A Project Report On

"TIME TABLE GENARATOR USING GENETIC ALGORITHM"

Submitted to Osmania University in

Partial fulfillment for the Award of the Degree of

B.Sc. (MPCs-B)

(2022 - 2023)

By

R.BHANU PRASAD – 105320468101

N.UDAY KIRAN – 105320468084

N.SAIGANESH – 105320468087

T.PAVAN KUMAR – 105320468120

Department of Computer Science

A.V. College of Arts, Science and Commerce

Gagan Mahal, Hyderabad.

(Affiliated to Osmania University)

ESTD: 1968

A.V. COLLEGE OF ARTS, SCIENCE & COMMERCE

(Affiliated to Osmania University)

Gagan mahal, Hyderabad-500029, Telangana

Phone: 040 – 27637751, www.avcollege.in

Accredited with 'B++' Grade by NAAC (3rd Cycle)

Vision: Empowerment Through Pursuit of Excellence

Date:

CERTIFICATE

This is to certify that the project work entitled "**TIME TABLE GENARATOR USING GENETIC ALGORITHM**" is being submitted by **Mr.R.Bhanu Prasad, N.Uday Kiran, N.Sai Ganesh, T.Pavan Kumar,** bearing **H.T. No. 105320468101, 10532468084,105320468087,10532468120,**respectively as a part of curriculum in the partial fulfilment for the Award of **B.Sc. (MPCs)** in **A.V. College of Arts, Science & Commerce** for the academic year 2022-2023. This has not been submitted to any other University or Institution for the Award of any Degree.

HOD **External Examiner**

(Mrs. D. Suhasini)

Internal Examiner

DECLARATION

I hereby declare that the project work entitled "**TIME TABLE GENARATOR USING GENETIC ALGORITHM**" has been developed by me under the supervision of Ms.P.Sowmya Sree **of A.V. College of Arts, Science & Commerce, Hyderabad** and submitted to **Osmania University** for the award of the degree of **B.Sc. (MPCs).**

The work done is original and has not been submitted in part or in full to any other University or Institution for the award of any Degree.

R.BHANU PRASAD – 105320468101

N.UDAY KIRAN – 105320468084

N.SAIGANESH – 105320468087

T.PAVAN KUMAR – 105320468120

PRINCIPAL

ACKNOWLEDGEMENT

I would like to express my deep gratitude and respect to all those people behind the scene who guided, inspired and helped me in the completion of this project work.

I wish to express my heartfelt thanks to my Internal Guide **Mrs. P. Sowmya** and **HOD Mrs. D. Suhasini** for their valuable suggestions and advice throughout the course and **Dr. CH. Rajalingam, Principal of A.V. College of Arts, Science & Commerce** for the indispensable support and encouragement. I also extend my thanks to other faculties and supporting staff for their cooperation during my course.

It would be void unless I salute to my parents, I am greatly indebted to my parents and family members for their tireless support to me in all my endeavors, without whose un sustained support, I could not have made this career in computer science.

Last but not the least, I would like to thank my friends for being a constant source of encouragement all the time throughout my career.

R.BHANU PRASAD – 105320468101

N.UDAY KIRAN – 105320468084

N.SAIGANESH – 105320468087

T.PAVAN KUMAR – 105320468120

PRINCIPAL

INTRODUCTION

The class timetabling problem is a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year. In earlier days, time table scheduling was done manually with a single person or some group involved in task of scheduling it manually, which takes a lot of effort and time. Planning timetables is one of the most complex and error-prone applications. Timetabling is the task of creating a timetable while satisfying some constraints. There are basically two types of constraints, soft constraints and hard constraints. Soft constraints are those if we violate them in scheduling, the output is still valid, but hard constraints are those which if we violate them; the timetable is no longer valid. The search space of a timetabling problem is too vast, many solutions exist in the search space and few of them are not feasible. Feasible solutions here mean those which do not violate hard constraints and as well try to satisfy soft constraints. We need to choose the most appropriate one from feasible solutions. Most appropriate ones here mean those which do not violate soft constraints to a greater extent.

Using Genetic Algorithm (GA), a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily. Moreover, each of the obtained solutions has been found much better than a manually prepared solution which is in use. The problem of timetable scheduling is described as a highly constrained NP-hard problem. A lot of complex constraints need to be addressed for development of an efficient algorithm to solve this problem.

P (Polynomial Time):

Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3).

Example: Finding maximum element in an array or to check whether a string is palindrome or not. So, there are many problems which can be solved in polynomial time

NP (Non deterministic Polynomial Time):

Problems which can't be solved in polynomial time like TSP (Travelling Salesman Problem) or an easy example of this is subset sum: given a set of numbers, does there exist a

subset whose sum is zero? But NP problems are checkable in polynomial time means that given a solution of a problem, it can be checked whether the solution is correct or not in polynomial time.

Consider two problems A and B:

Reducibility:

If one instance of a problem A can be converted into problem B (NP problem) then it means that A is reducible to B.

NP-Hard:

Suppose, if A is reducible to B, then it means that B is at least as hard as A.

NP-Complete:

The group of problems which are both in NP and NP-hard are known as NP-Complete problem.Timetabling is the task of creating a timetable while satisfying some constraints. There are basically two types of constraints, soft constraints and hard constraints. Soft constraints are those which are violated in scheduling, the output is still valid, but hard constraints are those which if violated, the timetable is no longer valid. The search space of a timetabling problem is too vast, many solutions exist in the search space and few of them are not feasible. Feasible solutions here mean those which do not violate hard constraints and as well try to satisfy soft constraints. The most appropriate solution has to be chosen from feasible solutions. Most appropriate ones here mean those which do not violate soft constraints to a greater extent. In this project hard-constraints have been taken care of strictly and it has been ensured that soft-constraints are as well followed as much as possible.

Soft-Constraints (Flexible)

- More or less equal load is given to all faculties
- Required time (hours per week) is given to every Batch.

Hard-Constraints

- There should not be any single instance of a faculty taking two classes simultaneously.
- A class group must not have more than one lectures at the same time.

1.1 OBJECTIVE:

The main objective of this project is to generate the timetable without any clashes or collisions with the constraints such as:

a) To generate timetable using genetic algorithm in order to reduce the complexity and reduce large no. of permutations and combinations.
b) Reduces time and man work.
c) Avoids conflicts between the classwork.
d) Implements substitution for faculty in their absence.

Objects of Timetable Scheduler with respect to Genetic Algorithm

Students Group:

The student group class has the id, name of the student group, number of subjects, array of subject names and hours of study required for each subject per week. It also contains the id of teachers who will teach those subjects.

Teacher:

It is a class to hold the faculty information. It has an id, name of faculty, subject that he/she teaches and an in assigned which represents the no. of batches assigned to the teacher.

Slot:

A slot here is the most basic unit of Genetic algorithm. It represents a single characteristic of a Gene.

Timetable:

This class object holds an array of Slot. This is basically a class to generate new slots initially for each Student group.

Gene:

It is the main constituent of a Chromosome and is made up of a sequence of slot numbers. It represents a Time table of a single class group

Chromosome:

A chromosome here is a collection or an array of Genes. It is the main class of algorithm and it undergoes crossover and mutation to furnish fitter individuals.

Utility:

It is basically for testing purpose only. It contains some methods like printSlots() which help to keep track of algorithm through console.

Input Data:

This is a class mainly to fetch the input from user either through text file or through form and provide it to the working classes of the algorithm.

1.2 PROBLEM STATEMENT:

The class timetabling problem is a typical scheduling problem that appears to be a tedious job in every academic institute.

- It takes lots of time and effort.
- Timetable is one of the most complex and error prone application.

