A MINI PROJECT

on

# AUTOMATIC TIMETABLE GENERATION USING GENETIC ALGORITHM

Submitted

In partial fulfillment for the requirement for the award of the degree of

**BACHELOR OF TECHNOLOGY**

in

## Computer Science and Engineering

by


**MODITHALA ABHISHEK** **20641A0560**


Under the Guidance of

**Mrs. Rajitha Bonagiri**

Assistant Professor



**Department of Computer Science & Engineering**

**Vaagdevi College of Engineering**

(UGC Autonomous, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban —506005(T.S)


(2020-2024)

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VAAGDEVI COLLEGE OF ENGINEERING**

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta , Khila Warangal (Mandal), Warangal Urban –506005(T.S)

# CERTIFICATE

This is to certify that the mini project entitled **"AUTOMATIC TIMETABLE GENERATION USING GENETIC ALGORITHM"** is submitted by **M.ABHISHEK (20641A0560)** in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in Computer Science and Engineering during the academic year 2023-2024.

**Project Guide**
Mrs.Rajitha Bonagiri

**Head of the Department**
Dr. N. Satyavathi

**External Examiner**

## ACKNOWLEDGEMENT

The development of the project though it was an arduous task, it has been made by the help of many people. We are pleased to express our thanks to the people whose suggestions, comments, criticisms greatly encouraged us in betterment of the project. We would like to express our sincere gratitude and indebtedness to my project.

We would like to express our sincre thanks and profound gratitude to Dr. K. Prakash,Principa1 of Vaagdevi College of Engineering, for his support, guidance and encouragement in the course of our project.

We are also thankful to the Head of the Department **Dr.N.Sathyavathi** for providing excellent infrastructure and a nice atmosphere for completing this project successfully. We are highly thankful to the Project Coordinators for their valuable suggestions, encouragement and motivations for completing this project successfully.

We are thankful to the Guide **Mrs.Rajitha Bonagiri** for her valuable suggestions and interest throughout the course of this project.

Finally, we would like to take this opportunity to thank our families for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

Modthala Abhishek 20641A0560

# ABSTRACT

The goal of this project is to create a time table generator for colleges. The creation of schedules is a very common issue that affects all educational institutions. The conflict between staff members' preferences is precisely where the issue arises. Every semester, colleges are required to create time tables, which used to be an extremely time-consuming task. Once the timetables are set for a given semester, the student is allowed to access them. Once the timetables are established for a particular semester, employees are also permitted to check the class allotment schedule.

The Time Table Assignment for Any Department project's goal was to create an application that would allow staff and student allotment subject to classes. Following information was added by the administrator for Add the student, the staff, the subject, enter the timetable, and update the timetable. The majority of colleges offer a variety of programmes, each of which has a number of disciplines.

There are now a limited number of faculties, each of which teaches many disciplines. Therefore, the timetable now has to include the instructors at the appropriate times. the timetable schedule, which makes the most use of all faculty subject demands, slots so that their timings do not cross. For this, a genetic algorithm is employed. We suggest using a timetable object in our method for creating timetables.

This object consists of classroom objects, their respective schedules, and a fitness rating for the schedule. Additionally, in order to further describe the imperatives, we used a composite configuration design that is easily expandable to include or uproot as many duties. Every obligation class now checks the condition found in our investigation between two timetable objects. In the unlikely event that the requirement is met, the score is raised by one if a crash is available.

# CONTENTS

# FIGURE INDEX

# CHAPTER 1

# INTRODUCTION

Automatic Timetable Generator system is an automated system which generates time table according to the data given by the user. Time table scheduling has been in human requirements since they thought of managing time effectively. It is widely used in schools, colleges and other fields of teaching and working like crash courses, couching centers, training programs etc.

Timetable scheduling is the process of creating timetables that fit the constraint of the scenario. It is used in a magnitude of industry from scheduling transportations up to creating complex schedule for highly optimized automated factories. Majority of small scale scheduling are done manually while larger operations require computer assisted scheduling.

Artificial intelligence is one of the rising computing solution due to an increase in computing power. It can be applied to different types of problems which can help optimize existing solutions or create never been tried solutions due to multiple limitations. Artificial intelligence helps create solutions for non-deterministic polynomial time problems which businesses and organizations will always have.

Genetic algorithm is a metaheuristic that mimics the process of natural selection. It can be performed in multiple different ways with different types but it will all follow the same concept. This research aims to create an artificial intelligence through the use of evolutionary algorithm, specifically genetic algorithm combined with adaptive and elitist traits that can generate a university schedule timetable with the goal of generating a valid and as optimal as possible solution with certain constraints.

