**1. INTRODUCTION**

Time Table Management system is an automated system which generates time table according to the data given by the user. The main requirement of the application is to provide the details about the branch, subjects, number of labs, total number of period and details about the lab assistance. Then the application generates the time table as per the requirement.

**1.1 Terminologies and Definition**

Development of **College** **Timetable Management System** to generate and manage timetable for an institution, the following terminologies were used.

* Faculty Id: Each faculty has a unique id, which cannot be altered. It is used by a teacher to login to the system and use it.
* Subject Id: Each subject has a unique id given by the university.
* Lab Id: Each lab has a unique id which cannot be altered.
* Admin Rights: Admin can only generate the timetable no other faculty can generate timetable. Admin rights is not given to all faculty it may or may not be given to all head of the department.

**1.2 Problem Definition**

**College Timetable Management System** is an automated system developed for any institution. Timetabling is the allocation, subject to constraints, of given resources to objects in space-time domain to satisfy a set of desirable objectives as nearly as possible. Particularly, the university timetabling problem for courses can be viewed as fixing in time and space a sequence of meetings between instructors and students, while simultaneously satisfying a number of various essential conditions or constraints. Planning timetables is one of the most complex and error-prone applications. There are still serious problems occurring and these problems are repeating frequently. Therefore there is a great requirement for an application distributing the courses evenly and without collisions. Graph coloring algorithm is one of the most used algorithms. [1]

Manually creating a timetable and managing it is a very tedious job and maintain all the databases required to design the timetable is very time consuming. So we intend to design an automated system which will generate the timetable considering faculty details, teacher details, semester details etc.

* 1. **Objectives**

To create a **College** **Timetable Management System** to be used by any Institution and will reduce time consumption and the labor of manually designing the timetable and managing it. To create a leave management system to manage the leaves of faculties and to post a notice for substitute class or extra class.

* 1. **Tools & Platform**

**1.5. Project Organization**

**1.6. Assumptions**

The application is designed for St. Thomas’ College of Engineering and Technology on that basis following assumptions are made while designing the system:

1. Classroom are assigned to each year and department manually for their theoretical lecture.
2. Classes are held from 9:30am -5:30pm on weekdays and have 7 periods for lectures/labs to be assigned excluding recess time.
3. Classes are held from 9:30am -1:45pm on Saturday and have 4 periods for lecture/labs to be assigned excluding recess time.
4. On some days classes may start late or finish early so starting or ending periods can be empty.
5. Faculty are assigned the subjects they will teach by Head of the Department manually.
6. Users have basic knowledge of computer

**2. Literature Review**

There are few research working papers on university timetable that are published. Most of the timetable system produced can only be used at the university involved in the research, as each university has its own needs and requirements that differs specifically.

**2.1. Graph Coloring:**

Most timetable software is based on graph delegation. Graph can be defined as vertex connected by lines where dots will represent events and line will correlate with those events. A vertex in Graph represents a subject, an edge represents a pair of courses that conflict, and a color represents the period in which that particular course is to be scheduled. The objective is to find minimum color and make a suitable timetable using it. The minimum coloring problem and the timetabling problem have been classified as NP-hard problems in the general case. This means that it is unlikely that it will be possible to find fast (i.e., polynomial-time) algorithms to solve these problems. In order to find optimal solutions to such NP-hard problems, it is usually necessary to consider all possible solutions to choose the best one.[2]

Strength: No conflict occurred because different colors were used for every subject

Weakness: Could not represent certain elements such as connecting two different subjects with different time.

**2.2. Network flow technique:** Dyer and Mulvey (1976), Mulvey (1982), Chahal and de Werra (1989), Dinkel et al (1989) propose to use a network model as the core of the timetabling algorithm. The network employed by Dinkel et al. contains three levels, plus a source and a sink vertex. The first level is the Department Level which includes a vertex for each department, such that all of these vertices are connected to the source. The second level is the Faculty/Staff Level which includes a vertex for each possible combination of teacher and course taught by the teacher; these vertices are connected to the vertices representing the departments to which the teachers belong. The third level is the Room Size/Time Level, which contains a vertex for each combination of room and time. Each vertex of this level is connected to a vertex of the second level only if the size of the room represented by the vertex in compatible with the number of students of the course represented by the other vertex. An edge between levels 2 and 3 represents a possible lecture. The capacities and the lower bounds of edges representing the lectures are 0 and 1 respectively and due to uni-modularity this ensures that the optimal solution to the problem will possess all integer values. The coefficients of the objective function are assigned based on availabilities of teachers and rooms and preferences of the teachers. [3]

Strength: Allocate time for the combination of lecturer-classroom whereby no conflicts generated in the schedule.

