**CHAT BOT SYSTEM FOR GCE USING**

**ARTIFICIAL INTELLIGENCE**

**A MINI PROJECT REPORT**

|  |  |
| --- | --- |
| ***Submitted by*** | |
| **SHUNMUGA RAJESHWARN S** | **950820104044** |
| **RAMASAMY** | **9508201040** |
| **KAMESH** | **950818104018** |

**ARUL RAJAVEL 95082010**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GOVERNMENT COLLEGE OF ENGINEERING, TIRUNELVELI**

# ANNA UNIVERSITY : CHENNAI 600 025

**JUNE 2021**

# BONAFIDE CERTIFICATE

Certified that this mini project report **“CHAT BOT SYSTEM FOR GCE USING AI”** is the bonafide work of **“SHUNMUGA RAJESHWARAN S-950820104044, RAMASAMY M -950818104028 ,KAMESH R 95082010 and ARUL RAJAVEL - 9508201”** who carried out the project work under my supervision**.**

**SIGNATURE, SIGNATURE, Dr.G.TAMILPAVAI,M.E.,Ph.D., MRS. M. MAHIL,M.E. , HEAD OF THE DEPARTMENT, ASSISTANT PROFESSOR,**

Dept. of Computer Science and Engg, Dept. of Computer Science and Engg, Government College of Engineering, Government College of Engineering, Tirunelveli-627007. Tirunelveli-627007.

Submitted for mini-project viva- voce held at Government College of Engineering , Tirunelveli on .

**Internal Examiner External Examiner**

# ACKNOWLEDGEMENT

We render special thanks to **Dr.P.LATHA M.E,Ph.D.,** Principal of Government College of Engineering, Tirunelveli for permitting us to do this project successfully.

We express our sincere thanks to our Head of the department **Dr.G.TAMILPAVAI,M.E.,PhD.,** Department of Computer science and engineering for permitting us to do this project.

We take this opportunity to express our thanks to our guide **Mrs. M. MAHIL M.E.,** Assistant Professor, Computer Science and Engineering, for her encouragement and valuable support in completing this project work in time.

We would also like to thank our faculty adviser **Dr.E.SIVA SANKARI M.E.,Ph.D.,** Assistant Professor, Department of Computer Science & Engineering for her valuable suggestions in bringing out this project successfully

We extend our sincere thanks to **Mrs. SIVA PRIYA, M.E.,** for her encouragement and valuable support in completing the project work.

We sincerely thank all the teaching and non-teaching staff members of this department for their assistance in completing our project.

Finally, we thank our family members and friends for their co-operation and selfless help and support.

**ABSTRACT**

Time table scheduling is an essential part of all educational institutions. It sure is a time and work consuming process. The manual process has high possibilities of errors as well. This project focusses on the automation of the process in an effective way. The main requirement of the application is to provide the details about the branch, subjects, number of labs, total number of period and details about the lab assistance. Then the Application generates the time table according to the need. The problem is to design and implement an algorithm to create a semester course time table by assigning timeslots and rooms to a given set of courses to be run that semester under given constraints. The constraints include avoiding clashes of time-slots and rooms, assigning appropriate rooms and appropriate number of slots and contact hours to the courses etc. Here we have used “Genetic Algorithm” for the process. This algorithm provides effective results when compared with the other contemporary algorithms.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

[Acknowledgement iii](#_TOC_250004)

Abstract iv

1. INTRODUCTION 1
2. LITERATURE REVIEW 2
3. PROBLEM STATEMENT 11

[EXISTING SYSTEM 12](#_TOC_250003)

[PROPOSED SYSTEM 13](#_TOC_250002)

[BLOCK DIAGRAM 16](#_TOC_250001)

1. REQUIREMENTS SPECIFICATION 17

[HARDWARE REQUIREMENTS 17](#_TOC_250000)