## 1. Existing System

Normally timetable generation done manually. As we know all Institutions or organizations have its own timetable, managing and maintaining these will not be difficult. Considering workload with this scheduling will make it more complex.

Each operation must be completed manually under the current system, and processing is a time- consuming effort as well. In the prior method, institutions had to manually manage their timetable information on paper and ink, which took time and money. The organisation can't meet its demands in a timely manner, and the outcomes might not be reliable. Numerous issues and short comings with the system are caused by manual upkeep.

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.

## 1. Limitations of Existing System

- If any student, staff entry is wrongly made then the maintenance becomes very difficult.
- It requires a lot amount of time to sort all their schedules without any collisions. Moreover, there is a high chances of collisions between classes.
- It also requires human power but whereas in this automatic timetable generator all you need is few clicks.

## 2. Proposed System

Automatic Timetable manger is a Java based software used to generate timetable automatically it will help you to manage all the periods automatically. Proposed system will help to generate it automatically also helps to save time. There is no need for Faculty to worry about their period details and maximum workload. It is a comprehensive timetable management solution for Colleges which helps to overcome the challenges in current system.

The purpose of this work is to illustrate how effectively evolutionary algorithms can be used to find the best solutions for scheduling Timetables in general. Despite the abundance of commercial scheduling software, its lack of generality makes it difficult for it to Satisfy the needs of varied institutions. The main challenge to overcome is the demand of particular coding as per the distinct colleges. An institute must deal with a number of restrictions when constructing a schedule. These constraints can be categorised as Either "hard" or "soft" based on whether they are necessary or desirable.

## 1. Advantages of Proposed System

### No paper Work

- The toughest thing about manual school management is the endless list of paperwork.

### No More Confusions

- Timetable software, it is possible for you to automatically generate timetables effectively, sticking by the requirements.

### Error Free

- When you manually prepare a timetable, you tend to make errors You may end up with allotting two teachers for a single class at a time or vice versa or you may miss the allotment of certain periods Timetable management were helps you to avoid such errors.

## 1.3 Software Requirements

The prerequisites software &libraries for the real time communication system are:-

- The operating system on which the software can run easily windows, Linux/Unix, macOS.
- The front-end software is JavaScript.
- The back end of this is Java.
- VScode

## 1.4 Hardware Requirements

The prerequisites hardware for the real time communication system project are:-

- Laptop or PC
- Windows 7 or higher
- 13 processor system or higher
- 8 GB RAM or higher
- 100 GB ROM or higher

## 1.5 Literature Survey

1.D. Nguyen, K. Nguyen, K. Trieu, and N. Tran (2010), have automated the scheduling issue at universities using the Tabu search technique. In the search space that the Tabu Search Algorithm uses, viable results are seen. The search space is the set of feasible solutions to the issue. Author uses "Tabu," or basic building blocks. Tabus allows you to stop cycling and move awayfrom non- improving motions and local optimums. The Benefit of Tabu Search process prevents using memories to cycle back to the prior findings, allowing for additional development, But evaluating resources is expensive and formulating the problem is hard which is the drawback of this approach.

2.N. M. Hussin and A. Azlan (2013), have put into practise the graph colouring heuristic method for building process scheduling difficulty stages. Schedule challenge is addressed as a graphical representation issue. The most important component is the building stage, which results in a resident of ideal solutions. The following phase is the improvement phase, where the best possible answer is produced. This approach does not schedule soft restrictions and takes a very lengthy time to solve a problem.

3.W. F. Mahmudy and R. E. Febrita (2017), Use fuzzy logic to create and carry out timetable scheduling that incorporates multiple genetic operators. The proportion of truth of a statement may vary between 0 and 1, depending on the membership values of formal fuzzy logic variables, which may not always be 0 or 1. By employing linguistic factors, a more stable state is reached in a shorter amount of time. Membership function evaluation is difficult, hard to create a fuzzy logic model, more calibrate tuning and simulation is required before using for any application. These are some of the major drawbacks of this method.

# CHAPTER 2

# DESIGN OF THE PROJECT

## 2.1 Block Diagram of the System

The software can simply be compared to a schedule computing tool that accepts simple data sets as input and produces organised results. Using the straightforward input-process model We fill out the form with all the information, and a genetic algorithm applies it to the data on the backend to create class schedules.

