A Project Report

on

Smart Class Scheduler

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under the Supervision of
Mr. Arvind Kumar
Associate Professor
Department of Computer Science and Engineering

Submitted By

19SCSE1010667 - Aman Bisht

19SCSE1010914 - Mohit Bajaj

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled "**Smart Class Scheduler**" in partial fulfillment of the requirements for the award of the

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of JULY-2021 to DECEMBER-2021, under the supervision of **Mr. Arvind Kumar, Associate Professor**, Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

19SCSE1010667 - Aman Bisht

19SCSE1010914 - Mohit Bajaj

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor
Mr. Arvind Kumar, Associate Professor

<u>CERTIFICATE</u>

The Final Thesis/Project/ Dissertation Viva-Voce examination of 19SCSE1010667 - Aman Bisht ,19SCSE1010914 - Mohit Bajaj has been held on _________________ and his/her work is recommended for the award of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

ABSTRACT

Timetabling Problem is Hard problem which is very difficult to solve by using conventional methods. A lot of complex constraints need to be addressed for development of an efficient algorithm to solve this timetabling problem. Therefore, there is a great requirement for an application distributing the course evenly and without collisions. There are various tools available for generating timetable. This tool can reduce our manual work of generating timetable but limitations of this tool is that it requires more time, gives less accuracy and also error rate is high. So, our aim here is to develop a simple, easily understandable, efficient and portable application, which could automatically generate good quality timetable within seconds. In the present scenario, all the college related work such as making the defaulter list of students and measuring the performance of a teacher according to feedback given by students is done manually. All these tasks are time consuming and also require a lot of efforts and resources. To solve these problems our system uses Sentiment analysis API for generating feedback of teacher. System generates teacher's performance graph according to feedback given by student. The system will also send alert message to students if their attendance is less than 75%. The main purpose of our system is to reduce the workload of teachers and also be a cost effective and a quick respondent system.

This study presents a generic solution to the problem of scheduling. The majority of the previously presented heuristics address the topic from the perspective of the students. This solution, on the other hand, is based on the availability of professors for a certain time slot. While all hard restrictions (such as teacher availability) are carefully resolved, the scheduling method given in this work is adaptive, with the primary goal of resolving the issue of lecture and subject clashes involving teachers.

Introduction

The class timetabling problem is a scheduling method that has received a lot of attention and has a lot of consequences in the domains of operational research and artificial intelligence. Gotlieb was the first to investigate the issue, formulating a class-teacher scheduling problem based on the assumption that each lecture had one group of students and one teacher, with the combination of teacher and pupils being freely chosen.[1].Dynamic alterations in the context of scheduling issues has been researched [17]. you'll find a survey of existing approaches to dynamic scheduling.[16]. Due to the magnitude of the real-world problem, practically all effective solutions are heuristic in nature and do not ensure optimality. repo [13], which deal with various situations of problem settings [14], are among the well-known outcomes. When creating a schedule, consideration is paid to the efficient use of resources such as the classroom, the teacher, and so on.' This becomes a highly time-consuming process.

Timetabling concerns all activities with regard to making a timetable that must be subjective to different constraints. 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, Educational institutes 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 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 focus of this report is mainly on tertiary institutes because almost all tertiary institutes face the problem of timetable scheduling [9]. In any case, the current method of manual timetable scheduling is considered very inadequate as it is time consuming and provides very high chances violations on the timetable. Timetable scheduling is described as the sharing out of resources for factors under predefined constrictions so that it maximizes the likelihood of allocation or reduces the violation of restrictions set [Shengxiang and Sadaf (2009)]. In the new approach, genetic algorithms seem to be a way forward to solving this problem. GAs is a dominant force in the overall purpose as optimization tools which model principles of evolution [Davis (2007)], in another aspect they can be said to be adaptive systems that inspired by the nature of evolution. [6], also mentioned that they are most often capable of finding globally optimal solutions even in the most complex of search spaces, thus in this case GAs are used to automate the scheduling of classes, and in contrast they are known to keep several distinct outcomes in the form of a population [14]. [18] Continues to say that, the distinct outcomes known as parents are selected from the total population and mated together to form a new offspring called a child. [2] Substantiated on this by mentioning that GAs work on a populated strategy and by combining together to form new optimal solutions. The new offspring generated is further mutated, adopting the biological concept, in order to bring about diversity into the total population [7] and [4].

Literature Review

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 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. Trying to develop a software which helps to generate Timetable for an Institution automatically. By looking at the existing system we can understand that timetable generation is done manually. Manually adjust the timetable when any of the faculty is absent, and this is the big challenge for Automatic Timetable Generator that managing the timetable automatically when any of the faculty is absent. As we know all institutions/organizations have its own timetable, managing and maintaining these will not be difficult. Considering workload with this scheduling will make it more complex. As mentioned, when Timetable generation is being done, it should consider the maximum and minimum workload that is in a college. In those cases, timetable generation will become more complex. Also, it is a time-consuming process.

Timetabling is recognized to be a non-polynomial complete issue, which means that there is no efficient technique to find a solution. Furthermore, the most striking feature of NP-complete problems is that no optimum solution to them is known. As a result, a heuristic technique is used to solve a timetabling problem. This heuristic approach yields a set of good solutions (but not necessarily the best solution).

A set of events (e.g., courses and exams) are allotted into a given number of timeslots (time periods) pursuant to a set of constraints in a generic educational timetabling problem, which often makes the problem highly difficult to solve in real-world settings [2]. Indeed, large-scale schedules, such as university timetables, may necessitate many hours of labor by qualified individuals or teams in order to develop high-quality timetables with optimal constraint satisfaction [7] and timetable optimization.

There are two kinds of restrictions. Constraints, both hard and soft. Hard limitations are those that cannot be violated while a timeline is being calculated. For example, in order for a teacher to be booked for a time slot, the teacher must be available at that time. Only when no hard limitation is violated is a solution acceptable. Soft constraints, on the other hand, are those that should be addressed as much as possible in the solution. For example, while a teacher's schedule is important, the emphasis is on creating a legitimate timetable, which can result in a teacher being available for a time slot. As a result, while addressing the timetabling problem, hard constraints must be followed while also attempting to satisfy as many soft requirements as feasible. Because of the problem's intricacy, the majority of the work has focused on heuristic algorithms

Heuristic optimization approaches are expressly focused at good viable solutions that may not be optimal in cases when the complexity of the problem or the limited time available do not allow for an accurate solution. In general, two questions arise: I How quickly is the solution computed? (ii) How near is the answer to being optimal? A tradeoff between speed and quality is frequently required, which is addressed by running simpler algorithms several times, comparing results acquired with more intricate ones, and comparing the effectiveness of different heuristics. The empirical evaluation of the heuristic technique is based on the analytical difficulties involved in the worst-case conclusion of the problem. In its most basic form, the scheduling task consists of mapping pre-allocated class, instructor, and room combinations onto time slots.