Weakness: The network model does not prevent the solution from assigning a single teacher to multiple lectures at the same time.

**3. Concept and problem analysis**

Timetable is the first and foremost tool in the academic and administrative management of the educational systems. Since generation of a time table is a tedious process and there are various permutations and combinations required to get the correct class-wise flow, the timetable software aims to automate this process. The timetable software tracks all subjects and classes and offers various combinations to the user to choose from. The timetable software is a user-friendly automated process that gives flexibility and a combination of options.

Our basic function is to create a time table for a college including different departments and semester. The main problem that occurred during the project is to create and maintain the databases of different entities involved in this process. The database contains the information about the various semesters, subjects, lab, teachers etc. So maintain such a large database is a big challenge for us. The problem we face during our project is how the collision of two subjects or the teachers can avoid. Every project has some drawbacks. Off period were coming in between two period. So, these are some problem which we face in our project.

In a typical semester, the courses are required to be scheduled at different times in order to avoid conflict. The formal definition for university timetable is given in detailed by Carter et al (1995).In event that the following resources and time slot exist as follows:

• A set of lecturers {t1, t2,.............,tn}

• A set of subjects {c1, c2,.......,cm}

• A set of classes {r1, r2,...............,rq}

• A set of time {p1, p2..................ps}

As a preliminary step, construct a three verse set to represent the requirement which are (i=1, 2....n), (j=1, 2..........m) and (k=1, 2....q). Each set indicates lecturer ti lecturing the subject cj in class rk. Further, determine each three verse set to time p1 which is (i =1, 2.....s). This will form a four verse set < ti, cj, rk, p1> which represents the stipulation that lecturer t, lectures the subject c, class r and at p time. For every two four verse set, for example < te, cf, rg, ph> < tw, cx, ry, pz> and the constriction below must be fulfilled. :

If te = tw hence ph ≠ pz (1)

If cf = cx hence ph ≠ pz (2)

If rg = ry hence ph ≠ pz (3)

Constriction (1) ensures that no lecturer will scheduled for more than once in each period, constriction (2) ensures that no subject will be scheduled for the same time and constriction (3) ensures that no class is scheduled for more than once for each period. [2]

The problem is to design an algorithm to create a semester course timetable by assigning time slots and rooms to given set of courses to be run that semester under given constraints. The constraints include avoiding clashes of time slots and rooms, assigning appropriate rooms and appropriate number of slots and contact hours to the courses etc. Although most of the college administrative work has been computerized. The lecture timetable scheduling is still mostly done manually due to its technical difficulties. The manual scheduling of lecture-timetable required considerable time and efforts. The lecture-timetable scheduling is constraints satisfaction problem in which we find an optimal solution that satisfies the given set of constraints.

The primary objective in preparing timetable is obtaining conflict free for each activity sharing the same resources. Most of the method faces difficulties in generating a feasible timetable which faces long processing period without complying with several conditional constraints. The aim of the project is to create timetable in an efficient way which will be conflict free and abiding by all the constraints.

**3.1 Need:**

In any educational institution, creating a timetable or scheduling the classes and other activities is of prime importance but is a very tedious process and to overcome this worry, So we have developed an automated timetable management system. Developed with a user-friendly interface, the timetable management system uses algorithms to generate subject and course combinations. The timetable management system plays a crucial role in creating a structured timetable for classes across subjects. The timetable management software prioritizes different subjects and also gives the option to modify allotment based on requirements. The basic aim of the timetable management system is to optimize resource allocation and reduce manpower and time wasted in creating an error free schedule with zero data clashes. The software for the timetable management can be used to ensure the most effective time schedules are fixed according to the college's needs and suggestions.

**3.2Constraints:**

Constraints are the fundamental elements of a timetable that the software use to create the timetable and must be abided by.