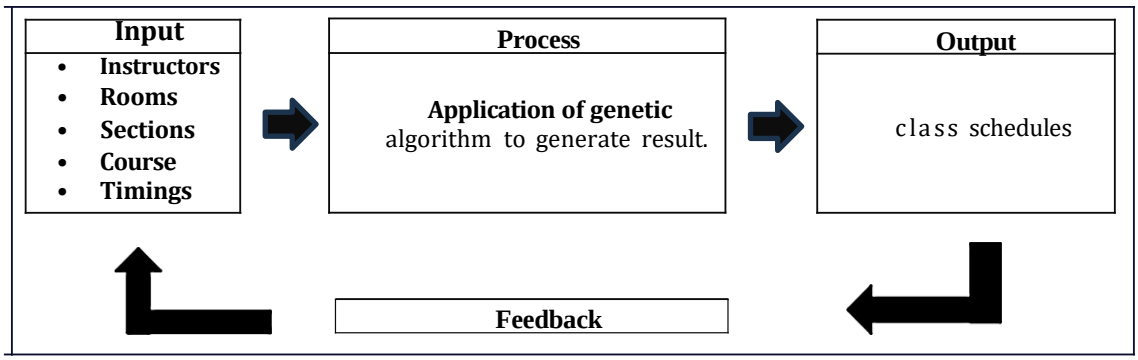



Fig 2.1: Simplified IPO model of the Program

## 2.2 Genetic Algorithm

Natural selection, a process in biological evolution, is a process that the genetic algorithm mimics. The best way to visualise the repeating process is through a flowchart. Instead of using the term "person," the term "chromosome" is used in genetic algorithms.

The fundamentals of a genetic algorithm are depicted in the diagram below. However, in actual use, the system adds an additional phase known as environment modification following evaluation. A strong programming tool for issue solving is the genetic algorithm (GA). It belongs to the class of evolutionary algorithms, which is a subset of artificial intelligence's evolutionary computation. Professor John Holland of the University of Michigan created it in 1960. In the 1970s, his book Adaptation in Natural and Artificial Systems helped launch the field of genetic algorithm (GA) study.

The Darwinian theory of natural evolution, which holds that species in the world multiply in geometric proportions resulting to a battle for life mostly owing to a lack of food and space, served as the inspiration for this technique. The strongest will prevails in this conflict. The variants that are most suited to survival are those whose accumulation led to the evolution of species. Organisms with harmful mutations have very little chance of surviving. Natural selection is the mechanism that leads to evolution.

A random beginning point will be chosen for the population generation, and a mixture of greedy and random approaches will be used to fill the table with any suitable entities. All rigid restrictions must be adhered to throughout population generation. While soft constraints are completely disregarded, medium requirements will go through several tries to be followed.

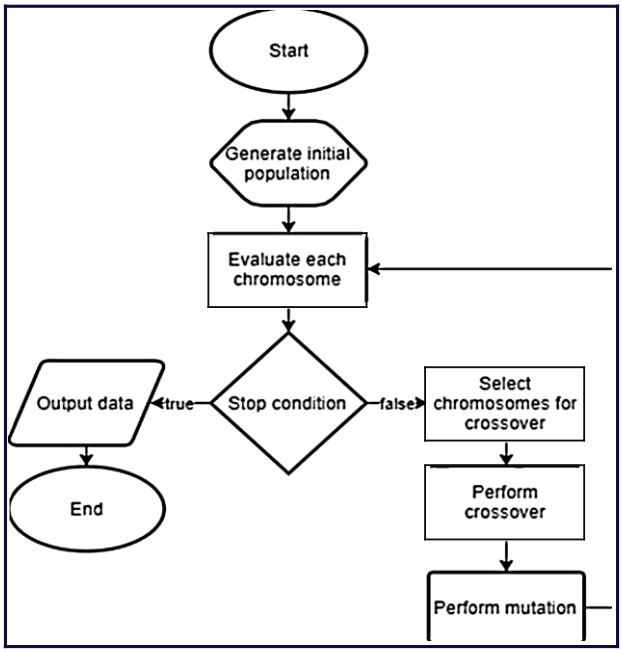



Fig 2.2: Flowchart of Genetic Algorithm

## 2.3 Steps involved in Genetic Algorithm

The whole method of scheduling based on genetic algorithm is explained in detail in this section. A scheduling procedure is divided into several important modules are as follows,

**1. Data Encoding and Decoding.**

The initial step before beginning a Genetic Algorithm is data encoding. To obtain a straightforward value, such as a string, it converts a solution into a chromosome. It is used to increase the algorithm's speed. Making the data into a binary string is a simple technique to accomplish this. A gene is a component of a chromosome and can also be transformed into a binary string.

**2. Initial Population**

It is the initial phase of GA. It involves producing a large number of randomly chosen people under strict limits. The user's needs determine the population choice. Due to evolution, a tiny population will eventually become even smaller and exterminate the entire population. On the other side, a huge population will produce better outcomes but will be slower and need more resources.

**3. Evaluation of Population**

A solution's fitness is an assessment of its quality made utilising soft restrictions. The answer is appropriate for this range. The core of the genetic algorithm is population evaluation. The fitness can be expressed using a range from 0 to 1, with 1 being the population's best estimate, and using the other individuals to range them. There will always be a solution where the fitness is 1 in this situation, and a solution where the fitness is 0.

**4. Crossover Evolution**

A technique for creating a new population based on an older population is crossover evolution. The two-chromosome simple crossover evolution allows for the creation of X additional chromosomes. The two chromosomes are divided into pieces, and new chromosomes are made from various pieces. Mutation: The algorithm is made to move by means of mutation. By randomly altering a gene's values, a novel, unexpected solution is produced.

**5. Mutation**

Mutation is used to get the algorithm moving. It consists of changing the values of a gene randomly, resulting in a new unexpected solution. These solutions offer a new point of view for the fitness function. The mutation changes only the chromosome, without affecting others solutions.

**5. New Population**

The crossover and the mutation permit to create a new population of original solutions.

# CHAPTER 3

# IMPLEMENTATION

## 3.1 Packages Used

**Java Util Package** ( **java.util** ) : Java util package provides the basic utility classes to java programmers. It is one of the most useful package for java programmers, it helps them to achieve different types of requirements easily by using it's predefined classes. The program below uses class of this package to sort an array of integers. The class provides many API's(methods) for different array requirements.

**Java Io Package** ( **java.io** ) : Java IO(Input/Output) package provides classes and interfaces for handling system(computer, laptop etc) input and output operations. Using these classes programmer can take the input from user and do operations on that and then display the output to user. Generally input is given using keyboard/keypad. We can also do file handling(read/write) using the classes of this package.The program below uses the Console class of java.io package to take the input from user and then print that input to user's screen(command prompt or any other terminal used).

**Java AWT package (java.awt** ) : Java AWT(Abstract Window Toolkit) package contains classes and interfaces used to develop graphical user interface or window based applications in java. Java AWT components are platform-dependent since they uses operating system resources to create components like textbox, button, checkbox etc. The program below creates a frame containing a label using java.awt classes.

**Java Swing pacakge ( java.swing)** : Java Swing is a popular and powerful Graphical User Interface (GUI) toolkit that is used for developing desktop applications. It is a part of the Java Foundation Classes (JFC) and provides a rich set of components and layout managers for creating a variety of **GUIs.** Java Swing is platform-independent and can be used on any operating system that supports Java.

## 3.2 CODE

## Main.java

```
import  java.awt.*;
import  java.util.*;
import java.awt.event.*;
import javax.swing.*;
import java.awt.image.*;
import java.io.*;
import javax.imageio.ImageIO;

class Main{
public static void main(String[] args) throws IOException {
    Map<String,Batch> B = new HashMap<String,Batch>();
    Map<String,Course> CO = new HashMap<String,Course>();
    Map<String,Professor> PR = new HashMap<String,Professor>();
    JFrame f=new JFrame();
    JFrame s=new JFrame();
    JPanel home=new JPanel();
    JLabel l1=new JLabel("Welcome to Timetable Generator");
    l1.setBounds(250,50,850,200);
    l1.setFont(new Font("TimesRoman",Font.BOLD,35));
    BufferedImage image = ImageIO.read(new File("timetable.jpeg"));
    JLabel label = new JLabel(new ImageIcon(image));
    label.setSize(600,500);
    label.setBounds(300,20,500,600);
    home.setLayout(null);
     home.add(l1);
    JPanel view=new JPanel();
    String[] Timet={"Select",};
    String[] profs={"Select"};
    String[] credits={"1","2","3","4","5","6"};
```

```
String[] course={"Select"};
   JComboBox cr=new JComboBox(credits);
   JComboBox cb=new JComboBox(Timet);
   JComboBox pf=new JComboBox(profs);
   JComboBox co=new JComboBox(course);
   JTabbedPane t=new JTabbedPane();
                JLabel vl=new JLabel("Timetable");
                vl.setBounds(530,5,250,70);
   vl.setFont(new Font("TimesRoman",Font.BOLD,20));
                final String timet[][] = { {"Monday","","","","","","","","","},
                  {"Tuesday","","","","","","","","","},
                  {"Wednesday","","","","","","","","","},
                  {"Thursday","","","","","","","","","},
                  {"Friday","","","","","","","","","},
                  {"Saturday","","","","","","","","","}};
                String column[]={"","1","2","3","4","5","6","7","8"};
   JTable jt=new JTable(timet,column);
   jt.setSize(1000,100);
   JScrollPane sp=new JScrollPane(jt);
   sp.setSize(1000,150);
   sp.setBounds(100,150,1000,125);
   cb.setBounds(475,70,100,20);
   pf.setBounds(600,70,100,20);
   cb.addActionListener(new ActionListener(){
                    public void actionPerformed(ActionEvent e){
        for(int i=0;i<6;i++){
           for(int j=0;j<8;j++){

timet[i][j+1]=B.get(cb.getItemAt(cb.getSelectedIndex()).toString()).a[i][j];
           }
        }
        view.revalidate();
        view.repaint();
      }
   });
```

```
view.setLayout(null);
    view.setBackground(Color.orange);
                    view.add(vl);
                    view.add(cb);
    view.add(pf);
    jt.setVisible(true);
    sp.setVisible(true);
    view.add(sp);
    JPanel delete=new JPanel();
    JLabel d1 = new JLabel("Delete Course");
    d1.setBounds(50,50,500,50);
    d1.setFont(new Font("TimesRoman",Font.BOLD,30));
    JLabel d2 = new JLabel("Course");
    d2.setFont(new Font("TimesRoman",Font.BOLD,20));
    d2.setBounds(50,125,200,50);
    JComboBox d3 = new JComboBox(course);
    d3.setBounds(250,125,200,50);
    JButton d4 = new JButton("Delete");
    d4.setBounds(550,125,200,50);
    d4.setBackground(Color.cyan);
    JLabel d5 = new JLabel("Delete Professor");
    d5.setFont(new Font("TimesRoman",Font.BOLD,30));
    d5.setBounds(50,200,500,50);
    delete.add(d1);
    delete.add(d2);
    delete.add(d3);
    delete.add(d4);
    delete.add(d5);
    JPanel createBatch=new JPanel();
    JLabel cb1= new JLabel("Name");
    cb1.setFont(new Font("TimesRoman",Font.BOLD,20));
    cb1.setBounds(50,100,200,50);
    JTextField cb2 = new JTextField();
```

```
cb2.setBounds(250,100,200,50);
    JLabel cb3 =new JLabel("Courses");
    cb3.setFont(new Font("TimesRoman",Font.BOLD,20));
    cb3.setBounds(50,275,200,50);
    JComboBox cb4 = new JComboBox(course);
    cb4.setBounds(250,275,200,50);
cb5.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            if(cb4.getSelectedIndex()!=0 && cb7.getSelectedIndex()!=0){
if(B.get(cb2.getText()).add(CO.get(cb4.getItemAt(cb4.getSelectedIndex()).toString()),PR.get(cb7.ge
tItemAt(cb7.getSelectedIndex()).toString()))){
                JOptionPane.showMessageDialog(s,"A new course has been added to the batch
"+cb2.getText()+"\n"+"Course:
"+cb4.getItemAt(cb4.getSelectedIndex()).toString()+"\n"+"Professor:
"+cb7.getItemAt(cb7.getSelectedIndex()).toString());
                }
                else{
                JOptionPane.showMessageDialog(s,"Periods are insufficient for adding this course to
the batch");
                }
            else{
                JOptionPane.showMessageDialog(s,"Select a valid course and valid professor");
            }
    });
    cb4.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            cb7.removeAllItems();
            cb7.addItem("Select");
            for(int i=0;i<CO.get(cb4.getItemAt(cb4.getSelectedIndex()).toString()).getProfsSize();i++){
                cb7.addItem(CO.get(cb4.getItemAt(cb4.getSelectedIndex()).toString()).getProfsItem(i));
            }
        }
 });
```

```
createBatch.setLayout(null);
    createBatch.setBackground(Color.orange);
    createBatch.add(cb1);
    createBatch.add(cb2);
    createBatch.add(cb3);
    createBatch.add(cb4);
JPanel createProfessor=new JPanel();
    JLabel cp1=new JLabel("Name");
    cp1.setFont(new Font("TimesRoman",Font.BOLD,20));
    cp1.setBounds(50,100,200,50);
    JTextField cp2=new JTextField();
    cp2.setBounds(250,100,200,50);
    JLabel cp3=new JLabel("Course");
    cp3.setFont(new Font("TimesRoman",Font.BOLD,20));
    cp3.setBounds(50,275,200,50);
    JPanel createCourse=new JPanel();
    JLabel cc1= new JLabel("Name");
    cc1.setFont(new Font("TimesRoman",Font.BOLD,20));
    cc1.setBounds(50,100,200,50);
    JButton cc5 = new JButton("Create");
    cc5.setBounds(250,450,200,50);
    cc5.setBackground(Color.cyan);
    cc5.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent e){
          d3.addItem(cc2.getText());
          cb4.addItem(cc2.getText());
          cp4.addItem(cc2.getText());
          CO.put(cc2.getText(),new Course());
          CO.get(cc2.getText()).setName(cc2.getText());
});
    createCourse.setLayout(null);
    createCourse.setBackground(Color.orange);
    createCourse.add(cc1);
    createCourse.add(cc2);
    createCourse.add(cc3);
```

```
t.add("Home",home);
   t.add("View",view);
   t.add("Create Batch",createBatch);
   t.add("Create Course",createCourse);
                  t.add("Create Professor",createProfessor);
   t.add("Delete",delete);
   t.setBackground(Color.green);
   f.add(t);
   f.setSize(1500,800);
   t.setBounds(50,50,1200,600);
   f.setLayout(null);
   f.setVisible(true);
}
}
```