One possible strategy is as follows: A tuple is defined as a specific combination of identifiers such as class, teacher, and room that is submitted as an input to the problem. [20] The issue now becomes one of mapping tuples onto period slots in such a way that tuples occupying the same period slot are disjoint (have no identifiers in common). If tuples are arbitrarily assigned to periods, there will be a number of collisions in all but the most trivial circumstances. The frequency of collisions in a timetable can be used as an objective assessment of the schedule's quality. As a result, we consider the number of clashes to be the cost of any particular timetable. A schedule's cost is easy to calculate. We keep track of the number of occurrences of each class, instructor, and room identification for each week period. The total cost of the timetable is the sum of the various costs. Abramson [21] goes into much detail about this process.

The proposed algorithm facilitates in the solution of the timetabling problem while emphasizing teacher availability. This program employs a heuristic approach to provide a generic solution to the problem of school scheduling. It requires the user to enter a number of subjects, a number of teachers, the subjects that each teacher teaches, the number of days in a week for which the timetable must be created, the number of time slots in a day, and the maximum number of lectures a teacher can give in a week.

It initially creates a temporary time table using a randomly generated subject sequence. When creating this sequence, effort is taken to avoid subject duplication throughout the course of a day. Following that, the availability of teachers for each of the subjects assigned to the relevant slot is checked. When a teacher is available for the topic during the allotted time slot, the subject and teacher are placed into the output data structure and designated as final. Before assigning this subject to the output data structure, a check is made to see how many maximum lectures a teacher can provide. If the teacher has been assigned more than the maximum number of lectures, the subject is moved to a Clash data structure. This variable selection criterion can be randomized to avoid cycling and to improve the search. There are numerous ways [22] that can be used.

As an example, consider the random walk technique (with the given probability p a random variable is selected) – not the worst variable, but a random selection of a variable that is bad enough (e.g., from the top N worst variables), or – a probability-based selection of a variable based on the aforementioned factors (e.g., roulette wheel selection).

However, in planning for timetables, considerations have to be made such that each lecture period must have at least one lecturer or professor, a time slot and venue. This is generally considered to be a highly constrained and hard problem to solve. Another challenge is how to generate an optimal timetable solution. To this end, approaches based on evolutionary algorithm using problem-specific domains, heuristics and context-based reasoning have been developed by various researchers. Parallel frameworks such as the PTMSS and genetic artificial immune networks have also been developed all in a bid to produce optimal solutions for timetable generation. The Scheduler implemented in this paper uses tools such as: XHTML, HTML6, CSS6, PHP, JAVASCRIPTS, JQUERY, MYSQL and Tomcat Apache. Furthermore, there are quite a number of approaches that have been used to create various systems with similar features, but not with exact functions as the SCHEDULER.

Problem Statement

The existing issues with traditional timetable generation includes: difficulty in execution, time consuming, and is considered an arduous process. When generating a manual timetable, lots of effort and man power is needed, and such timetables in most Nigeria Institutions are usually prone to human error.

Furthermore, a major problem that is associated with the manual lecture timetable system is the high rate of clashes in lecture times and venues. Amending an already generated timetable requires the scheduler to recreate the schedule manually over and over again. This certainly creates a series of retracing which is usually difficult to figure out or resolve as the case maybe. To overcome these problems, concerned institutions need an efficient automated, feasible and competent timetable scheduling system. Such a system should be capable of satisfying all the soft and hard constraints and conditions previously highlighted.

For instance, the same faculty lecturer taking two more courses cannot be assigned the same room, venue and time slot for the same lectures. Concurrently, two different courses which are to be delivered to the same students or group of students should also not be allowed. As such, there is a major requirement for an application appropriate lecture timetable without variation, such that collision in the scheduling process is totally eliminated. To this effect, the Scheduler system overcomes these problems particularly by saving more processing time and also by eliminating the traditional error-prone manual processes involved.

Background

The Logarithmic algorithm used in implementing the said system is the modified Quick sort algorithm. The algorithm is based on the Divide-and Conquer approach. The process is split into 6 parallel processes that are each running simultaneously. However, since there are 6 parallel processes, the best-case performance is Θ(6nlogn), which reduces to just Θ(long), while the worst-case performance is Θ(6n2), which also reduces to Θ(n2).

• Definition: Suppose there are P processes running in n time to generate the timetable. Then we can model the
problem as follows:

Condition 1: Pi ≠ Pi+1, where I=0,1, 2, n
Condition 2: Pn≠ Ø.

If and only if Condition 1 and Condition 2 are met, then we obtain Equation 1 as follows:

Pan= {P i-5, Pi-4, Pi-3, Pi-2, Pi-1, Pi} ………………………… (1)
where, I=0,1, 2, n

Each Pi is running independently and is generating specific days of the week having 8 separate periods between
8am – 4pm. To modify the algorithm, we first generate the timetable for both single and double lecture periods in

order to eliminate clashes in both courses and venues to be allocated as seen in Algorithm 1. The algorithm stores
the created slots and merges them together using the carefully designed function called merge Table (D, S).

• **Algorithm** 1: Generation of timetable

```
Create storage arrays D and S.
Create Arrays Dp and Sp for Double and Single periods of
lecture times respectively. Each array is of p periods (am-pm)

begin
a. creates an array of p empty periods, p = Ø.
b. check if Pi = Pi+1and Pi+1 <Pan,
  where =0,1, 2, n.
  then:
  generate Dp and Sp and store in D and S
respectively.
c. Query from D and S to generate the timetable by calling the merge Table (D, S)
function.
end
```

Proposed System

This section describes the proposed system structure, starting with the aim as follows:

Aim of the System

• Developing a paperless timetable system semester courses and examinations, together with other related
scheduled administrative operations.
• To develop a fast, trendy, unique and easy to use application that is deployable and efficient.
• To avoid lecture and venue clashes.

