class Group:

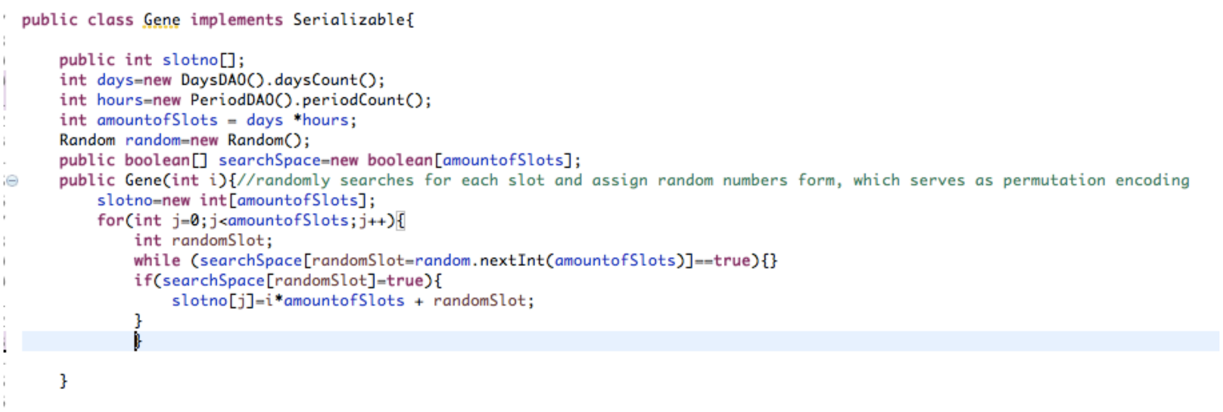
This class is a three dimensional array that stores subjects, hours and teachers. In logical terms, it is defined as [subjects][hours][teachers]. This is one of the most important aspect of the timetable, as it is used to create the allele for each gene. We have a classes called slots, which has two constructors, the first is a non-parameterized constructor, which is created for free periods. Free periods which are non-allocated timetable slot, for instance, the specified hours for that class is not equal to 35. The number 35 is generated from multiplying the amount of days which is 5 and the amount of periods which are 7. As for the second constructor, it shall take in three parameters, which is the class group, subjects and teachers



```
int indx=0,jindx;
for(String classValue:classesInfo){
    classgroup[indx] = new ClassGroup();
    classgroup[indx].id = indx;
    classgroup[indx].name = classValue;
    classgroup[indx].nosubject = 0;
    ArrayList<String> subjectInfo=sourceTest.geClassesSubjecs(classValue);
    jindx=0;
    for(String subjectValue:subjectInfo){
        System.out.println("Class Name" + " "+ classValue +  " " +  "Subject Name " + subjectValue + "  " + "Hours:"+
        classgroup[indx].subject[jindx] =subjectValue;
        classgroup[indx].hours[jindx++] =sourceTest.gethours(sourceTest.getClassId(classValue),
                                                    sourceTest.getSubjectId(subjectValue));
        classgroup[indx].nosubject++;

    }
    indx++;
}
noclassgroup= indx;
ArrayList<TeacherObj> arrayList=sourceTest.getTeachers();
int i=0;
for(TeacherObj obj:arrayList){
    for(String s:obj.getSubject()){
        teacher[i] = new Teacher();
        teacher[i].id = obj.getId();
        teacher[i].name =obj.getName();
        teacher[i].subject =s;
        i++;
```

Figure 30:Initialize class Groups

In Figure 30, here we initialize the a two dimensional array, here we define a first array that holds the values for all classes information. Class information are a counter which serves as an identifier, the class name and number of subjects. Secondly, we define a second array that hold the information of a specific class subject. This array stores information regarding the subjects and hours of that subject being taught in a specific class. To define all teacher properties, we have the assign the teacher id, teacher name, a list of all subjects taught by one teacher and their registered teaching hours.

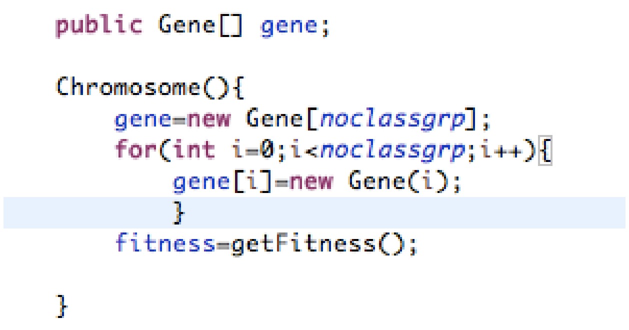


```
public class Gene implements Serializable{

    public int slotno[];
    int days=new DaysDAO().daysCount();
    int hours=new PeriodDAO().periodCount();
    int amountofSlots = days *hours;
    Random random=new Random();
    public boolean[] searchSpace=new boolean[amountofSlots];
    public Gene(int i){//randomly searches for each slot and assign random numbers form, which serves as permutation encoding
        slotno=new int[amountofSlots];
        for(int j=0;j<amountofSlots;j++){
            int randomSlot;
            while (searchSpace[randomSlot=random.nextInt(amountofSlots)]==true){}
            if(searchSpace[randomSlot]=true){
                slotno[j]=i*amountofSlots + randomSlot;
            }
        }

    }
```

Figure 31: Gene

5.3.1.3 Gene:

In the field of biology, a gene is described as the basic physical and functional unit of heredity. In this situation, from the perspective of computer science, the gene is the basic unit of a chromosome. In our approach, the gene is represented in permutation encoding, as a sequence of slot numbers. In the Figure 31, we have an array of slots, which is initialized by multiplying the days by periods. Then we create a gene constructor, which takes in a single integer parameter, we loop through the array of slots, then assign each integer with a randomization for a single class group timetable. We shall have 35 gene represented in permutation encoding.

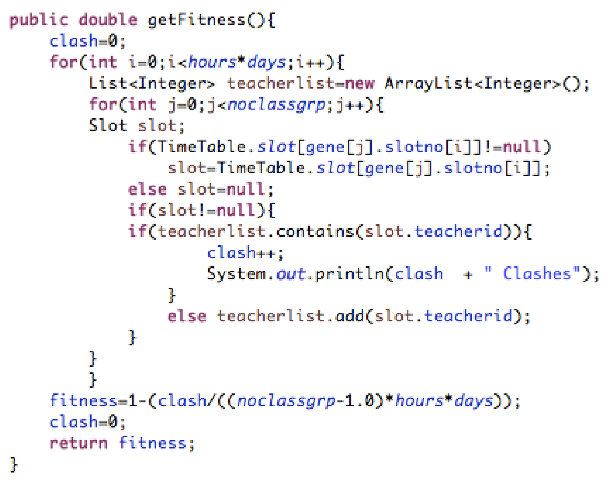


```
public Gene[] gene;

Chromosome(){
    gene=new Gene[noclassgrp];
    for(int i=0;i<noclassgrp;i++){
        gene[i]=new Gene(i);
        }
    fitness=getFitness();

}
```

Figure 32: Chromosome

5.3.1.4 Chromosome:

A chromosome is regarded as a solution to the given problem. Initially as the algorithm is executed, the chromosome is composed of an array of genes. Therefore, as each gene i is created, it is placed in a specific chromosome. In the chromosome, in order to get the best solution, it undergoes the fitness evaluation. The chromosome class calculates the best fitness solution for each class group. In the chromosome class, we have constructor Chromosome(), which calls the gene constructor gene(), assigns genes for each class group in the in our database. Finally after generation of genes, it calculates the fitness of each gene by calling a method getFitness().



```java
public double getFitness(){
    clash=0;
    for(int i=0;i<hours*days;i++){
        List<Integer> teacherlist=new ArrayList<Integer>();
        for(int j=0;j<noclassgrp;j++){
        Slot slot;
            if(TimeTable.slot[gene[j].slotno[i]]!=null)
                slot=TimeTable.slot[gene[j].slotno[i]];
            else slot=null;
            if(slot!=null){
            if(teacherlist.contains(slot.teacherid)){
                    clash++;
                    System.out.println(clash  + " Clashes");
                }
                else teacherlist.add(slot.teacherid);
            }
        }
        }
    fitness=1-(clash/((noclassgrp-1.0)*hours*days));
    clash=0;
    return fitness;
}
```

Figure 33 Fitness Function:

5.3.1.5 Fitness Function:

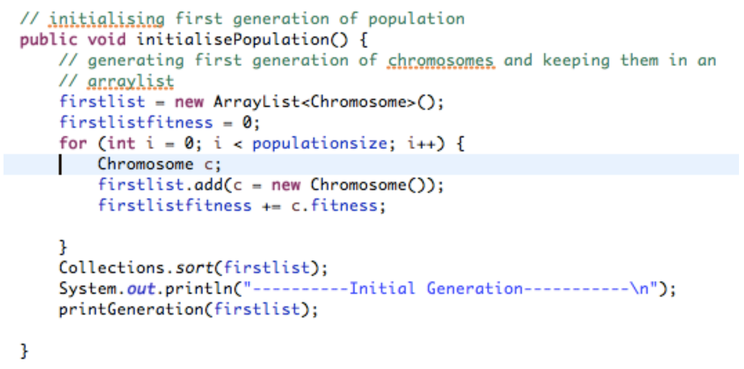
In genetic algorithm, the fitness measures the quality of the solution generated from each chromosome. In our system, this is regarded as the evaluation function, as it, measures how close the given optimal solution is to the timetabling problem. Each chromosome solution generated, it is subjected to evaluation by the getFitness() method. We award the fitness score as 1.0, the closer the solution is to the fitness score, the greater our optimal solution. To evaluate the fitness, we used the method in [24] to evaluate our fitness.

$$1 - \left(\frac{Clash}{(Classgroup\text{-}1.0)\times Days\times Hours}\right)$$

Equation 4: Fitness

The clash is the number of teachers that may occur at the same timetable slot in different class groups. If two or more occur, it increments the clash count.

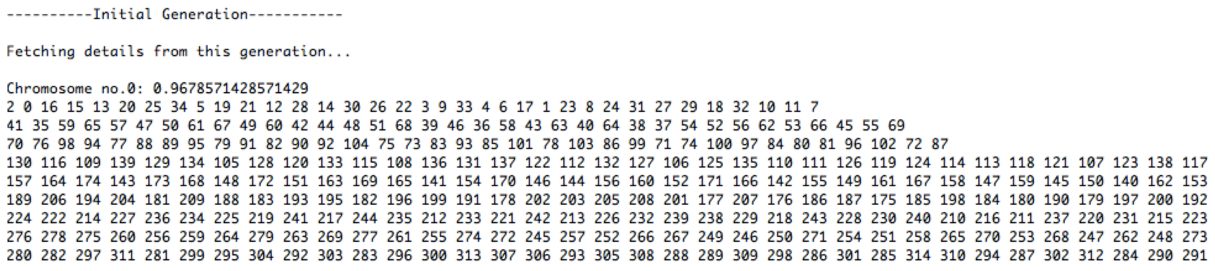
To further explain (Equation 4: Fitness), we shall introduce an example. For instance, given a clash of 6, with class groups of 9, days of 5 and hours of 7. We calculate 6 divided by the 35 multiplies by 9 – 1.0. The results generated shall be subtracted from 1, which is approximately 0.9786. This fitness score shall generate solution that is close to our optimal solution.



```
// initialising first generation of population
public void initialisePopulation() {
    // generating first generation of chromosomes and keeping them in an
    // arraylist
    firstlist = new ArrayList<Chromosome>();
    firstlistfitness = 0;
    for (int i = 0; i < populationsize; i++) {
        Chromosome c;
        firstlist.add(c = new Chromosome());
        firstlistfitness += c.fitness;

    }
    Collections.sort(firstlist);
    System.out.println("----------Initial Generation-----------\n");
    printGeneration(firstlist);

}
```

Figure 34: Initial Population

Therefor with all explained in this section, we shall describe how the initial population is created. First and foremost, the initial population is generated randomly, therefore allowing a range of possible solutions. We define a population size of a 1000, because, as fit solutions are given more opportunities to reproduce, we need more space for new arrivals. This is a basic elementary constraint used in GA implementation, as it restricts the algorithm from producing infeasible solutions. Within the range of initial population size, we create new chromosomes by calling the Chromosome() constructor, and retrieving the fitness value for the initial population. Within the chromosome class, we recall that it also calculates the fitness, we call the fitness as chromosome c.fitness. We store the possible set of chromosomes in a java array list and print them out for testing purposes.



```
----------Initial Generation-----------

Fetching details from this generation...

Chromosome no.0: 0.9678571428571429
2 0 16 15 13 20 25 34 5 19 21 12 28 14 30 26 22 3 9 33 4 6 17 1 23 8 24 31 27 29 18 32 10 11 7
41 35 59 65 57 47 50 61 67 49 60 42 44 48 51 68 39 46 36 58 43 63 40 64 38 37 54 52 56 62 53 66 45 55 69
70 76 98 94 77 88 89 95 79 91 82 90 92 104 75 73 83 93 85 101 78 103 86 99 71 74 100 97 84 80 81 96 102 72 87
130 116 109 139 129 134 105 128 120 133 115 108 136 131 137 122 112 132 127 106 125 135 110 111 126 119 124 114 113 118 121 107 123 138 117
157 164 174 143 173 168 148 172 151 163 169 165 141 154 170 146 144 156 160 152 171 166 142 155 149 161 167 158 147 159 145 150 140 162 153
189 206 194 204 181 209 188 183 193 195 182 196 199 191 178 202 203 205 208 201 177 207 176 186 187 175 185 198 184 180 190 179 197 200 192
224 222 214 227 236 234 225 219 241 217 244 235 212 233 221 242 213 226 232 239 238 229 218 243 228 230 240 210 216 211 237 220 231 215 223
276 278 275 260 256 259 264 279 263 269 277 261 255 274 272 245 257 252 266 267 249 246 250 271 254 251 258 265 270 253 268 247 262 248 273
280 282 297 311 281 299 295 304 292 303 283 296 300 313 307 306 293 305 308 288 289 309 298 286 301 285 314 310 294 287 302 312 284 290 291
```

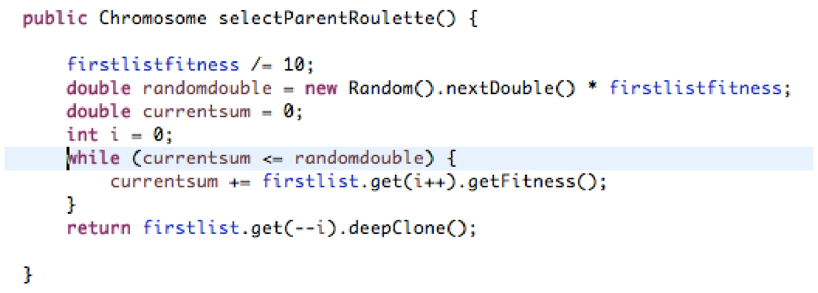
Figure 35: Generated Output Of Initial Population (Genotypic Representation)

Initial population of chromosome in genotypic representation with fitness value of 0.9678572……

5.3.2 Genetic operators

GA is based on a triangle of three principles, which includes the principle of evaluation, principle of reproduction and principle of selection based on fitness function. The principle of evaluation is selected from its fitness which was explained in 5.3.1.5. We select the two best chromosomes and mother and father in the principle of selection. Finally, principle of reproduction involves crossover and mutation operations, which we shall describe in this subsection.

5.3.2.1 Survivor Selection: Roulette Wheel Selection



```java
public Chromosome selectParentRoulette() {

    firstlistfitness /= 10;
    double randomdouble = new Random().nextDouble() * firstlistfitness;
    double currentsum = 0;
    int i = 0;
    while (currentsum <= randomdouble) {
        currentsum += firstlist.get(i++).getFitness();
    }
    return firstlist.get(--i).deepClone();

}
```

Figure 36: Roulette Wheel Selection

From the fitness value generated, we shall perform selection of two parent gene using the “roulette wheel selection”. Selection of parent chromosome is proportional to the fitness value of the chromosomes. Therefore, the higher the fitness value, the higher chances of being selected as the parent. We used the basic method for fitness calculation, which includes:

1. Calculate fitness from the best 10 %
2. Generate a random number between 0 and fitness
3. From the top population, increment the fitness of the of the partial sum C, till C less than the R
4. The fitness of the individual C which proceeds R shall be chosen as the parent.

Figure 37: Algorithm for Roulette Wheel Selection.

Just as in the Darwin’s theory of evolution for natural selection, two fit parents are required to mate in order to produce an offspring. The simulation is executed in order to select the best “father “and “mother” chromosome, which shall be used in the next genetic operators. We require the best chromosomes, in order to allow them pass their genes to the next generation. Best chromosomes solutions are required to produce better solutions. Just as in evolutional theorem, the least fit individual dies, in the genetic algorithm. We discard them as they are no longer required, else, they would result in worst solutions..

5.3.2.2 Crossover: Single Point Crossover

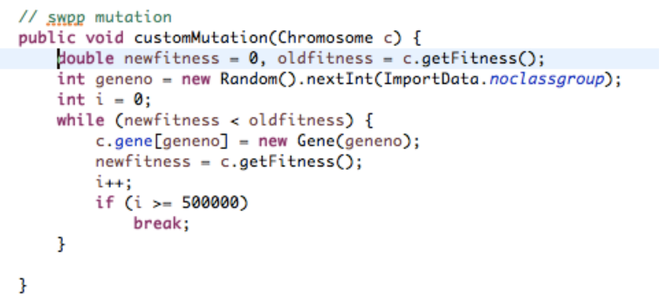
Figure 38: Crossover

In reference to the theory of evolution, two parents are required to mate, which results in creation two or more offspring. This is called reproduction, which is analogous to the genetic algorithm crossover operator. From selector operator, we require the best chromosomes, in order to allow them pass their genes to the next generation. Best chromosomes solutions are required to produce better solutions. The mother and father chromosome are subjected to single point crossover. In the single point crossover, a random point is selected from the two fit , and their tails are swapped, resulting in new . We applied this method in Figure 39.

1. Two chromosomes are chosen from the selection operator
2. Randomly chosen crossover point is generated.
3. Genes of the two parent chromosomes are exchanged at that point.
4. We then search for the highest fitness values from the swapped offspring, if the father gene is more, we return his else mother.
5. The returned dominant gene shall be present in the offspring.

Figure 39: Crossover Algorithm

5.3.2.3 Mutation Operator



```
// swpp mutation
public void customMutation(Chromosome c) {
    double newfitness = 0, oldfitness = c.getFitness();
    int geneno = new Random().nextInt(ImportData.noclassgroup);
    int i = 0;
    while (newfitness < oldfitness) {
        c.gene[geneno] = new Gene(geneno);
        newfitness = c.getFitness();
        i++;
        if (i >= 500000)
            break;
    }

}
```

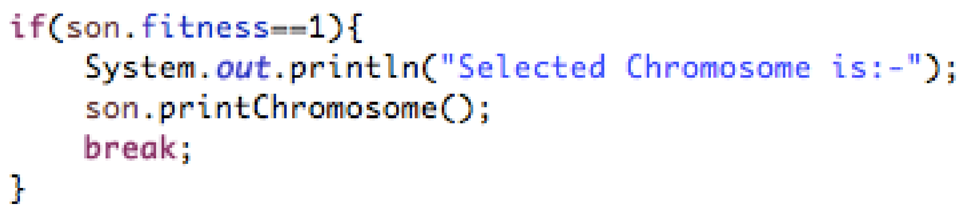
Figure 40: Mutation Operator

In order to maintain genetic diversity, from one chromosome to another, we apply the mutation operator. Just like crossover, this operator is also analogous to biological mutation, as changes small genes in the offspring, “small tweak in offspring chromosome”. In our approach, we employ the basic swap mutation, by literally swapping two random gene of the same child chromosomes. Figure 40, show the main method for the mutation operator. We have two variables newfitness and oldfitness, we initialize the newfitness value and oldfitness value with the actual fitness of the child chromosome. We also obtain gene value, which we select at random, if the new value is less than the oldfitness value, we exchange the random gene from offspring chromosome with a new gene. Then we return the new fitness values of the mutated offspring.

1. Select a random number as mutation point
2. Find the mutation point of the child chromosome
3. Exchange the gene of the child chromosome with a new gene.

Figure 41: Mutation Algorithm

5.3.3 Termination Condition

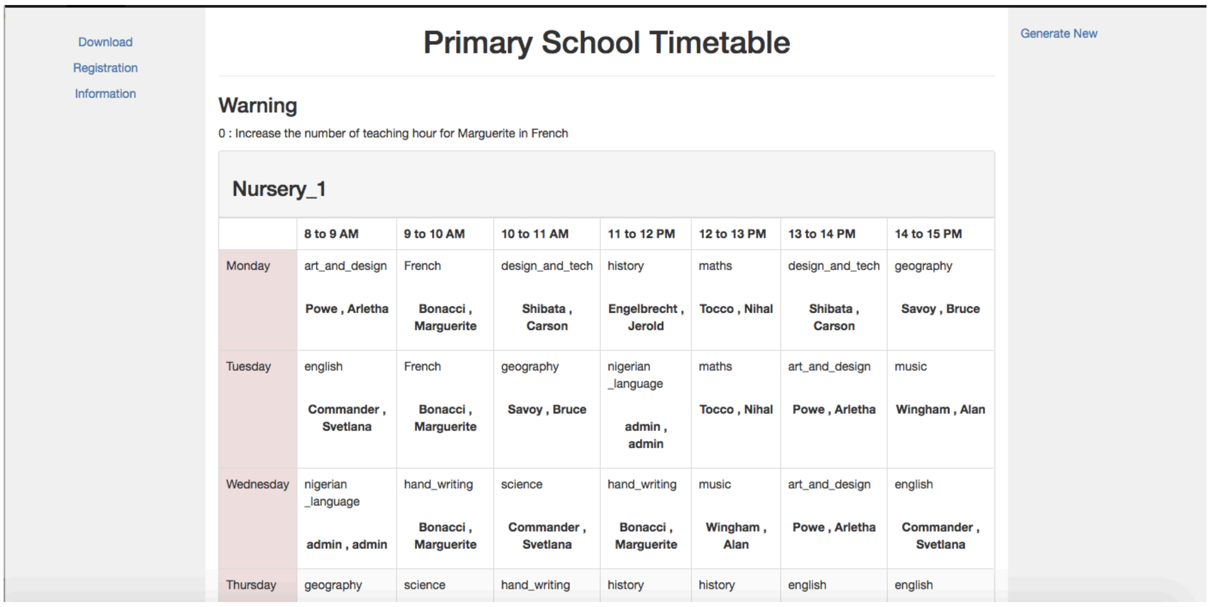


```
if(son.fitness==1){
    System.out.println("Selected Chromosome is:-");
    son.printChromosome();
    break;
}
```

Figure 42: Termination Condition

For the Genetic Algorithm to terminate, we define a termination criterion. If this is not defined, the algorithm would be set to run in an infinite loop. At the initial stage of progression, the Genetic Algorithm shall be relatively fast, as it generates possible solutions to the problem. At later stages, the process begins to saturate as the space for improvements become relatively small. We define our termination criteria, if fitness of the new generation (offspring) is equal to 1, the terminate.

5.4 Phenotypic Representation



Primary School Timetable

Warning

0 : Increase the number of teaching hour for Marguerite in French

Nursery_1

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	art_and_design **Powe , Arletha**	French **Bonacci , Marguerite**	design_and_tech **Shibata , Carson**	history **Engelbrecht , Jerold**	maths **Tocco , Nihal**	design_and_tech **Shibata , Carson**	geography **Savoy , Bruce**
Tuesday	english **Commander , Svetlana**	French **Bonacci , Marguerite**	geography **Savoy , Bruce**	nigerian _language **admin , admin**	maths **Tocco , Nihal**	art_and_design **Powe , Arletha**	music **Wingham , Alan**
Wednesday	nigerian _language **admin , admin**	hand_writing **Bonacci , Marguerite**	science **Commander , Svetlana**	hand_writing **Bonacci , Marguerite**	music **Wingham , Alan**	art_and_design **Powe , Arletha**	english **Commander , Svetlana**
Thursday	geography	science	hand_writing	history	history	english	english

Figure 43: Phenotypic Representation

The user interface represents the phenotype space. In here, decoding is performed, which converts the solution form permutation encoding to user readable information in Figure 43.

5.5 Summary

In this chapter we described the two phases of the application, which is the BACP and GA. We described the relevance of the BACP and how it can be used by the timetabler. The BACP is extremely important as it is used in every educational sector. If it is correctly balanced, it would assist in producing a best optimal solution. In this chapter we also described the back end code of the application with respect to satisfying the constraints. Finally, we described how the genetic algorithm was applied in respect to our application.

6 Chapter 6: Results and Discussion

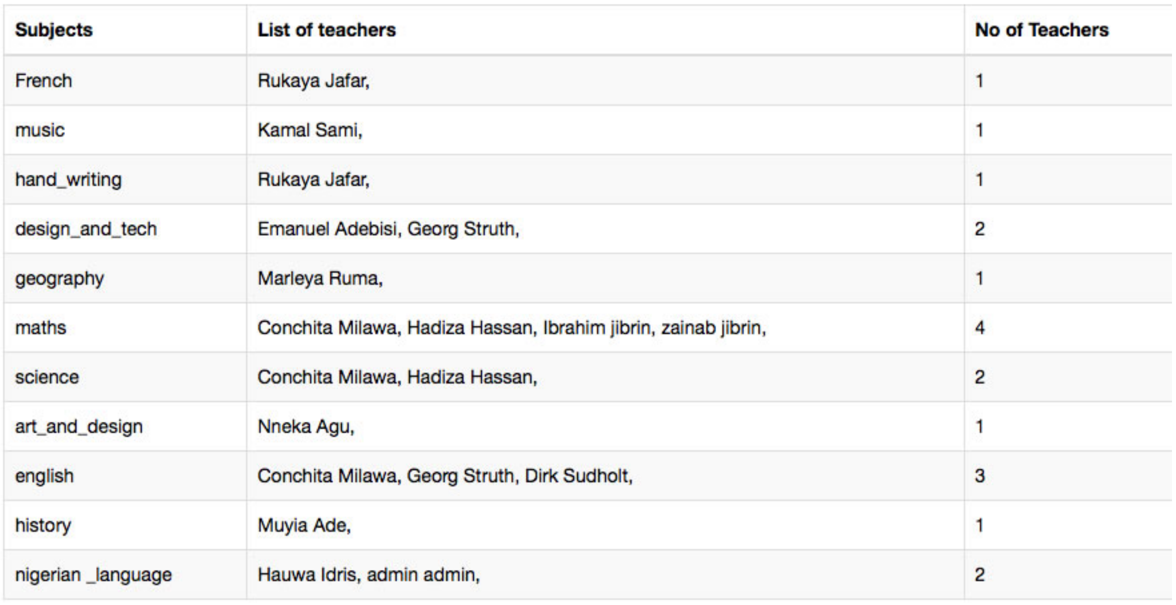
Initially stated in section 1.1, we proposed a system that was built to solve the timetabling problem for a small primary school. This section of the report shall focus on the results generated by our system. We shall critically evaluate the systems results with respect to the scheduling problem. The application was subjected 4 hard constraints and 1 soft constraint, which were described in section 3.3.1. For simplicity, we displayed our results on the interface which we describe in great detail.

6.1 Classification of Data

No of Teachers	16
No of Subjects	11
No of Classes	9

Figure 44: Test Data

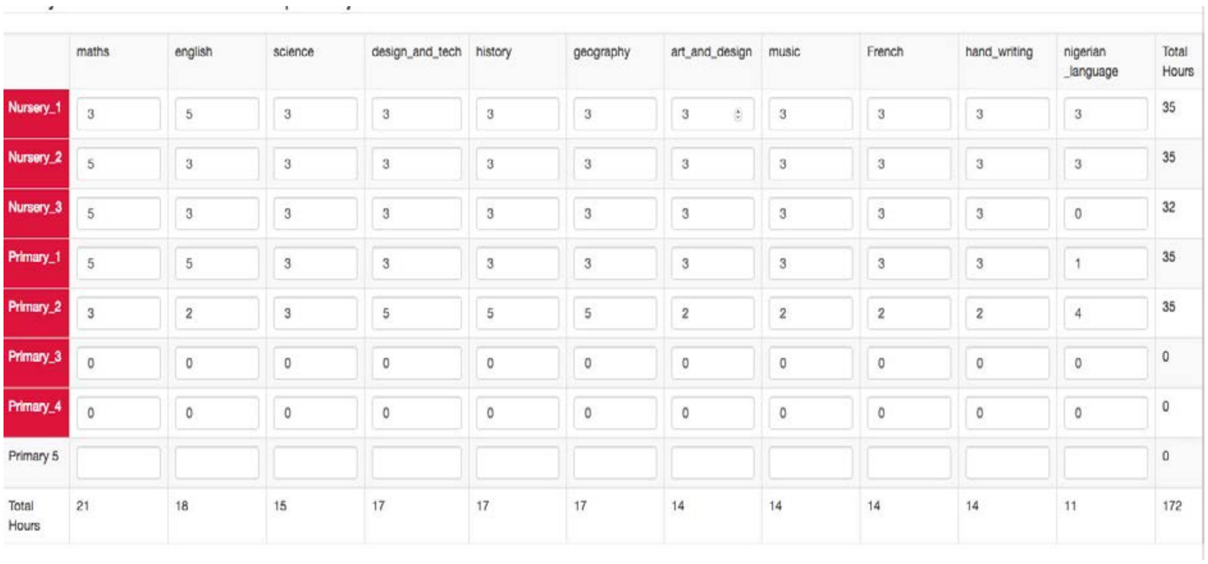
Our system was designed to be deployed in a "small primary school", therefor, for test purposes, we considered using data that was relevant to our domain. It is important to indicate that unlike past academic research works mentioned in the literature, we used random test data. The test data used was not gotten form a real primary school, we designed a flexible system, which the timetabler can manipulate as they desire.



Subjects	List of teachers	No of Teachers
French	Rukaya Jafar,	1
music	Kamal Sami,	1
hand_writing	Rukaya Jafar,	1
design_and_tech	Emanuel Adebisi, Georg Struth,	2
geography	Marleya Ruma,	1
maths	Conchita Milawa, Hadiza Hassan, Ibrahim jibrin, zainab jibrin,	4
science	Conchita Milawa, Hadiza Hassan,	2
art_and_design	Nneka Agu,	1
english	Conchita Milawa, Georg Struth, Dirk Sudholt,	3
history	Muyia Ade,	1
nigerian _language	Hauwa Idris, admin admin,	2

Figure 45: Timetable Information

This image show the timetbaler, information regarding the list of subjects, teachers registered to teach these subjects and the number of teachers allocated for a specific subject.



	maths	english	science	design_and_tech	history	geography	art_and_design	music	French	hand_writing	nigerian _language	Total Hours
Nursery_1	3	5	3	3	3	3	3	3	3	3	3	35
Nursery_2	5	3	3	3	3	3	3	3	3	3	3	35
Nursery_3	5	3	3	3	3	3	3	3	3	3	0	32
Primary_1	5	5	3	3	3	3	3	3	3	3	1	35
Primary_2	3	2	3	5	5	5	2	2	2	2	4	35
Primary_3	0	0	0	0	0	0	0	0	0	0	0	0
Primary_4	0	0	0	0	0	0	0	0	0	0	0	0
Primary 5												0
Total Hours	21	18	15	17	17	17	14	14	14	14	11	172

Figure 46: Subject Hours

6.2 Findings

Initially when we were awarded the project, we saw this from an easy point of view, we never expected the level of difficulty involved in this project. We agree with all evidence described in the introduction and literature review that scheduling is a very difficult problem. Naïve techniques such as backtracking and network flow, could be applied to the problem, but there is no evidence that this would provide a best optimal solution such as genetic algorithm. We support this claim because, initially we tried using a simple back tracking approach to solve the problem, but this generated infeasible solutions towards the end of timetable. Honestly our initial approach was not the genetic algorithm, but our final decision to use this algorithm was best, as it satisfied the constraint based scheduling problem.

Additionally, we were given the task to develop a web application, we decided to take a course specific to javaEE because, we choose to use java as the programing language. This was the constructive foundation to our project. Then we broadly researched on timetabling in general, all, we found methods generated that could solve the problem. Researched on how past academic researchers developed their version of the application. With most academic research pointing to the success of the genetic algorithm, we finally decided to base our work on this. Most results were fast, effective and all generated feasible solutions to the timetabling problem. The breaking point for our decision was the genetic algorithm used in solving the ever hard University timetabling problem.
Peter Uchenna [25] developed a brilliant application to solving the hardest timetabling problem in the educational sector. In this, we noticed the power of the genetic algorithm, as it always produces a solution. Which we confirmed by his results and saw that this was a resent development done in 2015. His outstanding results proved to be summary of our decision.

It should also be indicated that the more values which are inserted into the database, leads to more time spent in generating a feasible solution. This cannot be regarded as a limitation of the application, as this is due to the fact that the system tries to find a solution amongst infeasible solution, which takes times. Therefore, the system applies the genetic algorithm till it finds a solution.

6.3 Goals Achieved

6.3.1 Hard Constraints

6.3.1.1 Teacher Clash and Overlap:

The timetable generated feasible results by satisfying the teacher clash constraint. The timetable was designed to allocate at most a single teacher for a specific class. There was no instance of two teachers found present in the timetable. The timetable produced a feasible timetable which avoided assigning a specific teacher to two classes simultaneously. The results of this constraint are displayed in the appendix section of class timetable.