## Timetable.java

```
import java.util.*;
public class Timetable{
private String name;
    String a[][] = new String[6][8];
     void setName(String s)(
       this.name=s;
       }
            String getName(){
                return this.name;
           }
         }
```

## Batch.java

```
Import java.uti1.*;
 Public class Batch extends Timetable( Static Random rand = new Random();
```

```
  Static ArrayList<Integer> list = new ArrayList<Integer>();
    Boolean add(Course x,Professor y)(
       Int I,j,k,m,n;
       For(i=0;i<6;i++)( For(j=0;j<8;j++){
                If(y.a[i][j]==null && this.a[i][j]—=null){
                List.add(i*10 + j);
            }
        }
   }

If(list.size()>=x.getCredits())(
   For(i=0;i<x.getCredits();i++)(
      K=rand.nextInt(list.size());
      N=1ist.get(k)%10; M=1ist.get(k)/10;
   This.a[mI[n]=x.getName()+"("+y.getName(
                              )+")";
      y.a[m][n]=this.getName()+"("+x.getName()+")";
      list.remove(k);
      }
 List.clear();
 Return true;
 }
       List.clear();
          Return false;
      }
 }
```

## Course.java

```
import java.util.*;
public class Course(
    private String
    name; private
    int credits;
```

```
    private ArrayList<String> profs = new ArrayList<String>(); void setName(String s)(
        this.name = s;
        }
void setCredits(int d)(
   this.credits=d;
   }
String getName()(
   return this.name;
   }
int getCredits()(
   return this.credits;
   }
void addProfs(String s)(
   this.profs.add(s);
   }
 int getProfsSize(){
  return this.profs.size();
  }
 String getProfsItem(int d)(
 return this.profs.get(d);
       }
 }
```

**Professor.java**

```
 import java.uti1.*;
  public class Professor extends Timetable(
     private ArrayList<String> courses = new
     ArrayList<String>();
     void addCourses(String s)(
        this.courses.add(s);
        }
        }
```

# CHAPTER 4
# TESTING

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, subassemblies, 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 tests. Each test Type addresses a specific testing requirement.

## 5.1 TYPES OF TESTS

**Unit testing:**Unit testing involves the testing of each unit or an individual component of the software Application. It is the first level of functional testing. The aim behind unit testing is to validate unit Components with its performance. A unit is a single testable part of a software system and tested during the development phase of the application software.

**Integration testing:**Integration testing is the second level of the software testing process comes after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the Integration testing level is to expose defects at the time of interaction between integrated components or units. Unit Testing uses modules for testing purpose, and these modules are combined and tested in Integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

**Functional testing:**It is a type of software testing which is used to verify the functionality of the software Application, whether the function is working according to the requirement specification. In functional Testing, each function tested by giving the value, determining the output, and verifying the actual output With the expected value. Functional testing performed as black-box testing which is presented to Confirm that the functionality of an application or system behaves as we are expecting. It is done to Verify the functionality of the application.

# CHAPTER 5

# RESULT

This system generates separate timetables for each class, faculty member, and lab automatically. The project is designed to prevent slot conflicts and provides tools for customising the schedule as needed.