• To provide an interface that supports other related activities required but not necessarily related to timetable
scheduling processes such as: calendar of events, news feed etc.
• To provide an interactive chat forum for the administrators, lecturers and students alike in real-time.

There are a number of advantages considered while building the Scheduler scheduling system are
as follows:

Advantages

• The paperless timetable system reduces the human stress attributed to the manual process of creating
timetables.
• The automation provides Subjects (Course Title), Course Code, Department, Faculty, Lecturer, School Years,
Semester's, Room, Venue, Time, and Live Chats (for Administrators, Lecturers and Students). In addition,
Timetable Alerts for Students and Lecturers is provided by the SCHEDULER.
• The system eliminates all manual paperwork.
• The system also drastically reduces the man power and time consumed while executing the timetable
scheduling tasks.
• The Scheduler can be used either as a stand-alone or real-time system.

Disadvantages of existing system

• Lack of acceptance and awareness of the scheduling systems such as the Scheduler at most Nigeria Universities,
Polytechnics, and Colleges is partly attributed to ICT phobia and other related causes.
• Increased running costs of ICT infrastructure such as bandwidth and power are a real source of concern.

• The Applications are usually customized and copyrighted, making distribution and reuse difficult

Components Functionality:

The Scheduler allows for functionalities considered to be appropriate for implementation especially in cases where the description of the functionality is not adequate. In such cases, generally appropriate assumptions are made to
the following rules as follows

Student’s Classification Rules:

• Each class may have an interval of a maximum of 2hrs per slot.
• At least a maximum of 7 to 8 hours of lectures is permitted in a day.
• Time for practical work is also inclusive.
• Lecture time slots can be allotted and changed with ease by each course Head of Departments (HODs) often based on request by either lecturers or students.
• Courses are to be allocated in any of the following categories:
• Core (mandatory) courses.
• Electives (optional) courses.
• Maximum and minimum number of courses to be offered by students should be specified.

Lecturer Classification Rules:

Lecturer requirements are applied thus:

• Lecturers may reschedule already allotted time slots by agreeing and arranging with their students.
• A lecturer is restricted to no more than 3 lectures in a day, and no more than 2 hours at a stretch for every taught course the deliver. The extra hour should be allotted a separate time slot. However, free slots can still be utilized for extra lectures by a willing lecturer as agreed with their students.

Administrator Classification Rules:

These rules are determined by both HODs and administrators thus:

- A student may produce a copy of the timetable.
- Emails are to be sent to both students and lecturers containing the timetable.
- Break periods are to be properly captured.
- The timetable should not contain clashes

Data Input Process:

This defines the type of data that is inserted, retrieved and updated from a database as follows:

- Lecturer: define details of information that describe the lecturers involved.
- Course Title: Title that describes the course.
- Course Code: Code that describes the course.
- Department: Is the name of a given department.
- Faculty: Is a specified faculty for each department.
- School Years: The duration of the course of study such as: 4, 5, 6 or 7yrs.
- Semester: The semester in question i.e. first semester and second semester.
- Room: Is the lecture room or hall name or any related description.
- Venue: Describes building or location where the room is situated.
- Time Slot: The time allotted for each lecture.
- Time Interval: It is the time that a lecture is expected to last

System Constraints:

A constraint is a condition that a solution to a problem must satisfy. Two major constraints are stated in this paper namely: hard and soft constraints.

Hard Constraints:
Hard constraints are those constraints which set conditions for the variables that are required to be satisfied are as follows:

- Duplicate lectures must be eliminated.
- Experiments must be held in a Laboratory and not in lecture classes.
- Lecture rooms must not be booked twice at the same time.
- Venue of lectures should not be doubly-booked during the same period to avoid clashes.
- All lecture venues and rooms must be scheduled once not twice.
- A lecture room must be large enough to contain all students before allocation.

Soft Constraints:

Soft constraints are constraints that are easier to adjust as follows:

- Lectures for each given course should be evenly spread within the week.
- Departmental courses borrowed from different departments must be evenly distributed.
- Break periods must be allocated slots first before other courses.
- Faculty general courses must be allocated slots first before departmental courses.

Violation of Validated Constraints:

These are constraints that are required to be satisfied, so that there will be assurance of validated timetables.
However, this defines process such as;

- No more than 2 consecutive lectures by the same lecturer in any given period.
- The vital constraint is that both lecturers and their students can not appear in more than one lecture venue at the
same time.

System Requirement and Specification:

The system requirement and specification (SRS) document is summarized and laid out in a template as seen in Table 1. Some essential flow requirements are entered into it to show how to use the template. Care must be taken to ensure that even the smallest and most trivial requirements are written. Such requirements would help in validating the system during testing. The following are the specific software and hardware requirement:

Software Specification Requirements:

User Interface: PHP, XHTML, CSS, JQUERY
Client-side Scripting: JavaScript, PHP Scripting
Programming Language: PHP, ASP
IDE/Workbench/Tools: Adobe Dreamweaver CS6
Database: MySQL (My SQLite, Optional, Oracle 10g)
Server Deployment: Apache/2.2.4, Tomcat Apache.

Hardware Specification Requirements:

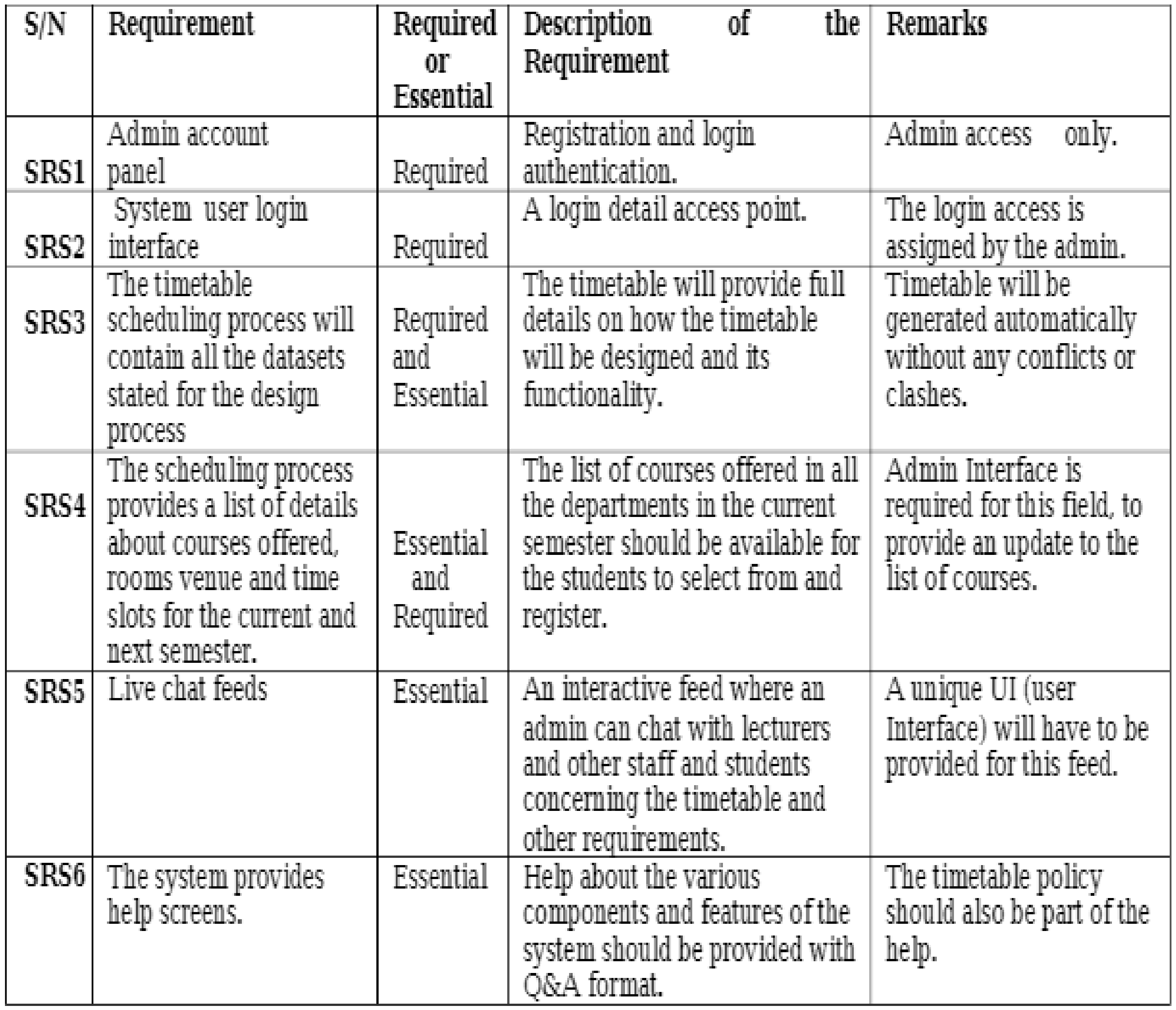
Monitor: 17-inch LCD Screen (optional).
Processor: Pentium 3, 4, dual-core, Intel, Core i7.
Hard Disk: 500GB or 1, 2 or 4 Terabyte.
RAM: 4GB or more.

The System Design

The system has a major component that forms the basis for the design. These components are: admin account panel (i.e. for registration and login authentication), user login panel, and timetable data entry with full details of lecturers for the scheduling process. The workflow of the system enables the user to have easy understanding of the process for creating the timetable. However, the system provides a user with a robust graphical user interface (GUI). This is a simple interactive interface for entering the details concerning a course such as: course title, course code, venue, rooms, department, and faculty, lecturer, school years, semesters, time slots, and lecturer information. The system administrator provides the users access to get registered, and have access to the system.

Note that the scheduler is the assigned officer who collates all data for entries, from various sources and afterwards generate the timetable