6.3.1.2 Subject Clash:

In every school timetable, a class is assigned to take at most one subject at a specific time. Our system proved to be successful as it also generated results that satisfied this constraint. With respect to the curriculum, the timetable assigns at most one subject to each timeslot. It evenly spreads all subjects according to the hours for that class. Evidence confirming to the results are shown in appendix section of class timetable.

6.3.1.3 Teaching Hours

Another constraint defined in Chapter 3 was the issue of teaching hours. In every organization, a set of working hours are dynamically assigned to each registered teacher. The system was successful in assigning teachers to classes without exceeding the specific teaching our. Evidence is displayed in appendix section . In the user interface, the timetable was designed to assign the subject without a teacher in that particular slot. For instance, in the databse, we have only 1 french teacher for the whole school, with 20 assigned as teacher T working hours. The system keeps assigning teacher T till the hours exceeded, if the number of timeslots is equal to hours exceeded, then allocation for that teacher stop.
In a situation where by we have more than one teacher for a particular teacher, the system selects the first teacher who is eleigibale to teach that subject and assigns for that particular class group . After allocation has been done, the system searches for the next teacher teaching that particular subject, and assigns to a specific class group. It keep repeating the sequence of allocation till they are no more available teachers. After being previously assigned, if a class still requires a teacher to teach that subject, it for free teachers, who have not exceeded their

teaching limit. If found then it allocates for that specific subject, else if no teacher found then it leaves it as blank. Relative evidence described here can be found in in the entire timetable, as it makes allocation but leaves the slot of French as blank, because no teacher was found.

6.3.1.4 Class Meetings

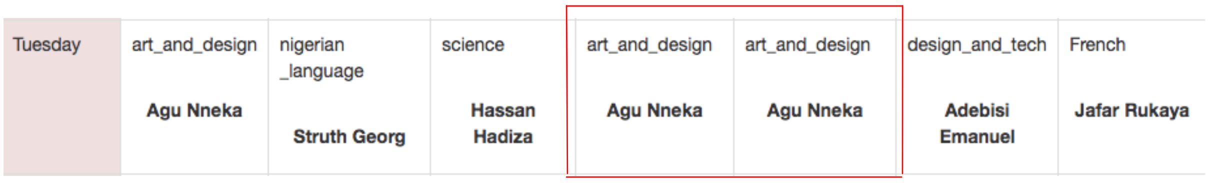
The system was designed to correctly allocate subjects and teachers to each class. The system spreads the subjects according to the number of hours placed by the timetabler. The system also distributes teachers across all slots for each class. Evidence that conforms that the system performs balancing of subjects and teachers across classes is evident in appendix in the entire timetable.

6.3.2 Soft Constraint

6.3.2.1 Teacher balancing

The system keeps assigning teachers and subjects till the teaching hours limit has been reached. The system comes up with a solution where by it assigns the subject but leaves the room for the teacher. Therefore to indicate to the timetabler that improvements have to be made in order to improve a near optimal solution to the best optimal solution.

6.4 Negative results



Tuesday	art_and_design **Agu Nneka**	nigerian _language **Struth Georg**	science **Hassan Hadiza**	art_and_design **Agu Nneka**	art_and_design **Agu Nneka**	design_and_tech **Adebisi Emanuel**	French **Jafar Rukaya**

As much as we would like to describe the overall success of the system, we must be extremely honest in our evaluation by documenting the systems negative results. Unlike other educational sectors, which has the double period for a subject in a specific day, the primary school sector prohibits that. As the genetic algorithm perform assignments, it fails to allocate double period for subjects at random.

6.5 Evaluation

6.5.1 Functional Requirement Evaluation:

Through the use of the genetic algorithm, the system generates the best and near best solution. Immediately the system is initiated, it satisfies all hard and soft constraints therefore creating the best feasible solution. In some instances, the generate a near best solution, which indicates that it satisfies all hard constraints and some soft constraints. In these kinds of situations, the system makes suggestions the timetable might need to consider. If suggestions are considered, therefore changes are made by the timetable to reflect the suggestions, leading to best optimal solution . The system has proved to perform what it was designed to it.

6.5.2 Non-Functional Requirements:

In this subsection we shall evaluate the system performance in relation to performance. In one test carried on the system, it generated a number of 114 chromosome with 5 generations before reaching an optimal solution. Also it should be noted as the population of timetable elements increase, the time required to generate a feasible solution would be increased, therefore leading to a corresponding increase in number of chromosomes and generates Alternatively if we decrease the amount of teaching hours for teachers, it makes it difficult to find a feasible timetable, therefore the system increases the number of runs and time it takes to generate a solution. The reason why it takes longer to search for a solution is because after the system exhausted all possible solutions, the system shall still try to find solutions in no solutions at all. Our system is extremely fast when timetable data is small, but it becomes relatively slow when data is large.

In the aspect of usability, the system was designed to be as simplistic for any user to understand. For instance, when the system is not able to satisfy all soft constraints, the system provides suggestions which confirms the “ease of use”. Additionally, the system preforms color change in timetable slots to indicate the amount soft constraints violated.

6.5.3 Overall Research Evaluation

With respect to the aims and objectives of the research problem, our system performed excellently by generating solving the timetabling problem. To justify our evaluation, recall a feasible timetable is one that satisfies all “hard constraints”,. The feasible timetable is also one that performs assignment of teachers to a given class at a particular timeslot for a proposed subject. Through results generated in section 6.1, it is clearly evident that our system has conformed to its purpose by solving the Small Primary School Timetabling problem. With the use of genetic algorithm, the system assign at most one teacher and subject to a specific timeslot without exceeding the amount of teaching hours assigned to a teacher. It also performs balancing of subjects according to the number of hours for a specific class. Additionally, we would like to indicate the system’s functionality of performing feasible suggestions to enable timetabler improve their generated results. With all these mentioned, we conclude that system requirements and objectives have been met.

6.6 Further work

Even with the outstanding success rate of the system, there are still more which could have been done to make it better. These improvements shall be made based on a real primary school.
Regardless of the educational sector, most schools use a feasible timetable used in the last academic year of semester, then make minor changes to generate a new feasible timetable. Because of the nature of the genetic algorithm, which generates possible solutions for all classes, it is difficult indicate a particular point of change, generate a solution for the specific class based on the changes. Therefore, additional research is required to create a solution that reflects such changes. In relation to the system, the feasible timetable could be downloaded, minor changes could be physically made by the timetabler. This solution is still under investigation, because, it is impossible for the timetable to keep track of every teacher assigned to a timeslot. Due to this limitation, if the timetable makes changes, there is a high probability of teacher clash occurrence in the timetable.

Additionally, harder subjects are taken in the mornings then comes easier subjects. For instance, mathematics is usually the first subject of the day, then comes English. An additional constraint such as this, should be incorporated into this constraint satisfaction problem.

7 Chapter 7: Conclusion

In this chapter ,we shall provide a summarized description of all chapter descripted in this report. First and foremost, we shall start by referring our main aims and objective descried in Chapter 1, which proposed to build an automated primary school timetabling system using the Genetic Algorithm. We succeeded in developing the application through the success of academic literatures listed in Chapter 2. In chapter 2, we described the meta-heuristic approaches for solving the timetabling system, they include the simulated annealing, genetic algorithm, and tabu search. We described each method, the algorithmic procedure, and critically analysed two method academic works based on the algorithms. Through research, it is evident that the genetic algorithm proved to be efficient and more effective compared to other meta heuristic approaches. After selecting the most effective approach for the timetabling problem, we provided system requirements, which were subdivided into functional, non functional and algorithmic requirements. The functional requirements were regarded as the mandatory functionalities of the system, in relation to timetabling, here we defined the hard and soft constraints, which the system must satisfy to produce a feasible timetable. The non functional requirements described system qualities such as performance, usability and security. We further described the algorithmic functionality of the system, which described how the genetic algorithm was used by the system. In addition to this chapter, we listed principles of the (BSC) which we respected while developing the project. In relation to the system requirements, we described design techniques in the system. Justifiable evidence such as UML diagrams, ERD and SDLC models were provided, to provide better understanding to any reader of the project. With relation to the complexity of the system, we described the system technology stack, which was implemented using object oriented programing language, java , MySQL and deployed on apache tomcat server 8.5. In Chapter 5, we describe out described how the system was implemented with respect to chapter 4, which describes genotypic to phenotypic transformation involved in the process of our feasible timetable generation. Finally we critically evaluate and provide all results that justify our timetable reflect information provided in all chapters.