During analysis of this problem, we can remark than an automatic creation of timetables has two aspects. First problem is in complexity of the solution and searching algorithm for the solution. This is a combinatorial problem with a large number of variables. Only small percent of them are feasible timetables, and some of them can be considered as good ones. The problem is partially solvable using a variety of heuristic and optimization methods, integer linear programming, taboo search, genetic algorithms etc. Much less explored problem is in defining requirements of the timetable ,the names of teachers and classes in the matrix header, filled with weekly number of hours that the teacher teaches in the particular class

1.3 EXISTING SYSTEM:

At present the Time table is prepared manually in schools and colleges or in the corporate sector. Preparing time table is a tedious task which involves a lot of human activity and is a time-consuming task it also involves much of the permutations and combinations which make humans exhaustive at the end of the day.

The other way of making timetable is using an excel sheet which is not as effective and efficient. The time table might be required for an academic or any other purpose or even required for a day scheduler. Preparing the timetable can also be done by using html and scripting languages which is inefficient for the required purpose.

1.3.1 DRAWBACKS:

- Tedious task which involves a lot of human activity.
- Time consuming task it also involves much of permutations and combinations which make humans exhaustive at the end of the day.
- Making timetable using excel sheet is not as effective and efficient.
- Preparing timetable is also done by html and scripting languages which is inefficient for the required purpose.
- As all institutions/organisation have its own timetable managing and maintaining these will be difficult . Considering the workload of staff will make the scheduling part of timetable more complex . Because of these constraints managing staff with respect to their workloads will be difficult .

1.4 PROPOSED SYSTEM:

The proposed project Generating Timetable Using Genetic Algorithm has the functionality to produce the timetable based on the inputs given by the user and also has Notification Substitution Feature where the class can be given to other faculty on the absence of the other the Substitution and Notification feature acts as a reminder to the user or faculty as which classes have accepted by whom.Genetic Algorithm gives the optimal solution.In this context then any other even though it might hacve any other limitations

1.5 ADVANTAGES:

Faculty did not need to worry for time clashes,Authority now does not need to perform permutation and combination Authority can concentrate on other things rather than wasting their time on preparing Time-Table

To develop the proposed system, it needs no extra facilities and devices. All dependencies are satisfied from the open source projects. All tools used are free, open source and the programming language is JSP and hence its development is economically

Proposed system is technically feasible because the proposed system requires only those H/W and S/W tools that are available in the system. It requires the installation of JSP and

MYSQL which can be done for free. More over Automatic Timetable Generator expandability will be maintained in the new system. New modules can be added later on the application, if required in the future. Additionally, the application will have User-friendly Forms and Screens.

It determines how much effort will go in the proposed information system, and in educating and training the users on the new system. Since the user interface is very simple and easily understandable, no training is required for using this software. Making tasks that are beyond human capabilities easier. Handling heavy or large loads, manipulating tiny objects or the requirement to make products very quickly or slowly are examples of this. Production is often faster and labor costs less on a per product basis than the equivalent manual operations.

Automation systems can easily incorporate quality checks and verifications to reduce the number of out-of-tolerance parts being produced while allowing for statistical process control that will allow for a more consistent and uniform product. Economic improvement automation can serve as the catalyst for improvement in the economies of enterprises or society.

chapter-II

SYSTEM ANALYSIS

2.1 LITERATURE SURVEY:

➢ Henry Laurence gantt invented shedules.

➢ Timetable is a basic time management tool consists of a list of times at which possible tasks,events or actions intended to take place.

➢ A well constructed timetable establishes a natural rhythm and routine which can be comforting to teachers and students.

As it known that the Genetic Algorithms are used to get the worth of the solution to the given problem in a context. Genetic Algorithms are a way of solving problems by mimicking nature. The same combination of selection, recombination and mutation to evolve a set of candidates for resolving a given problem is used.

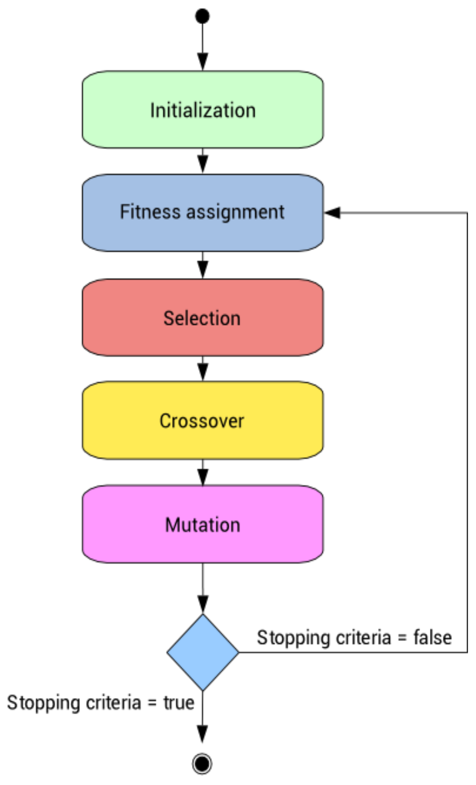


Figure 2.1 Working of Genetic Algorithm Flow chart

As shown in fig. 2.1 the algorithm ends when it satisfies the stopping criteria otherwise continues. 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 a 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. The 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 idea of searching among a collection of candidate solutions for a desired solution is so common in computer science that it has been given its own name as searching in a "search space." Here the term "search space" refers to some collection of candidate solutions to a problem and some notion of "distance" between candidate solutions. If same problem is being solved, usually it gives some solution, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution) is called search space (also state space). Each point in the search space represent one feasible solution. Each feasible solution can be "marked" by its value or fitness for the problem. Looking for the solution, which is one point (or more) among feasible solutions - that is one point in the search space.

The looking for a solution is then equal to a looking for some extreme (minimum or maximum) in the search space. The search space can be whole known by the time of solving a problem, but usually few points from it are known and generating other points as the process of finding solution continues.

2.2 REQUIREMENTS SPECIFICATION:

The following are the hardware and software requirements that have used to implement the proposed system

2.2.1 HARDWARE REQUIREMENTS :

Name of the Processor	:	Intel(i5) CORE
Hard disk capacity	:	1TB
Ram Capacity	:	8GB

2.2.2 SOFTWARE REQUIREMENTS:

OS	:	Windows
Languages	:	java
IDE	:	Net Beans
Database	:	MYSQL

Languages Used:

Front End	:	Html, CSS, Bootstrap, JavaScript
Back End	:	MySQL Yog

Tools:

MySQL yog:

Powerful backup wizard with a one clicks backup option and ability to schedule backups with ease. The data search feature which helps the user to find out the query without searching the whole document. The Schema Designer is a visual interface where tables and table structures can be defined, displayed and manipulated. The Schema Designer provides a convenient graphical way to perform common operations as an alternative to the menu-based way of operation that was always supported. Also, the Schema Designer provides a graphical overview of (complete or only parts of) the databases.

NetBeans IDE:

NetBeans IDE is the official IDE for Java 8+ version. With its editors, code analyzers, and converters, people can quickly and smoothly upgrade the applications to use new Java 8 language constructs, such as lambdas, functional operations, and method references. Batch analyzers and converters are provided to search through multiple applications at the same time, matching patterns for conversion to new Java 8 language constructs

2.3 FEASIBILITY STUDY:

Economic Feasibility:

To develop the proposed system, it needs no extra facilities and devices. All dependencies are satisfied from the open source projects. All tools used are free, open source and the programming language is JSP and hence its development is economically

Technical Feasibility:

Proposed system is technically feasible because the proposed system requires only those H/W and S/W tools that are available in the system. It requires the installation of JSP and MYSQL which can be done for free. More over Automatic Timetable Generator expandability will be maintained in the new system. New modules can be added later on the application, if required in the future. Additionally, the application will have User-friendly Forms and Screens.