Table 1: Detailed Design Breakdown

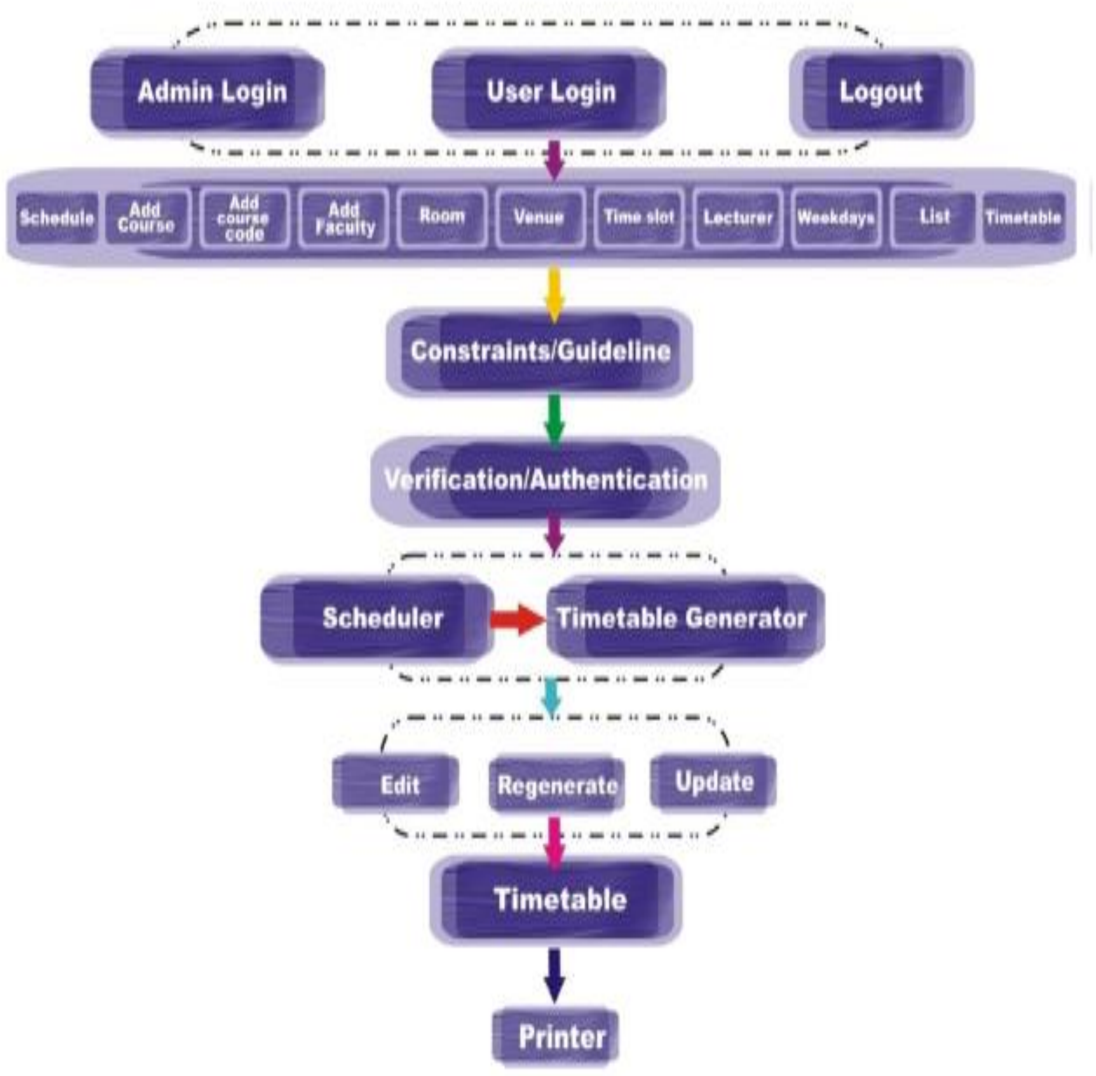


S/N	Requirement	Required or Essential	Description of the Requirement	Remarks
SRS1	Admin account panel	Required	Registration and login authentication.	Admin access only.
SRS2	System user login interface	Required	A login detail access point.	The login access is assigned by the admin.
SRS3	The timetable scheduling process will contain all the datasets stated for the design process	Required and Essential	The timetable will provide full details on how the timetable will be designed and its functionality.	Timetable will be generated automatically without any conflicts or clashes.
SRS4	The scheduling process provides a list of details about courses offered, rooms venue and time slots for the current and next semester.	Essential and Required	The list of courses offered in all the departments in the current semester should be available for the students to select from and register.	Admin Interface is required for this field, to provide an update to the list of courses.
SRS5	Live chat feeds	Essential	An interactive feed where an admin can chat with lecturers and other staff and students concerning the timetable and other requirements.	A unique UI (user Interface) will have to be provided for this feed.
SRS6	The system provides help screens.	Essential	Help about the various components and features of the system should be provided with Q&A format.	The timetable policy should also be part of the help.

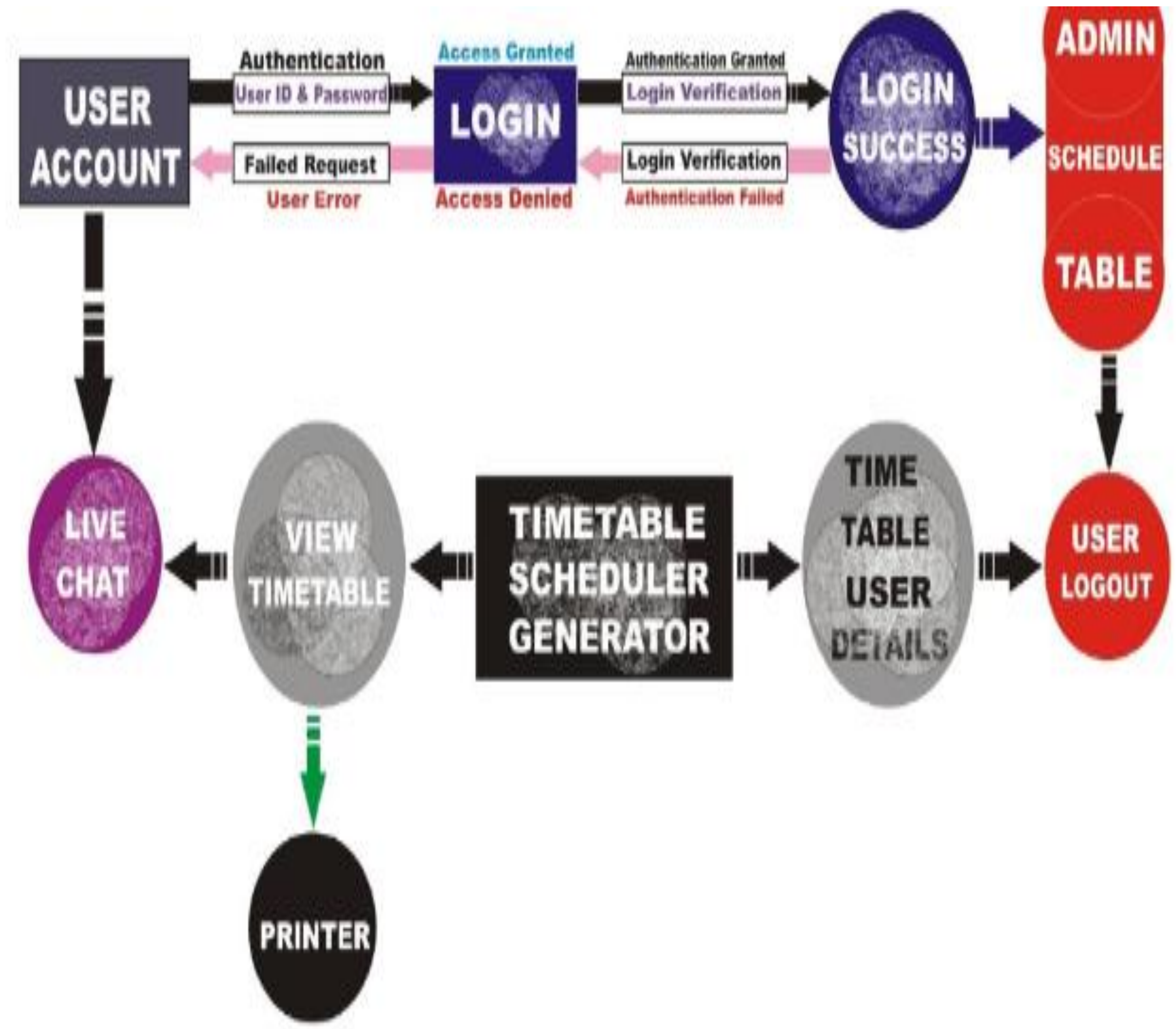
.

Data Flow Diagram

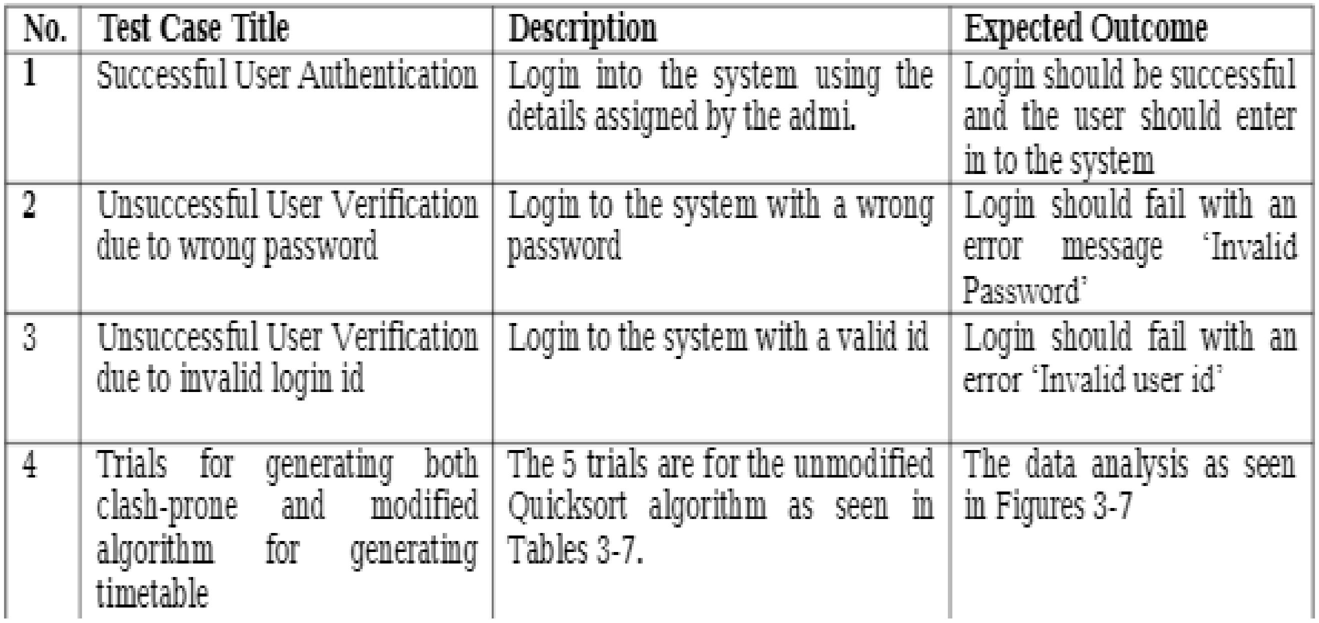
The flow diagram shows the user interface design of the Timetable Scheduling Process and how the timetable is to be generated.



Data Flow Diagram



Test Plan Description



No.	Test Case Title	Description	Expected Outcome
1	Successful User Authentication	Login into the system using the details assigned by the admi.	Login should be successful and the user should enter in to the system
2	Unsuccessful User Verification due to wrong password	Login to the system with a wrong password	Login should fail with an error message 'Invalid Password'
3	Unsuccessful User Verification due to invalid login id	Login to the system with a valid id	Login should fail with an error 'Invalid user id'
4	Trials for generating both clash-prone and modified algorithm for generating timetable	The 5 trials are for the unmodified Quicksort algorithm as seen in Tables 3-7.	The data analysis as seen in Figures 3-7

System Architecture Design

The architecture of the system is structurally designed to constitute three essential parts which includes: GUI, Front End ((FE) and Back End (BE). A description of the parts is as follows:

• The GUI defines the structural design regarding how the system will look like after implementation. This has a unique interactive platform that suite the user needs.
• The FE comprises of everything including the design and types of languages used in the design of the system. They include: PHP, PHPMYQL, HTML AND CSS etc.
• The BE otherwise called the server-side, deals with the system inputs, retrieval, editing and updates. This refers to everything the user cannot see in the FE such as the database and servers used.

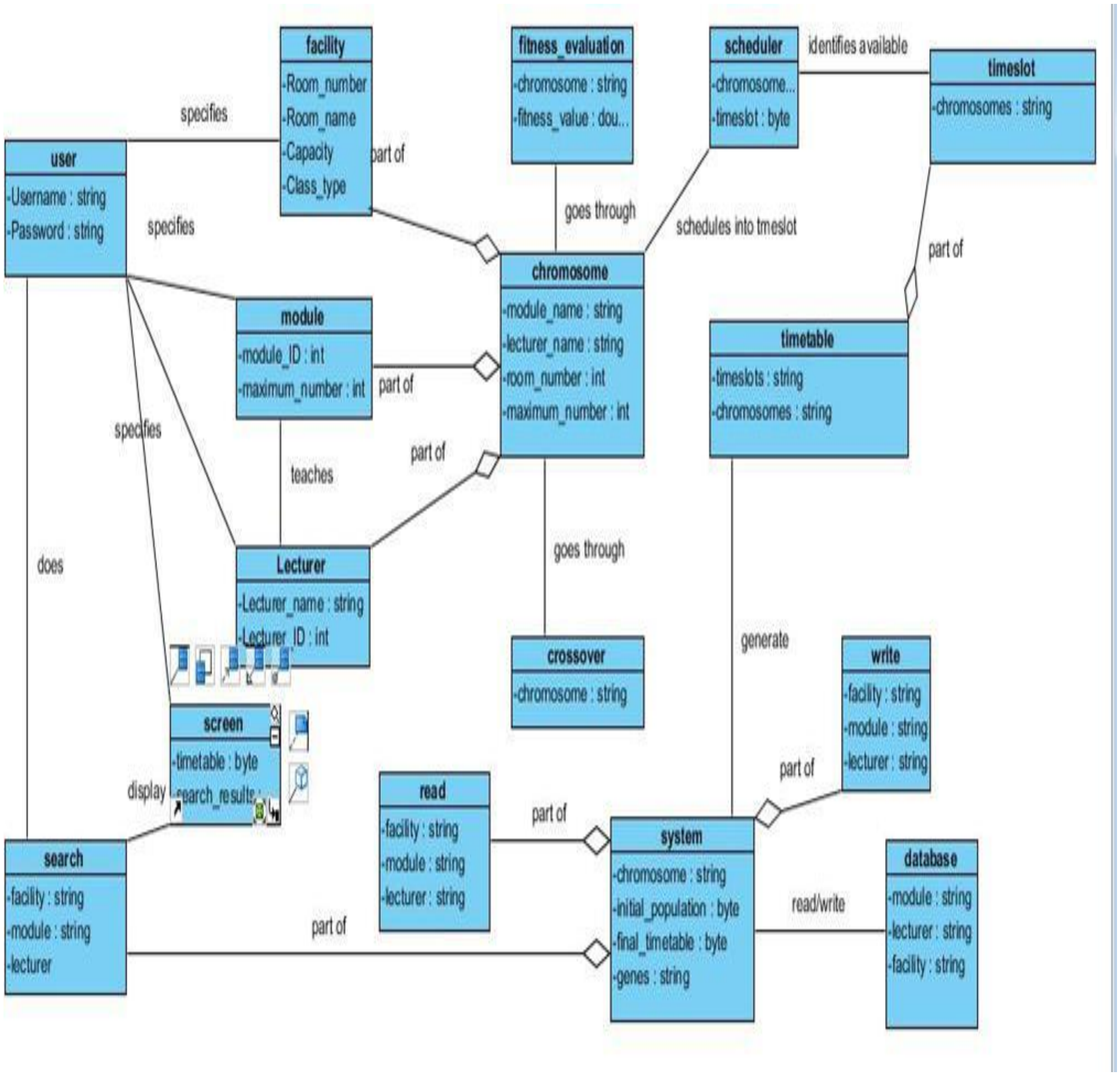
Learning Outcome

The learning outcomes as defined indicate what the group is planning to have achieved at the end of the undertaken project, which is genetic algorithm implementation in a timetable scheduling system. The learning outcomes are derived from the outcomes and deliverables stated in the above terms of reference table.

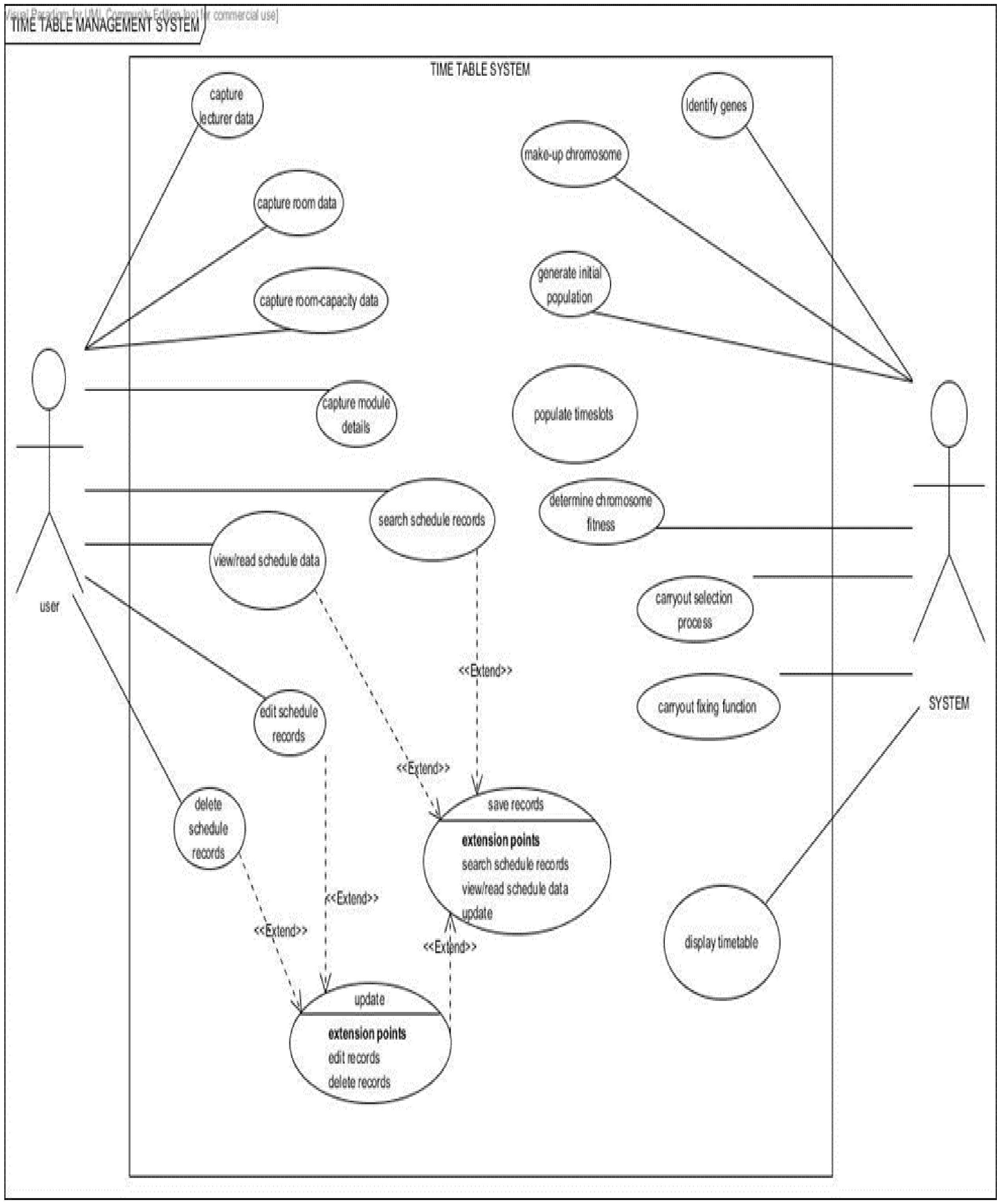
Having a functional system is not just the aim of this project but the group is aimed at learning about genetic algorithms, i.e. in general terms to their applications. It is highly empirical to know about the concept to be applied in order to model a better system and for future referencing about genetic algorithms. With the application of genetic algorithms, there are factors that may come into play as any other concept, so, understanding the time factor of genetic algorithms when implemented in real time situations is also to be reviewed. Genetic algorithms are also said to work hand in hand with other algorithms, hence the concept of mutation comes into play. Mutation is a broad topic, so understanding it when applied on a very specific topic is also under the group learning scope. In any project undertaken, there are violations that may occur.

Learning about these violations is also important and understanding these violations will also help make the project a success.