The results generated was an indication of our systems success. We developed an interactive design for both algorithmic and software process that produced successful results. The genetic algorithm performed excellently in producing best optimal and near best optimal solutions to solving the timetabling problem. A unique feature of our system is its ability to make feasible suggestions for the timetabler. These suggestions are very useful in situations where the genetic algorithm is unable to provide the best feasible solution. If suggestions are incorporated in the system, they grantee best optimal solutions in the next generation. We are estatic to present a system that solves its main purpose successfully, but we had to honestly provide improvements in further implementation. Additional constraints and research in the area of genetic algorithm have to done in to further improve the system.

With evidence showing, we present an easy approach to timetabling called “Small Primary School Timetable System”. We have successfully satisfied the purpose of our dissertation through the use a meta-heuristic approach called the Genetic Algorithm.

8 Works Cited

[1] d. Costa, "A Tabu Search Algorithm for computing an operational timetable," Holland, 1994.

[2] B. Kourousic-Seljak, "Timetabling Construction using general heuristic techniques," 2002.

[3] b. hamad, K. Jaber and H. Amin, "A Survey of approaches for University course timetabling problem," 2014.

[4] S. A, "A survey of Automated Timetabling," 1999.

[5] A. D, "Constrcuting School Timetables using Simulated Annealing: sequential and parallel algorithm," 1991.

[6] H. Tarawneh, A. Masri and A. Zulkofli, "A Hybrid Simnulated Annealing with Solutions Memory for Curriculum based Course Timetabling Ptroblem.," Malaysia, 2013.

[7] . Rushil and . Nelishia, "A Study of Genetic Algorithm to solve the School Timetabling problem," Berlin, 2013.

[8] C. Alberto, D. Marco and M. Vittorio, "A genetic Algorithm to solve the timetabling problem," 1994.

[9] H. Salman, B. Mehdi and OmidHamidi, "A Tabu serach Algorithm with effecient diversification strategy for high school timetabling problem," Iran, 2013.

[10] G. Micheal, P. Jean-Yves, M. Gendreau and P. J.-Y, "Tabu Search," Canada, 2010.

[11] K. S., C. Gelatt and M. Vecchi, "Optimization by Simulated Annealing," 1983.

[12] F. Glover, "Tabu Search: A Tutorial," Colorado, 1990.

[13] i. C. Ru, A. Armando and C. Adriano, Simulated Annealing Theory with Applications, Portugal: Sciyo, 2010, pp. 2-4.

[14] D. Abramson, "Constructing School Timetables using simulated annealing: Sequential and parallel algorithm," Australia, 1991.

[15] R. W. Steve, "Concepts of Scale in Simulated Annealing," California, 1984.

[16] N. Rangel-Valdez, J. Jasso-Luna, M. Rodriguez-Chavez and G. Bujano-Guzman, "Practical relaxation of a special case of the Curriculum-Based Course Timetabling problem," Berlin, 2014.

[17] L. Rhydian, P. Ben and R.-D. Olivia, "Metaheursitics for University course timetabling," Scotland, 2007.

[18] D. Costa, "A Tabu Search algorithm for computing on perational timetable," North-Holland, 1994.

[19] Z. Lu and J. K. Hao, "Adaptive tabu serach for course timetabling," 2010.

[20] H. H. John, "Genetic Algorithm: Computer programs that evolve in ways that resemble natural selectiuon can solve complex problems even their creators do not fully

understand," 2005. [Online]. Available: http://www2.econ.iastate.edu/tesfatsi/holland.GAIntro.htm.

[21] R. Rushil and N. Pillay, "A study of Genetic Algorithm to solve the School Timetabling Problem," `berlin, 2013.

[22] A. Colorni, M. Dorigo and V. Maniezzo, "A GENETIC ALGORITHM TO SOLVE THE TIMETABLE PROBLEM," 1994.

[23] C. Fernández and M. Santos, "A non-standard genetic algorithm approach to solve constrained school timetabling problems," 03029743, 2004.

[24] P. Khurana, "Github.Inc," April 29. [Online]. Available: https://github.com/pranavkhurana/Time-table-scheduler/blob/master/AutomatedTimeTableScheduler.pdf. [Accessed 15 August 2017].

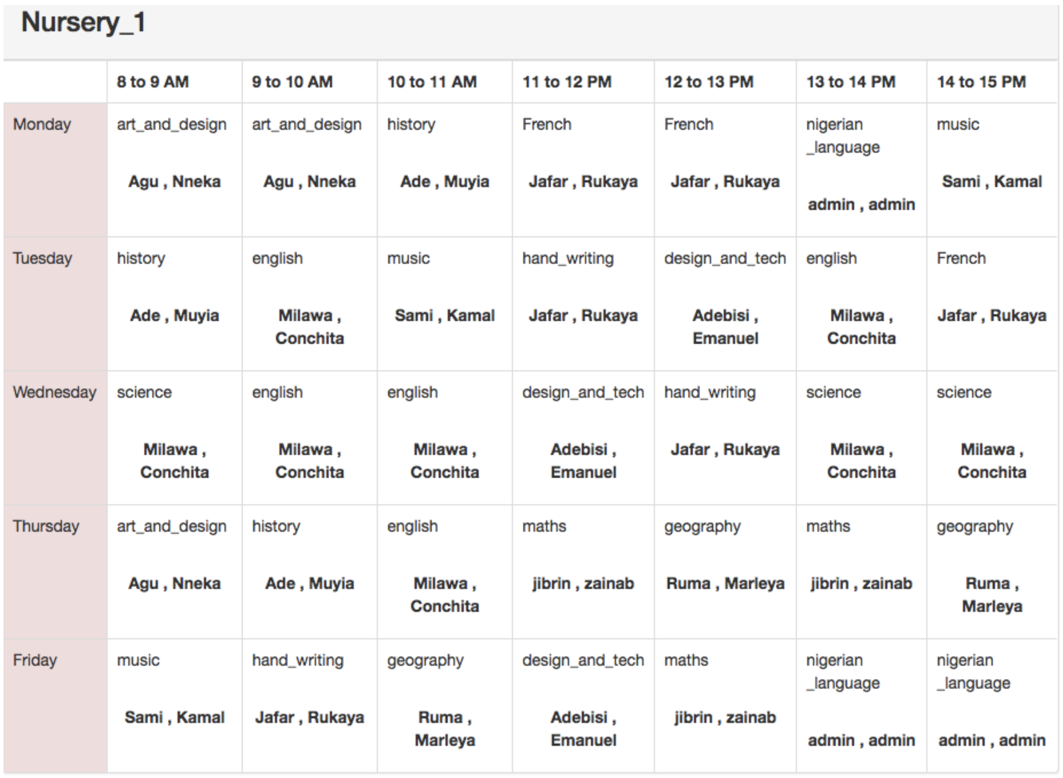
[25] P. U. Eze, "Constrcutive and Unconstructive Initialisation of a Genetic Algorithm for Solving a University Timetabling Problem," sheffield, 2015.

[26]

[27] N. Basir, W. Ismail and N. M. Norwawi, "A Simulated Annealing for Tahmidi Course Timetabling," 2013.

[28] B. Nurlida, I. Waidah and M. N. Norita, "A Simulated Annealing for Tahmidi Course Timetabling," Malaysia, 2013.