Behavior Feasibility:

Behavioral feasibility determines how much effort will go in the proposed information system, and in educating and training the users on the new system. Since the user interface is very simple and easily understandable, no training is required for using this software. Making tasks that are beyond human capabilities easier. Handling heavy or large loads, manipulating tiny objects or the requirement to make products very quickly or slowly are examples of this. Production is often faster and labor costs less on a per product basis than the equivalent manual operations.

Automation systems can easily incorporate quality checks and verifications to reduce the number of out-of-tolerance parts being produced while allowing for statistical process control that will allow for a more consistent and uniform product. Economic improvement automation can serve as the catalyst for improvement in the economies of enterprises or society.

For example, the gross national income and standard of living in Germany and Japan improved drastically in the 20th century, due in large part to embracing automation for the production of weapons, automobiles, textiles and other goods for export.

Chapter-III

SYSTEM DESIGN

3. System Architecture:

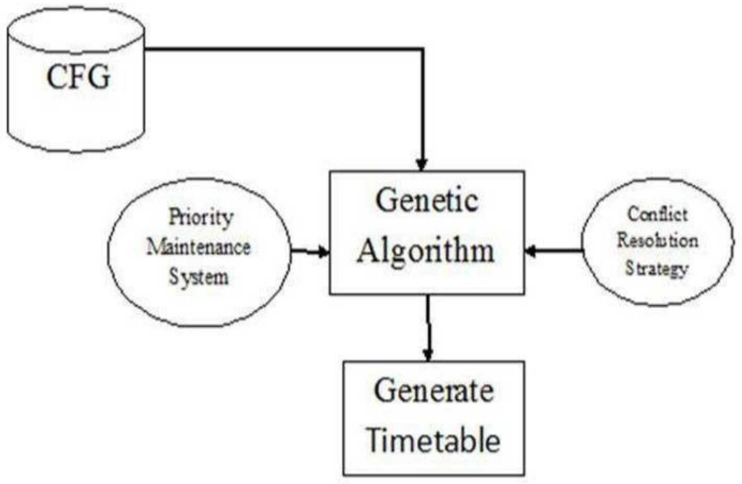


Figure :3. System Architecture

Fig 3 consist of a configuration file in which provide all the input fields required for timetable. Genetic algorithm takes this input and performs operation on it, resolve conflicts, satisfy priority and automatically generate timetable.

Constraint Data:

In order to test for each of the types of hard constraint it is necessary to store sufficient detail about the timetable requirements. This means that information concerning all lecturers, classrooms and classes must be maintained. The way in which each of these data types are implemented will now be given in detail. A lecturer is a structured data type with one field- an availability timetable. An availability timetable is an array which indicates (using 1s and 0s) whether the lecturer is available or unavailable to lecture during each hour in the week. In this way prior commitments can be taken into consideration. A class is a structured type with three fields. Each class has a lecturer number and a number indicating the code number of the group of related classes to which it belongs.

For example, core first year engineering subjects might form one set of related classes, and would each have the same number in their related class field. Each class can only be listed as belonging to one set of related classes.

This is a simplification which ignores more complex relations between classes, for example, where classes are studied by more than one faculty. The set of all lecturers is stored as an array. Similarly, there are arrays of classes and of classrooms. Each of these elements is identified uniquely by its position in the relevant array.

Repair Strategy:

A repair strategy is used which ensures that all classes appear exactly once. For robustness, this is done in two stages. Firstly, any classes which appear more than once are (non-deterministically) altered such that they appear only once. Secondly, any classes which did not appear at all are booked to a spare space (regardless of room size, etc.)

If this repair strategy is applied to an empty timetable the result is a timetable with each class booked to a random time and place. As such, the repair strategy is also used for initial a random population. The use of the repair strategy ensures that each class is booked exactly once. Hence, the number of hard constraints which must be considered when timetables are being evaluated is further reduced.

3.1 MODULES:

It is the important part of project which generate Timetable automatically. IN this module it develops module. Generation done by considering maximum and minimum workload for a Faculty (without less and without exceeding). This will be generated by admin and viewed by user and Faculty who's are the users of the system.

The operations that can be performed by the user are:

Step 1:- Start timetable
Step 2:- The details of the hours, subjects and faculty names handling each subject is taken.

- a. hours per day
- b. subjects per semester
- c. number of working days

- d. Number of faculties

Step 3:- Read the details provided.
Step 4:- Validate the details and a timetable is created accordingly.
Substitute timetable
Step 5:- The details of the substitute faculty is taken.
Step 6:- Read the date, reason and substitute faculty available for the subject.
Step 7:- The request is sent to the substitute facility.
Step 8:- The substitute faculty accepts or rejects the request.
Step 9:- The details are read.
Step 10:- Accordingly the timetable is generated.
Step 11:- Stop

3.2 DESIGN REPRESENTATION:

Designing an object oriented system is aided by a high-level, graphical representation of classes, objects and their relationships. The graphical nature of the representation allows easy visualization of the structural aspects of the system. The design is represented at a high level in that only the essential elements of the design are present independent of lexical details or the specifics of the implementation.

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

- Class diagram
- Use Case diagram
- Sequence diagram
- Activity diagram

3.2.1 CLASS DIAGRAM:

Class diagram represents the complete structure of the timetable generation and even the substitution is included in the generation as shown in fig. 3.2.1. Class diagram represents the classes as Faculty, Student and the Timetable Generator which emphasizes on the parameters used during registration and other functionalities.

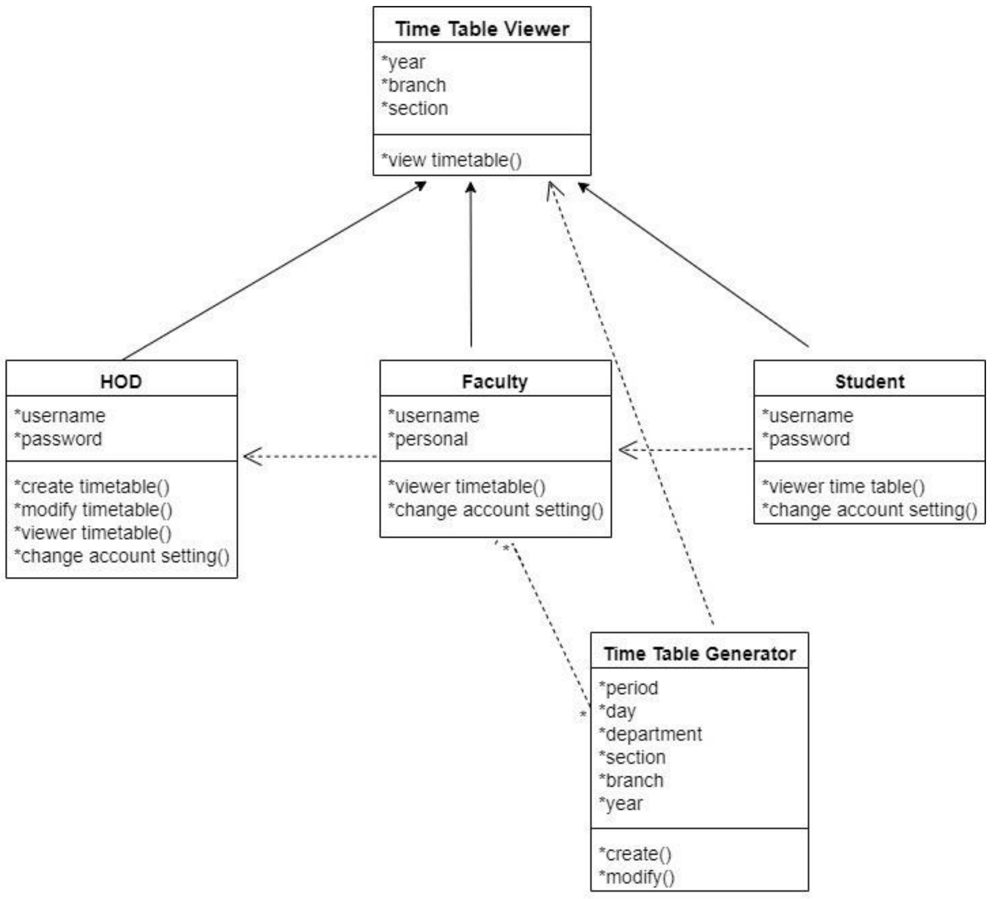


Figure 3.2.1: Class Diagram of Timetable Generation

3.2.2 USECASE DIAGRAM:

Use case diagram shows how the user gets interacted with the system and mainly it describes about how all the use cases in the system gets interacted with each other along with the user. This use case diagram gives the different uses cases and also includes the other types of diagrams, which provides the higher-level view of the system used.

