

# 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 are not given to all faculties 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.

## 1.3 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.

## 1.4 Tools & Platform

### 1.4.1 Software Requirements

#### Language used for implementation

- J2EE(Java Enterprise Edition)
- HTML5

- CSS
- Javascript

➤ For Developer's side:

- OS: Windows 7 or above
- Code: J2EE(Java Enterprise Edition)
- Platform : Eclipse IDE
- Database: MySQL 5.5 or higher
- Server: Wildfly
- Internet Browser: Any standard Internet Browser

➤ For Client's side:

- OS: Any operating system
- Browser: Any standard Internet browser

### **1.4.2 Hardware Requirements**

➤ For Developer's side:

- RAM: 2 GB or higher
- Hard disk: 80 GB or more free space
- Processor: 1.8 GHz or higher
- Intranet Connection

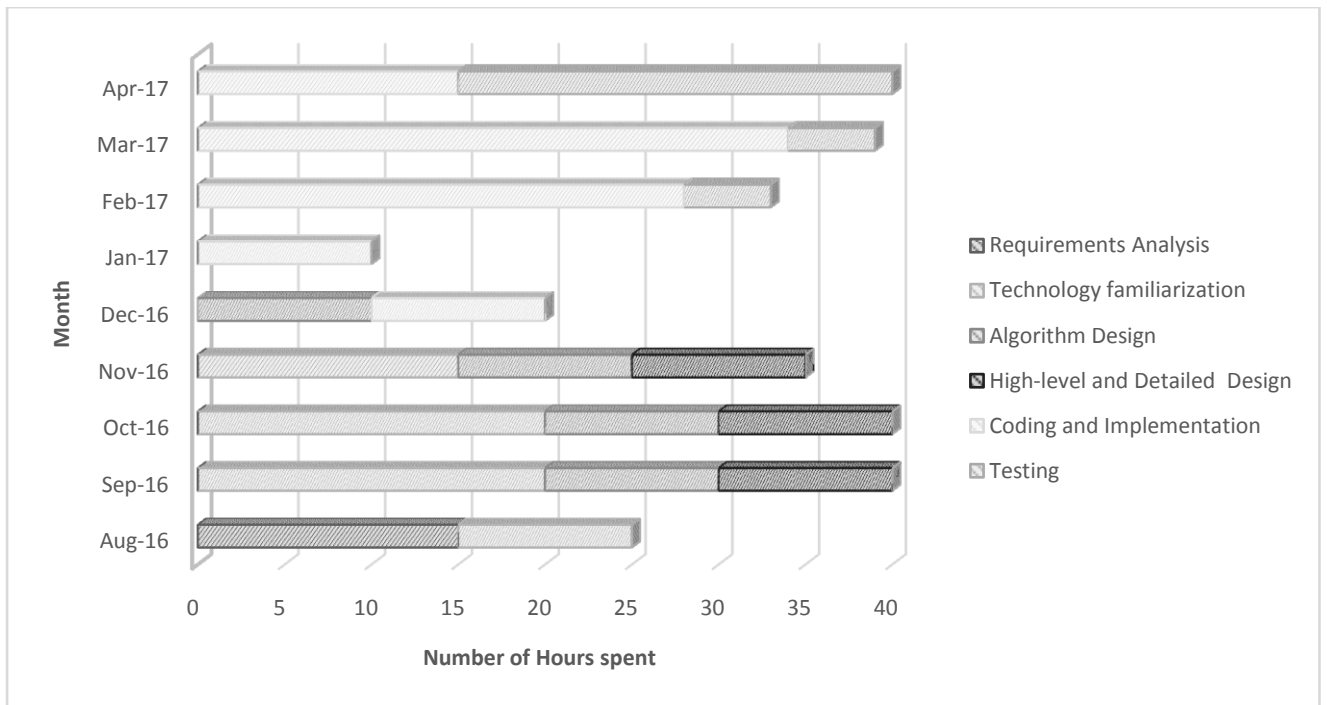
➤ For Client's side:

- RAM: 512MB or higher
- Processor: 1.8 GHz or higher

### **1.5. Project Organization**

The project started in August 2016 and ended in April 2017, the duration was 9 months and the following were phases for building the project.

- Requirement Analysis
- Technology familiarization
- Algorithm Design
- High level detailed design
- Coding and Implementation
- Testing.