We want to develop an algorithm for developing an effective and practical timetabling algorithm which will satisfy constraints. We primarily focused on developing algorithm which is easy to implement without compromising on its effectiveness and performance. As represented by the constraints designing a timetable is a very complex job. For this reason, creating a reliable automatic server which requires no manual intervention is a very difficult problem and most organizations do not have such a solution. Instead, most timetables are created manually by expert administrator who have deep knowledge of all requirements of all parties involved.

**3.2.1 Hard constraints**: All hard constraints must not followed, no hard constraints can be violated in any case.

1. A classroom is not assigned to more than one lecture at same time.
2. An instructor cannot teach more than one subject at the same time.
3. Two Courses for the same year-session students of a department cannot take place at same time except for elective subject in theoretical class and two different labs for which department are divided into sub groups may be scheduled together.
4. The lectures are not allotted to time slots which come under the lecturer’s prohibited time zones.
5. Practical subjects can start from 1st, 2nd and 5th period only and are held for 3 hours continuously so they end on 3rd, 4th and 7th period respectively.

**3.2.2 Soft Constraints**: It is desirable to satisfy all soft constraints but may be sacrificed to find a feasible timetable.

1. Practical classes of 4th year should get first preference and must be scheduled in the first half i.e. first four periods.
2. All non-laboratory classes should be scheduled in the second half i.e. last 3 periods for any year.
3. A subject should not have more than one class on a particular day.
4. Two consecutive class should not be taken by same teacher.
5. One subject should not assigned more than once in one day.
6. There should not be off period in between.
7. One teacher should have a non-teaching day in a week.
8. Theoretical Classes should not be in second half.

**3.3 Functionality:**

In our project we have three end users.

1) Admin

2) Head of the Department

3) Faculty

* Admin is provided with following features:
* Generate timetable
* View timetable
* Add user
* View user details
* Update user details
* Delete user
* Add lab
* Head of the Department is provided with following features:
* View timetable
* Assign subject to faculty
* Assign lab to faculty
* Update subject Details
* Insert Department
* Insert subject
* Delete Subject
* Faculty is provided with following features:
* View Timetable
* Give subject preference

**3.4 Nonfunctional requirements:**

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.   For example, attributes such as performance, security, usability, compatibility aren't a "feature" of the system, but are a required characteristic.

In our project nonfunctional requirements are:

N1: Dynamicity in management.  
N2: Ease of Maintenance.  
N3: Automate the manual work.   
N4: Platform Independence.  
N5: Availability of updated data.  
N6: Authentication to authorized user.

**3.5 Functional Requirements:**

Functional requirements define the fundamental actions that system must perform. The functional requirements for the system are divided into three main categories, User, Subject/Classes/Semester, and Master Timetable Management.

User Management:-The system records registrations.

Input: -registration details.

Output: -registration successfully.

The system records the subjects' details.

Input: Subject details.

Output: -recorded successfully.

The system records the labs' details.

Input: lab details.

Output: -recorded successfully.

The system updates all details of user.

Input: -enter id and new record.

Output: -Updated successfully.

The system deletes details of user.

Input: -enter id.

Output: -Deleted successfully.

The system displays all details of user.

Input: -see details.

Output: -All details of user

Retrieve the member information.

Input: -enter the member id.

Output: -retrieve the details.

Login Management:-

Login function provides the login to member and managed authentication.

The system shall track login details.

Input: - username and password.

Output: - login successfully.

Timetable Generation module:-

Timetable will show master details it occupies whole detail about faculty/classes/subject/branch etc.  
The system shall track all details.

Input: - Details of subject, faculty, lab.

Output: -timetable.

The system shall track all details about lab.

Input: - lab no.

Output: - show details about lab timetable

Subject Management:-

Subject Management function mainly used to show details about which subject has offered by university and optional subject chosen by branch.

The system displays records of all subjects.

Input: -enter subject code.

Output: - show details about subject.

The system provides HOD to assign teacher to subject.

Input: - enter assign details.

Output: - Recorded successfully.

The system records new subject details.

Input: - enter subject details.

Output: - Recorded successfully.

The system deletes subject details.

Input: - enter subject id.

Output: - deleted successfully.

The system updates subject details.

Input: - enter subject id and new records.

Output: - updated successfully.

The system provides faculties to enter their subject preference which they want to teach.

Input: -enter preferences.

Output: - Recorded successfully.

The system provides HOD to assign teacher to subject.

Input: - enter assign details.

Output: - Recorded successfully.

The system provides HOD to assign teacher to lab.

Input: - enter assign details.

Output: - Recorded successfully

The system shows timetable.

Input: - Generate.

Output: - Timetable.

User

**4.1 Data Flow Diagram**

**4.1.1 Data Flow Diagram Level 0**

A D M I N

View timetable

Generate timetable

View user details

Add user

H

O

D

F

A

C

U

L

T

Y

Assign subject

Post notice

View timetable

View the response

Give Subject Preference

Figure 1: Data Flow Diagram Level 0

Figure 2: Data Flow Diagram Level 1

**4.1.2. Data Flow Diagram Level 1**

HOD

Faculty

Preference

Subject assigned

**Admin**

**Faculty Details**

Timetable

Subject preference

Preference details

Faculty’s preference

Subject assignment details

Generate Request

Subject assignment details

Timetable

Timetable

View request

View request

View request

Login credentials

Login credentials

Login credentials

Login credential

Login credential

Login credentials

Login access Logcredentials

Login access Logcredentials

**4.2 Entity Relationship Diagram**

1

n

Faculty

Class

takes

1

1

n

Lab

preferred

assigned

n

n

Subject

Figure 3: Entity Relationship Diagram

**4.2.1. Attribute**

* Faculty( FacultyId, Name, Password, Post, Teaching\_hours, Admin\_rights)
* Class( Year, Dept, Semester, Period)
* Subject(Code, Sub\_name, Semester, Theory, Practical)
* Leave(FacultyId, Start\_date, Duration)
* Lab( Name, Code)

**5. Sample Code**

**Code of class definition in java:**

public class Subject {

String subjectID;

int sem;

String dept;

int deptcode;

int year;

String subjectName;

int theortical;

int practical;

int tutorial;

}

class Sclass{

int sem;

int year;

String dept;

Subject[][] timetable=new Subject[6][7];

Sclass()

{

for(int i=0;i<6;i++){

for(int j=0;j<7;j++){

timetable[i][j]=new Subject();

}

}

for(int i=4;i<7;i++)

{

timetable[5][i].subjectID="X";//Represents classes cannot be alloted.

}

}

}

class Teacher

{

String name;

String tID;

Subject[][] timetable=new Subject[6][7];

int teaching\_hours;

LinkedList<Subject> subQ =new LinkedList<Subject>();

Teacher()

{

for(int i=0;i<6;i++)

{

for(int j=0;j<7;j++)

{

timetable[i][j]=new Subject();

}

}

for(int i=4;i<7;i++)

timetable[5][i].subjectID="X";

}

public void notavailable(int day,int start\_period,int end\_period)

{

for(int i=start\_period;i<=end\_period;i++)

{

timetable[day][i].subjectID="X";

}

}

private boolean non\_teachingday(int day){

for(int i=0;i<6;i++)

{

for(int j=0;j<7;j++){

if(timetable[i][j].subjectID!=null&&timetable[i][j].subjectID!="X")

break;

if(j==7)

return false;

}

}

return true;

}

public int first\_period(){

int day=0;

for(int i=0;i<6;i++)

{

if(timetable[i][0].subjectID!=null && timetable[i][0].subjectID!="X")

day++;

}

return day;

}

private float day\_ratio(int day)

{

int unassigned=0,assigned=0;

for(int j=0;j<7;j++)

{

if(timetable[day][j].subjectID==null)

unassigned++;

else

assigned++;

}

return ((float)unassigned/(assigned+unassigned));

}

public int max\_day\_ratio()

{

float max=-1,x;

int day=6,free\_day=-1;

for(int i=0;i<6;i++)

{

x=day\_ratio(i);

if(x>max)

{

if(x==1 && free\_day==-1)

{

free\_day=i;

}

else

{

max=x;

day=i;

}

}

}

if(max==0)

return free\_day;

else

return day;

}

}

public class Lab

{

String labName;

Subject[][] timetable=new Subject[6][7];

LinkedList<Subject> labQ =new LinkedList<Subject>();

int labmax=10;

Lab()

{

for(int i=0;i<6;i++)

{

for(int j=0;j<7;j++)

{

timetable[i][j]=new Subject();

}

}

for(int i=4;i<7;i++)

timetable[5][i].subjectID="X";