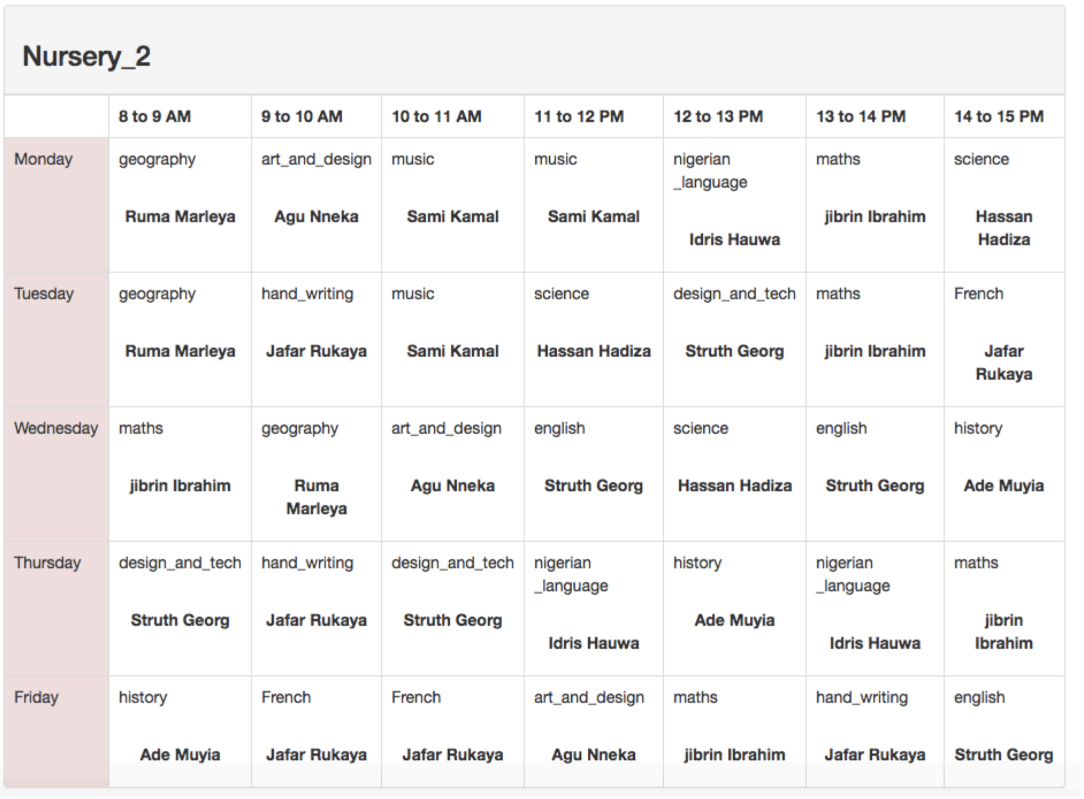
Appendix



Nursery_1

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	art_and_design **Agu , Nneka**	art_and_design **Agu , Nneka**	history **Ade , Muyia**	French **Jafar , Rukaya**	French **Jafar , Rukaya**	nigerian _language **admin , admin**	music **Sami , Kamal**
Tuesday	history **Ade , Muyia**	english **Milawa , Conchita**	music **Sami , Kamal**	hand_writing **Jafar , Rukaya**	design_and_tech **Adebisi , Emanuel**	english **Milawa , Conchita**	French **Jafar , Rukaya**
Wednesday	science **Milawa , Conchita**	english **Milawa , Conchita**	english **Milawa , Conchita**	design_and_tech **Adebisi , Emanuel**	hand_writing **Jafar , Rukaya**	science **Milawa , Conchita**	science **Milawa , Conchita**
Thursday	art_and_design **Agu , Nneka**	history **Ade , Muyia**	english **Milawa , Conchita**	maths **jibrin , zainab**	geography **Ruma , Marleya**	maths **jibrin , zainab**	geography **Ruma , Marleya**
Friday	music **Sami , Kamal**	hand_writing **Jafar , Rukaya**	geography **Ruma , Marleya**	design_and_tech **Adebisi , Emanuel**	maths **jibrin , zainab**	nigerian _language **admin , admin**	nigerian _language **admin , admin**

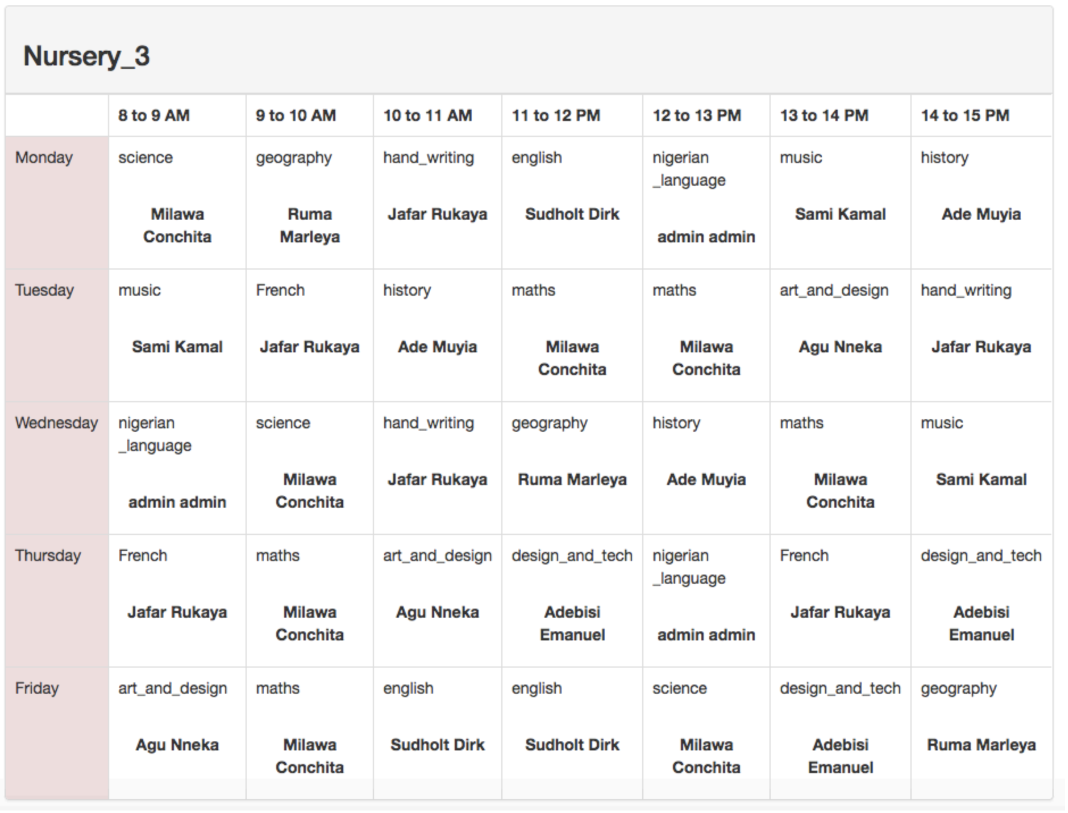
Figure 47: Nursery 1 Timetable



Nursery_2

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	geography **Ruma Marleya**	art_and_design **Agu Nneka**	music **Sami Kamal**	music **Sami Kamal**	nigerian _language **Idris Hauwa**	maths **jibrin Ibrahim**	science **Hassan Hadiza**
Tuesday	geography **Ruma Marleya**	hand_writing **Jafar Rukaya**	music **Sami Kamal**	science **Hassan Hadiza**	design_and_tech **Struth Georg**	maths **jibrin Ibrahim**	French **Jafar Rukaya**
Wednesday	maths **jibrin Ibrahim**	geography **Ruma Marleya**	art_and_design **Agu Nneka**	english **Struth Georg**	science **Hassan Hadiza**	english **Struth Georg**	history **Ade Muyia**
Thursday	design_and_tech **Struth Georg**	hand_writing **Jafar Rukaya**	design_and_tech **Struth Georg**	nigerian _language **Idris Hauwa**	history **Ade Muyia**	nigerian _language **Idris Hauwa**	maths **jibrin Ibrahim**
Friday	history **Ade Muyia**	French **Jafar Rukaya**	French **Jafar Rukaya**	art_and_design **Agu Nneka**	maths **jibrin Ibrahim**	hand_writing **Jafar Rukaya**	english **Struth Georg**