Progress Chart

## 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 is 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  $\{t_1, t_2, \dots, t_n\}$
- A set of subjects  $\{c_1, c_2, \dots, c_m\}$
- A set of classes  $\{r_1, r_2, \dots, r_q\}$
- A set of time  $\{p_1, p_2, \dots, p_s\}$

As a preliminary step, construct a three verse set to represent the requirement which are  $(i=1, 2, \dots, n)$ ,  $(j=1, 2, \dots, m)$  and  $(k=1, 2, \dots, q)$ . Each set indicates lecturer  $t_i$  lecturing the subject  $c_j$  in class  $r_k$ . Further, determine each three verse set to time  $p_l$  which is  $(l=1, 2, \dots, s)$ . This will form a four verse set  $\langle t_i, c_j, r_k, p_l \rangle$  which represents the stipulation that lecturer  $t_i$  lectures the subject  $c_j$ , class  $r_k$  and at  $p_l$  time. For every two four verse set, for example  $\langle t_e, c_f, r_g, p_h \rangle$  and  $\langle t_w, c_x, r_y, p_z \rangle$  and the constriction below must be fulfilled. :

If  $t_e = t_w$  hence  $p_h \neq p_z$  (1)

If  $c_f = c_x$  hence  $p_h \neq p_z$  (2)

If  $r_g = r_y$  hence  $p_h \neq p_z$  (3)

Constriction (1) ensures that no lecturer will be 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.

### 3.2 Constraints:

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 has deep knowledge of all requirements of all parties involved.

**3.2.1 Hard constraints:** All hard constraints must be 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 1<sup>st</sup>, 2<sup>nd</sup> and 5<sup>th</sup> period only and are held for 3 hours continuously so they end on 3<sup>rd</sup>, 4<sup>th</sup> and 7<sup>th</sup> 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 4<sup>th</sup> 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 classes should not be taken by same teacher.
5. One subject should not assign 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 behaviors. They are contrasted with functional requirements that define specific behavior 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:

- Dynamicity in management.
- Automate the manual work
- Platform independence
- Availability of updated data.
- Authentication to authorized user.
- The System should be reliable.

### 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 successful.

- 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/Wrong username or password.



- Timetable Generation module:-
  - The system will show timetable year and department wise.

Input: - Details of subject, faculty, lab.

Output: -Timetable.

- The system shall track all details about lab.

Input: - Lab number.

Output: - Show details about lab timetable

- Faculty can see his/her timetable.

Input: - Faculty id.

Output: - Show details about his/her timetable

- Subject Management:-
  - The system displays records of all subjects.

Input: -Enter subject code.

Output: - Show details about subject.

- 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 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.

- Subject Preference Management:-

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

Input: -Enter preferences.

Output: - Recorded successfully.

- Subject Assignment Management:-

- 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

### **3.6. 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. Classrooms 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. Faculties are assigned the subjects they will teach by Head of the Department manually.
6. Users have basic knowledge of computer

## 4.Design and Methodology

### 4.1 Data Flow Diagram

#### 4.1.1 Data Flow Diagram Level 0

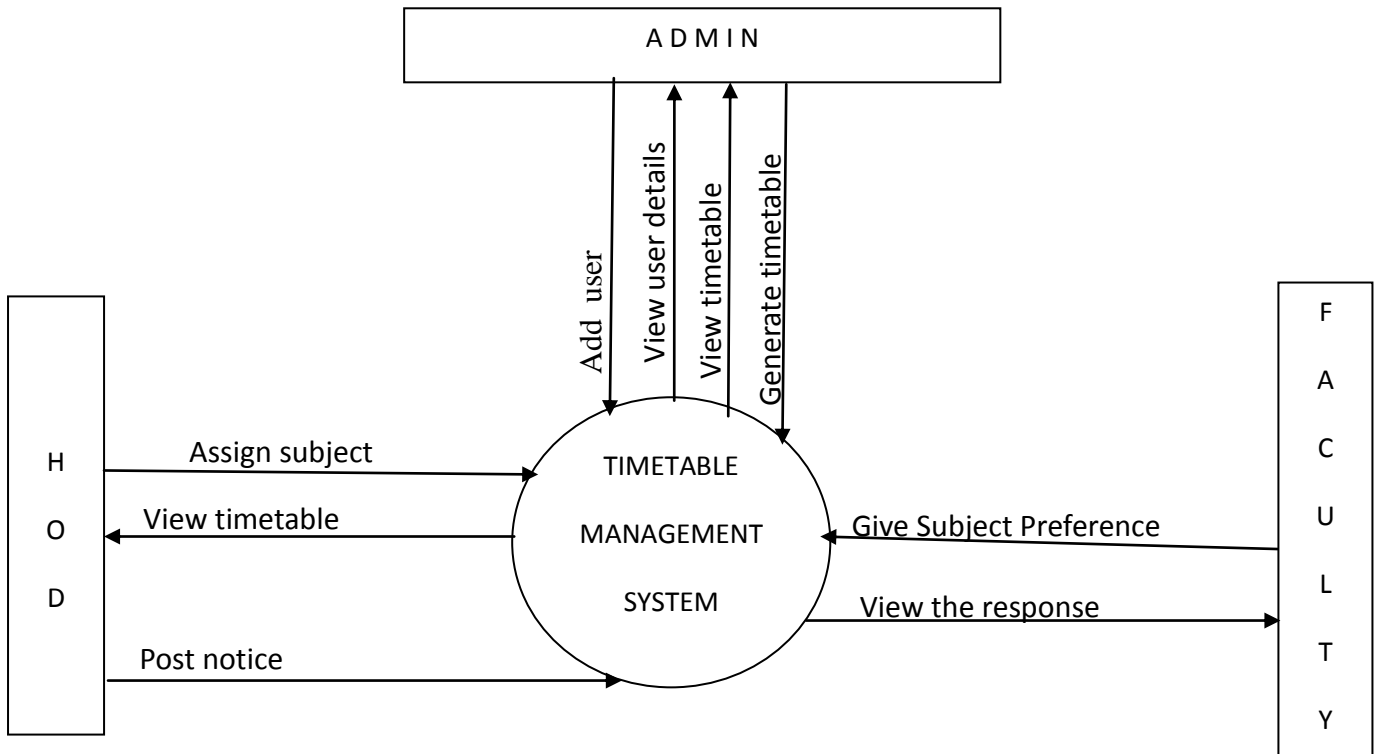


Figure 1: Data Flow Diagram Level 0

#### 4.1.2. Data Flow Diagram Level 1

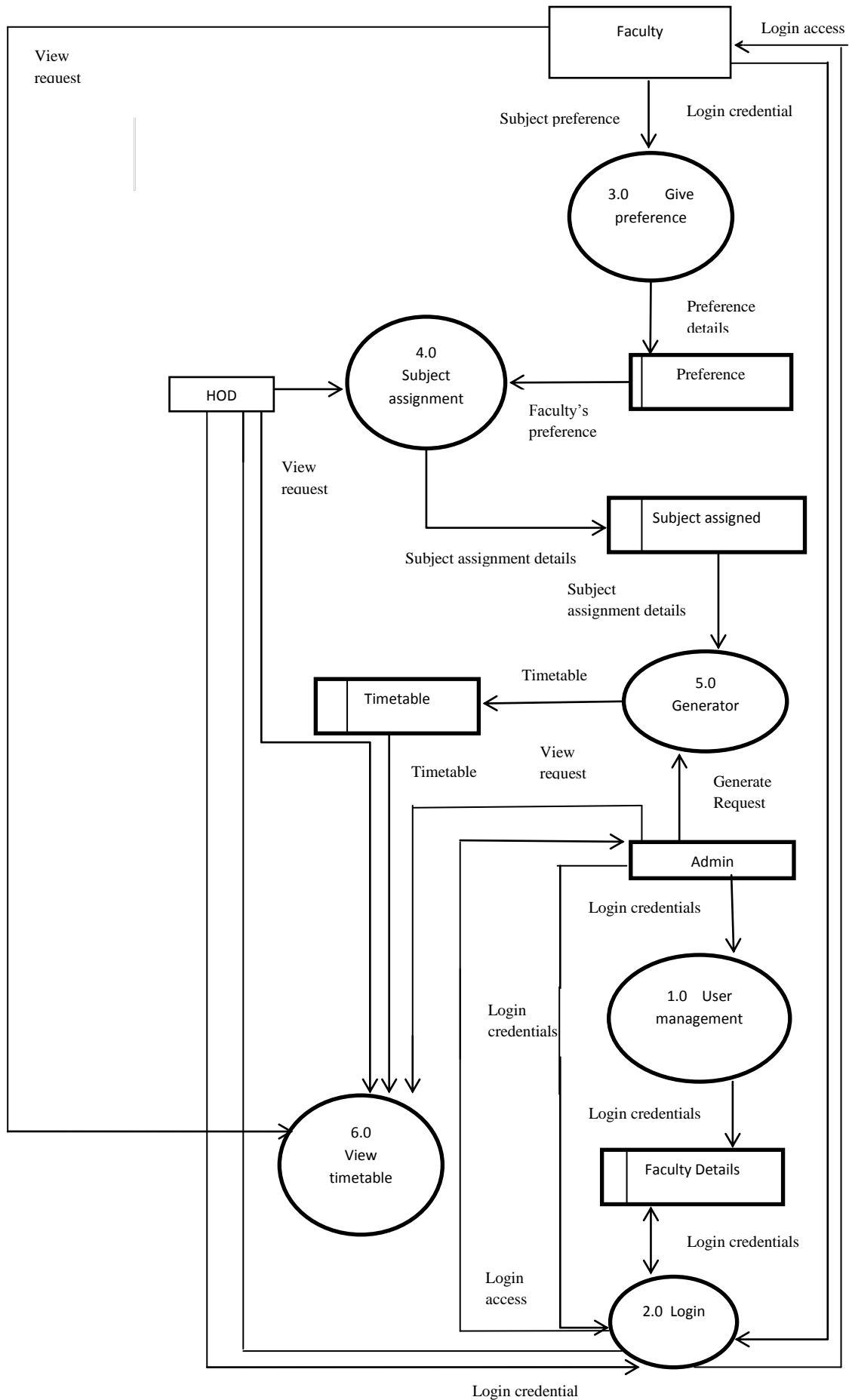


Figure 2: Data Flow Diagram Level 1

## 4.2 Entity Relationship Diagram

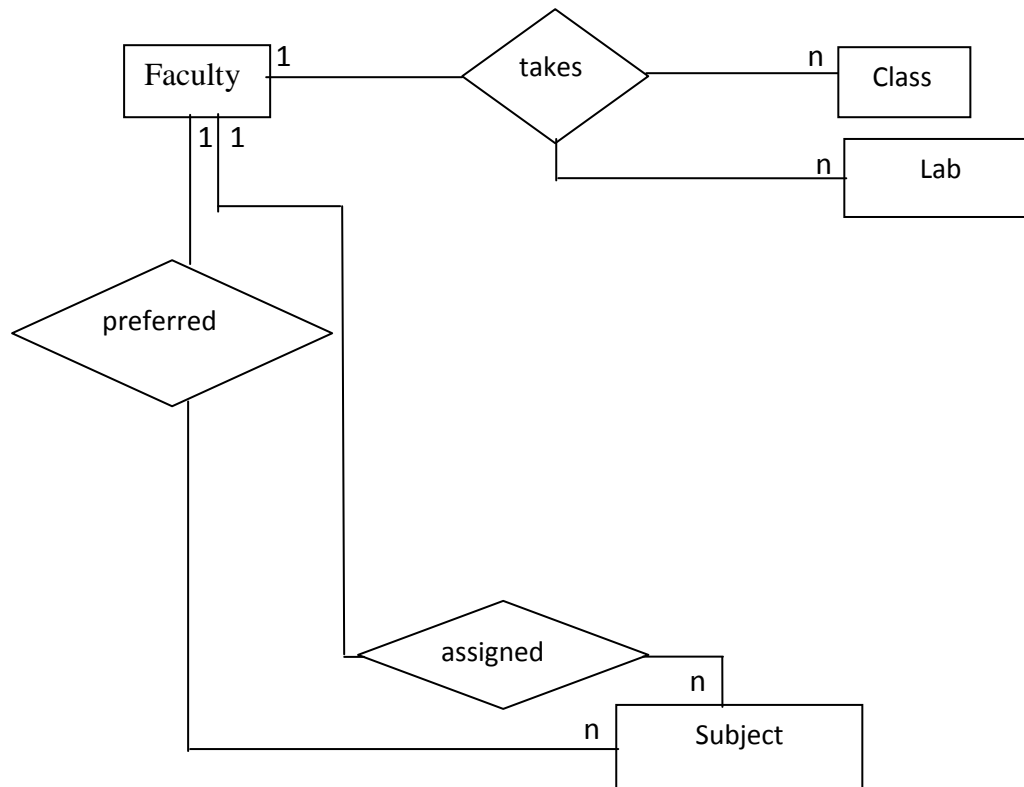


Figure 3: Entity Relationship Diagram

### 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)
- Lab( Name, Code)

## 4.3 Database Design

TABLE: FACULTY

Field Name	Type	Field Properties	Description
<u>FID</u>	Int	Primary key	Faculty ID
<u>F_FIRSTNAME</u>	Varchar	Not null	password
F_LASTNAME	Varchar	Not null	Faculty Name
F_PASSWORD	Varchar	Not null	password
ADMINRIGHTS	Varchar	'Y' or 'N'	Admin rights
POST	Varchar	Foreign key (TEACHER_POST)	Post of teacher
DEPARTMENT	Varchar	Not null	Department of teacher

**TABLE: FACULTY\_POST**

Field Name	Type	Field Properties	Description
POST	Varchar	Primary key	Teacher's Post
TEACHING_HOURS	Int	Not null	Teaching Hours

**TABLE: SUBJECT**

Field Name	Type	Field Properties	Description
<u>S_ID</u>	Varchar	Primary key	Subject ID
<u>DEPARTMENT</u>	Varchar	Primary key	Department
SUBJECT_NAME	Varchar		Subject Name
SEMESTER	Int		Semester number
THEORTICAL	Int		Number of theory class
PRACTICAL	Int		Number of practical class
TUTORIAL	Int		Number of tutorial class

**TABLE: PREFERENCE**

Field Name	Type	Field Properties	Type
FIRST	Varchar		Name of the 1 <sup>st</sup> subject
SECOND	Varchar		Name of the 2 <sup>nd</sup> subject
THIRD	Varchar		Name of the 3 <sup>rd</sup> subject
FOURTH	Varchar		Name of the 4 <sup>th</sup> subject
FIFTH	Varchar		Name of the 5 <sup>th</sup> subject
SIXTH	Varchar		Name of the 6 <sup>st</sup> subject
F_ID	Int	Foreign key(faculty)	Id of faculty
DEPARTMENT	Varchar		Name of the 1 <sup>st</sup> subject

**TABLE: ASSIGN**

Field Name	Type	Field Properties	Description
<u>S_ID</u>	Varchar	Foreign key(SUBJECT)	Subject ID
<u>F_ID1</u>	Int	Foreign key(FACULTY)	Faculty ID
THEORITICAL1	Int		
F_ID2	Int		
THEORITICAL2	Int		
TUTORIAL 1	Int		
TUTORIAL2	Int		
DEPARTMENT	Varchar		Department

**TABLE: LAB**

Field Name	Type	Field Properties	Description
<u>LAB_ID</u>	Varchar	Foreign key (SUBJECT)	Subject ID
<u>DESCRIPTION</u>	Varchar		Teacher ID
DEPT	Varchar		Department

**TABLE: ASSIGNLAB**

Field Name	Type	Field Properties	Description
<u>LABCODE</u>	Varchar	Foreign key(Lab)	Subject code
FID1	Int		Faculty Id
FID2	Int		Faculty Id
FID3	Int		Faculty Id
FID4	Int		Faculty Id
LABNAME	Varchar		Name of the lab
DEPARTMENT	Varchar		Department

**TABLE: S\_TIMETABLE**

Field Name	Type	Field Properties	Description
DAY	Varchar	Primary Key	Day of the week
PERIOD	Int	Primary Key	Period of the day
SEMESTER	Int	Primary Key	Year of class
DEPARTMENT	Varchar	Primary Key	Department of class
S_ID	Varchar	Foreign Key	Subject

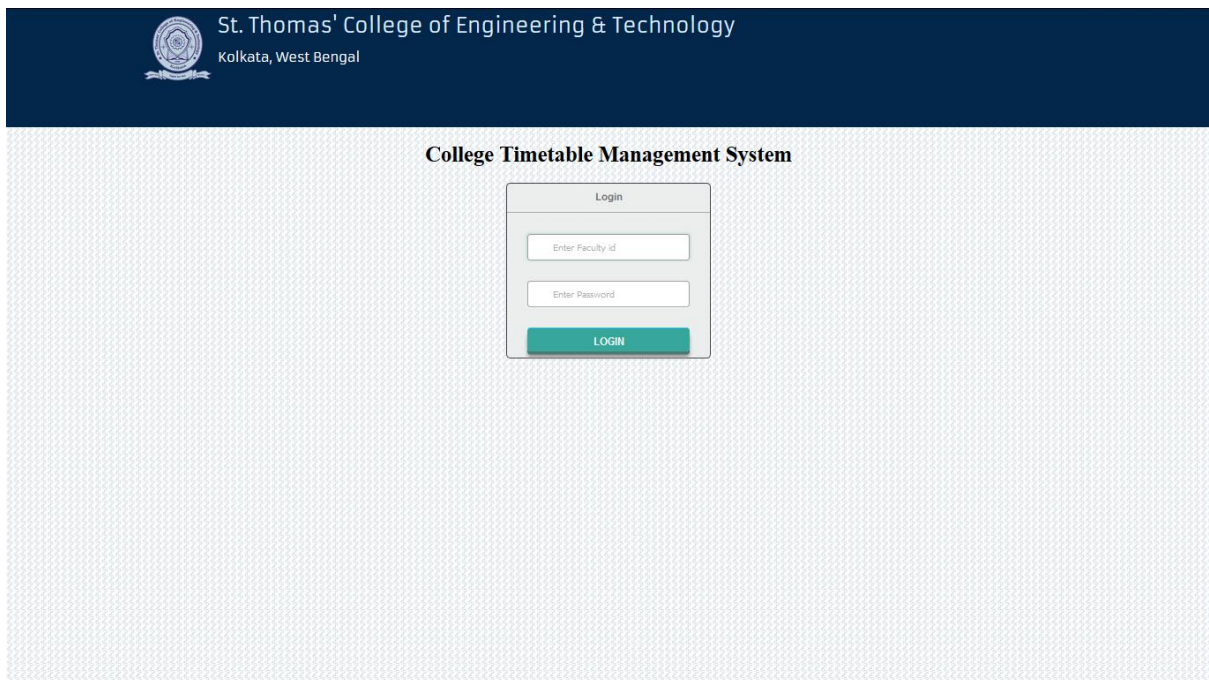
**TABLE: TEACHERTIMETABLE**

Field Name	Type	Field Properties	Description
DAY	Varchar	Primary Key	Day of the week
PERIOD	Int	Primary Key	Period of the day
DEPARTMENT	Varchar	Primary Key	Department of class
S_ID	Varchar	Foreign Key	Subject
F_ID	Int	Foreign Key	Teacher

**TABLE: DEPT**

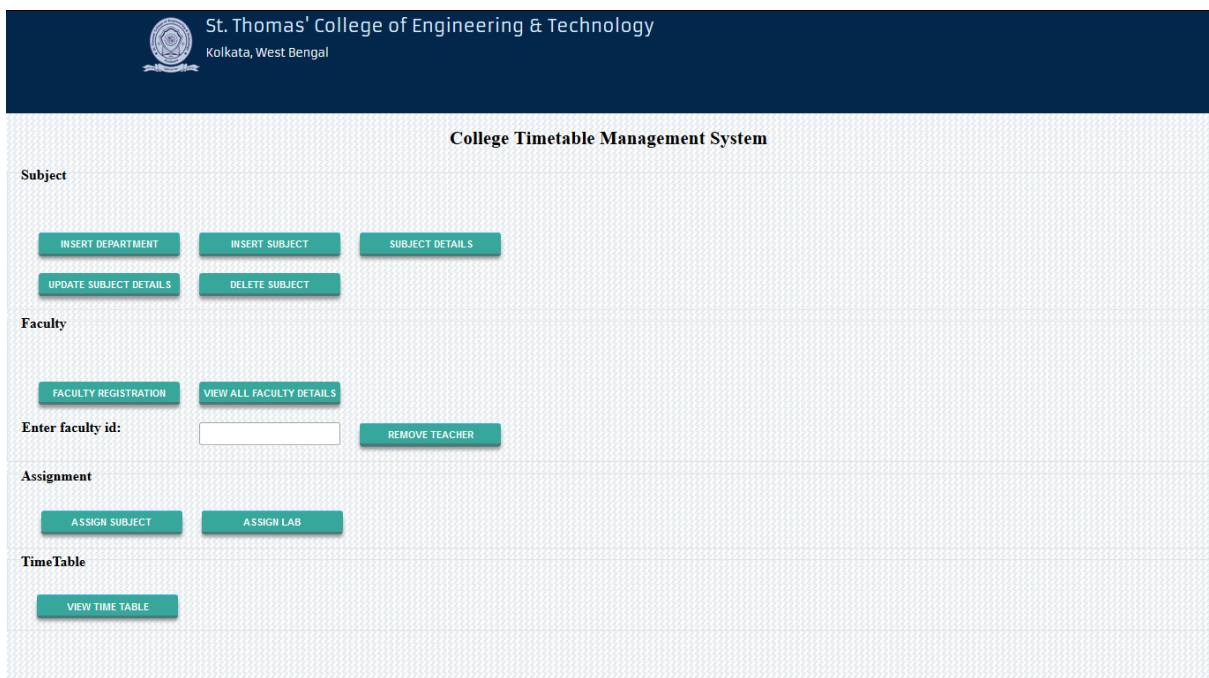
Field Name	Type	Field Properties	Description
<u>DEPARTMENT_ID</u>	Int	Primary Key	ID of Department
Department	Varchar	Unique	Department Name

## 4.4 FRONT END



The screenshot shows the login interface of the College Timetable Management System. At the top, there is a dark blue header with the college's logo and name: "St. Thomas' College of Engineering & Technology, Kolkata, West Bengal". Below the header, the title "College Timetable Management System" is centered. The main content area features a light gray box with a "Login" title. Inside this box, there are two input fields: "Enter Faculty id" and "Enter Password". Below these fields is a green button labeled "LOGIN".

Login Page for College Timetable Management System



The screenshot shows the home page for the admin of the College Timetable Management System. The header is identical to the login page. Below the header, the title "College Timetable Management System" is centered. The main content area is divided into several sections, each with a title and a set of buttons:

- Subject**: Includes buttons for "INSERT DEPARTMENT", "INSERT SUBJECT", "SUBJECT DETAILS", "UPDATE SUBJECT DETAILS", and "DELETE SUBJECT".
- Faculty**: Includes buttons for "FACULTY REGISTRATION" and "VIEW ALL FACULTY DETAILS". Below these buttons, there is a label "Enter faculty id:" followed by an input field and a "REMOVE TEACHER" button.
- Assignment**: Includes buttons for "ASSIGN SUBJECT" and "ASSIGN LAB".
- TimeTable**: Includes a button for "VIEW TIME TABLE".

Home page for admin





### College Timetable Management System

#### Give Subject Details

SubjectCode :

Subject Name :

Department :

Semester :

Theoretical Classes :

Practical Classes :

Tutorial Classes :

SUBMIT

Page for adding new Subject



### College Timetable Management System

#### Subject Details

Code	Name	Department	Semester	Theoretical	Practical	Tutorial
CH301	Basic Environmental Engineering & Elementary Biology	CSE	3	3	0	0
CS201	Basic Computation & Principles of Computer Programming	CSE	2	3	0	1
CS301	Analog	CSE	3	3	0	0
CS302	Data Structure & Algorithm	CSE	3	3	0	1
CS303	Computer Organisation	CSE	3	3	0	1
CS391	Analog & Digital Electronics	CSE	3	0	3	0
CS392	Data Structure & Algorithm	CSE	3	0	3	0
CS393	Computer Organisation	CSE	3	0	3	0
CS501	Design & Analysis of Algorithm	CSE	5	3	0	1
CS502	Microprocessor & MicroController	CSE	5	3	0	1
CS503	Discrete Mathematics	CSE	5	3	0	0
CS504D	Object Oriented Programming(IT)	CSE	5	3	0	1
CS591	Design & Analysis of Algorithm	CSE	5	0	3	0
CS592	Microprocessor & MicroController	CSE	5	0	3	0
CS593	Programming Practices using C++	CSE	5	0	3	0
CS594A	Circuit Theory & Network(ECE)	CSE	5	0	3	0
CS701	Software Engineering	CSE	7	3	0	0
CS702	Compiler Design	CSE	7	3	0	0
CS703C	Artificial Intelligence	CSE	7	3	0	0
CS704A	Distributed Operating System	CSE	7	3	0	0
CS705A	Internet Technology(IT)	CSE	7	3	0	0
CS781	Group Discussion	CSE	7	0	3	0
CS791	Software Engineering Lab	CSE	7	0	3	0
CS793C	Artificial Intelligence	CSE	7	0	3	0
CS794	Project	CSE	7	0	3	0
CS795A	Internet Technology(IT)	CSE	7	0	3	0
ES101	Engg. Mechanics	CSE	1	3	0	1
ES101	Basic Electrical & Electronic Engineering 1	CSE	1	0	3	0

Subject Details for all the subjects in the database



### College Timetable Management System

## Registration Form

Enter First Name :

Enter Last Name :

Enter Password :

Enter Post :

Enter Department :

Enter Admin\_Rights : ☐ Yes ☐ No

SUBMIT

## Add new teacher to the Database



### College Timetable Management System

#### All Faculty Details

ID	Name	Department	Post	Admin
1	Subarna Bhattacharjee	CSE	HOD	Yes
2	Aruna Chakraborty	CSE	Associate Professor	Yes
3	Samrat Sarkar	CSE	Assistant Professor	No
4	Subhankar Mallick	CSE	Assistant Professor	No
5	Amiya Halder	CSE	Assistant Professor	No
6	Debashis Chakraborty	CSE	Assistant Professor	No
7	Biswajita Datta	CSE	Assistant Professor	No
8	Ranjita Chowdhury	CSE	Assistant Professor	No
9	Amit Paul	CSE	Assistant Professor	No
10	Barnali GuptaBanik	CSE	Assistant Professor	No
11	Mousumi	CSE	Assistant	No

## Database View of all the teacher details



### College Timetable Management System

ID	FirstName	LastName	Department	post
2	Aruna	Chakraborty	CSE	Associate Professor
4	Subhankar	Mallick	CSE	Assistant Professor

Subject Code	<input type="text" value="HU301"/>
Department	<input type="text" value="CSE"/>
Teacher ID(Theoretical1)	<input type="text" value="2"/>
Teaching Hour(Theoretical1)	<input type="text"/>
Teacher ID(Theoretical2)	<input type="text" value="Not assigned"/>
Teaching Hour(Theoretical2)	<input type="text" value="0"/>
<input type="button" value="SUBMIT"/>	

Page for assignment of Subject

## 5.Sample Code

### Code of class definition in java:

```
public class Subject {  
    String subjectID;  
    intsem;  
    String dept;  
    intdeptcode;  
    int year;  
    String subjectName;  
    inttheoretical;  
    int practical;  
    int tutorial;  
}  
  
classSclass{  
    intsem;  
    int year;  
    String dept;  
    Subject[][] timetable=new Subject[6][7];  
    Sclass()  
    {  
        for(inti=0;i<6;i++){  
            for(int j=0;j<7;j++){  
                timetable[i][j]=new Subject();  
            }  
        }  
        for(inti=4;i<7;i++){  
            {  
                timetable[5][i].subjectID="X";//Represents classes cannot be allotted.  
            }  
        }  
    }  
}
```

```

class Teacher
{
    String name;
    String tID;
    Subject[][] timetable=new Subject[6][7];
    intteaching_hours;
    LinkedList<Subject>subQ =new LinkedList<Subject>();
    Teacher()
    {
        for(inti=0;i<6;i++)
        {
            for(int j=0;j<7;j++)
            {
                timetable[i][j]=new Subject();
            }
        }
        for(inti=4;i<7;i++)
            timetable[5][i].subjectID="X";
    }
    public void notavailable(intday,intstart_period,intend_period)
    {
        for(inti=start_period;i<=end_period;i++)
        {
            timetable[day][i].subjectID="X";
        }
    }
    privatebooleannon_teachingday(int day){
        for(inti=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 intfirst_period(){
    int day=0;
    for(inti=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));
}

publicintmax_day_ratio()
{
    float max=-1,x;
    int day=6,free_day=-1;
    for(inti=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"></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");
FacultyBeanfbean=(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= newSubjectDaoImpl();
        ArrayList<SubjectBean> sb1=sd1.getAllSubjectName();

        if(sb1.size()>0){

            for(SubjectBean sb:sb1){
                %>
                <option><%=sb.getId() %></option>

            <% } } %>

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

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

        if(sb1.size()>0){

            for(SubjectBean sb:sb2){
                %>
                <option><%=sb.getId()%></option>

            <% } } %>

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

<%
        SubjectDao sd3= newSubjectDaoImpl();

```

```

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

        if(sb1.size()>0){

            for(SubjectBean sb:sb3){
                %>
                <option><%=sb.getId() %></option>

            <% } } %>

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

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

        if(sb1.size()>0){

            for(SubjectBean sb:sb4){
                %>
                <option><%=sb.getId()%></option>

            <% } } %>

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

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

        if(sb1.size()>0){

            for(SubjectBean sb:sb5){
                %>
                <option><%=sb.getId() %></option>

            <% } } %>

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

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

        if(sb1.size()>0){

```

```

                for(SubjectBean sb:sb6){
                    %>
                    <option><%= sb.getId() %></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");

        intfacultyid=Integer.parseInt(fid);

        bean.setFid(facultyid);

        GivePreferenceDAOcalldao=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;
}
}

```

## **6. Testing, Results, Discussion on Results:**

### **6.1 Testing**

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 an integrated component are methodically verified to ensure they are error free and fully meet the requirement. Test cases are used to confirm that whether the system works perfectly under all expected input types.

#### **6.1.1 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.
2. 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.

#### **6.1.2 Login Module:**

A pop up message is shown for incorrect login credentials. Login module doesn't display whether the login id is wrong or the password is wrong. Login doesn't happen for the following combination:

- Correct faculty id and incorrect password.
- Incorrect faculty id and correct password.
- Incorrect faculty id and incorrect password.

#### **6.1.3Subject Preference Module:**

The Preference module allows Faculty to give preference of the subjects. Faculty can give choices of only those subjects present in database.

#### **6.1.4Subject Assignment Module:**

The assignment module allows Head of the Department only to assign the subjects and teaching hours to faculty.


Assign subjects to teacher: While assigning teacher to subjects the teacher should have enough teaching hour left to teach the subject and the summation of the number of teaching hours assigned to the subject should be equal to the total teaching hours.

### 6.1.5 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.

## 6.2 Result

 <b>St. Thomas' College of Engineering &amp; Technology</b> Kolkata, West Bengal							
<b>College Timetable Management System</b>							
Timetable							
Department: CS							
Semester: I							
Period	1	2	3	4	5	6	7
Monday	CS101	CS107	CS107	CS107	CS106	CS106	CS106
Tuesday	CS105	CS101	CS103	CS102	CS108	CS108	CS108
Wednesday	CS103	CS104	CS103	CS102	CS102	CS105	CS104
Thursday	CS104	CS103	CS101	CS101	---	---	---
Friday	CS105	CS104	CS102	CS105	---	---	---
Saturday	---	---	---	---	---	---	---

Page displaying timetable for students





### College Timetable Management System

Faculty Id:1

Faculty Name:Subarna Bhattacharjee

#### Timetable

Period	1	2	3	4	5	6	7
Monday	---	---	---	---	---	---	---
Tuesday	---	---	---	---	CS506	CS506	CS506
Wednesday	---	CS105	---	CS701	---	---	---
Thursday	CS701	---	---	IT704	---	---	---
Friday	---	---	---	---	CS507	CS507	CS507
Saturday	---	---	---	---	---	---	---

Page for displaying timetable for teachers

## 6.3 Discussion on Result

The timetable obtained had the following features:

- There was no off period between two classes for students.
- Most faculty got a non-teaching day for research works.
- No faculty had class on first period more than 3 days.
- No hard constraint was violated while few of the soft constraints were violated for few cases.

## 7. 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.

## 8. Reference:

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

### Website:

- <http://www.cs.qub.ac.uk/itc2007/>
- <https://www.tutorialspoint.com/jsp/>
- <https://www.tutorialspoint.com/mysql/>
- <https://www.tutorialspoint.com/servlets/>