Usecase diagram for timetable generation is shown in fig. 3.2.2 where the faculty interacts with the application system in-order to get the timetable. It includes registration, viewing the timetable, request for substitution and accessing the response.

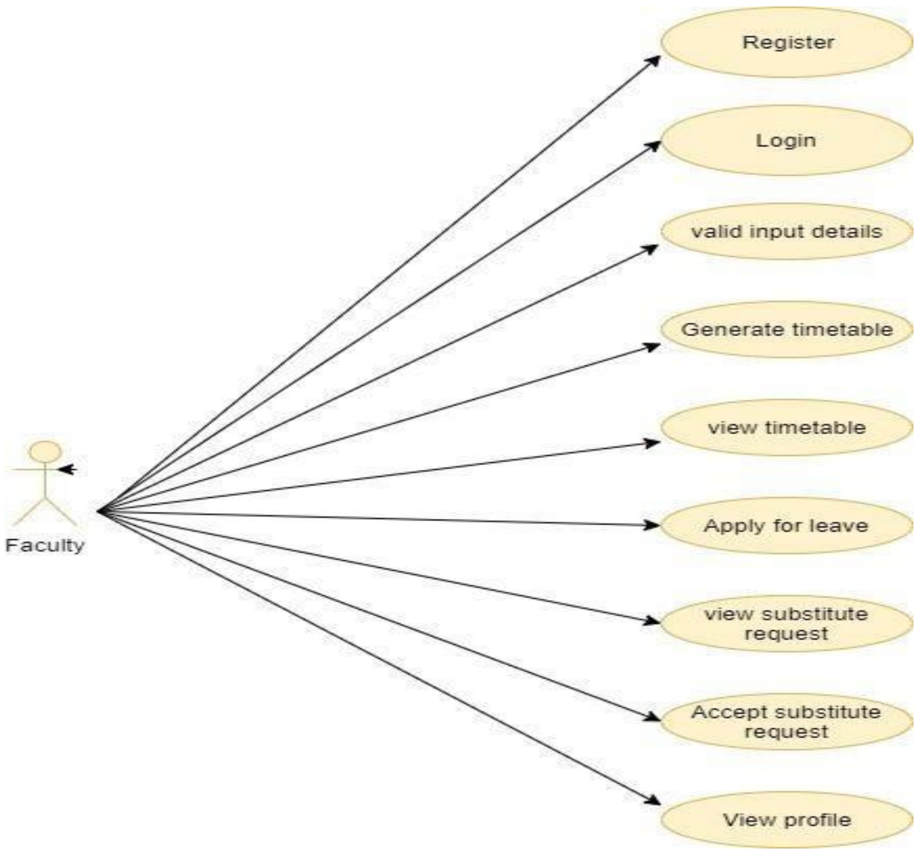


Figure 3.2.2: Usecase Diagram of Timetable Generation

3.2.3 ACTIVITY DIAGRAM:

The activity diagram represents the control flow of actions done by the user or held between the user and his task to achieve.

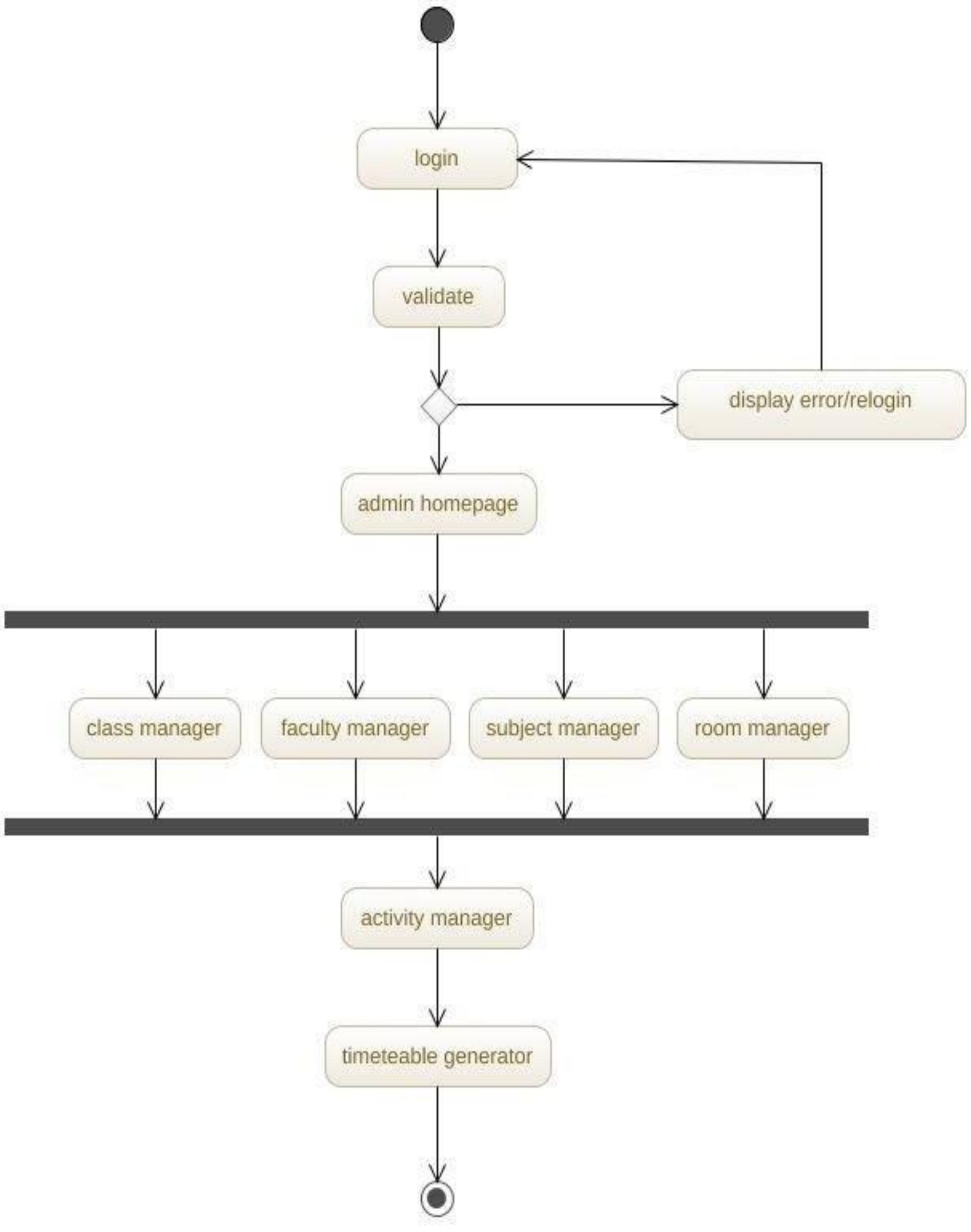


Figure 3.2.3: Activity Diagram of Timetable Generation

3.2.4 SEQUENCE DIAGRAM:

The sequence diagram represents the sequence of instructions given in an application as shown in fig 3.2.4Sequence diagram illustrates the sequence of actions followed during the use of timetable. It incorporates registration, timetable generation and substitution.