Fig 5.1: Homepage of Timetable Generator

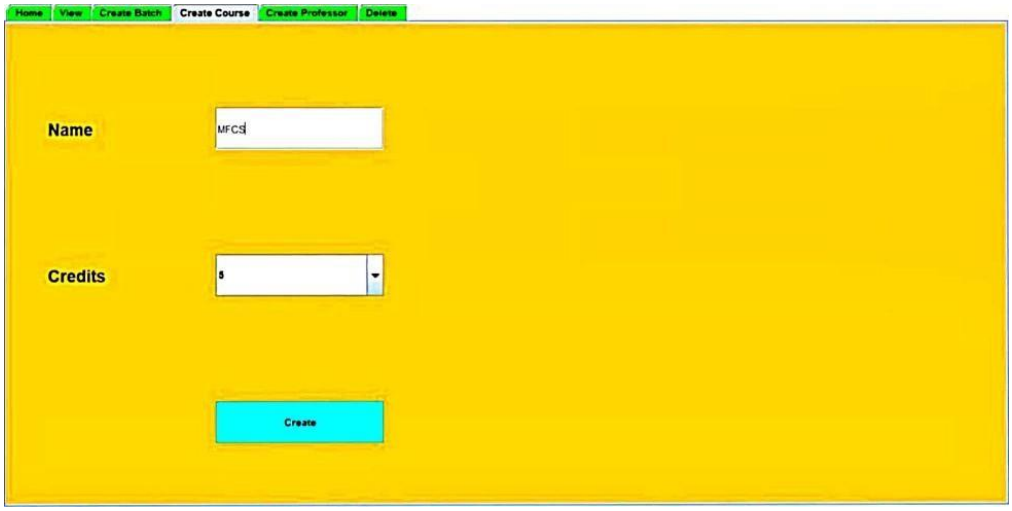



Fig 5.2 : Create Batch page of Timetable Generator

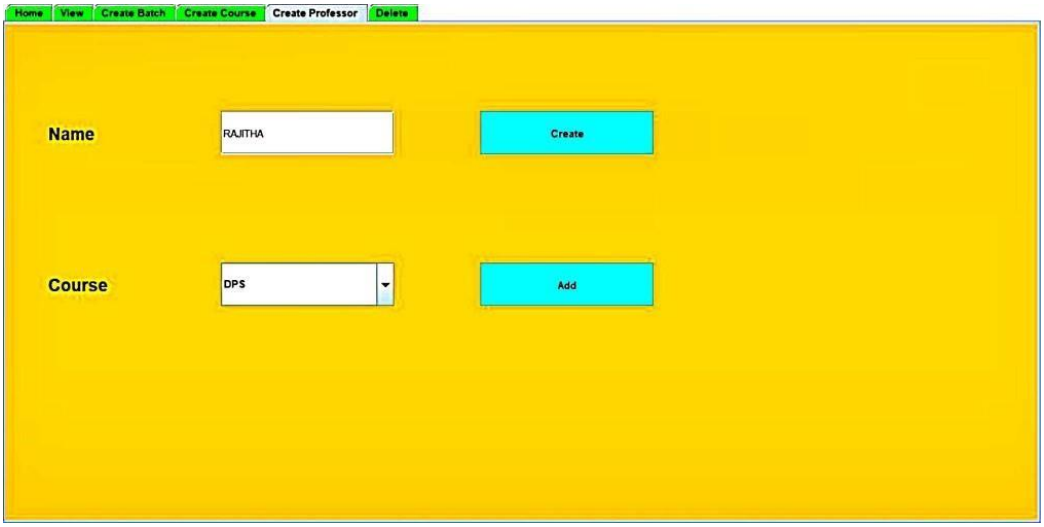
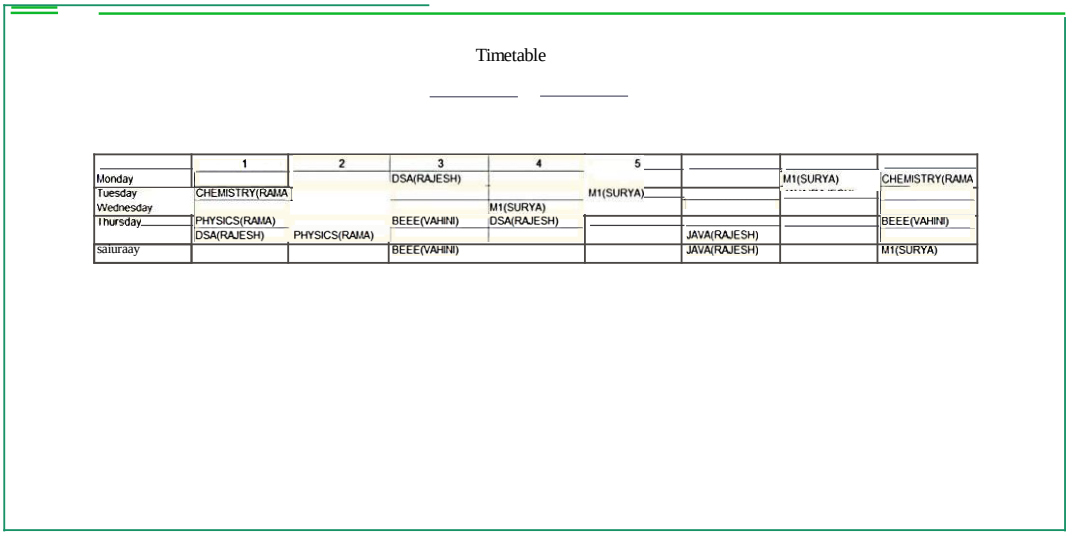



