

AUTOMATIC TIME TABLE GENERATOR

Computer Science and Engineering

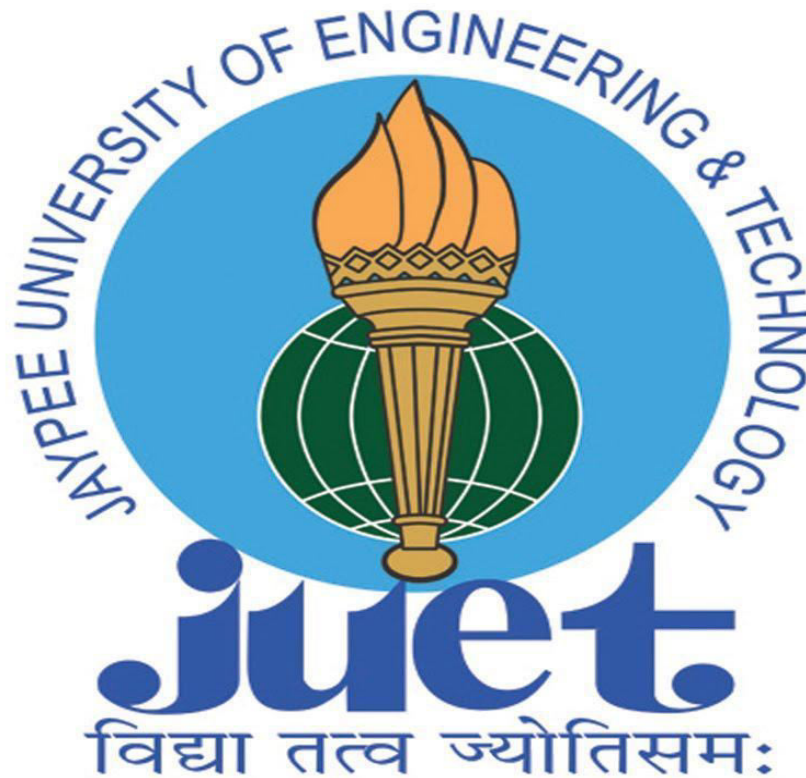
By

Abhay Garg (181B007)

Aditi Negi (181B013)

Simple Prasad (181B216)

Under the supervision of
Mrs. Babita Tiwari



NOV – 2020

Submitted in partial fulfilment of the requirement for the degree of
Bachelor of Technology

Department of Computer Science & Engineering

**Jaypee University of Engineering and Technology, A-B ROAD,
RAGHOGARH, DT. GUNA - 473226, M.P., INDIA**

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled “AUTOMATIC TIME TABLE GENERATOR AND COURSE MANAGEMENT SYSTEM FOR JUET” in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Engineering and Technology, Guna an authentic record of my our work carried out over a period from August 2020 to December 2020 under the supervision of Mrs. Babita Tiwari Assistant Professor (Grade-I) .

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Student Name: Abhay Garg

Rollno:181B007

(Student Signature)

Student Name: Aditi Negi

Rollno:181B013

(Student Signature)

Student Name: Simple Prasad

Rollno:181B216

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Supervisor Name: Miss Babita Tiwari

Designation: Assistant Professor (Grade-I).

Department name: C.S.E

(Supervisor Signature)

Dated: MAY2020

ACKNOWLEDGEMENT

This work is a synergistic product of many minds and I feel a deep sense of gratitude to my parents, for their encouragement and for being ever supportive. My sincere thanks goes to my supervisor Mrs. Babita Tiwari for her thorough assistance with this work. I also acknowledge the Computer Science Students for their active verbal participation and suggestions towards the evolvement of this project work.

TABLE OF CONTENTS

| | |
|---|----|
| CHAPTER ONE | |
| INTRODUCTION | 7 |
| CHAPTER TWO | |
| REVIEW OF RELEVANT LITERATURE | 11 |
| CHAPTER THREE | |
| DESCRIPTION | 17 |
| CHAPTER FOUR | |
| SYSTEM ANALYSIS AND DESIGN | 20 |
| CHAPTER FIVE | |
| SYSTEM IMPLEMENTATION | 28 |
| CHAPTER SIX | |
| SUMMARY, CONCLUSIONS AND RECOMMENDATIONS..... | 30 |
| CHAPTER SEVEN | |
| PROGRESS | 64 |
| REFERENCES | 42 |
| APPENDIX | 44 |

LIST OF FIGURES

General View of Automated Timetable

ER Diagram

Use Case Diagram

Class Diagram

SUMMARY

Scheduling course timetables for a large array of courses is a very complex problem which often has to be solved manually by the center staff even though results are not always fully optimal. Timetabling being a highly constrained combinatorial problem, this work attempts to put into play the effectiveness of evolutionary techniques based on Darwin's theories to solve the timetabling problem if not fully optimal but near optimal.

Genetic Algorithm is a popular meta-heuristic that has been successfully applied to many hard combinatorial optimization problems which includes timetabling and scheduling problems. In this work, the course sets, halls and time allocations are represented by a multidimensional array on which a local search is performed and a combination of the direct representation of the timetable with heuristic crossover is made to ensure that fundamental constraints are not violated.

Finally, the genetic algorithm was applied in the development of a viable timetabling system which was tested to demonstrate the variety of possible timetables that can be generated based on user specified constraints and requirements.

CHAPTER ONE

INTRODUCTION BACKGROUND STATEMENT

Timetabling concerns all activities with regard to making a timetable that must be subjective to different constraints. According to Collins Concise Dictionary (4th Edition) “a *timetable* is a *table of events arranged according to the time when they take place.*”

A critical factor in running a university or essentially an academic environment is the need for a well-planned and clash-free timetable. Back in the times when technology was not in wide use, academic timetables were manually created by the educational center staff.

Every year, JUET face the rigorous task of drawing up timetables that satisfies the various courses and their respective examinations being offered by the different department. The difficulty is due to the great complexity of the construction of timetables for lectures and exams, due to the scheduling size of the lectures and examinations periods and the high number of constraints and criteria of allocation, usually circumvented with the use of little strict heuristics, based on solutions from previous years.

A timetable management system is designed and created to handle as much course data as fed while ensuring the avoidance of redundancy. An educational timetable must meet a number of requirements and should satisfy the desires of all entities involved simultaneously as well as possible. The timing of events must be such that nobody has more than one event at the same time.

The proposed timetabling system is designed to handle events of course lectures offered at JUET.

Based on the above event, the system would have only one module which is the Course Lecture Timetable Module.

PROBLEM STATEMENT

The systems available currently can build or generate a set of timetables, but still have issues with generating a clash-free and complete timetable. The tedious tasks of data introduction and revision of usually incomplete solutions are the bottleneck in these cases. JUET (most educational institutions) have resorted to manual generation of their timetables which according to statistics takes over a month to get completed and optimal. Even at the optimal stage of the manually generated timetable, there are still a few clashes and it is the lecturer that takes a clashing course that works out the logistics of the course so as to avoid the clash.

OBJECTIVE

The literature on and implementation of educational timetabling problems is quite scattered. Different research papers that have been brought out on timetabling may refer to the same type of institution but they mostly deal with different kinds of assignments, i.e., decisions like the timing of events, sectioning students into groups, or assigning events to locations. Moreover, each institution has its own characteristics which are reflected in the problem definition. Yet, there have been no leveling grounds for developing a system that can work for most of these institutions.

The aim of this work is the generation of course schedules while demonstrating the possibility of building these schedules automatically through the use of computers in such a way that they are optimal and complete with little or no redundancy through the development of a viable lecture timetabling software.

The primary objective is to be able to optimize the algorithm used in today's timetable systems to generate the best of timetabling data with fewer or no clashes.

The secondary objective is to expand the scope of timetable automation systems by making it generic thereby bringing about uniformity in the creation of timetables as it applies to different universities or educational institutions i.e. will be able to generate timetables that fit the requirements of any academic institution.

RESEARCH METHODOLOGY

This research is concerned with the problem of constructing timetables for JUET. JUET timetable construction problems are interesting objects to study because neither modelling nor solving them is straightforward. It is difficult to make a clear-cut distinction between acceptable and not acceptable timetables. Because of the large diversity in acceptance criteria, realistic timetable construction problems are multidimensional. Each dimension may introduce its own characteristic aspects that add to the complexity of the problem. Therefore, only heuristic solution approaches without known performance guarantees are practically feasible.

As a result of the large data input a timetable management system is supposed to handle, a linear method or algorithm cannot be employed to handle such validation and generation, hence the usage of a heuristic method. The heuristic method to be used in this study is the genetic algorithm. The genetic algorithm is one that seeks to find the most optimal solutions where the search space is great and conventional methods is inefficient.; it works on a basis of the Darwinian evolution theory.

SCOPE OF THE STUDY

The boundaries of this work take into consideration all academic institutions from the primary level to the higher institution level. It will only involve the technical skills of one academic personnel and a few data collaborators to handle the data input and constraint specifications. The proposed system will be able to handle as much data input as possible i.e. the course data, halls data, lecturer's data etc.