SOFTWARE REQUIREMENTS 17

1. METHODOLOGY 18

TOURNAMENT SELECTION 19

|  |  |  |
| --- | --- | --- |
| **6**  **7**  **8** | **RESULTS**  **CONCLUSION REFERENCES** | **24**  **26**  **27** |

**LIST OF FIGURES**

**FIG NO. FIGURE NAME PAGE NO.**

|  |  |  |
| --- | --- | --- |
| 3.1 | Block Diagram | 11 |
| 5.1 | Example | 15 |
| 6.1 | Before Tournament selection screenshot | 16 |
| 6.2 | After Tournament selection screenshot | 10 |
| 6.3 | Final Timetable | 11 |

# CHAPTER 1

**INTRODUCTION**

Time table concerns all the activities with regards to producing a schedule that must be subjective from different constraints. Timetable can be defined as the optimization of given activities , actions or events to a set of objects in space-time matrix to satisfy a set of desirable constraints.

A key factor in running an educational centre or basically an academic environment is the need of well planned, well throughout and clash-free timetable. Back in the days when technology was not in wide use, timetables were manually created by the academic institution.

A timetable problem consists of 4 parameters and they are:

* Set of time
* Set of available resources
* Set of scheduled contacts
* Set of constraints

The problem assigns time and resources to the contacts on such a way that the constraints will be satisfied. In various timetabling problems

, educational timetabling has been generally examined from practical standpoint.

# CHAPTER 2

**LITERATURE REVIEW**

# ”Improving a Lecture Timetabling System for University-Wide Use.” -Ben paechter,R.C.Rankin,Andrew

During the academic year 1996/97 the authors were commissioned by their institution to produce an automated timetabling system for use by all departments within the Faculty of Science. The system had to cater for the varying requirements of all the departments, be easy to use, robust, expandable, and timetable 100% of events fully automatically within a reasonable time. The timetables produced had not only to be workable, but also had to be ‘good’ with respect to management defined criteria. The work was intended as a pilot study for later extension to the whole institution. This paper describes the enhancements to the user interface and timetabling engine that were found to be necessary to meet the more extensive needs of a faculty.

# “University timetable through conceptual modelling”.- Jonathan Lee, Shang-pin ma,Lien fu lai,Nien-lin Hsueh

A number of approaches have been proposed in tackling the timetabling problem, such as operational research, human- machine interaction, constraint programming, expert systems, and neural networks. However, there are still several key challenges to be addressed: easily reformulated to support changes, a generalized framework to handle various timetabling problems, and ability to incorporate knowledge in the timetabling system. In this article, they propose an automatic software engineering approach, called task-based conceptual graphs, to addressing the challenges in the timetabling problem. Task-based conceptual graphs provide the automation of software development processes including specification, verification, and automatic programming. Maintenance can be directly performed

on the specifications rather than on the source code; moreover, hard and soft constraints can be easily inserted or removed. A university timetabling system in the Department of Computer Science and Information Engineering at National Central University is used as an illustrative example for the proposed approach.

# “Building Teaching Timetables Using Random Variables: Algorithms and Techniques”- Christos Panagiotaukopoulas

As noted by a number of researchers, teaching timetable construction is a multi-parameter and complex task. Various techniques have been proposed in order to tackle timetable constraints. This study focuses on the School Timetable Builder (STB) application, which enables the construction of teaching timetables for secondary education. The STB application works with random variables and outputs results considerably fast, evaluating all the constraints and conditions that should be taken into account. Moreover, the algorithms and the techniques used by STB application are discussed.

# “Automated Timetable Generation using Genetic Algorithm” - Shraddha Thakare

In this paper they have proposed system to design an effective model for timetable scheduling using a genetic algorithm. The objective of the research is to create a model using genetic algorithm, which can be effectively used to resolve difficult combinatorial optimization problem. The algorithm was tested on small and large cases of the problem. Algorithm performance was significantly enhanced with modification of basic genetic operators, which restrain the creation of new conflicts in the individual. The scheduling solution presented in this paper is an adaptive one, with a primary aim of obtaining best the optimal solutions.