Fig 5.3 : Create Professor Page




| | 1 | 2 | 3 | 4 | 5 | | | |
|---|---|---|---|---|---|---|---|---|
| Monday | | | DSA(RAJESH) | | | | M1(SURYA) | CHEMISTRY(RAMA |
| Tuesday | CHEMISTRY(RAMA | | | | M1(SURYA) | | | |
| Wednesday | | | | M1(SURYA) | | | | |
| Thursday | PHYSICS(RAMA) | | BEEE(VAHINI) | DSA(RAJESH) | | | | BEEE(VAHINI) |
| | DSA(RAJESH) | PHYSICS(RAMA) | | | | JAVA(RAJESH) | | |
| saturaay | | | BEEE(VAHINI) | | | JAVA(RAJESH) | | M1(SURYA) |

Fig 5.4 : View Page

The above figures describe the process of creating Timetable by just giving the subject and it's corresponding credits and it generates a timetable by considering all the constraints.

# CHAPTER 6
# CONCLUSION AND FUTURE SCOPE

## 6.1 Conclusion

The Automatic Timetable Generator project is a vital tool for educational institutions, businesses, and organizations seeking to streamline and optimize their scheduling processes. By automating the complex task of timetable creation, this project offers a solution that not only saves time and effort but also enhances resource utilization, reduces conflicts, and improves overall scheduling accuracy

With a user-friendly interface, robust data management, and the use of advanced scheduling and optimization algorithms, our project aims to deliver a highly efficient system. It can handle a wide range of constraints and preferences, ensuring that the generated timetables meet the unique requirements of our users.

As educational institutions, businesses, and organizations increasingly demand precision, efficiency, and flexibility in their scheduling processes, the Automatic Timetable Generator project is positioned to become an indispensable tool, resulting in time and cost savings and enhancing overall operational effectiveness.

### 6.2 Future Scope

This project will be very beneficial to the university because managing numerous faculties and assigning courses to them Simultaneously by hand is a very challenging task that this project will assist in managing effectively. This faculty timetable can be Readily controlled while taking into account the maximum and lowest workload. The faculty data in the database can also be used to Keep track of the faculty's expertise in specific fields. Attribute The accuracy of the project will allow for a more corrective Approach to the creation of this schedule. This project will produce output that is mostly corrective and error-free. The project's Potential future improvement is the creation of a master schedule for the departments and the entire college. Further adjustments can Be made while maintaining the project's approach and methods to accomplish this improvement. Additionally, it can be utilised to Assign a certain time slot that the instructor prefers. The university website may incorporate this timetable maker, making it more useful.

# CHAPTER 7
# REFERENCES


1K. Nguyen, D. Nguyen, K. Trieu, and N. Tran, “Automating a real-world university timetabling problem with Tabu search algorithm,” in 2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010.

2Azlan and N. M. Hussin, “Implementing graph coloring heuristic in construction phase of curriculum- based course timetabling problem,” in 2013 IEEE Symposium on Computers & Informatics (ISCI), 2013.

3R. E. Febrita and W. F. Mahmudy, “Modified genetic algorithm for high school time-table scheduling with fuzzy time window,” in 2017 International Conference on Sustainable Information Engineering and Technology (SIET), 2017.

4T. Elsaka, “Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling,” in 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), 2017.

5F. D. Wihartiko, H. Wijayanti, and F. Virgantari, “Performance comparison of genetic algorithms and particle swarm optimization for model integerProgramming bus timetabling problem,” IOP Conf. Ser. Mater. Sci. Eng., vol. 332, p. 012020, 2018.

6D. Wang, D. Tan, and L. Liu, “Particle swarm optimization algorithm: an overview,” Soft Comput., vol. 22, no. 2, pp. 387—408, 2018.

7K. Y. Junn, J. H. Obit, and R. Alfred, “The study of genetic algorithm approach to solving university course timetabling problem,” in Lecture Notes in Electrical Engineering, Singapore: Springer Singapore, 2018, pp. 454—463.

8Aminu et al., “Design and implementation of an automatic examination timetable generation and invigilation scheduling system using genetic algorithm,” In 2"d International Conference on Applied Engineering (ICAE), 2019.