SIGNIFICANCE OF THE STUDY

Outlined below is the significance of the proposed system and the improvements on existing systems that are featured in this system:

- The proposed system will provide an attractive graphical front-end which acts as the main point of interaction with user and data collaborators.
- It will also improve flexibility in timetable construction.
- The system will be able to generate reports on conflicting constraint specifications. The system will seek to improve on previous versions of timetable generating system.

- Stress in creating timetable manually will be greatly reduced as output would be automated.
- The system will save time.
- Productivity will be improved.
- The system can be revised i.e. its backend can be revised.

RESEARCH OUTLINE

This report contains a further four sections. Chapter 2 gives further background information while reviewing in detail the workings of existing systems. Chapter 3 discusses the structure, design and internal workings of each module in the project, it also details the tasks required to complete the project, and a timescale to complete them in. Chapter 4 details the backend of the system and shows the development and testing of each stage in the project. Chapter 5 presents my summary, conclusions and recommendations. The final section lists the references used while writing this report

CHAPTER TWO

REVIEW OF RELEVANT LITERATURE

INTRODUCTION

A Timetable or schedule is an organized list, usually set out in tabular form, providing information about a series of arranged events: in particular, the time at which it is planned these events will take place. They are applicable to any institution where activities have to be carried out by various individuals in a specified time frame. From the time schools became organized environments, timetables have been the framework for all school's activities. As a result, schools have devoted time, energy and human capital to the implementation of nearly optimal timetables which must be able to satisfy all requirements constraints as specified by participating entities.

The class lecture timetabling problem is a typical scheduling problem that appears to be a tedious job in JUET twice a year. The problem involves the scheduling of classes, students, teachers and rooms at a fixed number of time-slots, subject to a certain number of constraints. An effective timetable is crucial for the satisfaction of educational requirements and the efficient utilization of human and space resources, which make it an optimization problem. Traditionally, the problem is solved manually by trial and hit method, where a valid solution is not guaranteed. Even if a valid solution is found, it is likely to miss far better solutions. These uncertainties have motivated the scientific study of the problem, and to develop an automated solution technique for it. The problem is being studied for more than four decades, but a general solution technique for it is yet to be formulated.

The automated timetabling and scheduling is one of the hardest problem areas already proven to be NP-Complete and it is worthy of note is that as educational institutions are challenged to grow in number and complexity, their resources and events are becoming harder to schedule, hence the choice of this project topic which entails investigating the performance of Genetic Algorithm on the optimality of timetabling problems under predefined constraints.

REVIEW OF RELEVANT EXISTING THEORIES AND TECHNOLOGIES

Solutions to timetabling problems have been proposed since the 1980s. Research in this area is still active as there are several recent related papers in operational research and artificial intelligence journals. This indicates that there are many problems in timetabling that need to be solved in view of the availability of more powerful computing facilities and advancement of information technology.

The problem was first studied by Gotlieb (1962), who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students, one teacher, and any number of times which could be chosen freely. Since then the problem is being continuously studied using different methods under different conditions. Initially it was mostly applied to schools. Since the problem in schools is relatively simple because of their simple class structures, classical methods, such as linear or integer programming approaches could be used easily. However, the gradual consideration of the cases of higher secondary schools and universities, which contain different types of complicated class-structures, is increasing the complexity of the problem. As a result, classical methods have been found inadequate to handle the problem, particularly the huge number of integer and/or real variables, discrete search space and multiple objective functions.

This inadequacy of classical methods has drawn the attention of the researchers towards the heuristic-based non-classical techniques. Worth mentioning non-classical techniques that are being applied to the problem are genetic algorithms, neural networks and tabular search algorithms. However, compared to other non-classical methods, the widely used are the genetic/evolutionary algorithms (GAs/EAs). The reason might be their successful implementation in a wider range of applications. Once the objectives and constraints are defined, EAs appear to offer the ultimate free lunch scenario of good solutions by evolving without a problem solving strategy.

Since 1995, a large amount of timetabling research has been presented in the series of international conferences on Practice and Theory of Automated Timetabling (PATAT). Papers on this research have been published in conference proceedings, see e.g., (Burke & Carter, 1997) and (Burke & Erben, 2000), and three volumes of selected papers in the Lecture Notes in Computer Science series, see (Burke & Ross, 1996), (Burke & Carter, 1998), and (Burke & Erben, 2001). Additionally, there is a EURO working group on automated timetabling (EURO-WATT) which meets once a year regularly sends out a digest via e-mail, and maintains a website with relevant information on timetabling problems, e.g., a bibliography and several benchmarks.

Fang (1994), in his doctoral thesis, investigates the use of genetic algorithms to solve a group of timetabling problems. He presents a framework for the utilization of genetic algorithms in solving timetabling problems in the context of learning institutions. This framework has the following important points, which give you considerable flexibility: a declaration of the specific constraints of the problem and use of a function for evaluation of the solutions, advising the use of a genetic algorithm, since it is independent of the problem, for its resolution.

Gröbner (1997) presents an approach to generalize all the timetabling problems, describing the basic structure of this problem. Gröbner proposes a generic language that can be used to describe timetabling problems and its constraints.

Chan (1997) discusses the implementation of two genetic algorithms used to solve class-teacher timetabling problems for small schools.

Oliveira (Oliveira and Reis, 2000) presents a language for representation of the timetabling problem, the UniLang intends to be a standard suitable as input language for any timetabling system. It enables a clear and natural representation of data, constraints, quality measures and solutions for different timetabling (as well as related) problems, such as school timetabling, university timetabling and examination scheduling.

Fernandes (2002) classified the constraints of class-teacher timetabling problems in constraints strong and weak. Violations to strong constraints (such as scheduling a teacher in two classes at the same time) result in an invalid timetable. Violations to weak constraints result in a valid timetable, but affect the quality of the solution (for example, the preference of teachers for certain hours).

Eley (2006) in PATAT'06 presents a solution to the exam timetable problem, formulating it as a problem of combinatorial optimization, using algorithms Ant, to solve

Analyzing the results obtained by the various works published, we can say what the automatic generation of schedules is capable of achieving. Some works show that when compared with the schedules manuals in institutions of learning real, the times obtained by the algorithms for solving the class-teacher timetabling problem are of better quality, since, uses some function of evaluation.

There are two main problems in timetabling. The first one is related to the combinatorial nature of the problems, where it is difficult to find an optimal solution because it is impossible to enumerate all nodes in such a large

search space. The second one is related to the dynamic nature of the problems where variables and constraints are changing in accordance with the development of JUET. Therefore, a timetabling system must be flexible, adaptable and portable, otherwise the users will not use the system optimally or even as decision aids such as for storing, retrieving, and printing timetables, when the timetable planning decisions are made manually. In addition, JUET has adopted a semester system which gives freedom to students to choose subjects provided that all prerequisites are satisfied. This situation further complicates the construction of a timetable.

Various techniques have been proposed to solve timetabling problems. These techniques are neural networks, heuristics, graph coloring, integer programming, genetic algorithms, knowledge-based, and constraint logic programming. The models formulated by some of these techniques cannot be easily reformulated or customized to support changes, hence the selection of the genetic algorithm for the implementation of this project.

TIMETABLING AS AN NP-COMPLETE PROBLEM

In computational complexity theory, the complexity class NP-complete (abbreviated NP-C or NPC, NP standing for Nondeterministic Polynomial time) is a class of problems having two properties:

- ☐ Any given solution to the problem can be verified quickly (in polynomial time); the set of problems with this property is called NP.
- ☐ If the problem can be solved quickly (in polynomial time), then so can every problem in NP.

Although any given solution to the timetabling problem can be verified quickly, there is no known efficient way to locate a solution in the first place; indeed, the most notable characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases very quickly as the size of the problem grows.

When solving the timetabling problem, we are usually looking for some solution, which will be the best among others. The space of all feasible solutions (series of desired solutions with some more desirable than others) is called search space (also state space). Each point in the search space represents one feasible solution which can be "marked" by its value or fitness for the problem. The solution is usually one point in the search space.

As a result of comparative fact finding and exhaustive study of existing systems, Genetic Algorithms have been the most prominently used in generating near-optimal solutions to timetabling problems, hence its usage in the implementation of this project.

THOROUGH EXAMINATION OF THE GENETIC ALGORITHM

A BRIEF HISTORY OF GENETIC ALGORITHMS

The earliest instances of what might today be called genetic algorithms appeared in the late 1950s and early 1960s, programmed on computers by evolutionary biologists who were explicitly seeking to model aspects of natural evolution. It did not occur to any of them that this strategy might be more generally applicable to artificial problems, but that recognition was not long in coming: "Evolutionary computation was definitely in the air in the formative days of the electronic computer". By 1962, researchers such as G.E.P. Box, G.J. Friedman,

W.W. Bledsoe and H.J. Bremermann had all independently developed evolution-inspired algorithms for function optimization and machine learning, but their work attracted little follow-up. A more successful development in this area came in 1965, when Ingo Rechenberg, then of the Technical University of Berlin, introduced a technique he called evolution strategy, though it was more similar to hill-climbers than to genetic algorithms. In this technique, there was no population or crossover; one parent was mutated to produce one offspring, and the better of the two was kept and became the parent for the next round of mutation. Later versions introduced the idea of a population. Evolution strategies are still employed today by engineers and scientists, especially in Germany.