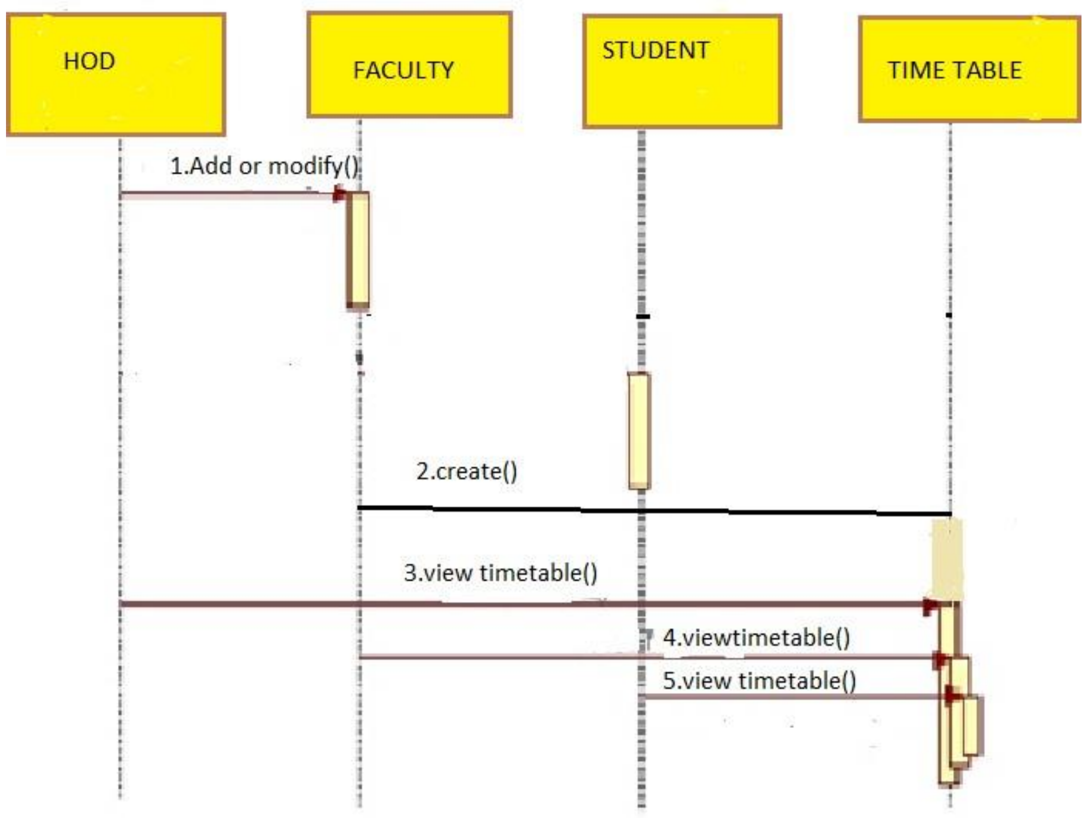


Figure 3.2.4: Sequence Diagram of Time Table Generation

Chapter-IV

IMPLEMENTATION

The implementation of the project is mainly using java, the zip file which contains all files of the project, when executed all of the files executes together not as an individual file. The main principle of the project is to generate the timetable according to the configuration file.

The timetable is generated by the internal calculation of the fitness function, crossover and mutation of the configuration file. The timetable is generated efficiently when the CFG (Configuration File) Given locally.

4.1 TECHNOLOGIES:

4.1.1 JAVA:

Java is a programming language originally developed by Sun Microsystems and released in 1995 as a component of Sun Microsystems platform. The language derives much of its syntax from C and C++, but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. The Java programming language is a high-level language that can be characterized by all the following:

- Simple
- Object-Oriented
- Distributed
- Multi-threaded
- Dynamic
- Portable
- Architecture Neutral

Java is a class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA),[meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but has fewer low-level facilities than either of them. The Java runtime provides dynamic capabilities (such as reflection and runtime code modification) that are typically not available in traditional compiled languages. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle) and released in 1995 as a core component of Sun Microsystems' Java platform. The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun had relicensed most of its Java technologies under the GNU General Public License. Oracle offers its own HotSpot Java Virtual Machine, however the official reference implementation is the OpenJDK JVM which is free open source software and used by most developers and is the default JVM for almost all Linux distributions.

As of September 2020, the latest version is Java 15, with Java 11, a currently supported long-term support (LTS) version, released on September 25, 2018. Oracle released the last zero-cost public update for the legacy version Java 8 LTS in January 2019 for commercial use, although it will otherwise still support Java 8 with public updates for personal use indefinitely. Other vendors have begun to offer zero-cost builds of OpenJDK 8 and 11 that are still receiving security and other upgrades.

Oracle (and others) highly recommend uninstalling outdated versions of Java because of serious risks due to unresolved security issues. Since Java 9, 10, 12, 13 and 14 are no longer supported, Oracle advises its users to immediately transition to the latest version (currently Java 15) or an LTS release.

Java JVM and bytecode

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate run time support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their machine for standalone Java applications, or in a web browser for Java applets.

Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions made interpreted programs almost always run more slowly than native executables. Just-in-time (JIT) compilers that compile byte-codes to machine code during runtime were introduced from an early stage. Java itself is platform-independent and is adapted to the particular platform it is to run on by a Java virtual machine (JVM) for it, which translates the Java bytecode into the platform's machine language.

Programs written in Java have a reputation for being slower and requiring more memory than those written in C++ . However, Java programs' execution speed improved significantly with the introduction of just-in-time compilation in 1997/1998 for Java 1.1 the addition of language features supporting better code analysis (such as inner classes, the StringBuilder class, optional assertions, etc.), and optimizations in the Java virtual machine, such as HotSpot becoming the default for Sun's JVM in 2000. With Java 1.5, the performance was improved Programs written in Java have a reputation for being slower and requiring more with the addition of the java.util.concurrent package, including lock free implementations of the ConcurrentMaps and other multi-core collections, and it was improved further with Java

Java uses an automatic garbage collector (AGC) to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use. Once no references to an object remain, the unreachable memory becomes eligible to be freed automatically by the garbage collector. Something similar to a memory leak may still occur if a programmer's code holds a reference to an object that is no longer needed, typically when objects that are no longer needed are stored in containers that are still in use. If methods for a non-existent object are called, a null pointer exception is thrown.

One of the ideas behind Java's automatic memory management model is that programmers can be spared the burden of having to perform manual memory management. In some languages, memory for the creation of objects is implicitly allocated on the stack or explicitly allocated and deallocated from the heap. In the latter case, the responsibility of managing memory resides with the programmer. If the program does not deallocate an object, a memory leak occurs. If the program attempts to access or deallocate memory that has already been deallocated, the result is undefined and difficult to predict, and the program is likely to become unstable or crash. This can be partially remedied by the use of smart pointers, but these add overhead and complexity. Note that garbage collection does not prevent logical memory leaks, i.e. those where the memory is still referenced but never used.

Garbage collection may happen at any time. Ideally, it will occur when a program is idle. It is guaranteed to be triggered if there is insufficient free memory on the heap to allocate a new object; this can cause a program to stall momentarily. Explicit memory management is not possible in Java.

Java does not support C/C++ style pointer arithmetic, where object addresses can be arithmetically manipulated (e.g. by adding or subtracting an offset). This allows the garbage collector to relocate referenced objects and ensures type safety and security.

As in C++ and some other object-oriented languages, variables of Java's primitive data types are either stored directly in fields (for objects) or on the stack (for methods) rather than on the heap, as is commonly true for non-primitive data types (but see escape analysis). This was a conscious decision by Java's designers for performance reasons.

Java contains multiple types of garbage collectors. By default, HotSpot uses the parallel scavenge garbage collector. However, there are also several other garbage collectors that can be used to manage the heap. For 90% of applications in Java, the Concurrent Mark-Sweep (CMS) garbage collector is sufficient.Oracle aims to replace CMS with the Garbage-First Collector

Terminologies Involved:

1.Permutations 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.