ER diagram:


facility
-Room_number
-Room_name
-Capacity
-Class_type
fitness_evaluation
-chromosome : string
-fitness_value : dou...
scheduler
-chromosome...
-timeslot : byte
identifies available
timeslot
-chromosomes : string
specifies
user
-Username : string
-Password : string
part of
goes through
schedules into tmeslot
specifies
chromosome
-module_name : string
-lecturer_name : string
-room_number : int
-maximum_number : int
part of
module
-module_ID : int
-maximum_number : int
part of
timetable
-timeslots : string
-chromosomes : string
specifies
teaches
part of
goes through
does
Lecturer
-Lecturer_name : string
-Lecturer_ID : int
generate
crossover
-chromosome : string
write
-facility : string
-module : string
-lecturer : string
screen
-timetable : byte
-search_results : ...
display
part of
read
-facility : string
-module : string
-lecturer : string
part of
system
-chromosome : string
-initial_population : byte
-final_timetable : byte
-genes : string
search
-facility : string
-module : string
-lecturer
part of
read/write
database
-module : string
-lecturer : string
-facility : string

Use Case Diagram:



1.

Learning Outcomes:

The learning outcomes as defined indicate what the group is planning to have achieved at the end of the undertaken project, which is genetic algorithm implementation in a timetable scheduling system. The learning outcomes are derived from the outcomes and deliverables stated in the above terms of reference table.

Having a functional system is not just the aim of this project but the group is aimed at learning about genetic algorithms, i.e. in general terms to their applications. It is highly empirical to know about the concept to be applied in order to model a better system and for future referencing about genetic algorithms. With the application of genetic algorithms, there are factors that may come into play as any other concept, so understanding the time factor of genetic algorithms when implemented in real time situations is also to be reviewed. Genetic algorithms are also said to work hand in hand with other algorithms, hence the concept of mutation comes into play. Mutation is a broad topic, so understanding it when applied on a very specific topic is also under the group learning scope. In any project undertaken, there are violations that may occur. Learning about these violations is also important and understanding these violations will also help make the project a success.

Result and Discussion

The test-plan is basically a list of test cases that need to be run on the system. Some of the test cases can be run independently for some components such as report generation from the database which is tested independently.
However, some of the test cases require the whole system to be ready before execution. It is better to test each component as at when it is ready before integrating the components, which is a unit test before the system test.
It is important to note that the test cases cover all the aspects of the system (i.e. all the requirements stated in the SRS template in Table 1). Table 2 contains a sample authentication test plan.

Conclusion

The workflow of this proposed system makes use of collision avoidance technique and process in scheduling an automated timetable system. This made it easier and faster to completely eliminate the manual process of generating timetable. This paper presented a Logarithmic Quicksort algorithm for solving a highly constrained timetable generation problem. The approach used a problem-specific domain representation context-based reasoning for obtaining feasible solution ate reasonable computing time. The future work will entail the use of a
real-time generation of timetables, with content-based analysis and reports to be generated through an embedded management information system (MIS).

Automatic Timetable Generator is a web-based application for generating timetable automatically. It is a great difficult task that to manage many Faculty's and allocating subjects for them at a time manually. So proposed system will help to overcome this disadvantage. Thus, we can generate timetable for any number of courses and multiple semesters. This system will help to create dynamic pages so that for implementing such a system we can make use of the different tools are widely applicable and free to use also.

References:

[1]. D. Datta, Kalyan Moy Deb, Carlos M. Fonseca, "Solving Class Timetabling Problem of IIT Kanpur using Multi- Objective Evolutionary Algorithm." Kangal 2005.

[2]. Edmund K Burke, Barry McCollum, Amnon Meisel's, Sonja Perovic, Rong Qu, "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems." European Journal Operational Research, 176: 177-192, 2007.

[3]. Awed and Cheneck, "Proctor Assignment" at Carleton University (1998).

[4]. Ching B., Li H., Lim A. and Rodrigues B. 2003, "Nurse Rostering Problems: A Bibliographic Survey." European Journal of Operational Research, 151(3) 447-460.

[5]. Easton K., Newhouse G. and Trick M. 2004, "Sports Scheduling." In: Leung J. (ed.) in Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapter 52, CRC Press.

[6]. S. and Burke E.K. 2004. University Timetabling In: Leung J. (ed.) "Handbook of Scheduling: Algorithms", "Models, and Performance Analysis." Chapter 45. CRC Press.

[7]. W. Leviers, "Constraint-based Techniques for the University Course Timetabling Problem", CPDC, (2005), pp.59-63.

[8]. S. Abdullah, E. K. Burke and B. McCollum, "A Hybrid Evolutionary Approach to the University Course Timetabling Problem", Proceedings of the IEEE Congress Evolutionary Computation, Singapore, (2007).

[9]. J. F. Gonçalves and J. R. De Almeida, "A Hybrid Genetic Algorithm for Assembly Line Balancing", Journal of Heuristics, Vol.8, (2002), pp.629-642.

[10]. G. Kendall and N. M. Hussain, "A Taboo Search Hyper Heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology", Lecture Notes in Computer Science, Springer Verlag, vol.3616, (2005), pp.270-293.

[11]. M. A. Saleh and P. Coddington, "A Comparison of Annealing techniques for Academic Course Scheduling", Lecture Notes in Computer Science, Springer Verlag, vol. 1408, (1998), pp.92-114.

[12]. Aubin J, Furl and J. A, "A Large-Scale Timetabling Problem", Compute. & Opry. Res., vol.16, no.1, pp.67-77, 1989.

[13]. Dempster M. A. H, "Two algorithms for the timetable problem," Proc, of Conference, Oxford, July 1969, pp. 63-8.

[14]. Dinkel J. J, Mote J, Venkata Ramanan M. A, "An efficient decision support system for academic course scheduling," Operations Research 37(6), 1989, pp. 853-864.

[15]. Goalie C.C, "The construction of class - teacher timetables", IFIP Congress 62, 1962, pp.73- 77.

[16]. Do Xuan Duong, Pham Hay Dine, "Solving the Lecture Scheduling Problem by the Combination of Exchange Procedure and Taboo Search Techniques," Studio Informatica Universalis, Vol.4, Number 2

[17]. Easton, K., Newhouse, G. and Trick, M, "The traveling tournament problem: description and benchmarks." In Proc of the 7th. International Conference on Principles and Practice of Constraint Programming, Pathos, 580–584, 2001

[18]. K. Nurmi, D. Gosens, T. Bartsch, F. Bonomi, D, Brisker, G. Duran, J. Kings, J. Marengo, C.C. Ribeiro, F. Spokesman, S. Urrutia and R. Wolf, "A Framework for a Highly Constrained Sports Scheduling Problem," In Proc of the International Multiconference of Engineers and Computer Science 2010 Vol III, IMECS 2010,Hong Kong