The next important development in the field came in 1966, when L.J. Fogel, A.J. Owens and M.J. Walsh introduced in America a technique they called evolutionary programming. In this method, candidate solutions to problems were represented as simple finite-state machines; like Rechenberg's evolution strategy, their algorithm worked by randomly mutating one of 9

these simulated machines and keeping the better of the two (Mitchell, 1996; Goldberg, 1989). Also like evolution strategies, a broader formulation of the evolutionary programming technique is still an area of ongoing research today. However, what was still lacking in both these methodologies was recognition of the importance of crossover.

As early as 1962, John Holland's work on adaptive systems laid the foundation for later developments; most notably, Holland was also the first to explicitly propose crossover and other recombination operators. However, the seminal work in the field of genetic algorithms came in 1975, with the publication of the book *Adaptation in Natural and Artificial Systems*. Building on earlier research and papers both by Holland himself and by colleagues at the University of Michigan, this book was the first to systematically and rigorously present the concept of adaptive digital systems using mutation, selection and crossover, simulating processes of biological evolution, as a problem-solving strategy. The book also attempted to put genetic algorithms on a firm theoretical

footing by introducing the notion of schemata. That same year, Kenneth De Jong's important dissertation established the potential of GAs by showing that they could perform well on a wide variety of test functions, including noisy, discontinuous, and multimodal search landscapes).

These foundational works established more widespread interest in evolutionary computation. By the early to mid-1980s, genetic algorithms were being applied to a broad range of subjects, from abstract mathematical problems like bin-packing and graph coloring to tangible engineering issues such as pipeline flow control, pattern recognition and classification, and structural optimization (Goldberg, 1989).

At first, these applications were mainly theoretical. However, as research continued to proliferate, genetic algorithms migrated into the commercial sector, their rise fuelled by the exponential growth of computing power and the development of the Internet. Today, evolutionary computation is a thriving field, and genetic algorithms are "solving problems of everyday interest" in areas of study as diverse as stock market prediction and portfolio planning, aerospace engineering, microchip design, biochemistry and molecular biology, and scheduling at airports and assembly lines. The power of evolution has touched virtually any field one cares to name, shaping the world around us invisibly in countless ways, and new uses continue to be discovered as research is ongoing. And at the heart of it all lies nothing more than Charles Darwin's simple, powerful insight: that the random chance of variation, coupled with the law of selection, is a problem-solving technique of immense power and nearly unlimited application.

CHAPTER THREE

DESCRIPTION

- Our body is made up of trillions of **cells**. Each cell has a core structure (**nucleus**) that contains your **chromosomes**.
- Each **chromosome** is made up of tightly coiled strands of deoxyribonucleic acid (**DNA**). **Genes** are segments of DNA that determine **specific traits**, such as eye or hair color. You have more than 20,000 genes
- A gene **mutation** is an alteration in your DNA. It can be inherited or acquired during your lifetime, as cells age or are exposed to certain chemicals. Some changes in your genes result in genetic disorders
- Natural Selection: Darwin's theory of evolution: only the organisms best adapted to their environment tend to survive and transmit their genetic characteristics in increasing numbers to succeeding generations while those less adapted tend to be eliminated.
- A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators
- Genetic algorithms are implemented as a computer simulation in which population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions.
- Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.
- Evolution usually starts from a population of randomly generated individuals and happens in generations.
- In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population.
- The new population is then used in the next iteration of the algorithm.
- Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

- If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

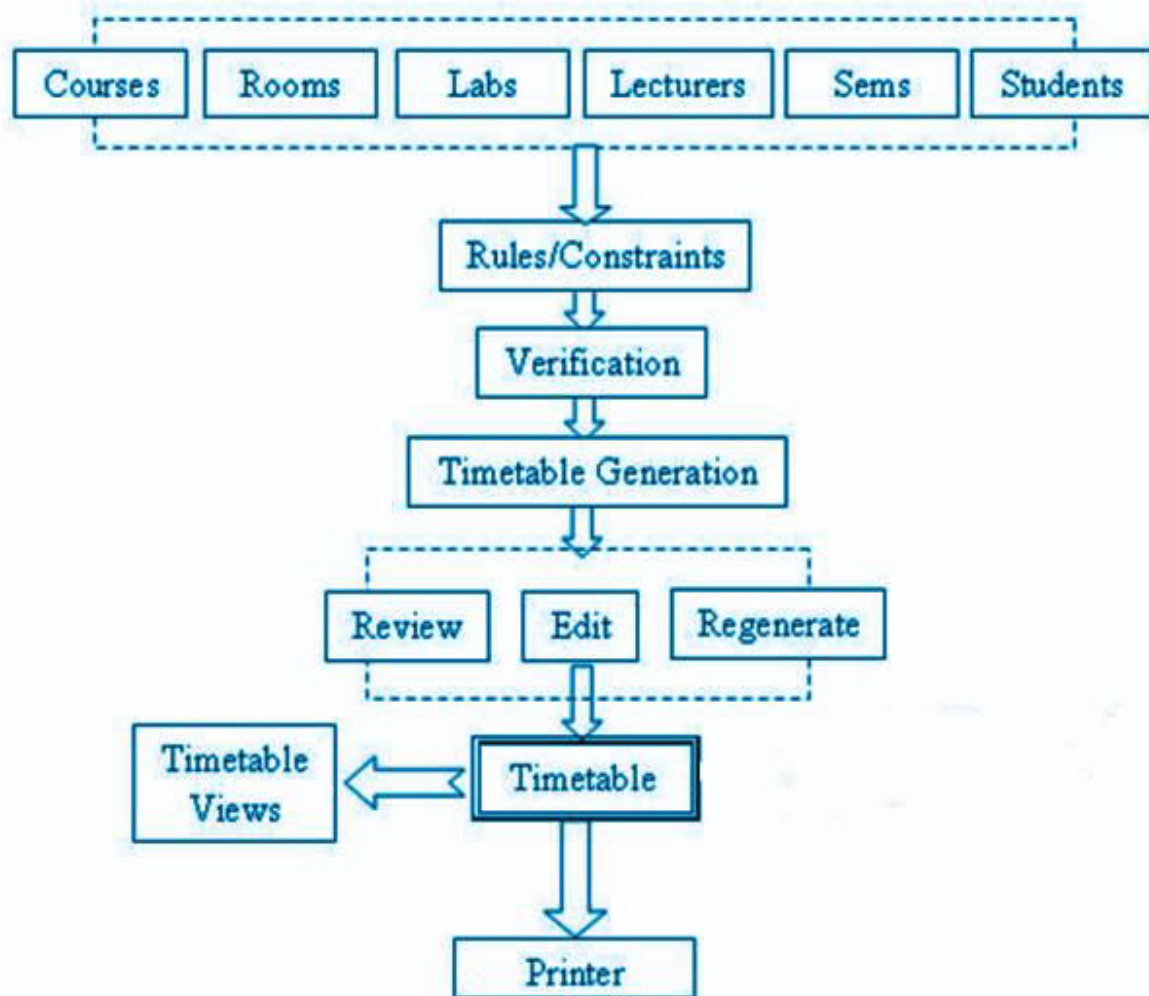
The Genetic Algorithm - a brief overview

Before we can use a genetic algorithm to solve a problem, a way must be found of encoding any potential solution to the problem. This could be as a string of real numbers or, as is more typically the case, a binary bit string. We will refer to this bit string from now on as the chromosome. A typical chromosome may look like this:

10010101110101001010011101101110111111101

At the beginning of a run of a genetic algorithm a large population of random chromosomes is created. Each one, when decoded will represent a different solution to the problem at hand. Let's say there are N chromosomes in the initial population. Then, the following steps are repeated until a solution is found.

- Test each chromosome to see how good it is at solving the problem at hand and assign a **Fitness Score** accordingly. The fitness score is a measure of how good that chromosome is at solving the problem to hand.
- Select two members from the current population. The chance of being selected is proportional to the chromosomes fitness. **Roulette Wheel Selection** is a commonly used method.
- Dependent on the **Crossover rate** crossover the bits from each chosen chromosome at a randomly chosen point.
- Step through the chosen chromosomes bits and flip dependent on the **Mutation rate**.
- Repeat step 2, 3, 4 until a new population of N members has been created.
- Keep repeating until required fitness is achieved.



CHAPTER FOUR

SYSTEM ANALYSIS AND DESIGN

INTRODUCTION

System Analysis is the study of a business problem domain to recommend improvements and specify the business requirements and priorities for the solution. It involves analysing and understanding a problem, then identifying alternative solutions, choosing the best course of action and then designing the chosen solution.