}

}

**Code for Assignment of Subjects to Teachers**

**assignSubject.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ include file=*"/Header & Background.html"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<form action=*"AssignSubjectServlet"* method=*"get"*>

<table align=*"center"*>

<tr><td>Enter Subject</td>

<td><input type=*"text"* name=*"subject"* ></br></td></tr>

<tr><td><input type=*"submit"* value=*"submit"*></td></tr>

</table>

</form>

</body>

</html>

**Code of Teacher’s Subject Preference**

**GivePreference.jsp**

<%@page import=*"com.Bean.FacultyBean"*%>

<%@page import=*"com.Dao.SubjectDao"*%>

<%@page import=*"com.Dao.SubjectDaoImpl"*%>

<%@page import=*"com.Bean.SubjectBean"*%>

<%@page import=*"java.util.ArrayList"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<%@ include file=*"/Header & Background.html"* %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Give Preference</title>

</head>

<body>

<%

Object ob= session.getAttribute("faculty");

FacultyBean fbean=(FacultyBean)ob;

%>

<form action=*"givePreferenceServlet"* method=*"post"*>

<table>

<tr><td></td><td>:</<td> <input type=*"text"* name=*"fid"* value=<%=fbean.getF\_Id() %> hidden=*"hidden"*>

<tr><td></td><td>:</<td> <input type=*"text"* name=*"department"* value=<%= fbean.getDepartment() %> hidden=*"hidden"*>

<tr><td>1st Prefernce</td><td>:</<td> <select name=*"Subject1"*>

<%

SubjectDao sd1= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb1=sd1.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb1){

%>

<option><%=sb.getsId() %></option>

<%}} %>

</select></td><tr>

<tr> </tr>

<tr><td>2nd Prefernce</td><td>:</<td> <select name=*"Subject2"*>

<%

SubjectDao sd2= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb2=sd2.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb2){

%>

<option><%=sb.getsId()%></option>

<%}} %>

</select></td><tr>

<tr> </tr>

<tr><td>3rd Prefernce</td><td>:</<td> <select name=*"Subject3"*>

<%

SubjectDao sd3= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb3=sd3.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb3){

%>

<option><%=sb.getsId() %></option>

<%}} %>

</select></td><tr>

<tr> </tr>

<tr><td>4th Prefernce</td><td>:</<td> <select name=*"Subject4"*>

<%

SubjectDao sd4= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb4=sd4.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb4){

%>

<option><%=sb.getsId()%></option>

<%}} %>

</select></td><tr>

<tr> </tr>

<tr><td>5th Prefernce</td><td>:</<td> <select name=*"Subject5"*>

<%

SubjectDao sd5= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb5=sd5.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb5){

%>

<option><%=sb.getsId() %></option>

<%}} %>

</select></td><tr>

<tr> </tr>

<tr><td>6th Prefernce</td><td>:</<td> <select name=*"Subject6"*>

<%

SubjectDao sd6= **new** SubjectDaoImpl();

ArrayList<SubjectBean> sb6=sd6.getAllSubjectName();

**if**(sb1.size()>0){

**for**(SubjectBean sb:sb6){

%>

<option><%= sb.getsId() %></option>

<%}} %>

</select></td><tr>

<tr><td></td><td><input style="height:*30px*; font-size: *6*; width : *161px*;" type=*"submit"* value=*"Submit"*></td></tr></table>

</table>

</form>

</body>

</html>

**givePreferenceServlet.java**

package com.Srv;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.Bean.GivePreferenceBean;

import com.Dao.GivePreferenceDAO;

@WebServlet("/givePreferenceServlet")

public class givePreferenceServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public givePreferenceServlet() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

PrintWriter out=response.getWriter();

GivePreferenceBean bean=new GivePreferenceBean();

bean.setFirst(request.getParameter("Subject1"));

bean.setSecond(request.getParameter("Subject2"));

bean.setThird(request.getParameter("Subject3"));

bean.setFourth(request.getParameter("Subject4"));

bean.setFifth(request.getParameter("Subject5"));

bean.setSixth(request.getParameter("Subject6"));

bean.setDepartment(request.getParameter("department"));

String fid=request.getParameter("fid");

int facultyid=Integer.parseInt(fid);

bean.setFid(facultyid);

GivePreferenceDAO calldao=new GivePreferenceDAO();

boolean b=calldao.insertData(bean);

if(b==true)

out.println("Inserted Successfully");

else

out.println("error");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**GivePreferenceDao.java**

package com.Dao;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import com.Bean.GivePreferenceBean;

public class GivePreferenceDAO {

private Connection con=null;

public static Connection getMySQLConnection()throws ClassNotFoundException,SQLException

{

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/FinalYear","root","monalisa");

System.out.println("Connected");

return con;

}

public boolean insertData(GivePreferenceBean ob)

{

boolean f= false;

try

{

con = getMySQLConnection();

PreparedStatement pst= con.prepareStatement("insert into preference values(?,?,?,?,?,?,?,?)");

{

pst.setString(1, ob.getFirst());

pst.setString(2, ob.getSecond());

pst.setString(3, ob.getThird());

pst.setString(4, ob.getFourth());

pst.setString(5, ob.getFifth());

pst.setString(6, ob.getSixth());

pst.setInt(7, ob.getFid());

pst.setString(8, ob.getDepartment());

int i1=pst.executeUpdate();

if(i1>0)

f=true;

}

}catch(Exception e)

{System.out.println(e.toString());}

finally

{

try

{

con.close();

}

catch (SQLException e)

{

e.printStackTrace();

}

}

return f;

}

}

1. **Testing, Results, Discussion on Results**:

An error is a human action that produces an incorrect result. A fault is a manifestation of an error in software (also known as a defect or bug). A fault, if encountered may cause a failure, which is deviation of the software from its expected delivery or service. Reliability is the probability that the software will not cause the failure of a system for a specified time under specified conditions. Errors occur because we are not perfect and, even if we were, we are working under constraints such as delivery deadlines.

Testing identifies faults, whose removal increases the software quality by increasing the software’s potential reliability. Testing is the measurement of software quality. We measure how closely we have achieved quality by testing the relevant factors such as correctness, reliability, usability, maintainability, reusability, testability, etc.

In this page both individual components and integrated components are methodically verified to ensure they are error free and fully meets the requirement. Test cases are used to confirm that whether the system works perfectly under all expected input types.

* 1. **User Management Module:**

The user management module allows new faculty to be added and remove the existing teacher. The following test cases were checked:

1. Add a new faculty

* The first name was tested with numerical values, special characters none of them were accepted. A message was shown only letters are allowed.
* The last name was also tested with numerical values, special characters none of them were accepted. A message was shown only letters are allowed.

1. Remove a faculty

While removing a faculty, faculty id is used. If the faculty id is not present or is missing then a pop error message is shown.

**6.2 Login Module:**

**6.3 Subject Preference Module:**

**6.4 Subject Assignment Module:**

**6.5 Timetable Generation Module:**

The generation module is checked using an auto assign function which automatically assigns different subject to different teacher and different lab. It was checked several times the following were the result of the test case:

* The algorithm never went into an infinite loop, a timetable was always generated for all the randomized input.
* Soft Constraint was satisfied for over 75% of the total assignment for generation of timetable in all the test cases.

1. **Conclusion:**

The primary objective in preparing timetable is obtaining conflict free for each activity sharing the same resources. Most of the method faces difficulties in generating a feasible timetable which faces long processing period without complying with several conditional constraints. The aim of the project is to create timetable in an efficient way which will be conflict free and abiding by all the constraints.

Future Work

* Leave Management module to assign substitute teacher when one teacher is on leave or absent.
* Room Management for tutorial/elective subjects.
* Assigning a teacher to more than one department for a combined class of more than one department.
* Adding more Soft constraint to the timetable.

1. **Reference:**
2. Andrea Schaerf. A survey of automated timetabling.Technical Report CS-R9567, CWI - Centrum voor Wiskunde en Informatica, 1995
3. Student Time Table By using Graph Colouring Algorithm: Baki Koyuncu ,Mahmut Seçir.
4. Dinkel,J,J.,Mote,J., & Venkataraman, M.A.(1989). An efficient decision support system for academic course scheduling.
5. Database System Concepts by Avi Silberschatz,Henry F.Korth & S.Sudarshan.
6. Fundamentals of Database System by Elmasri Navathe.

Website:

* http://www.cs.qub.ac.uk/itc2007/