# Automatic Timetable Generation using Genetic Algorithm- Williams Kehinde Oladipo

The generation of timetables has always been tedious right from time and apart from being tedious, the timetable created has always been filled with series of errors and mistakes. So many techniques have been put forward to solving this problem. In this paper, genetic Algorithm was used by creating a group of time series randomly from a given time and courses in other to find a solution to the timetable problems. The courses thus formed are evaluated with the help of the evaluation function. system administrator logs into the system and then the administrator input the courses with their codes and the unit. At that point, the admin will keep adding until the number of courses needed has been inputted. The admin can remove a course that has been inputted in the case of error. After inputting the courses, it moves to the next page where all the lecture halls or rooms that will be used will be inputted. After inputting these, the system then generates the timetable system. This technique (genetic algorithm) used helps in reducing to barest minimum, errors and mistakes in encountered in developing an automatic timetable.

# Senior high school course scheduling using genetic algorithm- N G A P H Saptarin

Manual course scheduling can be very complex and take a long time, even sometimes violate hard and soft constraints. Soft constraints usually relate to teacher and student preferences as part of the schedule. The purpose of this study was to apply genetic algorithms (GA) to prevent the violation of hard constraints and minimize the violation of soft constraints. The GA in this study distributing population in some groups. The distributed GA generate groups of population and then after each iteration, the migration between groups will be conducted based on given probability of migration. The distributed GA applied to avoid the premature convergence that could occur in original GA. The probability of migration observed in this study was 0, 0.1, 0.2, 0.3, 0.4, and 0.5. The study shown that the \ distributed GA succeed to prevent violation of hard constraints, minimize the soft constraints violation and avoid the premature convergence.

# CHAPTER 3 PROBLEM STATEMENT

# EXISTING SYSTEM:

# Manual mode:

The process is done in manual mode which firstly consumes most of the time of the individual. Efficiency of the solution is not always guaranteed as human mind is vulnerable to errors. With more number of constraints, the difficulty and the **time** of the completion of process **increases majestically**. Re-correction of timetables are much difficult in manual mode.

# NP- complete Algorithm:

Np- complete algorithm can be used to construct a time table. NP-complete problems are the hardest problems in NP set. A decision problem L is NP-complete if: L is in NP (Any given solution for NP-complete problems can be verified quickly, but there is **no efficient known solution**).

# PROPOSED SYSTEM:

Here we propose that, **‘GENETIC ALGORITHM’** can be used to provide the most efficient solution for this problem.

* The algorithm has a generation to generation approach in which we can consider the nodes as ‘**population**’, which are individually termed as ‘**chromosomes**’.
* To move to the next generation, the population undergoes ‘**crossover**’ mutation.
* Each generation is compared with the previous generation by evaluating the ‘**fitness values**’.

# BLOCK DIAGRAM

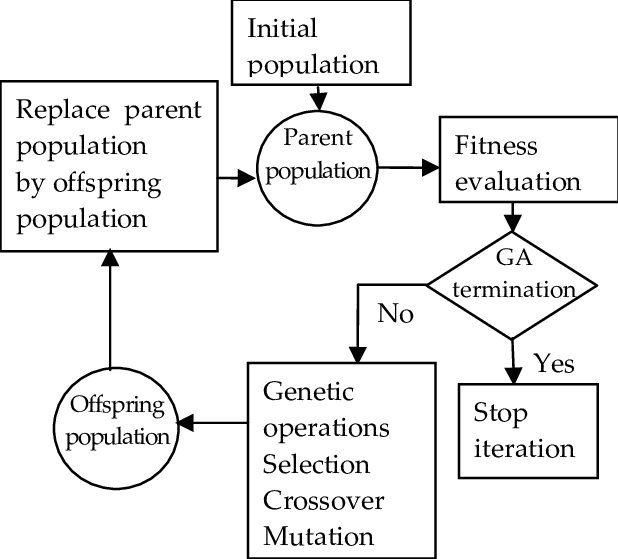


Fig 3.1 Block Diagram

# WORKING PRINCIPLE:

Genetic algorithms (GAs) are stochastic search methods based on the principles of natural genetic systems. They perform a search in providing an optimal solution for evaluation (fitness) function of an optimization problem. GAs deal simultaneously with multiple solutions and use only the fitness function values. While solving an optimization problem using GAs, each solution is coded as a string (called "chromosome") of finite length over a finite alphabet A. Each string or chromosome is considered as an individual. A collection of M ( M is finite) such individuals is called a population. GAs start with a randomly generated population of size M. In each iteration, a new population of same size is generated from the current population using three basic operations on the individuals of the population. The operators are (i) Selection, (ii) Reproduction/Crossover and (iii) Mutation. The new population obtained after selection, crossover and mutation is then used to generate another population. Note that the number of possible populations is always finite since A is a finite set and M is finite. Sometimes the knowledge about the best string obtained so far is preserved either within the population or in a separate location outside the population; in that way the algorithm would report the best value found, among all possible coded solutions obtained during the whole process. Such a model is called ELITIST model

# A. Genetic Algorithm

We used Genetic Algorithm to solve class scheduling problem with custom selection methods. Genetic algorithm is population based heuristic method largely applied to solve the scheduling problems. It is search methods based on principles of natural selection and genetics. The basic working of genetic algorithm is given by following algorithm.

As stated algorithm Genetic Algorithm starts with initializing the population with some random solutions. After initialization we evaluated these random solutions to determine the survival capacity of solution. Selection is an important activity used here to select proper parents to generate an individual called ospring which improves probability of survival of good solution. We applied crossover and mutation with probability [0.-0.8] to gener-

ate ospring from selected solutions so called parents and then again evaluate this ospring for its quality. If this ospring is better than available individuals in population then will replace this ospring into population otherwise discarded. We are repeating this process till termination condition is reached. Termination condition is total number of times you want to execute this loop activity

# TOURNAMENT SELECTION:

Tournament Selection is a Selection Strategy used for selecting the fittest candidates from the current generation in a Genetic Algorithm. These selected candidates are then passed on to the next generation. In a K-way tournament selection, we select k-individuals and run a tournament among them. Only the fittest candidate amongst those selected candidates is chosen and is passed on to the next generation.

In this way many such tournaments take place and we have our final selection of candidates who move on to the next generation. It also has a parameter called the selection pressure which is a probabilistic measure of a candidate’s likelihood of participation in a tournament. If the tournament size is larger, weak candidates have a smaller chance of getting selected as it has to compete with a stronger candidate.

The selection pressure parameter determines the rate of convergence of the GA. More the selection pressure more will be the Convergence rate. GAs are able to identify optimal or near-optimal solutions over a wide range of selection pressures. Tournament Selection also works for negative fitness values.

# ALGORITHM:

Algorithm --

1. Select k individuals from the population and perform a tournament amongst them
2. Select the best individual from the k individuals
3. Repeat process 1 and 2 until you have the desired amount of population

Let us have a 3-way tournament selection and our desired population size is 6 and the initial population with their fitness scores is [1, 2, 3, 4, 5, 6]. Our first tournament will look something like this (see the diagram) and the winner candidate with fitness value 6 moves on to the next generation.

After the first tournament we have our selected population as [6]. After a few such tournaments, we might have a selected population as [6, 6, 6, 5, 4, 3]. It might even be [6, 6, 5, 4, 3, 2]. Hence the fittest candidate is more likely to be selected for the next generation.

If the best candidate is selected with probability p then the next best candidate will be selected with a probability of p\*(1-p) and the next one with p\* (1-p)2 and so on

# Example:

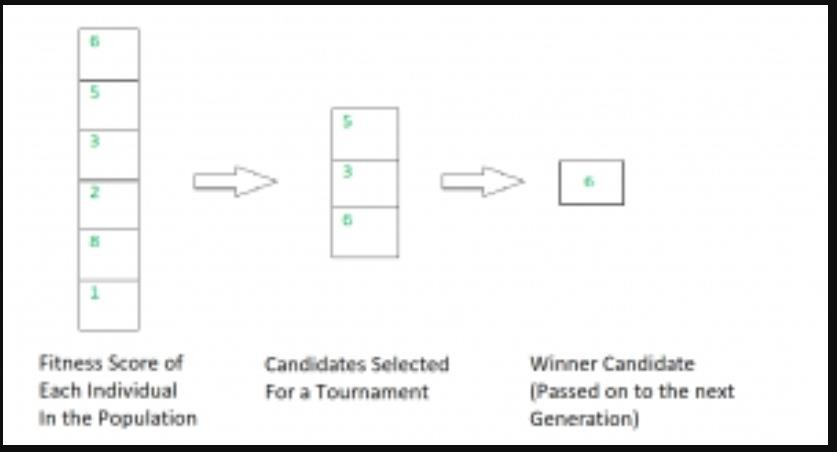


Fig 5.1

# CHAPTER 4 REQUIREMENT SPECIFICATIONS

The requirement specifications for the Time table management system are listed below:

# SOFTWARE REQUIREMENTS:

Database : mysql Frameworks : Java- Netbeans

# HARDWARE REQUIREMENTS:

Processor :Pentium(R) Dual-core CPU RAM :256 MB

Hard Disk :40 GB

# CHAPTER 5

**METHODOLOGY**

# MODULE 1:

* Each and every hour of a day is considered as a chromosome.
* The first generation is randomly assigned.
* MUTATION RATE:
  + To explore and exploit.
  + 0.1
  + CROSSOVER RATE:
  + Probability in which two chromosomes crossover
  + 0.9
* FITNESS VALUE:
  + Chromosome with no conflict will have fitness value=1.
  + Fitness value = 1/(no. of conflicts + 1).

# MODULE 2:

* TOURNAMENT SELECTION:
  + To explore and exploit.
  + Fitter candidates are selected and are participated in a tournament to select the fittest.

Tournament selection size = 3

# Genetic Algorithm:

Scheduling classes for a college timetable is often encountered with constraints (hard and soft) due to diversity as compared to a school timetable where the requirements are highly limited [29]. The problems associated with hard constraints need to be resolved to produce a functional solution. To optimize the performance of the scheduling it is important to address the issues linked to soft constraints. However, a careful approach should be formulated without compromising the solutions to hard constraints to minimize serious disruptions to the system. As such, a straightforward scheduling system as for a small school system cannot be successfully applied to a complex organization without turning to a different approach for quick and optimum results

There may be conflicts between multiple soft constraints and as a result a tradeoff will need to be reached between them .As an example, a class might have 12 students, and as such the soft constraint will assign a suitable classroom that can accommodate the students. However, there may be a classroom that the professor prefers which can hold up to 45 students. The class scheduler will hopefully find a preferred configuration as professor preference is considered in the soft constraints.

The Problem: The class-scheduling problem will be based on the following data.

* Available Professors
* Available Rooms
* Timeslots
* Student groups

A college timetable is different from a grade school timetable in that there may be free time periods on a college student’s schedule. This depends on the number of courses taken by the student. Each class will be assigned a timeslot, a professor, a room, and a student group by the class scheduler. The total number of classes that need to be scheduled can be obtained by summing the number of student groups multiplied by the number of modules each student group is enrolled in. For each class scheduled by this application the following hard constraints will be considered. • Classes can only be scheduled in free classrooms. • A professor can only teach one class at any one time. • Classrooms must be big enough to accommodate the student group. When encoding the class schedule, certain class properties are needed. They are: the timeslot the class is scheduled for, the professor teaching the class, and the classroom required for the class.

# Initial population of chromosomes:

A genetic algorithm creates an initial population of chromosomes with each chromosome (organism) representing a complete solution to a given problem. The genes which constitute the chromosomes are initialized to random values. Based on a function specific to a given

problem each chromosome is evaluated for its “fitness” which defines the quality of the solution.

# Suitability of chromosomes to mate:

In a genetic algorithm, creation of an improved population is the aim of mating the most suitable chromosomes in a population. It results in the production of offspring (chromosomes) which join the existing population of chromosomes to become part of a mating population in subsequent generations. For the genetic algorithm to function effectively a “fitness function” in the form of a numeric score has to be designed to evaluate the solutions (chromosomes). In nature there is no assignment of a score the -- organisms just die or survive.

# Implementation of Algorithm:

A timetable needs to be built around the following criteria : the rooms, professors, timeslots, courses/modules, and student groups. The room class will contain information about the classroom, such as the roomID, room number and the capacity of students that can be accommodated. This class will accept a roomId, a room number and the capacity as well as provide methods to get the room’s properties. The timeslot class represents the day of the week and time that a class takes place. It contains the timeslotId and the timeslot details. The professor class accepts a professorID and professor name properties. It also makes an allowance to retrieve this information as well.

A timetable class will encapsulate all these objects and will co-

ordinate how different constraints interact with each other. This class will also parse a chromosome and create a candidate timetable to be appraised and recorded. The timetable class serves two purposes. First, it is aware of all the available rooms, timeslots, professors, modules and groups. Second, it can read a chromosome, generate a subset of classes from that chromosome, and help evaluate the fitness of the chromosome. This class consists of two significant methods. The create class method and the calculation of clashes method. When creating a class for the timetable, an individual (chromosome) must be accepted, read and assigned information (timeslot, room, professor) to each class. As a result, the create class method uses a genetic algorithm and the subsequent chromosome to try different combinations of timeslots, rooms, and professors. The timetable class stores this information for future use. The clashes method then checks each one of the classes that has been built and count the number of clashes. A clash is a hard constraint violation.

Example of clashes: room is too small, conflict with professor and timeslot, conflict with room and timeslot. Clashes are used later in the genetic algorithm to calculate the fitness value. As each class is compared to all other classes, a “clash” is added, if any of the hard constraints are violated. The total number of clashes are calculated. This is then used to calculate the fitness value. The fitness value is the inverse of the number of clashes (1/clashes+1). If there are no clashes then the fitness value will be 1. The main method in the timetable class creates the timetable and initializes it with all of the available courses,

timeslots, rooms, modules, groups and professors. As a result, tournament selection and uniform crossover is used for the genetic algorithm. A termination check is set up such that the deciding factors are the number of generations and the fitness factor. Combining both of these factors will terminate the genetic algorithm either after a certain number of generations or if it finds a valid solution. As such the fitness value depends on the number of broken constraints. As a result, the perfect solution will have a fitness value of 1.

# Module 1:

The first generation of the table is randomly assigned and the conflicts in the time tables of the generation can be seen in the right corner along with fitness value.

# Module 2:

The next generation is assigned keeping the timetable with highest fitness value unchanged. The other chromosomes undergo mutation and crossover with the respective rates mentioned above for the best possible outcome. The process is repeated until we get an optimal solution for which the fitness value is 0.

# Module 3:

The process is terminated once we get a chromosome whose fitness value is 1. The optimal solution is arrived and the timetable is finally displayed as the result

# CHAPTER 6 RESULTS

**MODULE 1:**

# Before Tournament Selection:

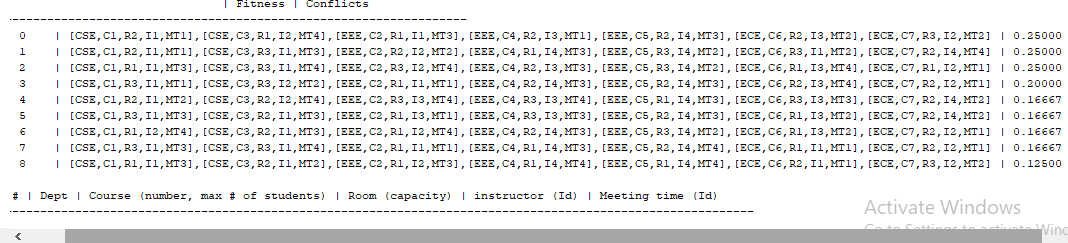


Fig 6.1

# MODULE 2:

**After Tournament Selection:**

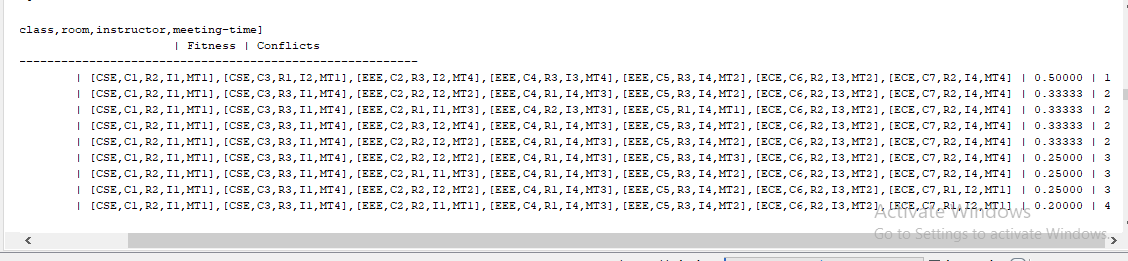


Fig 6.2

# MODULE 3:

**FINAL SCHEDULE:**

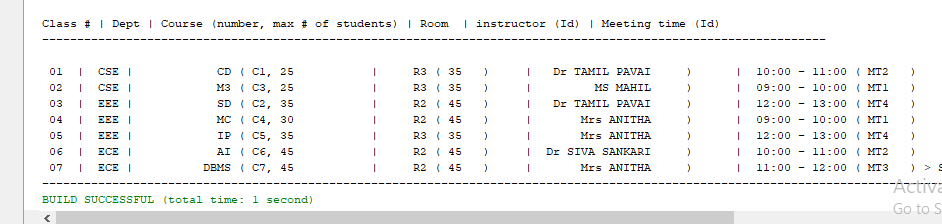


Fig 6.3

# CHAPTER 7 CONCLUSION

The timetable management system generated using genetic algorithm provides effective results which could take months to solve by human brain in seconds. It is quite user friendly and can be used by anybody who can access it with accessibility. It leads to instant generation of timetable and also instant correction of it too. It also gives us the possibility of multiple timetables for the same constraints and data every time we run the algorithm. And it is also great that it is an efficient solution each and every time. Researches are being held for the improvisation of the algorithm in every possible means. Since it gives option for us to change the data and constraints anytime , it is more efficient for multiple use.

# CHAPTER 8 REFERENCES

1. Jose Joaquim Moreira(2008)- “A System for Automatic construction of

exam timetable using Genetic Algorithm”

1. Edmund K. Burke,Sanja Petrovic(2002)- “ Recent research directions in

automated timetabling”.

1. N.D. Thanh, “Solving Timetabling Problem Using Genetic and Heuristic Algorithms”,

Eighth ACIS International Conference on Software Engineering, Artificial

Intelligence, Networking, and Parallel/Distributed Computing, hal. 472-477,

2007.

1. Enmund Burke, David Ellimand and Rupert Weare. (1994). “A Genetic Algorithm Based

University Timetabling System“, August 2011,

1. Q. Zhao, An Introduction to Genetic Algorithms, 2007.
2. Rich DC (1995): A Smart Genetic Algorithm for University Timetabling. In Burke E and

Ross P (Eds)