It involves determining how existing systems work and the problems associated with existing systems. It is worthy to note that before a new system can be designed, it is necessary to study the system that is to be improved upon or replaced, if there is any.

THE EXISTING SYSTEM

REVIEW OF THE EXISTING SYSTEM

Timetabling is the whole process concerned with making a timetable having events arranged according to a time when they take place which must be subject to the timing constraints of each entity placed in the table. University timetabling in this context refers to the rigorous task educational center staff in a Covenant University undergo to draw up timetables that satisfies various courses that should compulsorily be inherent in the final timetable solution.

These courses are usually taught by varied lecturers in different departments who may also wish to specify some timing constraints on their courses. Given all the courses and course details, the academic staff is charged with the responsibility of creating a near optimal timetable which would serve as a guide for academic activities in the university.

The traditional manual timetabling system is time-consuming, resource-intensive, involves many steps and requires re-processing the same data several times.

ADVANTAGES OF THE EXISTING SYSTEM

The timetable generation process by the education center staff is:

- Subjective and can be made better through collaboration with the different entities involved.

LIMITATIONS OF THE EXISTING SYSTEM

The traditional manual generations of timetables encounter a lot of problems which may include the following:

- Repeated time allocations may be made for a particular course thereby leading to data redundancy.
- A lot of administrative errors may occur as a result of confusing time requirements.
- Timetable generation by center staff may have a slow turnaround.
- Final generated timetable may not be near optimal as a result of clashing course requirements and allocations.
- It generates a lot of paperwork and is very tasking.
- It is not flexible as changes may not be easily made.

THE PROPOSED SYSTEM

REVIEW OF THE PROPOSED SYSTEM

The proposed systems were developed to solve the timetabling problem being faced by universities every academic year and reduce high cost and slow turnaround involved in the generation of near-optimal timetables.

The system has capabilities for input of the various courses, halls of lectures, departments, programs, buildings, lecturers and the specification of a few constraints from which the timetable is constructed. The proposed timetabling system for this project seeks to generate near optimal timetables using the principles of genetic algorithm (selection and crossover).

ADVANTAGES OF THE PROPOSED SYSTEM

The timetable generation process by the academic staff is:

- · Unlike the manual timetabling system, the system offers flexibility.
- · It utilizes minimal processing/computing power.
- · It greatly reduces the time needed to generate near-optimal timetables.
- · It provides an easy means for data entry and revision through an intuitive interface.
- · It increases productivity.
- · Timetables generated are between to 60% - 80% optimum.
- · It almost eliminates paperwork.

- · It simplifies the timetabling process.

LIMITATIONS OF THE PROPOSED SYSTEM

The following are the challenges in the system due to time constraints:

- · The proposed system can only generate timetables based on a few hard course constraints.
- · The proposed system can only generate timetables for two streams.
- · Timetable generated by this system is still subject to revision by academic center staff.
- · Not all of the genetic algorithm principles are implemented in the system.

SYSTEMS DESIGN

System design is the specification or construction of a technical, computer-based solution for the business requirements identified in a system analysis. It gives the overall plan or model of a system consisting of all specifications that give the system its form and structure i.e. the structural implementation of the system analysis.

MODELLING THE SYSTEM

Modelling a system is the process of abstracting and organizing significant features of how the system would look like. Modelling is the designing of the software applications before coding. Unified Modelling Language (UML) tools were used in modelling this system.

UML (UNIFIED MODELLING LANGUAGE) MODELLING

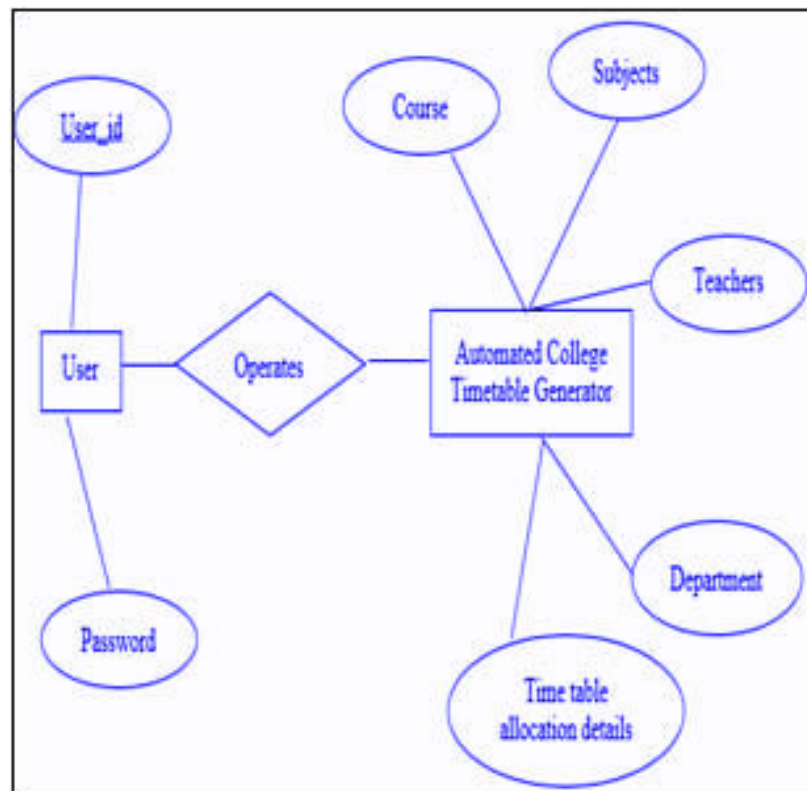
The Unified modelling language is an object-oriented system notation that provides a set of modelling conventions that is used to specify or describe a software system in terms of objects. The Unified Modelling Language (UML) has become an object modelling standard and adds a variety of techniques to the field of systems analysis and development hence its choice for this project.

UML offers ten different diagrams to model a system. These diagrams are listed below:

- Use case diagram
- Class diagram
- Object diagram
- Sequence diagram
- Collaboration diagram
- State diagram
- Activity diagram
- Component diagram
- Deployment diagram
- Package Diagram

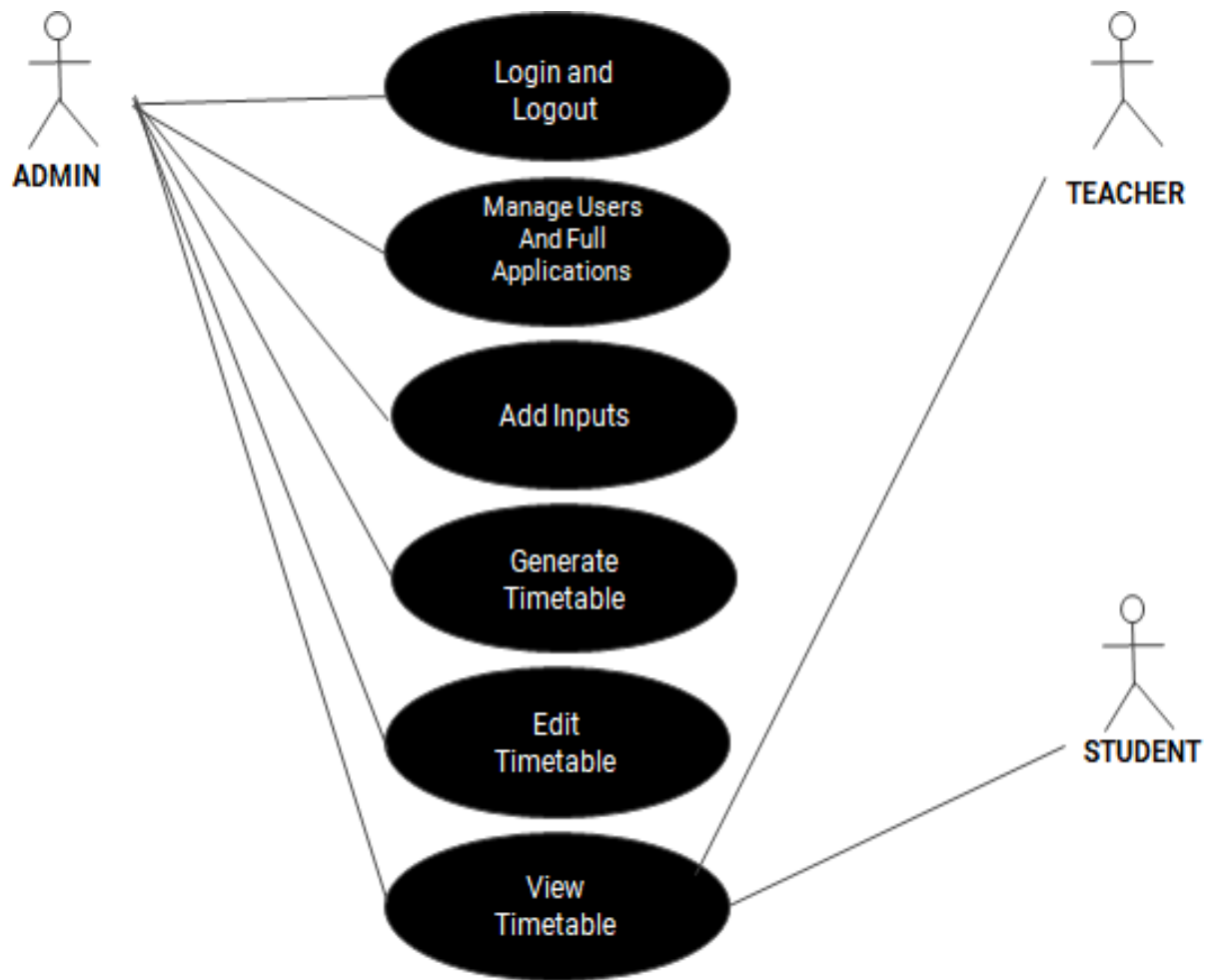
In this project, the Use case diagram, Class diagram, Sequence diagram, Activity diagram, Collaboration diagram, Component diagram and State diagram will be used for system modelling.

ER DIAGRAM



User Case Diagram

Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis of use case diagrams is on what a system does rather than how. They are used to show the interactions between users of the system and the system. A use case represents the several users called actors and the different ways in which they interact with the system.



USER CASES

USERS/ACTORS

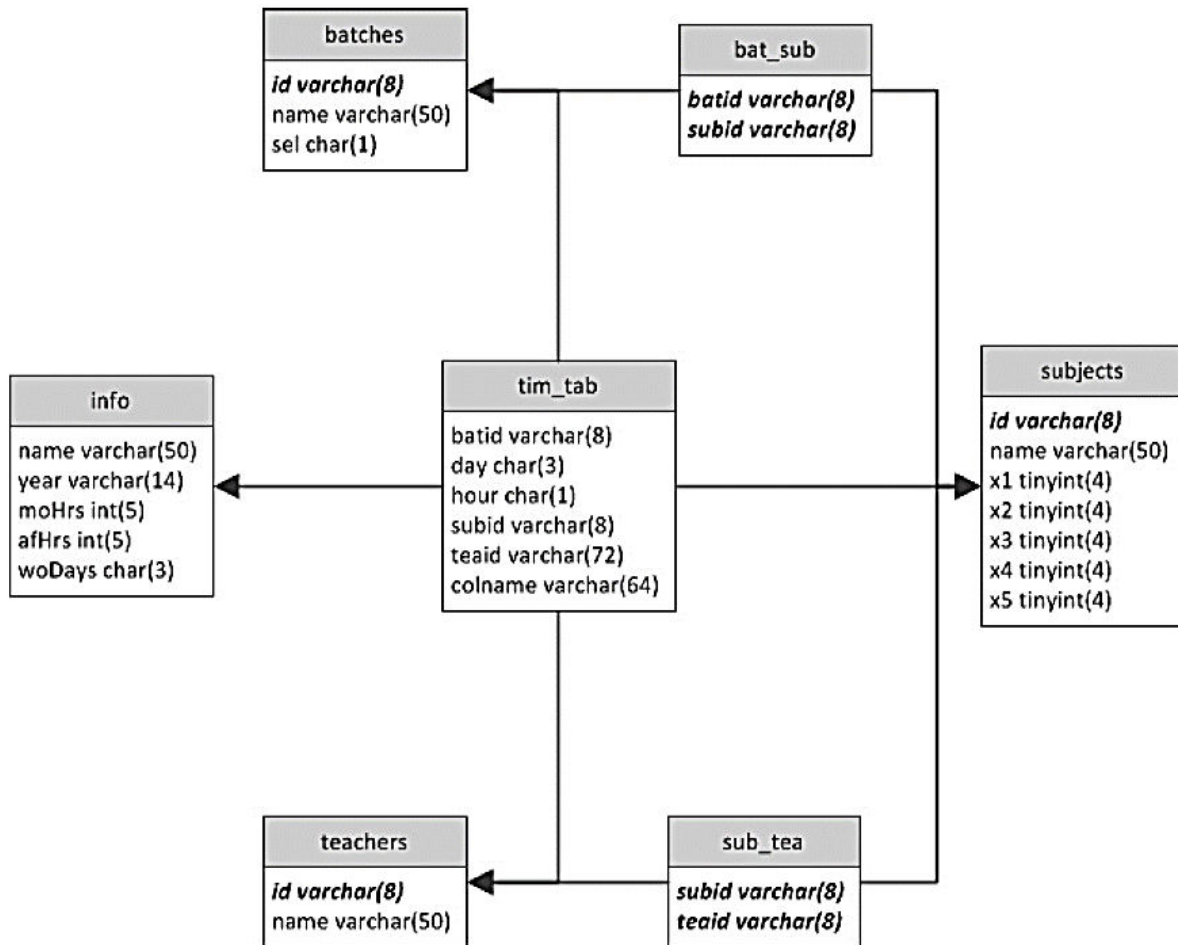
- Admin
- Teachers
- Students

Class Diagram

A class diagram is an organization of related objects. It gives an overview of a system by showing its classes and the relationships among them. Class diagrams only display what interacts but not what happens during the interaction hence they are static diagrams.

CLASSES

- · Lecturers
- · Buildings
- · Halls
- · Programs
- · Courses
- · Levels
- · Allocations
- · Front End Staff
- · Generator Module
- · File Writer



Sequence Diagram

A sequence graphically depicts how objects interact with each other via messages in the execution of a use case or operation. They illustrate how messages are sent and received between objects and the sequence of message transfer. It also details how operations are carried out according to the time of operation.

CLASSES

- Front End Staff
- Interface
- Course Allocation
- Generator Module
- Timetable Writer

MESSAGES

- Invoke
- Specify Input

- Input Buildings
- Input Halls
- Input Departments
- Input Programs
- Input Lecturers
- Input Courses
- Specify Timing Constraints
- Specify Other Constraints
- Add Input
- Set. allocations
- Generate Timetable
- Send Generated Timetable
- Print Out Generated Timetable

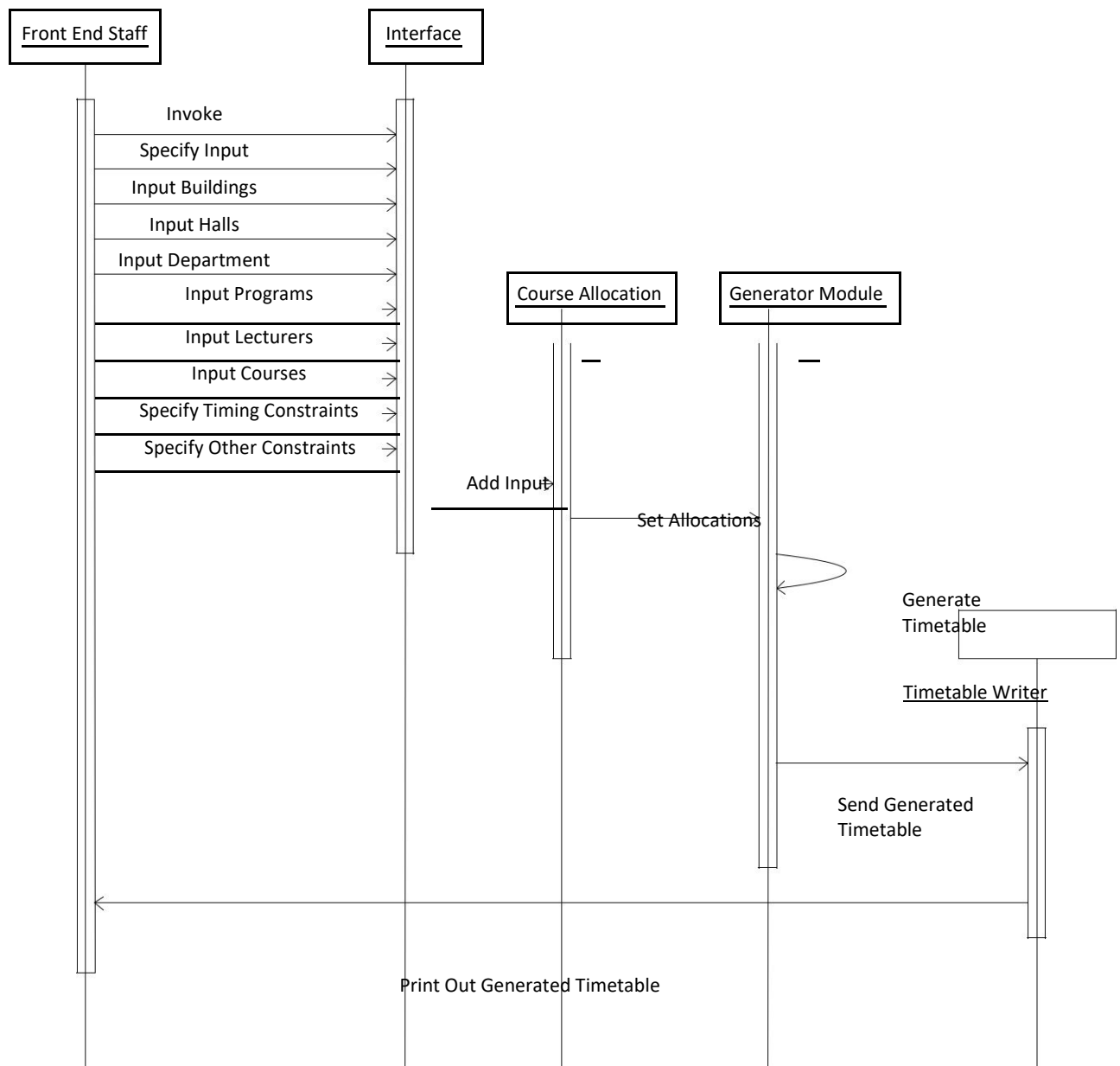


Figure 3.3.: Sequence Diagram to show how the different objects interact during the execution of the system

CHAPTER FIVE

SYSTEM IMPLEMENTATION

INTRODUCTION

The system implementation defines the construction, installation, testing and delivery of the proposed system. After thorough analysis and design of the system, the system implementation incorporates all other development phases to produce a functional system.

CHOICE OF PROGRAMMING LANGUAGE

Java is the language of choice for this project because of its high speed and low memory usage. The timetabling problem is combinatorial in nature hence the need for a programming language that has enhanced CPU optimization capabilities for the development of an algorithm like the genetic algorithm which optimizes search space and avoids local optima.

PROGRAM WRITING

The software implementation contains two major modules:

- **Crossover Module:** has a functionality of simulating the crossover of course population whose constraints are violated. Crossing over individual courses in the population attempts to reduce and or eliminate constraints violations. See Appendix A for source code.
- **Random Population Generation Module:** generates the initial random course population from the input supplied by the user.

SYSTEMS REQUIREMENTS

Below are the conditions a computer system on which the timetable software will be run:

HARDWARE REQUIREMENTS

- Processor should be Pentium 5 and above
- 512 Megabytes of RAM (or more)
- 1 Gigabyte of Free Disk Space

SOFTWARE REQUIREMENTS

- Windows XP (or higher)
- Java 8
- Database MySql
- HTML 5
- Cascading style sheets (CSS) J
- Javascript
- Bootstrap

CHAPTER SIX

SUMMARY, CONCLUSION AND RECOMMENDATIONS

SUMMARY

This study was carried out as is to reduce the intense manual effort being put into creating and developing university timetables. The timetable automation system currently is a conceptual work in progress but has the capability to generate near optimal timetables based on two unit courses with minimized course constraints.

CONCLUSION

Timetabling problem being the hard combinatorial problem that it is would take more than just the application of only one principle. The timetabling problem may only be solved when the constraints and allocations are clearly defined and simplified thoroughly and more than one principle is applied to it i.e. a hybrid solution (a combination of different solution techniques).

This research has been able to actualize a sub-implementation of a genetic algorithm which can be applied to input of 2-units courses.

RECOMMENDATIONS

In furtherance of this work, the following are recommended:

- The timetable system developed as the outcome of this project should be made open to avid students of computing who can collaborate and improve on the techniques and ideas inherent in this project.
- Further works on developing a timetabling system should be based on this research work so as to utilize the incremental model of software development.
- A collaborative model of timetabling system which utilizes a computer network can also be built which entails different departments and entities allocating courses and constraints concurrently while the system threads and reports clashes.

CHAPTER SEVEN

PROGRESS

CERTIFICATES



CERTIFICATE

SOLOLEARN

Issued 19 May, 2020

This is to certify that

Aditi Negi

has successfully completed the
PHP Tutorial course



Yeva Hyusyan
Chief Executive Officer

Certificate #1059-18564736

CERTIFICATE

SOLOLEARN

Issued 05 June, 2020

This is to certify that

Aditi Negi

has successfully completed the

CSS Fundamentals course



Yeva Hyusyan
Chief Executive Officer

Certificate #1023-18564736

CERTIFICATE

SOLOLEARN

Issued 25 May, 2020

This is to certify that

Aditi Negi

has successfully completed the

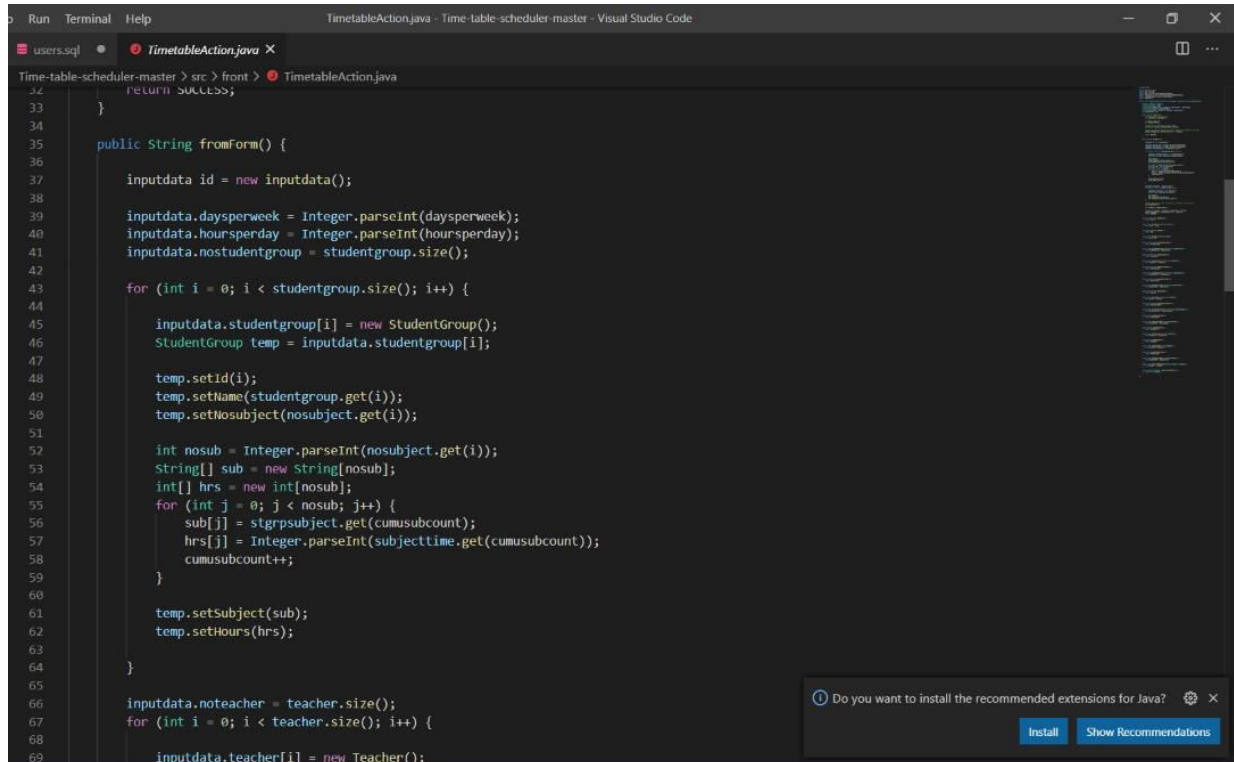
JavaScript Tutorial course



Yeva Hyusyan
Chief Executive Officer

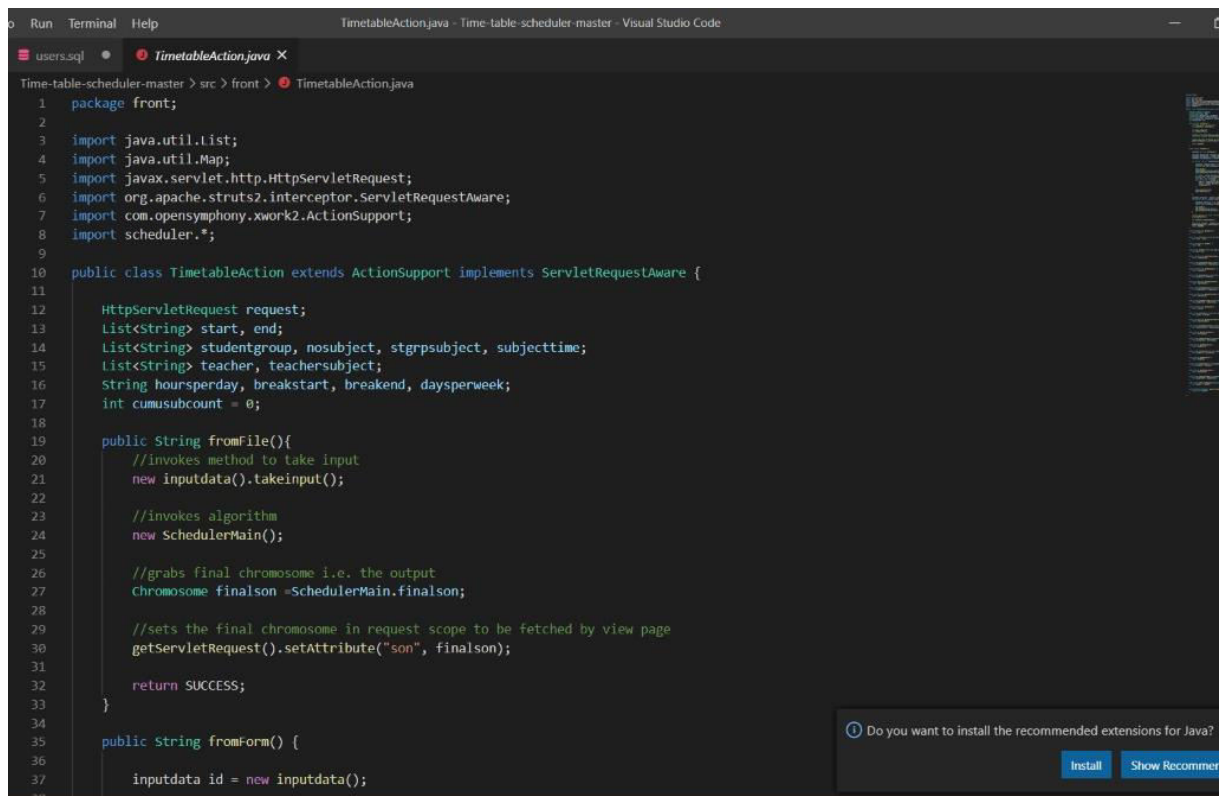
Certificate #1024-18564736

SCREENSHOTS



```
Time-table-scheduler-master > src > front > TimetableAction.java
32     return SUCCESS;
33 }
34
35 public String fromForm() {
36     inputdata id = new inputdata();
37
38     inputdata.daysperweek = Integer.parseInt(daysperweek);
39     inputdata.hoursperday = Integer.parseInt(hoursperday);
40     inputdata.nostudentgroup = studentgroup.size();
41
42     for (int i = 0; i < studentgroup.size(); i++) {
43         inputdata.studentgroup[i] = new StudentGroup();
44         StudentGroup temp = inputdata.studentgroup[i];
45
46         temp.setId(i);
47         temp.setName(studentgroup.get(i));
48         temp.setNosubject(nosubject.get(i));
49
50         int nosub = Integer.parseInt(nosubject.get(i));
51         String[] sub = new String[nosub];
52         int[] hrs = new int[nosub];
53         for (int j = 0; j < nosub; j++) {
54             sub[j] = stgrpsubject.get(cumusubcount);
55             hrs[j] = Integer.parseInt(subjecttime.get(cumusubcount));
56             cumusubcount++;
57         }
58
59         temp.setSubject(sub);
60         temp.setHours(hrs);
61     }
62
63     inputdata.noteacher = teacher.size();
64     for (int i = 0; i < teacher.size(); i++) {
65         inputdata.teacher[i] = new Teacher();
66     }
67 }
```

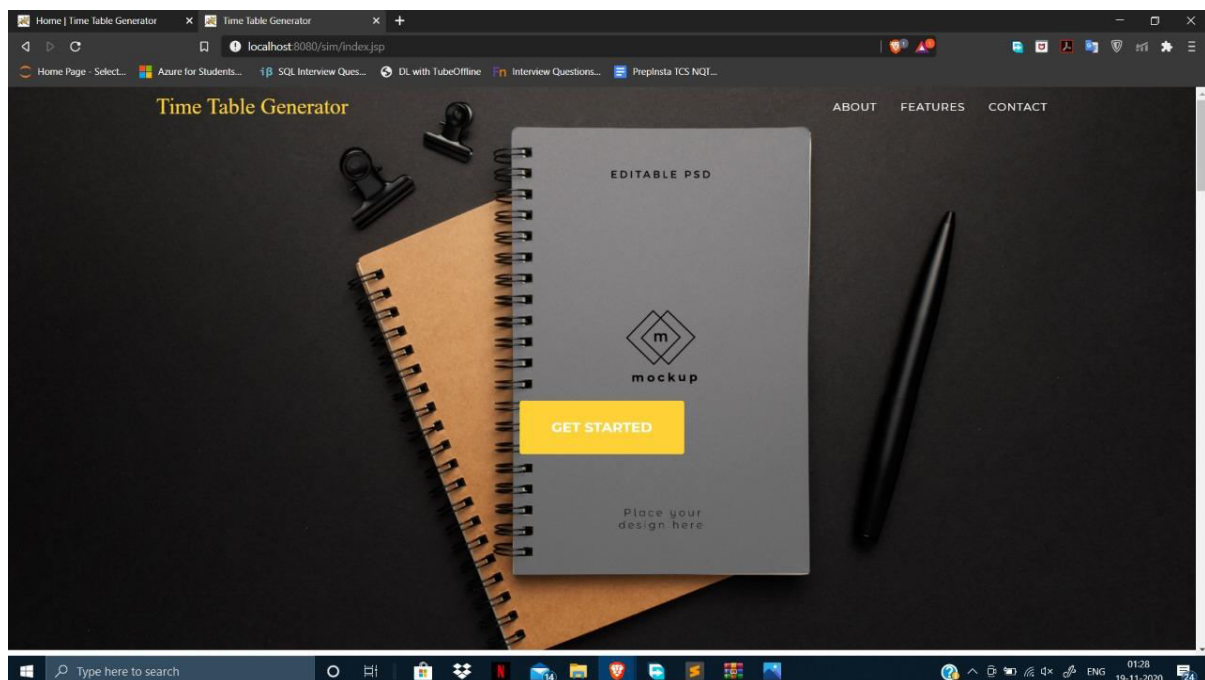
Do you want to install the recommended extensions for Java? [Install](#) [Show Recommendations](#)



```
Time-table-scheduler-master > src > front > TimetableAction.java
1 package front;
2
3 import java.util.List;
4 import java.util.Map;
5 import javax.servlet.http.HttpServletRequest;
6 import org.apache.struts2.interceptor.ServletRequestAware;
7 import com.opensymphony.xwork2.ActionSupport;
8 import scheduler.*;
9
10 public class TimetableAction extends ActionSupport implements ServletRequestAware {
11     HttpServletRequest request;
12     List<String> start, end;
13     List<String> studentgroup, nosubject, stgrpsubject, subjecttime;
14     List<String> teacher, teachersubject;
15     String hoursperday, breakstart, breakend, daysperweek;
16     int cumusubcount = 0;
17
18     public String fromFile(){
19         //invokes method to take input
20         new inputdata().takeinput();
21
22         //invokes algorithm
23         new SchedulerMain();
24
25         //grabs final chromosome i.e. the output
26         Chromosome finalson = SchedulerMain.finalson;
27
28         //sets the final chromosome in request scope to be fetched by view page
29         getServletRequest().setAttribute("son", finalson);
30
31         return SUCCESS;
32     }
33
34     public String fromForm() {
35         inputdata id = new inputdata();
36     }
37 }
```

Do you want to install the recommended extensions for Java? [Install](#) [Show Recommendations](#)

```
Run Terminal Help • users.sql - Time-table-scheduler-master - Visual Studio Code
users.sql
Time-table-scheduler-master > users.sql
28
29 CREATE TABLE IF NOT EXISTS `users` (
30   `id` int(4) NOT NULL,
31   `username` varchar(50) NOT NULL,
32   `password` varchar(50) NOT NULL,
33   `country` varchar(50) NOT NULL,
34   `email` varchar(50) NOT NULL
35 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
36
37 --
38 -- Dumping data for table `users`
39 --
40
41 INSERT INTO `users` (`id`, `username`, `password`, `country`, `email`) VALUES
42 (1, 'simple', 'prasad', 'india', 'simple11prasad@gmail.com');
43
44 --
45 -- Indexes for dumped tables
46 --
47
48 --
49 -- Indexes for table `users`
50 --
51 ALTER TABLE `users`
52   ADD PRIMARY KEY (`id`);
53
54 --
55 -- AUTO_INCREMENT for dumped tables
56 --
57
58 --
59 -- AUTO_INCREMENT for table `users`
60 --
61 ALTER TABLE `users`
62   MODIFY `id` int(4) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
63 /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
64 /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
65 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```



Home | Time Table Generator

localhost:8080/sim/form.jsp

HOME HOW TO USE? LOGOUT

PLEASE FILL-IN ALL REQUIRED DETAILS TO GENERATE YOUR COLLEGE TIME-TABLE.

Slots or Periods of study (per day):

Insert Break after period number (skip if no break):

No. of days (per week):

No of Teachers:

No. of Batches:

*subject names allotted to a batch must be from the subjects taught by teachers (Ignore case)
 **total hours of study for any batch must not exceed total available hours

Type here to search

(72) how to im x nodejs - MyS x MySQL = MyS x MySQL = MyS x View Time-Ta x (117) WhatsA x Meet - mi x Home | Time x

localhost:8080/Time_Table_Scheduler/main

HOME HOW TO USE? LOGOUT

☒ Friday
☒ Saturday
☐ Sunday

No of Teachers:

| | |
|---|--|
| Name: <input type="text" value="PS Banerjee"/> | Subject: <input type="text" value="NLP"/> |
| Name: <input type="text" value="Ajay Kumar"/> | Subject: <input type="text" value="DS"/> |
| Name: <input type="text" value="Prateek Pandey"/> | Subject: <input type="text" value="AP LAB"/> |
| Name: <input type="text" value="Babita Tiwari"/> | Subject: <input type="text" value="TOC"/> |
| Name: <input type="text" value="Rahul Pachauri"/> | Subject: <input type="text" value="SDF"/> |
| Name: <input type="text" value="Ratnesh Litoriya"/> | Subject: <input type="text" value="APS"/> |

No. of Batches:

ENG 23:43

Time Table Generator

HOME HOW TO USE? LOGOUT

Slots or Periods of study (per day):

4

Start: 10:30 End: 11:30

Start: 12:00 End: 01:00

Start: 02:00 End: 03:00

Start: 04:00 End: 05:00

Insert Break after period number (skip if no break):

15

No. of days (per week):

6

☒ Monday

☒ Tuesday

☒ Wednesday

☒ Thursday

View Time-Table

localhost:8080/Time_Table_Scheduler/view_timetable

BX

| | 10:30-11:30 | 12:00-13:00 | 15:00-16:00 | 16:15-17:15 |
|-------|---------------------|--------------------------|------------------------|------------------|
| Day 1 | | TOC Babita Tiwari | | |
| Day 2 | NLP PS Bannerjee | | OS LAB PS Bannerjee | |
| Day 3 | | | | |
| Day 4 | | AP LAB Prateek Pandey | | |
| Day 5 | | | | DS Ajay Kumar |
| Day 6 | | | | |

BY

| | 10:30-11:30 | 12:00-13:00 | 15:00-16:00 | 16:15-17:15 |
|-------|------------------------|-------------|-------------|-------------|
| Day 1 | OS LAB PS Bannerjee | | | |
| Day 2 | | | | |
| Day 3 | | | | |

BY

| | 10:30-11:30 | 12:00-13:00 | 15:00-16:00 | 16:15-17:15 |
|-------|--------------------------|-------------------|----------------------|------------------|
| Day 1 | OS LAB PS Bannerjee | | | |
| Day 2 | | | | |
| Day 3 | | | | |
| Day 4 | AP LAB Prateek Pandey | NLP Amit Rathi | | DS Ajay Kumar |
| Day 5 | | | TOC Babita Tiwari | |
| Day 6 | | | | |

BZ

| | 10:30-11:30 | 12:00-13:00 | 15:00-16:00 | 16:15-17:15 |
|-------|------------------|-------------|-------------|---------------------|
| Day 1 | | | | |
| Day 2 | DS Ajay Kumar | | | NLP PS Bannerjee |

Day 4

| | | | | |
|-------|--------------------------|-------------------|----------------------|------------------|
| | AP LAB Prateek Pandey | NLP Amit Rathi | | DS Ajay Kumar |
| Day 5 | | | TOC Babita Tiwari | |
| Day 6 | | | | |

BZ

| | 10:30-11:30 | 12:00-13:00 | 15:00-16:00 | 16:15-17:15 |
|-------|--------------------------|-------------|-------------|---------------------|
| Day 1 | | | | |
| Day 2 | DS Ajay Kumar | | | NLP PS Bannerjee |
| Day 3 | OS LAB PS Bannerjee | | | |
| Day 4 | | | | |
| Day 5 | TOC Babita Tiwari | | | |
| Day 6 | AP LAB Prateek Pandey | | | |

REFERENCES

1. A. Cornelissen, M.J. Sprengers and B.Mader (2010). "OPUS-College Timetable Module Design Document" Journal of Computer Science 1(1), 1-7.
2. Abramson D. & Abela J. (1992). "A parallel genetic algorithm for solving the school timetabling problem." In Proceedings of the 15th Australian Computer Science Conference, Hobart, 1-11.
3. Adam Marczyk (2004). "Genetic Algorithms and Evolutionary Computation ". Available online at <http://www.talkorigins.org/faqs/genalg/genalg.html>.
4. Al-Attar A.(1994). White Paper: "A hybrid GA-heuristic search strategy." AI Expert, USA.
5. Alberto Colomi, Marco Dorigo, Vittorio Manniezzo (1992). "A Genetic Algorithm to Solve the Timetable Problem" Journal of Computational Optimization and Applications, 1, 90-92.
6. Bufo M., Fischer T., Gubbels H., Hacker C., Hasprich O., Scheibel C., Weicker K., Weicker N., Wenig M., & Wolfangel C. (2001). Automated solution of a highly constrained school timetabling problem - preliminary results. EvoWorkshops, Como-Italy.
7. Burke E, Elliman D and Weare R (1994)."A genetic algorithm for university timetabling system." Presented at the East-West Conference on Computer Technologies in Education, Crimea, Ukraine.
8. Carrasco M.P.& Pato M.V.(2001). "A multiobjective genetic algorithm for the class/teacher timetabling problem." In Proceedings of the Practice and Theory of Automated Timetabling (PATAT'00), Lecture Notes in Computer Science, Springer, 2079, 3-17.
9. Chan H. W. (1997). "School Timetabling Using Genetic Search." 2th International Conference on the Practice and Theory of Automated Timetabling, PATAT'97.
10. Coello Carlos (2000). "An updated survey of GA-based multiobjective optimization techniques." ACM Computing Surveys, 32(2), 109-143.
11. Costa D.(1994). "A tabular search algorithm for computing an operational timetable." European Journal of Operational Research, 76(1), 98-110.
12. Datta D., Deb K., & Fonseca, C.M.(2006). Multi-objective evolutionary algorithm for university class timetabling problem, In Evolutionary Scheduling, Springer-Verlag Press.
13. David A Coley (1999). An Introduction to Genetic Algorithms for Scientists and Engineers, 1st ed. World Scientific Publishing Co. Pte. Ltd.

14. Dawkins Richard (1996). *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design*. W.W. Norton.
15. De Gans O.B.(1981). "A computer timetabling system for secondary schools in the Netherlands". *European Journal of Operations Research*,7, 175-182.
16. Deb K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons Ltd, England.
17. Deb K., Agarwal S., Pratap A., & Meyarivan T. (2002). "A fast and elitist multi-objective genetic algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation*, 6(2), 182-197.
18. Eley M. (2006). "Ant Algorithms for the Exam Timetabling Problem." 6th International Conference on the Practice and Theory of Automated Timetabling, PATAT'06.
19. Fang H. L. (1994). "Genetic Algorithms in Timetabling Problems." PhD Thesis, University of Edinburgh.
20. Fernandes C. (2002). "Infected Genes Evolutionary Algorithm for School Timetabling." WSES International Conference.
21. Fleming Peter and R.C. Purshouse (2002). "Evolutionary algorithms in control systems engineering: a survey." *Control Engineering Practice*, 10, 1223-124

APPENDICES

Appendix A: Terminologies involved

Permutation Encoding

In **permutation encoding**, every chromosome is a string of numbers, which represents number in a **sequence**. Other encoding techniques are Binary encoding Value encoding tree Encoding.

Elitism

A practical variant of the general process of constructing a new population is to allow the best organism(s) from the current generation to carry over to the next, unaltered. This strategy is known as *elitist selection* and guarantees that the solution quality obtained by the GA will not decrease from one generation to the next.

Roulette Wheel Selection

It is a selection procedure in which the possibility of selection of a chromosome is directly proportional to its fitness.

Single Point Crossover

It is that type of crossover between two chromosomes in which the chromosomes are broken at a single point and then crossed. When single point crossover happens in this project, it is made sure that chromosome is not so cut that it intersects the timetable of any student group.

Swap Mutation

It is the type of mutation technique in which the chromosomes are so mutated that two portions of the chromosome get exchanged resulting in a new chromosome.

Appendix B: Algorithm

- First of all an initial generation of chromosomes is created randomly and their fitness value is analysed.
- New Generations are created after this. For each generation, it performs following basic operations:
 - a. First of all preserve few fittest chromosomes from the previous generation as it is. This is called Elitism and is necessary to preserve desired characteristics in the coming generations .
 - b. Randomly select a pair of chromosomes from the previous generation. Roulette wheel selection method has been used here in this project.
 - c. Perform crossover depending on the crossover rate which is pretty high usually. Here single point crossover has been used.

d. Perform mutation on the more fit chromosome so obtained depending on the mutation rate which is kept pretty small usually.

- Now analyze the fitness of the new generation of chromosomes and order them according to fitness values.
- Repeat creating new generations unless chromosomes of desired fitness value i.e. fitness=1, are obtained.