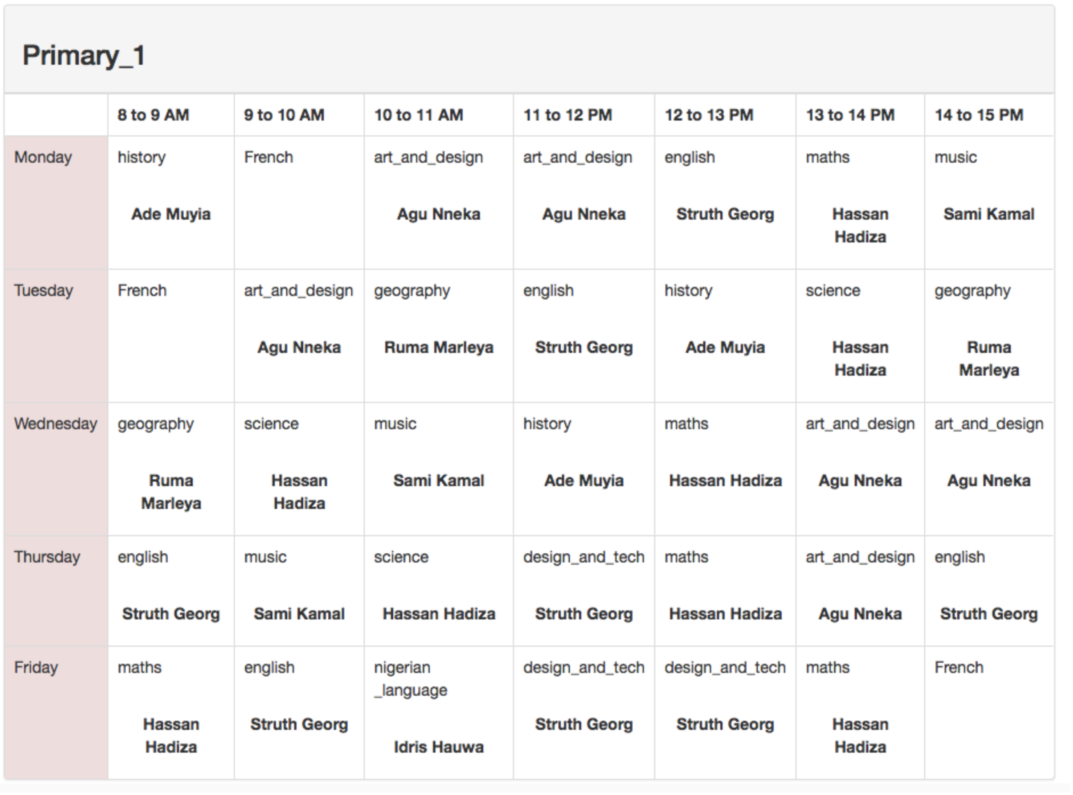
Figure 48: Nursery 2 timetable



Nursery_3

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	science **Milawa Conchita**	geography **Ruma Marleya**	hand_writing **Jafar Rukaya**	english **Sudholt Dirk**	nigerian _language **admin admin**	music **Sami Kamal**	history **Ade Muyia**
Tuesday	music **Sami Kamal**	French **Jafar Rukaya**	history **Ade Muyia**	maths **Milawa Conchita**	maths **Milawa Conchita**	art_and_design **Agu Nneka**	hand_writing **Jafar Rukaya**
Wednesday	nigerian _language **admin admin**	science **Milawa Conchita**	hand_writing **Jafar Rukaya**	geography **Ruma Marleya**	history **Ade Muyia**	maths **Milawa Conchita**	music **Sami Kamal**
Thursday	French **Jafar Rukaya**	maths **Milawa Conchita**	art_and_design **Agu Nneka**	design_and_tech **Adebisi Emanuel**	nigerian _language **admin admin**	French **Jafar Rukaya**	design_and_tech **Adebisi Emanuel**
Friday	art_and_design **Agu Nneka**	maths **Milawa Conchita**	english **Sudholt Dirk**	english **Sudholt Dirk**	science **Milawa Conchita**	design_and_tech **Adebisi Emanuel**	geography **Ruma Marleya**

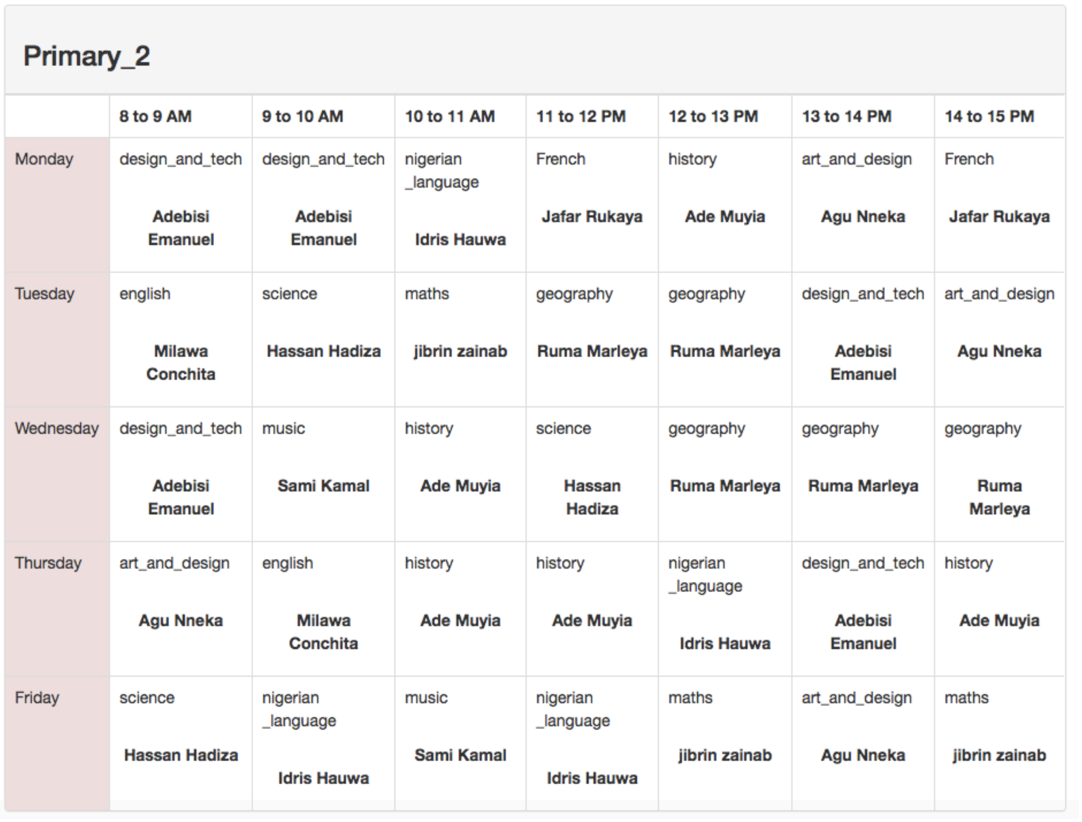
Figure 49: Nursery 3 Timetable



Primary_1

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	history **Ade Muyia**	French	art_and_design **Agu Nneka**	art_and_design **Agu Nneka**	english **Struth Georg**	maths **Hassan Hadiza**	music **Sami Kamal**
Tuesday	French	art_and_design **Agu Nneka**	geography **Ruma Marleya**	english **Struth Georg**	history **Ade Muyia**	science **Hassan Hadiza**	geography **Ruma Marleya**
Wednesday	geography **Ruma Marleya**	science **Hassan Hadiza**	music **Sami Kamal**	history **Ade Muyia**	maths **Hassan Hadiza**	art_and_design **Agu Nneka**	art_and_design **Agu Nneka**
Thursday	english **Struth Georg**	music **Sami Kamal**	science **Hassan Hadiza**	design_and_tech **Struth Georg**	maths **Hassan Hadiza**	art_and_design **Agu Nneka**	english **Struth Georg**
Friday	maths **Hassan Hadiza**	english **Struth Georg**	nigerian _language **Idris Hauwa**	design_and_tech **Struth Georg**	design_and_tech **Struth Georg**	maths **Hassan Hadiza**	French

Figure 50: Primary 1 Timetable



Primary_2

	8 to 9 AM	9 to 10 AM	10 to 11 AM	11 to 12 PM	12 to 13 PM	13 to 14 PM	14 to 15 PM
Monday	design_and_tech **Adebisi Emanuel**	design_and_tech **Adebisi Emanuel**	nigerian _language **Idris Hauwa**	French **Jafar Rukaya**	history **Ade Muyia**	art_and_design **Agu Nneka**	French **Jafar Rukaya**
Tuesday	english **Milawa Conchita**	science **Hassan Hadiza**	maths **jibrin zainab**	geography **Ruma Marleya**	geography **Ruma Marleya**	design_and_tech **Adebisi Emanuel**	art_and_design **Agu Nneka**
Wednesday	design_and_tech **Adebisi Emanuel**	music **Sami Kamal**	history **Ade Muyia**	science **Hassan Hadiza**	geography **Ruma Marleya**	geography **Ruma Marleya**	geography **Ruma Marleya**
Thursday	art_and_design **Agu Nneka**	english **Milawa Conchita**	history **Ade Muyia**	history **Ade Muyia**	nigerian _language **Idris Hauwa**	design_and_tech **Adebisi Emanuel**	history **Ade Muyia**
Friday	science **Hassan Hadiza**	nigerian _language **Idris Hauwa**	music **Sami Kamal**	nigerian _language **Idris Hauwa**	maths **jibrin zainab**	art_and_design **Agu Nneka**	maths **jibrin zainab**

Figure 51: Primary 2 Timetable