2.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.

3.Roulette Wheel Selection:

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

1.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.

2.Swap Mutations:

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.

Algorithm Steps:

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.

1. 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.
2. Select two members from the current population. The chance of being selected is proportional to the chromosome’s fitness. Roulette Wheel Selection is a commonly used method.

3. Depending on the Crossover rate crossover the bits from each chosen chromosome at a randomly chosen point.
4. Step through the chosen chromosomes bits and flip dependent on the Mutation rate.
5. Repeat step 2, 3, 4 until a new population of N members has been created. Keep repeating until required fitness is achieved.

IMPLEMENTATION OF CODE DETAILS

Each user has their own functionalities as follows.

1)Function: Subject allocation and timetable generation along with the substitution
2)Input : Subject, faculty
3)Output : Timetable

4.1.2 Steps to Generate Timetable:

The user or the faculty needs to follow the given steps to perform the action:

Step 1: The user or the faculty need to register the page in order to get access to the application as shown in fig 5.3.1.1.

It needs the following details like:

- User name
- Email
- Country

4.2 SAMPLE CODE:

```
//Mainprogram.java
package scheduler;
import java.util.*;
public class SchedulerMain
{
/*
          * Time Table scheduling is an np-hard problem which can best be solved
          * using Genetic Algorithms (of Artificial Intelligence).
          * Conceps used here are Permutation encoding, elitism, roulette wheel selection,
 * single pt crossover,swap mutation
 */
          List<Chromosome> firstlist;
         List<Chromosome> newlist;
         double firstlistfitness;
         double newlistfitness;
         int populationsize=1000;
         int maxgenerations=100;
         public static Chromosome finalson;
         public SchedulerMain() {
           //printing input data (on console for testing)
Utility.printInputData();
           //generating slots
 new TimeTable();
          //printing slots (testing purpose only)
 Utility.printSlots();
          //initialising first generation of chromosomes and puting in first arraylist
 initialisePopulation();
         //generating newer generation of chromosomes using crossovers and mutation
  createNewGenerations();
}
//Creating new Generations using crossovers and mutations
         public void createNewGenerations(){
                Chromosome father=null;
            Chromosome mother=null;
            Chromosome son=null;
                 int nogenerations=0;
//looping max no of generations times or until suitable chromosome found
                    while(nogenerations<maxgenerations){
                    newlist=new ArrayList<Chromosome>();
                    newlistfitness=0;
                    int i=0;
//first 1/10 chromosomes added as it is- Elitism
                    for(i=0;i<populationsize/10;i++)
                         {
                         newlist.add(firstlist.get(i).deepClone());
                         newlistfitness+=firstlist.get(i).getFitness();
                            }
//adding other members after performing crossover and mutation
                    while(i<populationsize)
                              {
                         father=selectParentRoulette();
                         mother=selectParentRoulette();
//crossover
                         if(new Random().nextDouble()<inputdata.crossoverrate)
                            {
                              son=crossover(father,mother);
                         }
                              else
                              son=father;
//mutation
                         customMutation(son);
                         if(son.fitness==1)
                              {
                              System.out.println("Selected Chromosome is:-");
                              son.printChromosome();
```

```
                                        break;
                                                }


                                newlist.add(son);
                                newlistfitness+=son.getFitness();
                                i++;
                        }
                        //if chromosome with fitness 1 found
                        if(i<populationsize){
                System.out.println("****************************************************
                ************************************");
                System.out.println("\n\nSuitable Timetable has been generated in the "+i+"th
                Chromosome of "+(nogenerations+2)+" generation with fitness 1.");
              System.out.println("\nGenerated Timetable is:");
                son.printTimeTable();
                finalson=son;
                break;
        }
```

Chapter-V

TESTING

5.1 TEST LEVELS:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.1.1 UNIT TESTING:

It is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

5.1.2 INTEGRATION TESTING:

It is a level of software testing where individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Some different types of integration testing are big-bang, mixed (sandwich), risky-hardest, top-down, and bottom-up. Other Integration Patterns are : collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

In the big-bang approach, most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. This method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

5.1.3 VALIDATION TESTING:

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

Validation testing is also known as dynamic testing, where we are ensuring that "we have developed the product right." And it also checks that the software meets the business needs of the client.

5.1.4 SYSTEM TESTING:

It is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

5.2 TEST CASES:

5.2.1 TEST CASE:1

Inputs	Expected Output	Actual Output	Result
User or faculty need to register the page	Registered Successfully	Registered Successfully	Success
User or faculty login with valid username or password	Login successfully	Login successfully	Success
User or faculty login with invalid Username or password	Invalid Username or Password	Invalid Username or Password	Success

Table 5.2.1: Test Case for Timetable generation

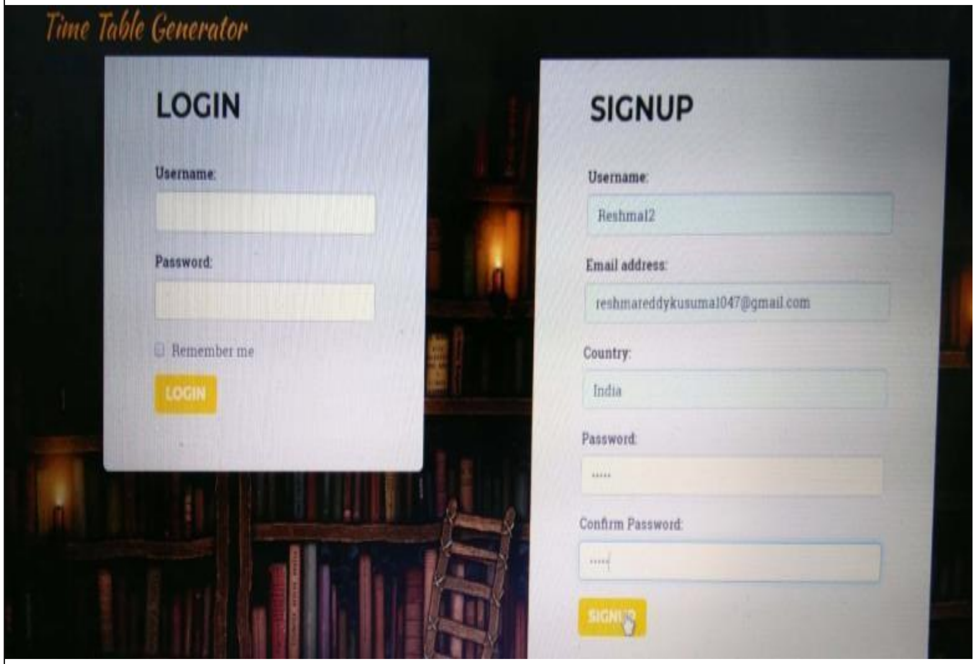
5.2.2 TEST CASE:2

Inputs	Expected Output	Actual Output	Result
Schedule a Timetable	Timetable Generation	Timetable Generation	Success
Invalid input details	Insufficient input details	Insufficient input details	Success

Table 5.2.2: Test Case for Timetable generation

Chapter-VI

RESULTS



LOGIN

Username:

Password:

SIGNUP

Username: Reshma12

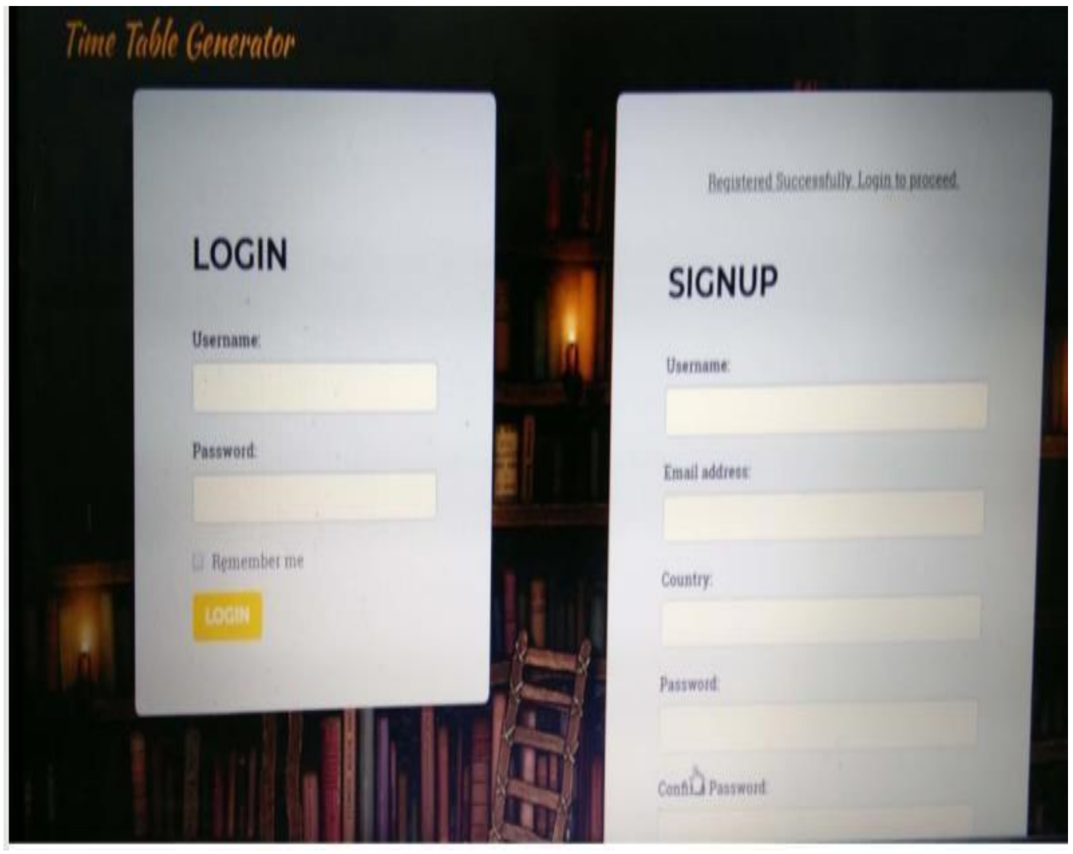
Email address: reshmareddykusuma1047@gmail.com

Country: India

Password: •••••

Confirm Password: •••••

Fig 6.1: Sign up page



Time Table Generator

LOGIN

Username:

Password:

Remember me

Registered Successfully. Login to proceed.

SIGNUP

Username:

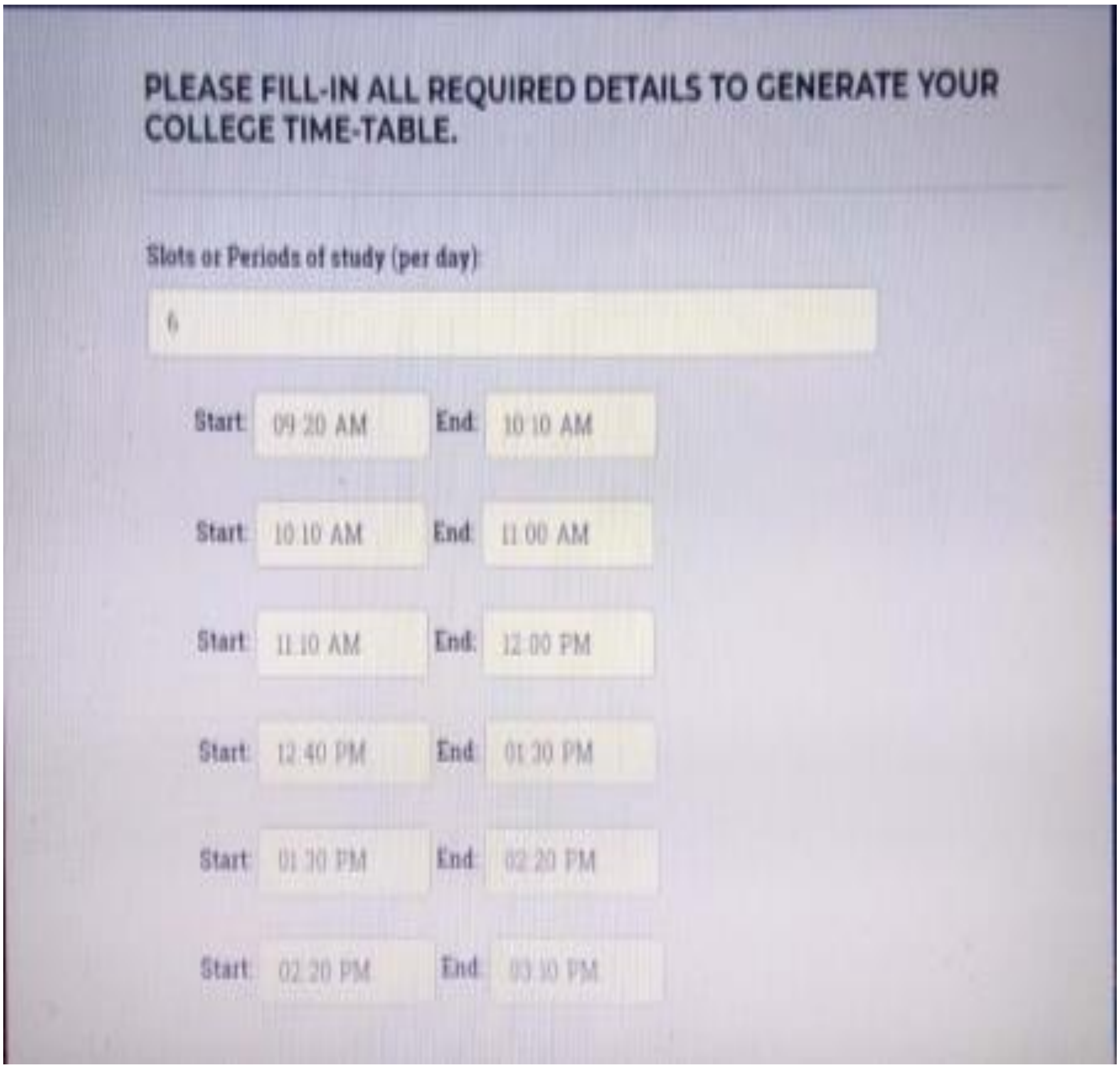
Email address:

Country:

Password:

Confirm Password:

Fig 6.2 :login page



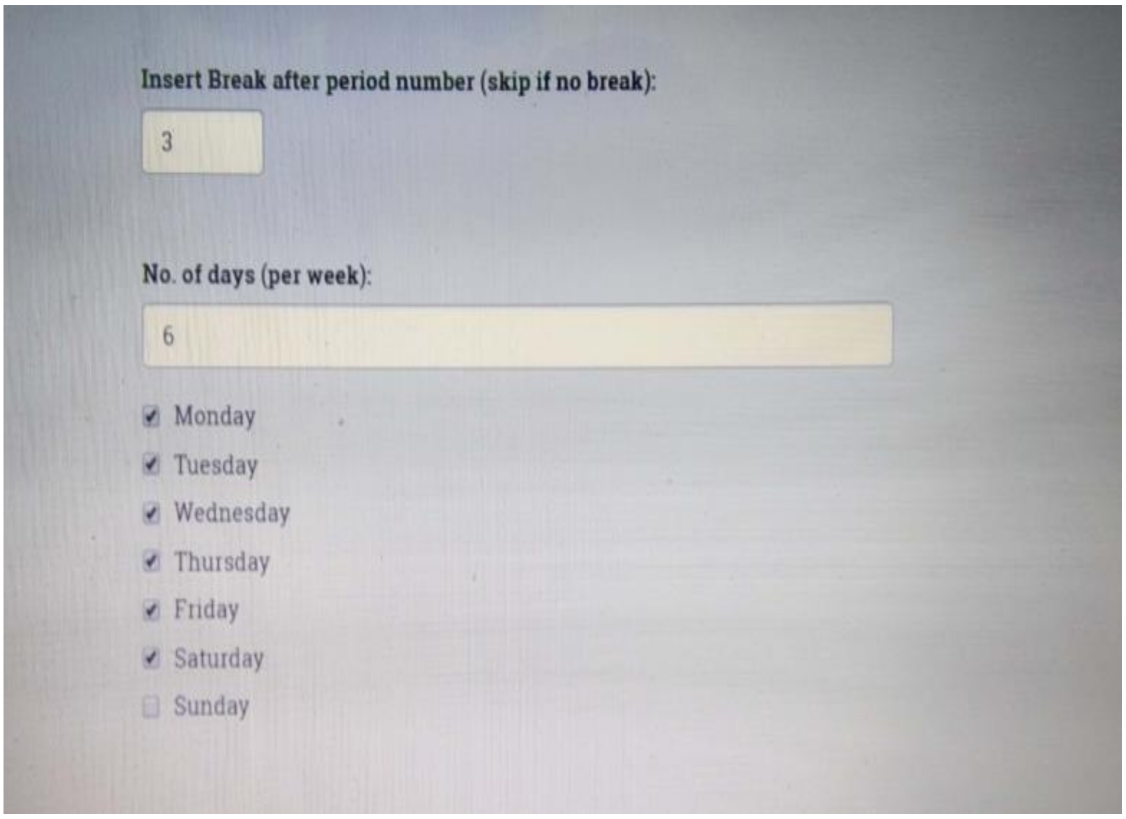
PLEASE FILL-IN ALL REQUIRED DETAILS TO GENERATE YOUR COLLEGE TIME-TABLE.

Slots or Periods of study (per day):

6

Start:	09:20 AM	End:	10:10 AM
Start:	10:10 AM	End:	11:00 AM
Start:	11:10 AM	End:	12:00 PM
Start:	12:40 PM	End:	01:30 PM
Start:	01:30 PM	End:	02:20 PM
Start:	02:20 PM	End:	03:10 PM

Fig 6.3 :No. of slots per day as input


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

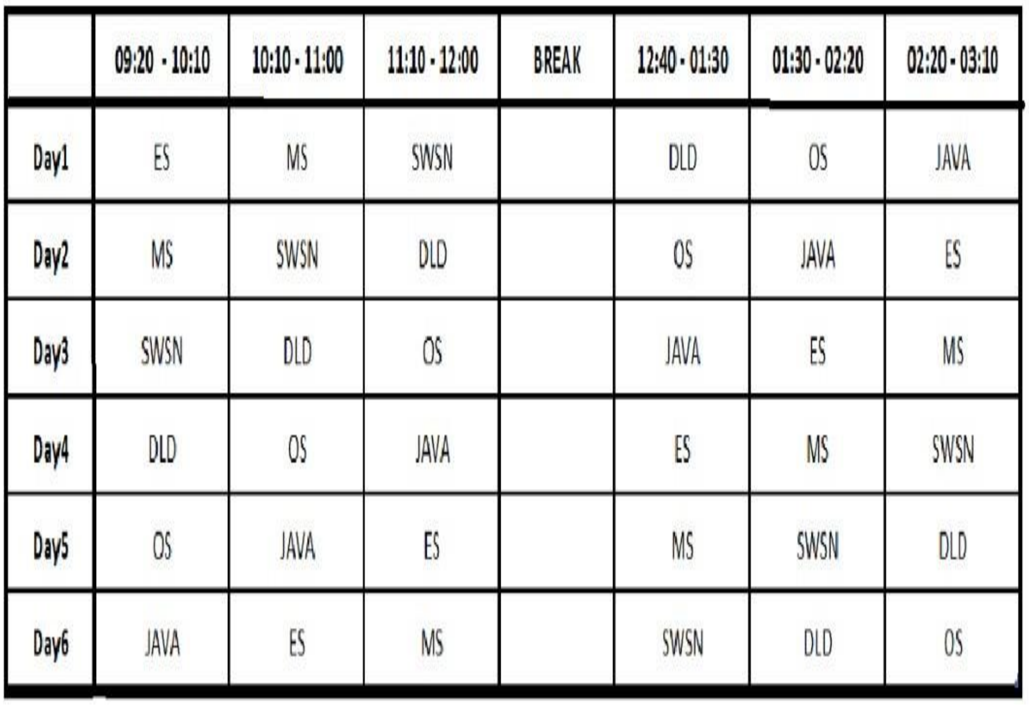
3

No. of days (per week):

6

- [✓ Monday]
- [✓ Tuesday]
- [✓ Wednesday]
- [✓ Thursday]
- [✓ Friday]
- [✓ Saturday]
- Sunday

Fig6.4: No. of days and break time as input



	09:20 - 10:10	10:10 - 11:00	11:10 - 12:00	BREAK	12:40 - 01:30	01:30 - 02:20	02:20 - 03:10
Day1	ES	MS	SWSN		DLD	OS	JAVA
Day2	MS	SWSN	DLD		OS	JAVA	ES
Day3	SWSN	DLD	OS		JAVA	ES	MS
Day4	DLD	OS	JAVA		ES	MS	SWSN
Day5	OS	JAVA	ES		MS	SWSN	DLD
Day6	JAVA	ES	MS		SWSN	DLD	OS

6.6 Timetable generation

Chapter-VII

CONCLUSION

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 or generating using html. So proposed system will help to overcome this disadvantage. Thus, timetable for any number of courses and multiple semesters can be generated. This system will help to create dynamic pages so that for implementing such a system make use of the different tools are widely applicable and free to use also.

The process of Time Table generation has been fully automated with this software. This web app can now cater to multiple colleges, universities and schools which can rely on it for their Time Table scheduling which earlier had to be done by hand.

Chapter-VIII

FUTURE ENHANCEMENTS

Though this web-app serves as a basic time table generator, there is a lot more which could be done to make this project even better in terms of consideration of soft constraints like professor giving preference to particular class. The up-gradations up to currently will be Classroom size considerations, lab facility consideration and multiple subject selection for faculty

More features such as schedule print for individual faculty etc. would also be involved to make this more useful as a final product.

BIBLIOGRAPHY

BOOKS:

[1] An Introduction to Genetic Algorithms, Melanie Mitchell, The MIT Press, 1999.

[2] Artificial Intelligence – A Guide to Intelligent Systems (Second Edition), Michael Negnevitsky, Addison-Wesley, 2005

[3] An Introduction to Genetic Algorithms for Scientists and Engineers, David A. Coley, World Scientific

[4] Introduction to Genetic Algorithms, S.N. Sivanandam and S.N. Deepa, Springer

[5] Multi-Objective Optimization using Evolutionary Algorithms, Kalyanmoy Deb, Wiley Student Edition

[6] Evolutionary Statistical Procedures, Roberto Baragona, Francesco Battaglia and

[7] Irene Poli, Springer-Verlag Berlin Heidelberg 2011

[8] Introduction to Evolutionary Computing, A.E. Eiben and J.E. Smith, Springer, 2003

WEBSITES:

- http://en.wikipedia.org/wiki/Genetic_algorithm
- http://ijsrcseit.com/paper/CSEIT1833406.pdf
- http://www.ai-junkie.com/ga/intro/gat3.html
- http://www.ijritcc.org/download/1416628378.pdf
- http://www.javatpoint.com
- http://www.obitko.com/tutorials/genetic-algorithms/encoding.php
- https://www.researchgate.net
- www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf