

 <p>Dr.T.J.Sawant D.E.B.E.(ELEC),MISTE Founder Secretary</p>	JAYWANTRAO SAWANT PRASARAK MANDAL	 <p>Dr.S.M.Deokar Ph.D(E&TC) Principal</p>
	<p align="center">JAYWANTRAOSAWANT POLYTECNIC</p>	
	<p align="center">(Approved by AICTE,New Delhi,Govt. of Maharashtra and affiliated to MSBTE,Mumbai.)</p> <p align="center">S.No.58, Handewadi Road,Hadapsar,Pune-28</p> <p align="center">Phone:+91-020-20970229 Telfax:+91-020-26970229</p> <p align="center">E-Mail: principal6425@gmail.com</p> <p align="center">Website:www.jspm.edu.in</p>	

Institute Vision Statement

To develop valuable human resources through technical education to satisfy the basic needs of industry and for the betterment of society

Institute Mission Statements

1. To provide and maintain an environment of academic excellence in field of technical education by creating awareness about recent trends in technology.
2. To inculcate the moral values in the students.
3. To provide the infrastructure with modern facilities

 <p>Dr.T.J.Sawant D.E.B.E.(ELEC),MISTE Founder Secretary</p>	JAYWANTRAO SAWANT PRASARAK MANDAL	 <p>Dr.S.M.Deokar Ph.D(E&TC) Principal</p>
	<p>JAYWANTRAOSAWANT POLYTECNIC</p>	
	<p>(Approved by AICTE,New Delhi,Govt. of Maharashtra and affiliated to MSBTE,Mumbai.)</p> <p>S.No.58, Handewadi Road,Hadapsar,Pune-28</p> <p>Phone:+91-020-20970229 Telfax:+91-020-26970229</p> <p>E-Mail: principal6425@gmail.com</p> <p>Website:www.jspm.edu.in</p>	

Computer Engineering Department

Vision

To build creative programming mind in the students for adopting emerging technology in the field of computing, for betterment of society

Mission

M1: To impart sound theoretical and practical knowledge in the discipline of computing through effective teaching learning process.

M2: To provide industry and society oriented skilled engineers through co-curricular and extracurricular activities.

M3: To inculcate

PO-PSO Mapping

Title: Automatic Timetable Generator Windows Application Using C#.

Sr No.	PO-PSO	Mapping	Justification
1	Basic and Discipline specific knowledge: Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the engineering problems.	L	Used in School and colleges to reduce manual work
2	Problem Analysis: Identify and analyze well defined engineering problems using codified standard method.	M	We implemented to reduce manual work of generating timetable
3	Design/development of solution: Design solutions for well-defined technical problems and assist with the design of system components or processes to meet specified needs.	M	We designed multiple table in in database to store data in well manner
4	Engineering Tools, Experiment and Testing: Apply modern engineering tools and appropriate technique to conduct standard test and measurement.	M	We used tools like Visual Studio, SQL Server 2014 Management Studio, Crystal Report.
5	Engineering practice for society, sustainability and environment: Apply appropriate technology in context of society, sustainability, environment and ethical practice.	L	Can help in generating in schools and colleges and gives well scheduled timetable.
6	Project Management: Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.	L	Managed work equally and with good co-ordination
7	Life-long learning: Ability to analyze individual needs and engage in updating in the context of technological changes	H	Project was executed and implemented successfully.

TABLE OF CONTENTS

SR NO	CHAPTERS	PAGE NO
1	ABSTRACT	1
2	INTRODUCTION	2
3	LITERATURE SURVEY	3
4	REQUIREMENT ANALYSIS	6
5	WORKFLOW DIAGRAM	7
6	PROJECT DESIGN	8
7	IMPLEMENTATION AND CODING	10
8	TESTING	86
9	SCREENSHOTS	87
10	CONCLUSION	96
11	REFERENCES	97

ABSTRACT

Time table generation is tedious job for educationalist with respect to time and man power. Providing a automatic time table generator will help to generate time table automatically. Proposed system of our project will help to generate it automatically also helps to save time. It avoids the complexity of setting and managing Timetable manually. In our project we are going to use algorithms like genetic resource scheduling to reduce these difficulties of generating timetable. These algorithms incorporate a numeral of strategy, aimed to improve the operativeness of the search operation. The system will take various inputs like number of subjects, teachers, and workload of a teacher, semester, rooms, and labs. By relying on these inputs, it will generate possible time tables for working days of the week for teaching faculty. This will integrate by making optimal use of all resources in a way that will best suit the constraints. The methods discussed in this paper are capable of handling both soft and hard constraints. Once the final timetable has been generated, teachers and students can access it through three different views i.e., class view, lab view and combined view of classes and labs.

INTRODUCTION

Although majority college organization work has been mechanized, the lecture timetable preparation is still commonly done by hand due to its inherent difficulties. The physical lecture-timetable preparation demands significant time and efforts. The manual lecture-timetable scheduling is a limitation fulfillment problem in which we find a result that satisfies the given set of constraints. There have been numerals of approaches made in the earlier period to the difficulty of constructing timetables for colleges and schools. Timetabling problems may be solve by diverse methods inherited from operation study such as graph coloring, local search measures such as tabu search, simulated annealing, genetic algorithms or from backtracking based constraint fulfillment handling. In our project, timetable problem is formulated as a constraint fulfillment problem and we proposed a realistic timetable algorithm which is capable of taking care of both hard and soft constraints. Various approaches have been made in the past decade to solve the problem of constructing timetables for schools and colleges. In our paper this problem is formulated as a constraint satisfaction problem and we discuss the various approaches that are capable of handling both hard and soft constraints. Hard constraints cannot be violated under any circumstances. For example, two classes cannot be allocated to a single teacher at the same time period, two classes cannot be attended by a student at the same time, more than one class cannot be held at a room at the same time et cetera. Soft constraints are necessary but not absolutely critical. For example, a timetable must be made in such a way that a group of students don't have to come to college to attend only one class. It is a complete time table solution for Colleges which help to overcome the challenges in manually constructing the timetable.

LITERATURE SURVEY

A. Timetabling

It is large and highly constrained, but above all the problem differs greatly for diverse colleges and learning institutions. It is hard to write a universal agenda, fitting for all possible timetable problems. Even though manual creation of timetable is sustained, it is still universal, because of the lack of suitable computer programs (Gore & Poonam Sonawane, March 2017).

Timetable problems

There exist a lot of diverse timetable problems such as:

- University Timetable
- Exam Timetable
- School Timetable
- Sports Timetable
- Worker Timetable

Moreover, there exist a lot of problem solving methods, which typically use the concept of customary optimization algorithms such as genetic algorithms, Backtracking, Constraint Logic Programming.

In recent years two major approaches appear to have been victorious.

- Local Search Procedures
- Constraint Programming (CP) (Gore & Poonam Sonawane, March 2017).

B. The Local Search Procedures

The local search measures such as Simulated Annealing, Tabu Search and Genetic Algorithms. These methods convey constraints as various cost functions, which are minimized by a Heuristic Search of enhanced solution in a neighborhood of some opening realistic result (Gore & Poonam Sonawane, March 2017).

1. Simulated annealing (SA)

Simulated annealing is a probabilistic method used for similar to the global optimum of a given function. Purposely, it is a metaheuristic to fairly accurate global optimization in a huge search space. It is frequently used when the search space is distinct. Simulated annealing is a technique for finding a good result to an optimization dilemma. If there is a condition where we want to maximize or reduce something, our problem can likely be tackle with simulated annealing (Gore & Poonam Sonawane, March 2017).

2. Tabu Search

Tabu Search is a Global Optimization algorithm and a Metaheuristic or Meta-strategy for calculating a surrounded heuristic method. Tabu Search is a parent for huge relations of derivative approach that establish memory structure in Metaheuristic, such as Tabu Search and Parallel Tabu Search (S, A Kavya Reddy, & K Panimozhi, April 2015).

3. Genetic algorithm (GA)

Genetic Algorithms (GA) was imaginary by John Holland and has described this thought in his book “Adaptation in natural and artificial systems” in the year 1975. Genetic algorithm is a met heuristic motivated by the procedure of natural selection that belong to the bigger class of evolutionary algorithms (EA). Genetic Algorithms are motivated by Darwin’s evolutionary theory. GA comes below the class of Evolutionary algorithms that use the principle of natural collection to develop a set of solution towards the best result. It is a search heuristic which generates solutions to optimization problems using ntechnique motivated by natural evolution like mutation, inheritance, crossover and selection (S, A Kavya Reddy, & K Panimozhi, April 2015).

C. The Constraint Programming (CP)

Major advantage of constraint programming is declaratively a clear-cut statement of the constraints serves as part of the program. This makes the program easy to adjust, which is critical in timetable constraints are handled through a system of const propagation, which minimizes domains of variables, coupled with backtracking search. In modern CP languages, both features do not need to be planned explicitly.

The main disadvantages of this approach are:

- I. The difficulty with expressing soft constraints
- II. The potential problems with enhancing the initial feasible solution.

The capability to convey composite constraints in a simple, declarative way is critical for establishing of the colleges and university timetable problem into the program and is critical for their successful tailored delivery approach is able to introduce soft constraints during a search, leading quickly to a ”Good” timetable, integration of local search into CP gives the capability to optimize efficiently the timetable Poonam Sonawane, March 2017).

E. Constraints

There are a variety of constraints to be satisfied the time to instantiate variables about time slots and classrooms. The constraints can be categorized into Hard and Soft constraints.

Hard Constraints

A timetable which breaks a hard constraint is not a feasible solution. Hard constraints comprise “First Order Conflicts”,

HC1. A classroom is not assigned to more than one teacher at the same time.

HC2. A teacher cannot teach more than one class at the same time.

HC3. Courses for the similar year-session students of a department cannot take place at the same time.

HC4. The classroom for a course should have enough capacity to take students registered in the course.

HC5 .The classroom should be well set of equipment with required services for the classes.

Soft Constraints

Soft constraints are less significant than constraints, and it is typically not possible to avoid breaking at least some of them. Either timetable technique is functional, which calculates the level to which a timetable has violated its soft constraints. Some soft constraints are more vital than others, and this is often specified with a priority value.

SC1. The teachers are not assigned to time slots, which are in the teacher's prohibited time zones.

SC2. Teachers daily teach hours should be limited to be within the acceptable utmost hours.

SC3. As far as possible, classes are scheduled in the lecturer's preferred time zones.

SC4. A lunch break must be scheduled.

SC5. The practical courses are scheduled in morning session, and the theory courses are scheduled in afternoon session.

SC6. The lecture hours for a course should be scheduled consecutively.

SC7. As distant as possible, classes should be listed in their corresponding department's exclusive-use classrooms.

SC8. The classrooms should be allocated in a manner to reduce the distances between adjacent classes' classrooms.

It is desirable for timetables to satisfy all hard and soft constraints. It is typically hard to meet all these constraints.

Any hard constraint must not be despoiled in any case, but some soft constraints can be sacrificed to find reasonable timetables. The timetable practice is made harder by the fact that so many people are affected by its outcome.

REQUIREMENT ANALYSIS

Hardware Requirements: -

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- RAM: 2 GB.

Software Requirements: -

- Operating system: Windows XP/7
- Coding Language: C#.
- IDE: Visual Basic.
- Front End: C#.
- Back End: SQL Server 2014 Management Studio

WORK FLOW DIAGRAM

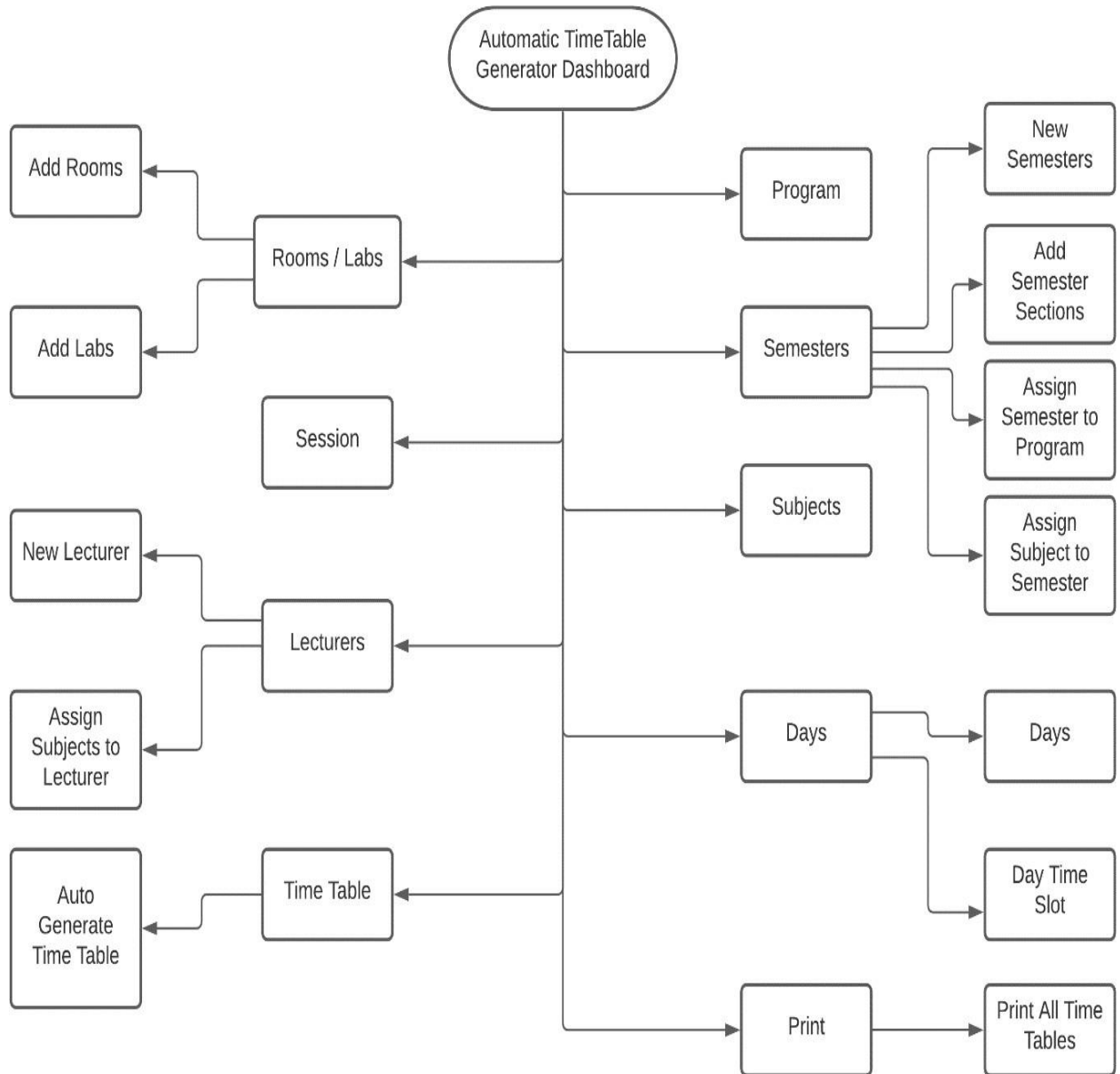


Fig. 1) Automatic Timetable Generator Flowchart

PROJECT DESIGN

Design of a scheme is basically a blueprint or plan for a system. It determines the part structure of the components.

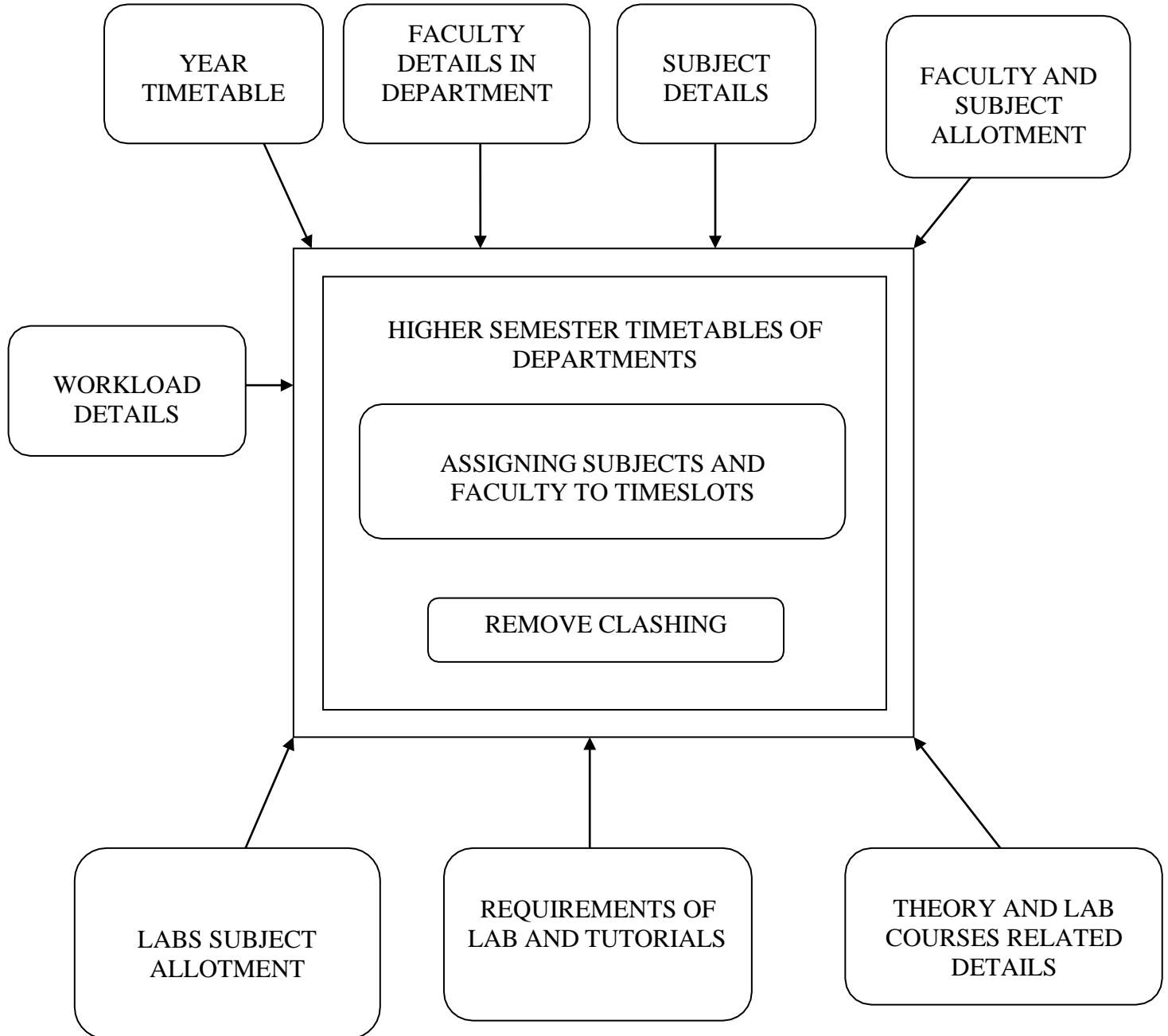


Fig 1: System Design of Timetable

Design contains the following things:

- 1) First year timetable it contains timetable of first year based on that we will create higher semester timetables.
- 2) Faculty details in department tells that the details of respective faculty in department.
- 3) A workload detail tells that the higher and lower workload that faculty has based on their designation. That is for professor has less work than the assistant professor.
- 4) Subject details as subject name, subject code.
- 5) Faculty and subject allotment table consist for which subject respective faculty is allotted based on timeslots.
- 6) Theory and lab courses related details contain the details of each subject that is handled by respective faculty.

IMPLEMENTATION AND CODING

HomeForm Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.Forms.ConfigurationForms;
using TimeTableGenerator.Forms.LectureSubjectForms;
using TimeTableGenerator.Forms.ProgramSemesterForms;
using TimeTableGenerator.Forms.TimeSlotForms;
using TimeTableGenerator.Forms.TimeTableForms;
using TimeTableGenerator.Reports;

namespace TimeTableGenerator.Forms
{
    public partial class HomeForm : Form
    {
        frmCourses CoursesForm;
        frmDays DaysForm;
        frmLabs LabsForm;
        frmLectures LecturesForm;
        frmProgram ProgramForm;
        frmRoom RoomForm;
        frmSemesters SemestersForm;
        frmSession SessionForm;
        FrmLecturesSubject LecturesSubjectForm;
        frmProgramSemesters ProgramSemestersForm;
        frmProgramSemesterSubject ProgramSemesterSubjectForm;
        frmDayTimeSlots DayTimeSlotsForm;
        frmSemesterSections SemesterSectionsForm;
        frmTimeTableManualEntry TimeTableManualEntryForm;
        frmAutoGenerateTimeTable AutoGenerateTimeTableForm;
        frmPrintAllTimeTables PrintAllTimeTablesForm;
        frmPrintTeacherWiseTimeTable PrintAllTeacherTimeTablesForm;
        frmPrintDaysWise PrintAllDaysTimeTablesForm;

        public HomeForm()
        {
            InitializeComponent();
        }
    }
}
```



```

        tsslblDateTime.Text = DateTime.Now.ToString("hh:mm tt dddd MMMM yyyy");
    }

private void toolStripButton2_Click(object sender, EventArgs e)
{
    if (ProgramForm == null)
    {
        ProgramForm = new frmProgram();
    }
    ProgramForm.ShowDialog();

}

private void toolStripButton3_Click(object sender, EventArgs e)
{
    if (SessionForm == null)
    {
        SessionForm = new frmSession();
    }
    SessionForm.ShowDialog();

}

private void toolStripButton4_Click(object sender, EventArgs e)
{
    if (CoursesForm == null)
    {
        CoursesForm = new frmCourses();
    }
    CoursesForm.ShowDialog();
}

private void newLecturesToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (LecturesForm == null)
    {
        LecturesForm = new frmLectures();
    }
    LecturesForm.ShowDialog();
}

private void assignSubjectsToLectureToolStripMenuItem_Click(object sender, EventArgs
e)
{
    if (LecturesSubjectForm == null)

```

```

    {
        LecturesSubjectForm = new FrmLecturesSubject();
    }
    LecturesSubjectForm.ShowDialog();
}

private void addRoomsToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (RoomForm == null)
    {
        RoomForm = new frmRoom();
    }
    RoomForm.ShowDialog();
}

private void addLabsToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (LabsForm == null)
    {
        LabsForm = new frmLabs();
    }
    LabsForm.ShowDialog();
}

private void newSemestersToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (SemestersForm == null)
    {
        SemestersForm = new frmSemesters();
    }
    SemestersForm.ShowDialog();
}

private void assignSemesterToProgrameToolStripMenuItem_Click(object sender,
EventArgs e)
{
    if (ProgramSemestersForm == null)
    {
        ProgramSemestersForm = new frmProgramSemesters();
    }
    ProgramSemestersForm.ShowDialog();
}

private void assignSubjectToSemesterToolStripMenuItem_Click(object sender, EventArgs
e)
{

```

```

        if (ProgramSemesterSubjectForm == null)
        {
            ProgramSemesterSubjectForm = new frmProgramSemesterSubject();
        }
        ProgramSemesterSubjectForm.ShowDialog();
    }

    private void toolStripMenuItem1_Click(object sender, EventArgs e)
    {
        if (DaysForm == null)
        {
            DaysForm = new frmDays();
        }
        DaysForm.ShowDialog();
    }

    private void toolStripMenuItem2_Click(object sender, EventArgs e)
    {
        if (DayTimeSlotsForm == null)
        {
            DayTimeSlotsForm = new frmDayTimeSlots();
        }
        DayTimeSlotsForm.ShowDialog();
    }

    private void addSemesterSectionsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (SemesterSectionsForm == null)
        {
            SemesterSectionsForm = new frmSemesterSections();
        }
        SemesterSectionsForm.ShowDialog();
    }

    private void manualTimeTableEntrytoolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (TimeTableManualEntryForm == null)
        {
            TimeTableManualEntryForm = new frmTimeTableManualEntry();
        }
        TimeTableManualEntryForm.ShowDialog();
    }

    private void autoGenerateTimeTableToolStripMenuItem_Click(object sender, EventArgs e)
    {

```

```

        if (AutoGenerateTimeTableForm == null)
        {
            AutoGenerateTimeTableForm = new frmAutoGenerateTimeTable();
        }
        AutoGenerateTimeTableForm.ShowDialog();
    }

    private void printAllTimeTablesToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        if (PrintAllTimeTablesForm == null)
        {
            PrintAllTimeTablesForm = new frmPrintAllTimeTables();
        }
        PrintAllTimeTablesForm.TopLevel = false;
        panelHeader.Controls.Add(PrintAllTimeTablesForm);
        PrintAllTimeTablesForm.Dock = DockStyle.Fill;
        PrintAllTimeTablesForm.FormBorderStyle = FormBorderStyle.None;
        PrintAllTimeTablesForm.BringToFront();
        PrintAllTimeTablesForm.Show();
    }

    private void printAllTeacherTimeTablesToolStripMenuItem_Click(object sender,
        EventArgs e)
    {
        if (PrintAllTeacherTimeTablesForm == null)
        {
            PrintAllTeacherTimeTablesForm = new frmPrintTeacherWiseTimeTable();
        }
        PrintAllTeacherTimeTablesForm.TopLevel = false;
        panelHeader.Controls.Add(PrintAllTeacherTimeTablesForm);
        PrintAllTeacherTimeTablesForm.Dock = DockStyle.Fill;
        PrintAllTeacherTimeTablesForm.FormBorderStyle = FormBorderStyle.None;
        PrintAllTeacherTimeTablesForm.BringToFront();
        PrintAllTeacherTimeTablesForm.Show();
    }

    private void HomeForm_Load(object sender, EventArgs e)
    {
    }

    private void printAllDaysWiseTimeTablesToolStripMenuItem_Click(object sender,
        EventArgs e)
    {

```

```

        if (PrintAllDaysTimeTablesForm == null)
        {
            PrintAllDaysTimeTablesForm = new frmPrintDaysWise();
        }
        PrintAllDaysTimeTablesForm.TopLevel = false;
        panelHeader.Controls.Add(PrintAllDaysTimeTablesForm);
        PrintAllDaysTimeTablesForm.Dock = DockStyle.Fill;
        PrintAllDaysTimeTablesForm.FormBorderStyle = FormBorderStyle.None;
        PrintAllDaysTimeTablesForm.BringToFront();
        PrintAllDaysTimeTablesForm.Show();
    }

    private void label1_Click(object sender, EventArgs e)
    {

    }

}
}

```

frmCourses:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmCourses : Form
    {
        public frmCourses()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
    }
}

```

```

{
    try
    {
        string query = string.Empty;
        if (string.IsNullOrEmpty(searchvalue.Trim()))
        {
            query = "select CourseID [ID], Title [Subject], CrHrs, RoomTypeID, TypeName
[Type], IsActive from v_AllSubjects";
        }
        else
        {
            query = "select CourseID [ID], Title [Subject], CrHrs, RoomTypeID, TypeName
[Type], IsActive from v_AllSubjects where (Title + ' ' + TypeName) like '%" +
searchvalue.Trim() + "%'";
        }

        DataTable Lablist = DatabaseLayer.Retrieve(query);
        dgvCourses.DataSource = Lablist;
        if (dgvCourses.Rows.Count > 0)
        {
            dgvCourses.Columns[0].Width = 80; // CourseID
            dgvCourses.Columns[1].Width = 250; // Title
            dgvCourses.Columns[2].Width = 60; // CrHrs
            dgvCourses.Columns[3].Visible = false; // RoomTypeID
            dgvCourses.Columns[4].Width = 250; // TypeName
            dgvCourses.Columns[5].Width = 80; // IsActive
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void frmCourses_Load(object sender, EventArgs e)
{
    cmbCrHrs.SelectedIndex = 0;
    ComboHelper.RoomTypes(cmbSelectType);
    FillGrid(string.Empty);
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

```

```

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSubjectTitle.Text.Length == 0)
    {
        ep.SetError(txtSubjectTitle, "Please Enter Subject Title!");
        txtSubjectTitle.Focus();
        txtSubjectTitle.SelectAll();
        return;
    }

    if (cmbSelectType.SelectedIndex == 0)
    {
        ep.SetError(cmbSelectType, "Please Select Type!");
        cmbSelectType.Focus();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from CourseTable where Title = 
    "" + txtSubjectTitle.Text.Trim() + "");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSubjectTitle, "Already Exist!");
            txtSubjectTitle.Focus();
            txtSubjectTitle.SelectAll();
            return;
        }
    }

    string insertquery = string.Format("insert into
    CourseTable(Title,CrHrs,RoomTypeID,IsActive) values('{0}','{1}','{2}','{3}"),
        txtSubjectTitle.Text.Trim(), cmbCrHrs.Text, cmbSelectType.SelectedValue,
    chkStatus.Checked);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Save Successfully!");
        SaveClearForm();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Details. Then Try Again!");
    }
}

```

```

    }
}

public void ClearForm()
{
    txtSubjectTitle.Clear();
    cmbSelectType.SelectedIndex = 0;
    cmbCrHrs.SelectedIndex = 0;
    chkStatus.Checked = false;
}

public void SaveClearForm()
{
    txtSubjectTitle.Clear();
    chkStatus.Checked = true;
    FillGrid(string.Empty);
}

public void EnableComponents()
{
    dgvCourses.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvCourses.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

```



```

private void btnCancel_Click(object sender, EventArgs e)
{
    DisableComponents();
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvCourses != null)
    {
        if (dgvCourses.Rows.Count > 0)
        {
            if (dgvCourses.SelectedRows.Count == 1)
            {
                txtSubjectTitle.Text =
Convert.ToString(dgvCourses.CurrentRow.Cells[1].Value);
                cmbSelectType.SelectedValue =
Convert.ToString(dgvCourses.CurrentRow.Cells[3].Value); // Program ID
                cmbCrHrs.Text = Convert.ToString(dgvCourses.CurrentRow.Cells[2].Value); //
Semester ID
                chkStatus.Checked =
Convert.ToBoolean(dgvCourses.CurrentRow.Cells[5].Value);
                EnableComponents();
            }
            else
            {
                MessageBox.Show("Please Select One Record!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
    else
    {
        MessageBox.Show("List is Empty!");
    }
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSubjectTitle.Text.Length == 0)
    {
        ep.SetError(txtSubjectTitle, "Please Enter Subject Title!");
    }
}

```

```

        txtSubjectTitle.Focus();
        txtSubjectTitle.SelectAll();
        return;
    }

    if (cmbSelectType.SelectedIndex == 0)
    {
        ep.SetError(cmbSelectType, "Please Select Type!");
        cmbSelectType.Focus();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from CourseTable where Title =
'" + txtSubjectTitle.Text.Trim() + "' and CourseID !=
'" + Convert.ToString(dgvCourses.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSubjectTitle, "Already Exist!");
            txtSubjectTitle.Focus();
            txtSubjectTitle.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("update CourseTable set Title = '{0}',CrHrs =
'{1}',RoomTypeID = '{2}',IsActive = '{3}' Where CourseID = '{4}' ",
        txtSubjectTitle.Text.Trim(), cmbCrHrs.Text, cmbSelectType.SelectedValue,
        chkStatus.Checked, Convert.ToString(dgvCourses.CurrentRow.Cells[0].Value));
    bool result = DatabaseLayer.Update(updatequery);
    if (result == true)
    {
        MessageBox.Show("Updated Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Details. Then Try Again!");
    }
}
}
}

```

frmDays:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmDays : Form
    {
        public frmDays()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select DayID [ID], Name [Day], IsActive [Status] from DayTable";
                }
                else
                {
                    query = "select DayID [ID], Name [Day], IsActive [Status] from DayTable where
Name like '%" + searchvalue.Trim() + "%'";
                }

                DataTable daylist = DatabaseLayer.Retrieve(query);
                dgvDays.DataSource = daylist;
                if (dgvDays.Rows.Count > 0)
                {
                    dgvDays.Columns[0].Width = 80;
                    dgvDays.Columns[1].Width = 150;
                    dgvDays.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                }
            }
        }
    }
}
```

```

        catch
        {
            MessageBox.Show("Some unexpected issue occure plz try again!");
        }
    }

    private void frmDays_Load(object sender, EventArgs e)
    {
        FillGrid(string.Empty);
    }

    private void txtSearch_TextChanged(object sender, EventArgs e)
    {
        FillGrid(txtSearch.Text.ToUpper());
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        ep.Clear();
        if (txtDayName.Text.Length == 0)
        {
            ep.SetError(txtDayName, "Please Enter Day Name!");
            txtDayName.Focus();
            txtDayName.SelectAll();
            return;
        }

        DataTable checktitle = DatabaseLayer.Retrieve("select * from DayTable where Name = '"
+ txtDayName.Text.ToUpper().Trim() + "'");
        if (checktitle != null)
        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtDayName, "Already Exist!");
                txtDayName.Focus();
                txtDayName.SelectAll();
                return;
            }
        }

        string insertquery = string.Format("insert into DayTable(Name,IsActive)
values('{0}','{1}')"
, txtDayName.Text.ToUpper().Trim(), chkStatus.Checked);
        bool result = DatabaseLayer.Insert(insertquery);
        if (result == true)
        {

```

```

        MessageBox.Show("Save Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
    }
}

```

```

public void ClearForm()
{
    txtDayName.Clear();
    chkStatus.Checked = false;

}

```

```

public void EnableComponents()
{
    dgvDays.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

```

```

public void DisableComponents()
{
    dgvDays.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

```

```

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

```

```

private void btnCancel_Click(object sender, EventArgs e)
{

```

```

        DisableComponents();
    }

    private void cmsedit_Click(object sender, EventArgs e)
    {
        if (dgvDays != null)
        {
            if (dgvDays.Rows.Count > 0)
            {
                if (dgvDays.SelectedRows.Count == 1)
                {
                    txtDayName.Text = Convert.ToString(dgvDays.CurrentRow.Cells[1].Value);
                    chkStatus.Checked = Convert.ToBoolean(dgvDays.CurrentRow.Cells[2].Value);
                    EnableComponents();
                }
                else
                {
                    MessageBox.Show("Please Select One Record!");
                }
            }
            else
            {
                MessageBox.Show("List is Empty!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
}

```

```

    private void btnUpdate_Click(object sender, EventArgs e)
    {
        ep.Clear();
        if (txtDayName.Text.Length == 0)
        {
            ep.SetError(txtDayName, "Please Enter Day Name!");
            txtDayName.Focus();
            txtDayName.SelectAll();
            return;
        }
    }

```

```

        DataTable checktitle = DatabaseLayer.Retrieve("select * from DayTable where Name = '"
+ txtDayName.Text.ToUpper().Trim() + "' and DayID != '" +
Convert.ToString(dgvDays.CurrentRow.Cells[0].Value) + "'");
        if (checktitle != null)

```

```

        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtDayName, "Already Exist!");
                txtDayName.Focus();
                txtDayName.SelectAll();
                return;
            }
        }

        string updatequery = string.Format("update DayTable set Name = '{0}',IsActive = '{1}'
where DayID = '{2}'",
            txtDayName.Text.ToUpper().Trim(), chkStatus.Checked,
            Convert.ToString(dgvDays.CurrentRow.Cells[0].Value));
        bool result = DatabaseLayer.Update(updatequery);
        if (result == true)
        {
            MessageBox.Show("Updated Successfully!");
            DisableComponents();
        }
        else
        {
            MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
        }
    }
}
}

```

frmLabs:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmLabs : Form
    {
        public frmLabs()
        {

```

```

        InitializeComponent();
    }

    private void txtCapacity_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }

    public void FillGrid(string searchvalue)
    {
        try
        {
            string query = string.Empty;
            if (string.IsNullOrEmpty(searchvalue.Trim()))
            {
                query = "select LabID [ID], LabNo [Lab], Capacity, IsActive [Status] from
LabTable";
            }
            else
            {
                query = "select LabID [ID], LabNo [Lab], Capacity, IsActive [Status] from
LabTable where LabNo like '%" + searchvalue.Trim() + "%'";
            }

            DataTable Lablist = DatabaseLayer.Retrieve(query);
            dgvLabs.DataSource = Lablist;
            if (dgvLabs.Rows.Count > 0)
            {
                dgvLabs.Columns[0].Width = 80;
                dgvLabs.Columns[1].Width = 150;
                dgvLabs.Columns[2].Width = 100;
                dgvLabs.Columns[3].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
            }
        }
        catch
        {
            MessageBox.Show("Some unexpected issue occure plz try again!");
        }
    }

    private void frmLabs_Load(object sender, EventArgs e)
    {
        FillGrid(string.Empty);
    }

```



```

    }

    private void txtSearch_TextChanged(object sender, EventArgs e)
    {
        FillGrid(txtSearch.Text.Trim());
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        ep.Clear();
        if (txtLabNo.Text.Length == 0 || txtLabNo.Text.Length > 31)
        {
            ep.SetError(txtLabNo, "Please Enter Lab No/Name Correct but must be less than 31
char!");
            txtLabNo.Focus();
            txtLabNo.SelectAll();
            return;
        }

        if (txtCapacity.Text.Length == 0)
        {
            ep.SetError(txtCapacity, "Please Enter Lab Capacity!");
            txtCapacity.Focus();
            txtCapacity.SelectAll();
            return;
        }

        DataTable checktitle = DatabaseLayer.Retrieve("select * from LabTable where LabNo = "
+ txtLabNo.Text.ToUpper().Trim() + "");
        if (checktitle != null)
        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtLabNo, "Already Exist!");
                txtLabNo.Focus();
                txtLabNo.SelectAll();
                return;
            }
        }

        string insertquery = string.Format("insert into LabTable(LabNo,Capacity,IsActive)
values('{0}','{1}','{2}']",
            txtLabNo.Text.ToUpper().Trim(), txtCapacity.Text.Trim(),
chkStatus.Checked);
        bool result = DatabaseLayer.Insert(insertquery);
        if (result == true)

```

```

    {
        MessageBox.Show("Save Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
    }
}

public void ClearForm()
{
    txtLabNo.Clear();
    txtCapacity.Clear();
    chkStatus.Checked = false;
}

public void EnableComponents()
{
    dgvLabs.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvLabs.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

private void btnCancel_Click(object sender, EventArgs e)

```

```

{
    DisableComponents();
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvLabs != null)
    {
        if (dgvLabs.Rows.Count > 0)
        {
            if (dgvLabs.SelectedRows.Count == 1)
            {
                txtLabNo.Text = Convert.ToString(dgvLabs.CurrentRow.Cells[1].Value);
                txtCapacity.Text = Convert.ToString(dgvLabs.CurrentRow.Cells[2].Value);
                chkStatus.Checked = Convert.ToBoolean(dgvLabs.CurrentRow.Cells[3].Value);
                EnableComponents();
            }
            else
            {
                MessageBox.Show("Please Select One Record!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
    else
    {
        MessageBox.Show("List is Empty!");
    }
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtLabNo.Text.Length == 0 || txtLabNo.Text.Length > 31)
    {
        ep.SetError(txtLabNo, "Please Enter Lab No/Name Correct but must be less than 31
char!");
        txtLabNo.Focus();
        txtLabNo.SelectAll();
        return;
    }

    if (txtCapacity.Text.Length == 0)

```

```

        {
            ep.SetError(txtCapacity, "Please Enter Lab Capacity!");
            txtCapacity.Focus();
            txtCapacity.SelectAll();
            return;
        }

        DataTable checktitle = DatabaseLayer.Retrieve("select * from LabTable where LabNo = '"
+ txtLabNo.Text.ToUpper().Trim() + "' and LabID != '" +
Convert.ToString(dgvLabs.CurrentRow.Cells[0].Value) + "'");
        if (checktitle != null)
        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtLabNo, "Already Exist!");
                txtLabNo.Focus();
                txtLabNo.SelectAll();
                return;
            }
        }

        string updatequery = string.Format("update LabTable set LabNo = '{0}',Capacity = '{1}',
IsActive = '{2}' where LabID = '{3}'",
            txtLabNo.Text.ToUpper().Trim(), txtCapacity.Text.Trim(),
chkStatus.Checked, Convert.ToString(dgvLabs.CurrentRow.Cells[0].Value));
        bool result = DatabaseLayer.Update(updatequery);
        if (result == true)
        {
            MessageBox.Show("Updated Successfully!");
            DisableComponents();
        }
        else
        {
            MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
        }
    }

    private void groupBox1_Enter(object sender, EventArgs e)
    {
    }
}

```

frmLectures:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmLectures : Form
    {
        public frmLectures()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select LectureID [ID], FullName [Lecturer], ContactNo [Contact No],
IsActive [Status] from LectureTable";
                }
                else
                {
                    query = "select LectureID [ID], FullName [Lecturer], ContactNo [Contact No],
IsActive [Status] from LectureTable where (FullName + ' ' +ContactNo) like '%" +
searchvalue.Trim() + "%'";
                }

                DataTable Lablist = DatabaseLayer.Retrieve(query);
                dgvLecturers.DataSource = Lablist;
                if (dgvLecturers.Rows.Count > 0)
                {
                    dgvLecturers.Columns[0].Width = 80;
                    dgvLecturers.Columns[1].Width = 150;
                    dgvLecturers.Columns[2].Width = 100;
                }
            }
            catch { }
        }
    }
}
```

```

        dgvLecturers.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    }
}
catch
{
    MessageBox.Show("Some unexpected issue occure plz try again!");
}
}

```

```

private void frmLectures_Load(object sender, EventArgs e)
{
    FillGrid(string.Empty);
}

```

```

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

```

```

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtLecturer.Text.Length == 0 || txtLecturer.Text.Length > 31)
    {
        ep.SetError(txtLecturer, "Please Enter Full Name!");
        txtLecturer.Focus();
        txtLecturer.SelectAll();
        return;
    }

```

```

    if (txtContactNo.Text.Length < 11)
    {
        ep.SetError(txtContactNo, "Please Enter Correct Contact No!");
        txtContactNo.Focus();
        txtContactNo.SelectAll();
        return;
    }

```

```

    DataTable checktitle = DatabaseLayer.Retrieve("select * from LectureTable where
FullName = '" + txtLecturer.Text.ToUpper().Trim() + "' and ContactNo =
'" + txtContactNo.Text.Trim() + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {

```

```

        ep.SetError(txtLecturer, "Already Exist!");
        txtLecturer.Focus();
        txtLecturer.SelectAll();
        return;
    }
}

string insertquery = string.Format("insert into
LectureTable(FullName,ContactNo,IsActive) values('{0}','{1}','{2}')"
        txtLecturer.Text.ToUpper().Trim(), txtContactNo.Text.Trim(),
chkStatus.Checked);
bool result = DatabaseLayer.Insert(insertquery);
if (result == true)
{
    MessageBox.Show("Save Successfully!");
    DisableComponents();
}
else
{
    MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
}
}

public void ClearForm()
{
    txtLecturer.Clear();
    txtContactNo.Clear();
    chkStatus.Checked = false;
}

public void EnableComponents()
{
    dgvLecturers.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvLecturers.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;

```

```

        btnCancel.Visible = false;
        btnUpdate.Visible = false;
        txtSearch.Enabled = true;
        ClearForm();
        FillGrid(string.Empty);
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        ClearForm();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        DisableComponents();
    }

    private void cmsedit_Click(object sender, EventArgs e)
    {
        if (dgvLecturers != null)
        {
            if (dgvLecturers.Rows.Count > 0)
            {
                if (dgvLecturers.SelectedRows.Count == 1)
                {
                    txtLecturer.Text = Convert.ToString(dgvLecturers.CurrentRow.Cells[1].Value);
                    txtContactNo.Text =
Convert.ToString(dgvLecturers.CurrentRow.Cells[2].Value);
                    chkStatus.Checked =
Convert.ToBoolean(dgvLecturers.CurrentRow.Cells[3].Value);
                    EnableComponents();
                }
                else
                {
                    MessageBox.Show("Please Select One Record!");
                }
            }
            else
            {
                MessageBox.Show("List is Empty!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }

```



```

    }

    private void btnUpdate_Click(object sender, EventArgs e)
    {
        ep.Clear();
        if (txtLecturer.Text.Length == 0 || txtLecturer.Text.Length > 31)
        {
            ep.SetError(txtLecturer, "Please Enter Full Name!");
            txtLecturer.Focus();
            txtLecturer.SelectAll();
            return;
        }

        if (txtContactNo.Text.Length < 11)
        {
            ep.SetError(txtContactNo, "Please Enter Correct Contact No!");
            txtContactNo.Focus();
            txtContactNo.SelectAll();
            return;
        }

        DataTable checktitle = DatabaseLayer.Retrieve("select * from LectureTable where
        FullName = '" + txtLecturer.Text.ToUpper().Trim() + "' and ContactNo =
        '"+txtContactNo.Text.Trim()+"' and LectureID != '" +
        Convert.ToString(dgvLecturers.CurrentRow.Cells[0].Value) + "'");
        if (checktitle != null)
        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtLecturer, "Already Exist!");
                txtLecturer.Focus();
                txtLecturer.SelectAll();
                return;
            }
        }

        string updatequery = string.Format("update LectureTable set FullName =
        '{0}',ContactNo = '{1}', IsActive = '{2}' where LectureID = '{3}'",
        txtLecturer.Text.ToUpper().Trim(), txtContactNo.Text.Trim(),
        chkStatus.Checked, Convert.ToString(dgvLecturers.CurrentRow.Cells[0].Value));
        bool result = DatabaseLayer.Update(updatequery);
        if (result == true)
        {
            MessageBox.Show("Updated Successfully!");
            DisableComponents();
        }
    }

```

```

        else
        {
            MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
        }
    }

    private void groupBox1_Enter(object sender, EventArgs e)
    {

    }
}

```

frmProgram:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmProgram : Form
    {
        public frmProgram()
        {
            InitializeComponent();
        }
        public void FillGrid(string searchvalue)
        {

            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select ProgramID [ID], Name [Progam], IsActive [Status] from
ProgramTable";
                }
                else
                {

```

```

        query = "select ProgramID [ID], Name [Progam], IsActive [Status] from
ProgramTable where Name like '%" + searchvalue.Trim() + "%'";
    }

    DataTable programlist = DatabaseLayer.Retrieve(query);
    dgvProgrames.DataSource = programlist;
    if (dgvProgrames.Rows.Count > 0)
    {
        dgvProgrames.Columns[0].Width = 80;
        dgvProgrames.Columns[1].Width = 150;
        dgvProgrames.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
    }
}
catch
{
    MessageBox.Show("Some unexpected issue occure plz try again!");
}
}

private void frmProgram_Load(object sender, EventArgs e)
{
    FillGrid(string.Empty);
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.ToUpper());
}

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtProgramName.Text.Length == 0)
    {
        ep.SetError(txtProgramName, "Please Enter Program Name!");
        txtProgramName.Focus();
        txtProgramName.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from ProgramTable where Name
= '" + txtProgramName.Text.Trim() + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)

```

```

        {
            ep.SetError(txtProgramName, "Already Exist!");
            txtProgramName.Focus();
            txtProgramName.SelectAll();
            return;
        }
    }

    string insertquery = string.Format("insert into ProgramTable(Name,IsActive)
values('{0}','{1}')"
        txtProgramName.Text.Trim(), chkStatus.Checked);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Save Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
    }
}

public void ClearForm()
{
    txtProgramName.Clear();
    chkStatus.Checked = false;
}

public void EnableComponents()
{
    dgvProgrames.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvProgrames.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
}

```

```

        btnUpdate.Visible = false;
        txtSearch.Enabled = true;
        ClearForm();
        FillGrid(string.Empty);
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        ClearForm();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        DisableComponents();
    }

    private void cmsedit_Click(object sender, EventArgs e)
    {
        if (dgvProgrames != null)
        {
            if (dgvProgrames.Rows.Count > 0)
            {
                if (dgvProgrames.SelectedRows.Count == 1)
                {
                    txtProgramName.Text =
Convert.ToString(dgvProgrames.CurrentRow.Cells[1].Value);
                    chkStatus.Checked =
Convert.ToBoolean(dgvProgrames.CurrentRow.Cells[2].Value);
                    EnableComponents();
                }
                else
                {
                    MessageBox.Show("Please Select One Record!");
                }
            }
            else
            {
                MessageBox.Show("List is Empty!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
}

```

```

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtProgramName.Text.Length == 0)
    {
        ep.SetError(txtProgramName, "Please Enter Program Name!");
        txtProgramName.Focus();
        txtProgramName.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from ProgramTable where Name
= '" + txtProgramName.Text.Trim() + "' and ProgramID != '" +
Convert.ToString(dgvProgrames.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtProgramName, "Already Exist!");
            txtProgramName.Focus();
            txtProgramName.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("update ProgramTable set Name = '{0}',IsActive =
'{1}' where ProgramID = '{2}'",
txtProgramName.Text.Trim(), chkStatus.Checked,
Convert.ToString(dgvProgrames.CurrentRow.Cells[0].Value));
    bool result = DatabaseLayer.Update(updatequery);
    if (result == true)
    {
        MessageBox.Show("Updated Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
    }
}
}

```

frmRoom:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmRoom : Form
    {
        public frmRoom()
        {
            InitializeComponent();
        }

        private void txtCapacity_KeyPress(object sender, KeyPressEventArgs e)
        {
            if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
            {
                e.Handled = true;
            }
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select RoomID [ID], RoomNo [Room], Capacity, IsActive [Status] from
RoomTable";
                }
                else
                {
                    query = "select RoomID [ID], RoomNo [Room], Capacity, IsActive [Status] from
RoomTable where RoomNo like '%" + searchvalue.Trim() + "%'";
                }

                DataTable roomlist = DatabaseLayer.Retrieve(query);
            }
            catch { }
        }
    }
}
```

```

        dgvRooms.DataSource = roomlist;
        if (dgvRooms.Rows.Count > 0)
        {
            dgvRooms.Columns[0].Width = 80;
            dgvRooms.Columns[1].Width = 150;
            dgvRooms.Columns[2].Width = 100;
            dgvRooms.Columns[3].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void frmRoom_Load(object sender, EventArgs e)
{
    FillGrid(string.Empty);
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtRoomNo.Text.Length == 0 || txtRoomNo.Text.Length > 10)
    {
        ep.SetError(txtRoomNo, "Please Enter Room No/Name Correct but must be less than
11 char!");
        txtRoomNo.Focus();
        txtRoomNo.SelectAll();
        return;
    }

    if (txtCapacity.Text.Length == 0)
    {
        ep.SetError(txtCapacity, "Please Enter Room Capacity!");
        txtCapacity.Focus();
        txtCapacity.SelectAll();
        return;
    }
}

```



```

        DataTable checktitle = DatabaseLayer.Retrieve("select * from RoomTable where
RoomNo = '" + txtRoomNo.Text.ToUpper().Trim() + "'");
        if (checktitle != null)
        {
            if (checktitle.Rows.Count > 0)
            {
                ep.SetError(txtRoomNo, "Already Exist!");
                txtRoomNo.Focus();
                txtRoomNo.SelectAll();
                return;
            }
        }

        string insertquery = string.Format("insert into RoomTable(RoomNo,Capacity,IsActive)
values('{0}','{1}','{2}')",
            txtRoomNo.Text.ToUpper().Trim(),
txtCapacity.Text.Trim(),chkStatus.Checked);
        bool result = DatabaseLayer.Insert(insertquery);
        if (result == true)
        {
            MessageBox.Show("Save Successfully!");
            DisableComponents();
        }
        else
        {
            MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
        }
    }

    public void ClearForm()
    {
        txtRoomNo.Clear();
        txtCapacity.Clear();
        chkStatus.Checked = false;
    }

    public void EnableComponents()
    {
        dgvRooms.Enabled = false;
        btnClear.Visible = false;
        btnSave.Visible = false;
        btnCancel.Visible = true;
        btnUpdate.Visible = true;
        txtSearch.Enabled = false;
    }

```

```

public void DisableComponents()
{
    dgvRooms.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

private void btnCancel_Click(object sender, EventArgs e)
{
    DisableComponents();
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvRooms != null)
    {
        if (dgvRooms.Rows.Count > 0)
        {
            if (dgvRooms.SelectedRows.Count == 1)
            {
                txtRoomNo.Text = Convert.ToString(dgvRooms.CurrentRow.Cells[1].Value);
                txtCapacity.Text = Convert.ToString(dgvRooms.CurrentRow.Cells[2].Value);
                chkStatus.Checked =
Convert.ToBoolean(dgvRooms.CurrentRow.Cells[3].Value);
                EnableComponents();
            }
            else
            {
                MessageBox.Show("Please Select One Record!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
}

```

```

    }
    else
    {
        MessageBox.Show("List is Empty!");
    }
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtRoomNo.Text.Length == 0 || txtRoomNo.Text.Length > 10)
    {
        ep.SetError(txtRoomNo, "Please Enter Room No/Name Correct but must be less than
11 char!");
        txtRoomNo.Focus();
        txtRoomNo.SelectAll();
        return;
    }

    if (txtCapacity.Text.Length == 0)
    {
        ep.SetError(txtCapacity, "Please Enter Room Capacity!");
        txtCapacity.Focus();
        txtCapacity.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from RoomTable where
RoomNo = '" + txtRoomNo.Text.ToUpper().Trim() + "' and RoomID != '" +
Convert.ToString(dgvRooms.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtRoomNo, "Already Exist!");
            txtRoomNo.Focus();
            txtRoomNo.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("update RoomTable set RoomNo = '{0}',Capacity =
'{1}', IsActive = '{2}' where RoomID = '{3}'",
txtRoomNo.Text.ToUpper().Trim(),
txtCapacity.Text.Trim(),chkStatus.Checked,
Convert.ToString(dgvRooms.CurrentRow.Cells[0].Value));

```

```

        bool result = DatabaseLayer.Update(updatequery);
        if (result == true)
        {
            MessageBox.Show("Updated Successfully!");
            DisableComponents();
        }
        else
        {
            MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
        }
    }

    private void groupBox1_Enter(object sender, EventArgs e)
    {

    }
}

```

frmSemesters:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmSemesters : Form
    {
        public frmSemesters()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {

            try
            {
                string query = string.Empty;

```

```

        if (string.IsNullOrEmpty(searchvalue.Trim()))
        {
            query = "select SemesterID [ID], SemesterName [Semester], IsActive [Status] from
SemesterTable";
        }
        else
        {
            query = "select SemesterID [ID], SemesterName [Semester], IsActive [Status] from
SemesterTable where SemesterName like '%" + searchvalue.Trim() + "%'";
        }

        DataTable semesterlist = DatabaseLayer.Retrieve(query);
        dgvSemester.DataSource = semesterlist;
        if (dgvSemester.Rows.Count > 0)
        {
            dgvSemester.Columns[0].Width = 80;
            dgvSemester.Columns[1].Width = 150;
            dgvSemester.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void frmSemesters_Load(object sender, EventArgs e)
{
    FillGrid(string.Empty);
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSemesterName.Text.Length == 0)
    {
        ep.SetError(txtSemesterName, "Please Semester Name!");
        txtSemesterName.Focus();
        txtSemesterName.SelectAll();
        return;
    }
}

```

```

    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from SemesterTable where
SemesterName = '" + txtSemesterName.Text.Trim() + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSemesterName, "Already Exist!");
            txtSemesterName.Focus();
            txtSemesterName.SelectAll();
            return;
        }
    }

    string insertquery = string.Format("insert into SemesterTable(SemesterName,IsActive)
values('{0}','{1}')" ,
        txtSemesterName.Text.Trim(), chkStatus.Checked);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Save Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
    }
}

public void ClearForm()
{
    txtSemesterName.Clear();
    chkStatus.Checked = false;
}

public void EnableComponents()
{
    dgvSemester.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

```

```

    }

    public void DisableComponents()
    {
        dgvSemester.Enabled = true;
        btnClear.Visible = true;
        btnSave.Visible = true;
        btnCancel.Visible = false;
        btnUpdate.Visible = false;
        txtSearch.Enabled = true;
        ClearForm();
        FillGrid(string.Empty);
    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        ClearForm();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        DisableComponents();
    }

    private void cmsedit_Click(object sender, EventArgs e)
    {
        if (dgvSemester != null)
        {
            if (dgvSemester.Rows.Count > 0)
            {
                if (dgvSemester.SelectedRows.Count == 1)
                {
                    txtSemesterName.Text =
Convert.ToString(dgvSemester.CurrentRow.Cells[1].Value);
                    chkStatus.Checked =
Convert.ToBoolean(dgvSemester.CurrentRow.Cells[2].Value);
                    EnableComponents();
                }
                else
                {
                    MessageBox.Show("Please Select One Record!");
                }
            }
            else
            {
                MessageBox.Show("List is Empty!");
            }
        }
    }

```

```

    }
}
else
{
    MessageBox.Show("List is Empty!");
}
}

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSemesterName.Text.Length == 0)
    {
        ep.SetError(txtSemesterName, "Please Enter Semester Name!");
        txtSemesterName.Focus();
        txtSemesterName.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from SemesterTable where
SemesterName = '" + txtSemesterName.Text.Trim() + "' and SemesterID != '" +
Convert.ToString(dgvSemester.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSemesterName, "Already Exist!");
            txtSemesterName.Focus();
            txtSemesterName.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("update SemesterTable set SemesterName =
'{0}',IsActive = '{1}' where SemesterID = '{2}'",
txtSemesterName.Text.Trim(), chkStatus.Checked,
Convert.ToString(dgvSemester.CurrentRow.Cells[0].Value));
    bool result = DatabaseLayer.Update(updatequery);
    if (result == true)
    {
        MessageBox.Show("Updated Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
    }
}

```



```

    }
  }
}

```

frmSession:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Forms.ConfigurationForms
{
    public partial class frmSession : Form
    {
        public frmSession()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select SessionID [ID], Title, IsActive [Status] from SessionTable";
                }
                else
                {
                    query = "select SessionID [ID], Title, IsActive [Status] from SessionTable where Title like '%" + searchvalue.Trim() + "%'";
                }

                DataTable sessionlist = DatabaseLayer.Retrieve(query);
                dgvSession.DataSource = sessionlist;
                if (dgvSession.Rows.Count > 0)

```

```

        {
            dgvSession.Columns[0].Width = 80;
            dgvSession.Columns[1].Width = 150;
            dgvSession.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void txtSessionTitle_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
}

private void frmSession_Load(object sender, EventArgs e)
{
    FillGrid(string.Empty);
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSessionTitle.Text.Length < 9)
    {
        ep.SetError(txtSessionTitle, "Please Enter Correct Session Title!");
        txtSessionTitle.Focus();
        txtSessionTitle.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from SessionTable where Title =
'" + txtSessionTitle.Text.Trim() + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {

```

```

        ep.SetError(txtSessionTitle, "Already Exist!");
        txtSessionTitle.Focus();
        txtSessionTitle.SelectAll();
        return;
    }
}

string insertquery = string.Format("insert into SessionTable(Title,IsActive)
values('{0}','{1}')" ,
        txtSessionTitle.Text.Trim(), chkStatus.Checked);
bool result = DatabaseLayer.Insert(insertquery);
if (result == true)
{
    MessageBox.Show("Save Successfully!");
    DisableComponents();
}
else {
    MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
}
}

public void ClearForm()
{
    txtSessionTitle.Clear();
    chkStatus.Checked = false;
}

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

public void EnableComponents()
{
    dgvSession.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()

```

```

{
    dgvSession.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvSession != null)
    {
        if (dgvSession.Rows.Count > 0)
        {
            if (dgvSession.SelectedRows.Count == 1)
            {
                txtSessionTitle.Text = Convert.ToString(dgvSession.CurrentRow.Cells[1].Value);
                chkStatus.Checked =
Convert.ToBoolean(dgvSession.CurrentRow.Cells[2].Value);
                EnableComponents();
            }
            else {
                MessageBox.Show("Please Select One Record!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
    else
    {
        MessageBox.Show("List is Empty!");
    }
}

private void btnCancel_Click(object sender, EventArgs e)
{
    DisableComponents();
}

private void btnUpdate_Click(object sender, EventArgs e)

```

```

{
    ep.Clear();
    if (txtSessionTitle.Text.Length < 9)
    {
        ep.SetError(txtSessionTitle, "Please Enter Correct Session Title!");
        txtSessionTitle.Focus();
        txtSessionTitle.SelectAll();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from SessionTable where Title = 
'" + txtSessionTitle.Text.Trim() + "' and SessionID != 
'" + Convert.ToString(dgvSession.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSessionTitle, "Already Exist!");
            txtSessionTitle.Focus();
            txtSessionTitle.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("update SessionTable set Title = '{0}',IsActive = '{1}' 
where SessionID = '{2}'",
        txtSessionTitle.Text.Trim(), chkStatus.Checked,
        Convert.ToString(dgvSession.CurrentRow.Cells[0].Value));
    bool result = DatabaseLayer.Update(updatequery);
    if (result == true)
    {
        MessageBox.Show("Updated Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Session Details. Then Try Again!");
    }
}
}
}

```

frmLecturesSubject:

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.LectureSubjectForms
{
    public partial class FrmLecturesSubject : Form
    {
        public FrmLecturesSubject()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select LectureSubjectID [ID], SubjectTitle [Subject Title], LectureID,
FullName [Lecture],CourseID, " +
                        " Title [Course],IsActive [Status] from v_AllSubjectsTeachers Where IsActive
= 1";
                }
                else
                {
                    query = "select LectureSubjectID [ID], SubjectTitle, LectureID, FullName
[Lecture],CourseID, " +
                        "Title [Course],IsActive [Status] from v_AllSubjectsTeachers " +
                        "WHERE IsActive = 1 AND (SubjectTitle + ' ' + FullName + ' ' + Title) like
'" + searchvalue.Trim()+"%";
                }

                DataTable semesterlist = DatabaseLayer.Retrieve(query);
                dgvTecherSubjects.DataSource = semesterlist;
                if (dgvTecherSubjects.Rows.Count > 0)
                {
                    dgvTecherSubjects.Columns[0].Visible = false; // LectureSubjectID
                    dgvTecherSubjects.Columns[1].Width = 250; //SubjectTitle
                }
            }
            catch { }
        }
    }
}

```

```

        dgvTecherSubjects.Columns[2].Visible = false; // LectureID
        dgvTecherSubjects.Columns[3].Width = 150; // FullName
        dgvTecherSubjects.Columns[4].Visible = false; // CourseID
        dgvTecherSubjects.Columns[5].Width = 300; // Title
        dgvTecherSubjects.Columns[6].Width = 100; // Status
    }
}
catch
{
    MessageBox.Show("Some unexpected issue occure plz try again!");
}
}

public void ClearForm()
{
    cmbTeacher.SelectedIndex = 0;
    cmbSubjects.SelectedIndex = 0;
}

public void EnableComponents()
{
    dgvTecherSubjects.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;

    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvTecherSubjects.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;

    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void FrmLecturesSubject_Load(object sender, EventArgs e)
{
    ComboHelper.AllSubjects(cmbSubjects);
    ComboHelper.AllTechers(cmbTeacher);
    FillGrid(string.Empty);
}

```

```

private void btnClear_Click(object sender, EventArgs e)
{
    ClearForm();
}

private void btnCancel_Click(object sender, EventArgs e)
{
    DisableComponents();
}

private void btnSave_Click(object sender, EventArgs e)
{
    try
    {
        ep.Clear();
        if (cmbTeacher.SelectedIndex == 0)
        {
            ep.SetError(cmbTeacher, "Please Select Teacher!");
            cmbTeacher.Focus();
            return;
        }

        if (cmbSubjects.SelectedIndex == 0)
        {
            ep.SetError(cmbSubjects, "Please Select Subject!");
            cmbSubjects.Focus();
            return;
        }

        DataTable dt = DatabaseLayer.Retrieve("select * from LectureSubjectTable where
LectureID = '"+cmbTeacher.SelectedValue+" ' and CourseID =
 '"+cmbSubjects.SelectedValue+" ' AND IsActive = 1");
        if (dt != null)
        {
            if (dt.Rows.Count > 0)
            {
                ep.SetError(cmbSubjects, "Already Registered!");
                cmbSubjects.Focus();
                return;
            }
        }

        string insertquery = string.Format("insert into
LectureSubjectTable(SubjectTitle,LectureID,CourseID,IsActive) values('{0}','{1}','{2}','1')",

```



```

cmbSubjects.Text+"("+cmbTeacher.Text+")",cmbTeacher.SelectedValue,cmbSubjects.Selected
Value);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Subject Assign Successfully!");
        DisableComponents();
        return;
    }
    else {
        MessageBox.Show("Some Unexpected issue is occure plz try Again!");
    }

}
catch
{

    MessageBox.Show("Please Check Sql Server Agent Connectivity!");
}
}

private void cmsedit_Click(object sender, EventArgs e)
{
    try
    {
        if (dgvTecherSubjects != null)
        {
            if (dgvTecherSubjects.Rows.Count > 0)
            {
                if (dgvTecherSubjects.SelectedRows.Count == 1)
                {
                    if (MessageBox.Show("Are you sure you want to Delete selected record?",
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                    {
                        string id =
Convert.ToString(dgvTecherSubjects.CurrentRow.Cells[0].Value);
                        bool status =
(Convert.ToBoolean(dgvTecherSubjects.CurrentRow.Cells[6].Value) == true ? false : true);
                        string updatequery = "Update LectureSubjectTable set IsActive = " + status
+ " where LectureSubjectID = " + id + """;
                        bool result = DatabaseLayer.Update(updatequery);
                        if (result == true)
                        {
                            MessageBox.Show("Deleted Successfully!");
                            DisableComponents();

```



```

public void FillGrid(string searchvalue)
{
    try
    {
        string query = string.Empty;
        if (string.IsNullOrEmpty(searchvalue.Trim()))
        {
            query = "select ProgramSemesterID [ID], Title, Capacity,ProgramSemesterIsActive
[Status], ProgramID, SemesterID,SessionID from v_ProgramSemesterActiveList";
        }
        else
        {
            query = "select ProgramSemesterID [ID], Title, Capacity,ProgramSemesterIsActive
[Status], ProgramID, SemesterID,SessionID from v_ProgramSemesterActiveList where Title
like '%" + searchvalue.Trim() + "%'";
        }

        DataTable Lablist = DatabaseLayer.Retrieve(query);
        dgvProgramSemesters.DataSource = Lablist;
        if (dgvProgramSemesters.Rows.Count > 0)
        {
            dgvProgramSemesters.Columns[0].Width = 60; // ProgramSemesterID
            dgvProgramSemesters.Columns[1].Width = 250; // Title
            dgvProgramSemesters.Columns[2].Width = 80; // Capacity
            dgvProgramSemesters.Columns[3].Width = 80; // ProgramSemesterIsActive
            dgvProgramSemesters.Columns[4].Visible = false; // ProgramID
            dgvProgramSemesters.Columns[5].Visible = false; // SemesterID
            dgvProgramSemesters.Columns[6].Visible = false; // SessionID
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

```

```

private void frmProgramSemesters_Load(object sender, EventArgs e)
{
    ComboHelper.Session(cmbSelectSession);
    ComboHelper.Semesters(cmbSelectSemester);
    ComboHelper.Programs(cmbSelectProgram);
    FillGrid(string.Empty);
}

```

```

    }

    private void cmbSelectProgram_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (!cmbSelectProgram.Text.Contains("Select"))
        {
            if (cmbSelectSemester.SelectedIndex > 0)
            {
                txtSemesterTitle.Text = cmbSelectSession.Text + " " + cmbSelectProgram.Text + "
" + cmbSelectSemester.Text;
            }
        }
    }

    private void cmbSelectSemester_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (!cmbSelectSemester.Text.Contains("Select"))
        {
            if (cmbSelectProgram.SelectedIndex > 0)
            {
                txtSemesterTitle.Text = cmbSelectSession.Text + " " + cmbSelectProgram.Text + "
" + cmbSelectSemester.Text;
            }
        }
    }

    private void txtSearch_TextChanged(object sender, EventArgs e)
    {
        FillGrid(txtSearch.Text.Trim());
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        ep.Clear();
        if (txtSemesterTitle.Text.Length == 0)
        {
            ep.SetError(txtSemesterTitle, "Please Select Again! Title is Empty!");
            txtSemesterTitle.Focus();
            txtSemesterTitle.SelectAll();
            return;
        }

        if (cmbSelectProgram.SelectedIndex == 0)
        {
            ep.SetError(cmbSelectProgram, "Please Select Program!");
            cmbSelectProgram.Focus();
        }
    }

```

```

        return;
    }

    if (cmbSelectSemester.SelectedIndex == 0)
    {
        ep.SetError(cmbSelectProgram, "Please Select Semester!");
        cmbSelectProgram.Focus();
        return;
    }

    if (txtCapacity.Text.Trim().Length == 0)
    {
        ep.SetError(txtCapacity, "Please Enter Semester Capacity!");
        txtCapacity.Focus();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from ProgramSemesterTable
    where ProgramID = " + cmbSelectProgram.SelectedValue + " and SemesterID = " +
    cmbSelectSemester.SelectedValue + " AND SessionID =
    "+cmbSelectSession.SelectedValue+"");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSemesterTitle, "Already Exist!");
            txtSemesterTitle.Focus();
            txtSemesterTitle.SelectAll();
            return;
        }
    }

    string insertquery = string.Format("insert into
    ProgramSemesterTable(Title,ProgramID,SemesterID,IsActive,Capacity,SessionID)
    values('{0}','{1}','{2}','{3}','{4}','{5}");
    txtSemesterTitle.Text.ToUpper().Trim(), cmbSelectProgram.SelectedValue,
    cmbSelectSemester.SelectedValue,
    chkStatus.Checked,txtCapacity.Text.Trim(),cmbSelectSession.SelectedValue);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Save Successfully!");
        SaveClearForm();
    }
    else
    {

```

```

        MessageBox.Show("Please Provide Correct Semester Details. Then Try Again!");
    }
}

public void ClearForm()
{
    txtSemesterTitle.Clear();
    cmbSelectSemester.SelectedIndex = 0;
    cmbSelectProgram.SelectedIndex = 0;
    chkStatus.Checked = false;

}

public void SaveClearForm()
{
    txtSemesterTitle.Clear();
    cmbSelectSemester.SelectedIndex = 0;
    chkStatus.Checked = true;
    FillGrid(string.Empty);

}

public void EnableComponents()
{
    dgvProgramSemesters.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvProgramSemesters.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void btnClear_Click(object sender, EventArgs e)
{

```

```

        ClearForm();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        DisableComponents();
    }

    private void cmsedit_Click(object sender, EventArgs e)
    {
        if (dgvProgramSemesters != null)
        {
            if (dgvProgramSemesters.Rows.Count > 0)
            {
                if (dgvProgramSemesters.SelectedRows.Count == 1)
                {
                    txtSemesterTitle.Text =
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[1].Value);
                    txtCapacity.Text =
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[2].Value);
                    cmbSelectProgram.SelectedValue =
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[4].Value); // Program ID
                    cmbSelectSemester.SelectedValue =
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[5].Value); // Semester ID
                    cmbSelectSession.SelectedValue =
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[6].Value); // Session ID
                    chkStatus.Checked =
Convert.ToBoolean(dgvProgramSemesters.CurrentRow.Cells[3].Value);
                    EnableComponents();
                }
                else
                {
                    MessageBox.Show("Please Select One Record!");
                }
            }
            else
            {
                MessageBox.Show("List is Empty!");
            }
        }
        else
        {
            MessageBox.Show("List is Empty!");
        }
    }
}

```

```

private void btnUpdate_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtSemesterTitle.Text.Length == 0)
    {
        ep.SetError(txtSemesterTitle, "Please Select Again! Title is Empty!");
        txtSemesterTitle.Focus();
        txtSemesterTitle.SelectAll();
        return;
    }

    if (cmbSelectProgram.SelectedIndex == 0)
    {
        ep.SetError(cmbSelectProgram, "Please Select Program!");
        cmbSelectProgram.Focus();
        return;
    }

    if (cmbSelectSemester.SelectedIndex == 0)
    {
        ep.SetError(cmbSelectProgram, "Please Select Semester!");
        cmbSelectProgram.Focus();
        return;
    }

    DataTable checktitle = DatabaseLayer.Retrieve("select * from ProgramSemesterTable
where ProgramID = '" + cmbSelectProgram.SelectedValue + "' and SemesterID = '" +
cmbSelectSemester.SelectedValue + "' and SessionID = '" + cmbSelectSession.SelectedValue + "'
and ProgramSemesterID != '" +
Convert.ToString(dgvProgramSemesters.CurrentRow.Cells[0].Value) + "'");
    if (checktitle != null)
    {
        if (checktitle.Rows.Count > 0)
        {
            ep.SetError(txtSemesterTitle, "Already Exist!");
            txtSemesterTitle.Focus();
            txtSemesterTitle.SelectAll();
            return;
        }
    }

    string updatequery = string.Format("Update ProgramSemesterTable set Title =
'{0}',ProgramID = '{1}',SemesterID = '{2}',IsActive = '{3}', Capacity = '{4}', SessionID = '{5}'
Where ProgramSemesterID = '{6}' ",
txtSemesterTitle.Text.ToUpper().Trim(), cmbSelectProgram.SelectedValue,
cmbSelectSemester.SelectedValue, chkStatus.Checked,

```



```

txtCapacity.Text.Trim(),cmbSelectSession.Selected.Value,Convert.ToString(dgvProgramSemester.CurrentRow.Cells[0].Value));
    bool result = DatabaseLayer.Update(updatequery);
    if (result == true)
    {
        MessageBox.Show("Updated Successfully!");
        DisableComponents();
    }
    else
    {
        MessageBox.Show("Please Provide Correct Details. Then Try Again!");
    }
}

private void txtCapacity_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
(e.KeyChar != '.'))
    {
        e.Handled = true;
    }
}

private void cmbSelectSession_SelectedIndexChanged(object sender, EventArgs e)
{
    if (!cmbSelectSession.Text.Contains("Select"))
    {
        if (cmbSelectSession.SelectedIndex > 0)
        {
            txtSemesterTitle.Text = cmbSelectSession.Text + " " + cmbSelectProgram.Text + "
" + cmbSelectSemester.Text;
        }
    }
}

private void groupBox1_Enter(object sender, EventArgs e)
{
}
}
}

```

frmProgramSemesterSubject:

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.ProgramSemesterForms
{
    public partial class frmProgramSemesterSubject : Form
    {
        public frmProgramSemesterSubject()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "Select [ProgramSemesterSubjectID]
[ID],[ProgramID],[Program],[ProgramSemesterID], Title [Semester], LectureSubjectID,SSTitle
[Subject], Capacity , IsSubjectActive [Status] From " +
                        " v_AllSemestersSubjects WHERE [ProgramSemesterIsActive] = 1 and
[ProgramIsActive] = 1 and [SemesterIsActive] = 1 and [SubjectIsActive] = 1 " +
                        " Order by ProgramSemesterID";
                }
                else
                {
                    query = "Select [ProgramSemesterSubjectID]
[ID],[ProgramID],[Program],[ProgramSemesterID], Title [Semester], LectureSubjectID,SSTitle
[Subject], Capacity, IsSubjectActive [Status] From " +
                        " v_AllSemestersSubjects WHERE [ProgramSemesterIsActive] = 1 and
[ProgramIsActive] = 1 and [SemesterIsActive] = 1 and [SubjectIsActive] = 1 " +
                        " AND (Program + ' ' + Title + ' ' + SSTitle) like '%" + searchvalue + "%' Order by
ProgramSemesterID";
                }

                DataTable semesterlist = DatabaseLayer.Retrieve(query);
                dgvTecherSubjects.DataSource = semesterlist;
            }
            catch { }
        }
    }
}

```

```

        if (dgvTecherSubjects.Rows.Count > 0)
        {
            dgvTecherSubjects.Columns[0].Visible = false; // ProgramSemesterSubjectID
            dgvTecherSubjects.Columns[1].Visible = false; //ProgramID
            dgvTecherSubjects.Columns[2].Width=120; // Program
            dgvTecherSubjects.Columns[3].Visible = false; // ProgramSemesterID
            dgvTecherSubjects.Columns[4].Width=150; // Semester
            dgvTecherSubjects.Columns[5].Visible = false; // LectureSubjectID
            dgvTecherSubjects.Columns[6].Width = 300; // Subject
            dgvTecherSubjects.Columns[7].Width = 80; // Capacity
            dgvTecherSubjects.Columns[8].Width = 80; // Status
            dgvTecherSubjects.ClearSelection();
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void frmProgramSemesterSubject_Load(object sender, EventArgs e)
{
    ComboHelper.AllProgramSemesters(cmbSemesters);
    ComboHelper.AllTeachersSubject(cmbSubjects);
    FillGrid(string.Empty);
}

private void cmbSemesters_SelectedIndexChanged(object sender, EventArgs e)
{
}

private void cmbSubjects_SelectedIndexChanged(object sender, EventArgs e)
{
    txtTitle.Text = cmbSubjects.SelectedIndex == 0 ? string.Empty : cmbSubjects.Text;
}

private void btnSave_Click(object sender, EventArgs e)
{
    ep.Clear();
    if (txtTitle.Text.Trim().Length == 0)
    {
        ep.SetError(txtTitle, "Please Enter Semester Subject Title!");
        txtTitle.Focus();
        txtTitle.SelectAll();
        return;
    }
}

```

```

    }

    if (cmbSemesters.SelectedIndex == 0)
    {
        ep.SetError(cmbSemesters, "Please Select Semester!");
        cmbSemesters.Focus();
        return;
    }

    if (cmbSubjects.SelectedIndex == 0)
    {
        ep.SetError(cmbSubjects, "Please Select Subject!");
        cmbSubjects.Focus();
        return;
    }

    string checkquery = "select * from ProgramSemesterSubjectTable where
ProgramSemesterID = '" + cmbSemesters.SelectedValue + "' and LectureSubjectID = '" +
cmbSubjects.SelectedValue + "'";
    DataTable dt = DatabaseLayer.Retrieve(checkquery);

    if (dt != null)
    {
        if (dt.Rows.Count > 0)
        {
            ep.SetError(cmbSubjects, "All Ready Exist!");
            cmbSubjects.Focus();
            return;
        }
    }

    string insertquery = string.Format("insert into
ProgramSemesterSubjectTable(SSTitle,ProgramSemesterID,LectureSubjectID)
values('{0}','{1}','{2}')" ,txtTitle.Text.Trim(),cmbSemesters.SelectedValue,
cmbSubjects.SelectedValue);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Subject Assign Successfully!");
        FillGrid(string.Empty);
        FormClear();
    }
    else {
        MessageBox.Show("Please Provide correct details, and try again!");
    }
}

```

```

private void FormClear()
{
    txtTitle.Clear();
    cmbSubjects.SelectedIndex = 0;
}

private void btnClear_Click(object sender, EventArgs e)
{
    FormClear();
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvTecherSubjects != null)
    {
        if (dgvTecherSubjects.Rows.Count > 0)
        {
            if (dgvTecherSubjects.SelectedRows.Count == 1)
            {
                if (MessageBox.Show("Are you sure you want to change status?",
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    bool existstatus =
Convert.ToBoolean(dgvTecherSubjects.CurrentRow.Cells[8].Value);
                    int semestersubjectid =
Convert.ToInt32(dgvTecherSubjects.CurrentRow.Cells[0].Value);
                    bool status = false;
                    if (existstatus == true)
                    {
                        status = false;
                    }
                    else {
                        status = true;
                    }
                    string updatequery = string.Format("update ProgramSemesterSubjectTable set
IsSubjectActive = '{0}' where ProgramSemesterSubjectID = '{1}'", status, semestersubjectid);
                    bool result = DatabaseLayer.Update(updatequery);
                    if (result == true)
                    {

```

```

        MessageBox.Show("Change Successfully!");
        FillGrid(string.Empty);
    }
    else {
        MessageBox.Show("Please Try Again!");
    }
}
}
else {
    MessageBox.Show("Please Select One Record!");
}
}
else {
    MessageBox.Show("List is Empty!");
}
}
else {
    MessageBox.Show("List is Empty!");
}
}
}
}

```

frmSemesterSections:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.ProgramSemesterForms
{
    public partial class frmSemesterSections : Form
    {
        public frmSemesterSections()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
    }
}

```

```

{
    try
    {
        string query = string.Empty;
        if (string.IsNullOrEmpty(searchvalue.Trim()))
        {
            query = " select SectionID, SectionTitle [Section], ProgramSemesterID, Title
[Semester],IsActive [Status] from v_AllSemesterSections order by ProgramSemesterID ";
        }
        else
        {
            query = " select SectionID, SectionTitle [Section], ProgramSemesterID, Title
[Semester],IsActive [Status] from v_AllSemesterSections " +
                " WHERE (SectionTitle + ' ' + Title) like '%" + searchvalue.Trim() + "%' order
by ProgramSemesterID";
        }

        DataTable sectionlist = DatabaseLayer.Retrieve(query);
        dgvSections.DataSource = sectionlist;
        if (dgvSections.Rows.Count > 0)
        {
            dgvSections.Columns[0].Visible = false; // SectionID
            dgvSections.Columns[1].Width = 200; //SectionTitle
            dgvSections.Columns[2].Visible = false; // ProgramSemesterID
            dgvSections.Columns[3].Width = 200; // Title
            dgvSections.Columns[4].Width = 80; // IsActive

            dgvSections.ClearSelection();
        }
    }
    catch
    {
        MessageBox.Show("Some unexpected issue occure plz try again!");
    }
}

private void FormClear()
{
    txtTitle.Clear();
    cmbSemesters.SelectedIndex = 0;
}

private void btnSave_Click(object sender, EventArgs e)
{
    try

```

```

{
    ep.Clear();
    if (txtTitle.Text.Trim().Length == 0)
    {
        ep.SetError(txtTitle, "Please Enter Section Title!");
        txtTitle.Focus();
        return;
    }

    if (cmbSemesters.SelectedIndex == 0)
    {
        ep.SetError(cmbSemesters, "Please Select Semester!");
        cmbSemesters.Focus();
        return;
    }

    DataTable dt = DatabaseLayer.Retrieve("select * from SectionTable where SectionTitle
= '" + txtTitle.Text.Trim() + "' and ProgramSemesterID = '" + cmbSemesters.SelectedValue +
'");
    if (dt != null)
    {
        if (dt.Rows.Count > 0)
        {
            ep.SetError(txtTitle, "Already Exists!");
            txtTitle.Focus();
            return;
        }
    }

    string insertquery = string.Format("insert into
SectionTable(SectionTitle,ProgramSemesterID) values('{0}','{1}')" , txtTitle.Text.Trim(),
cmbSemesters.SelectedValue);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == true)
    {
        MessageBox.Show("Save Successfully!");
        FillGrid(string.Empty);
        FormClear();
    }
    else
    {
        MessageBox.Show("Please Try Again!");
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}

```



```

    }

}

private void frmSemesterSections_Load(object sender, EventArgs e)
{
    ComboHelper.AllProgramSemesters(cmbSemesters);
    FillGrid(string.Empty);
}

private void btnClear_Click(object sender, EventArgs e)
{
    FormClear();
}

private void txtSearch_TextChanged(object sender, EventArgs e)
{
    FillGrid(txtSearch.Text.Trim());
}

private void cmsedit_Click(object sender, EventArgs e)
{
    if (dgvSections != null)
    {
        if (dgvSections.Rows.Count > 0)
        {
            if (dgvSections.SelectedRows.Count == 1)
            {
                if (MessageBox.Show("Are you sure you want to change status?",
"Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {
                    bool existstatus =
Convert.ToBoolean(dgvSections.CurrentRow.Cells[4].Value);
                    int sectionid = Convert.ToInt32(dgvSections.CurrentRow.Cells[0].Value);
                    bool status = false;
                    if (existstatus == true)
                    {
                        status = false;
                    }
                    else
                    {
                        status = true;
                    }
                    string updatequery = string.Format("update SectionTable set IsActive = '{0}'
where SectionID = '{1}'", status, sectionid);

```

```

        bool result = DatabaseLayer.Update(updatequery);
        if (result == true)
        {
            MessageBox.Show("Change Successfully!");
            FillGrid(string.Empty);
        }
        else
        {
            MessageBox.Show("Please Try Again!");
        }
    }
}
else
{
    MessageBox.Show("Please Select One Record!");
}
}
else
{
    MessageBox.Show("List is Empty!");
}
}
else
{
    MessageBox.Show("List is Empty!");
}
}

private void txtCapacity_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&
        (e.KeyChar != '.'))
    {
        e.Handled = true;
    }
}
}
}

```

frmDayTimeSlots:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.AllModels;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.TimeSlotForms
{
    public partial class frmDayTimeSlots : Form
    {
        public frmDayTimeSlots()
        {
            InitializeComponent();
        }

        public void FillGrid(string searchvalue)
        {
            try
            {
                string query = string.Empty;
                if (string.IsNullOrEmpty(searchvalue.Trim()))
                {
                    query = "select DayTimeSlotID,ROW_NUMBER() OVER (Order by
DayTimeSlotID) AS [S No],DayID,Name [Day],SlotTitle [Slot Title],StartTime [Start
Time],EndTime [End Time], IsActive [Status] from v_AllTimeSlots WHERE IsActive = 1 ";
                }
                else
                {
                    query = "select DayTimeSlotID,ROW_NUMBER() OVER (Order by
DayTimeSlotID) AS [S No],DayID,Name [Day],SlotTitle [Slot Title],StartTime [Start
Time],EndTime [End Time], IsActive [Status] from v_AllTimeSlots " +
                        " WHERE IsActive = 1 AND (Name + ' ' + SlotTitle) Like
'" + searchvalue.Trim() + "%'";
                }

                DataTable semesterlist = DatabaseLayer.Retrieve(query);
                dgvSlots.DataSource = semesterlist;
                if (dgvSlots.Rows.Count > 0)
                {
                    dgvSlots.Columns[0].Visible = false; // DayTimeSlotID
                    dgvSlots.Columns[1].Width = 80; // S NO
                    dgvSlots.Columns[2].Visible = false; // DayID
                    dgvSlots.Columns[3].Width = 130; // Name
                }
            }
            catch { }
        }
    }
}

```

```

        dgvSlots.Columns[4].Width = 150; // SlotTitle
        dgvSlots.Columns[5].Width = 100; // StartTime
        dgvSlots.Columns[6].Width = 100; // EndTime
        dgvSlots.Columns[7].Width = 80; // IsActive

    }
}
catch
{
    MessageBox.Show("Some unexpected issue occure plz try again!");
}
}

public void ClearForm()
{
    cmbDays.SelectedIndex = 0;
    cmbNumberOfTimeSlot.SelectedIndex = 0;
    chkStatus.Checked = true;

}

public void EnableComponents()
{
    dgvSlots.Enabled = false;
    btnClear.Visible = false;
    btnSave.Visible = false;
    btnCancel.Visible = true;
    btnUpdate.Visible = true;
    txtSearch.Enabled = false;
}

public void DisableComponents()
{
    dgvSlots.Enabled = true;
    btnClear.Visible = true;
    btnSave.Visible = true;
    btnCancel.Visible = false;
    btnUpdate.Visible = false;
    txtSearch.Enabled = true;
    ClearForm();
    FillGrid(string.Empty);
}

private void frmDayTimeSlots_Load(object sender, EventArgs e)
{

```

```

        dtpStartTime.Value = new DateTime(2020, 12, 12, 8, 0, 0);
        dtpEndTime.Value = new DateTime(2020, 12, 12, 16, 0, 0);
        ComboHelper.AllDays(cmbDays);
        ComboHelper.TimeSlotsNumbers(cmbNumberofTimeSlot);
        FillGrid(string.Empty);
    }

    private void groupBox1_Enter(object sender, EventArgs e)
    {

    }

    private void btnClear_Click(object sender, EventArgs e)
    {
        ClearForm();
    }

    private void btnCancel_Click(object sender, EventArgs e)
    {
        DisableComponents();
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        try
        {
            ep.Clear();
            if (cmbDays.SelectedIndex == 0)
            {
                ep.SetError(cmbDays, "Please Select Day!");
                cmbDays.Focus();
                return;
            }

            if (cmbNumberofTimeSlot.SelectedIndex == 0)
            {
                ep.SetError(cmbNumberofTimeSlot, "Please Select Time Slots Per Day!");
                cmbNumberofTimeSlot.Focus();
                return;
            }

            //string Updatequery = "update DayTimeSlotTable set IsActive = 0 where DayID = '" +
            cmbDays.SelectedValue + "'";
            //bool updatereult = DatabaseLayer.Update(Updatequery);
            //if (updatereult == true)
            //{

```

```

List<TimeSlotsMV> timeslots = new List<TimeSlotsMV>();

TimeSpan time = dtpEndTime.Value - dtpStartTime.Value;

int totalminuts = (int)time.TotalMinutes;

int numeroftimeslot = Convert.ToInt32(cmbNumberofTimeSlot.SelectedValue);

int slot = totalminuts / numeroftimeslot;

int i = 0;
do
{
    var timeslot = new TimeSlotsMV();
    var FromTime = (dtpStartTime.Value).AddMinutes(slot * i);
    i++;
    var ToTime = (dtpStartTime.Value).AddMinutes(slot * i);
    string title = FromTime.ToString("hh:mm tt") + "-" + ToTime.ToString("hh:mm
tt");

    timeslot.FromTime = FromTime;
    timeslot.ToTime = ToTime;
    timeslot.SlotTitle = title;
    timeslots.Add(timeslot);
} while (i < numeroftimeslot);
bool insertstatus = true;
foreach (TimeSlotsMV slottime in timeslots)
{
    string insertquery = string.Format("insert into
DayTimeSlotTable(DayID,SlotTitle,StartTime,EndTime,IsActive)
values('{0}','{1}','{2}','{3}','{4}')"
    cmbDays.SelectedValue, slottime.SlotTitle, slottime.FromTime,
slottime.ToTime, chkStatus.Checked);
    bool result = DatabaseLayer.Insert(insertquery);
    if (result == false)
    {
        insertstatus = false;
    }
}
if (insertstatus == true)
{
    MessageBox.Show("Slots Created Successfully!");
    DisableComponents();
}
else
{
    MessageBox.Show("Please Provide Correct Details, and Try Again!");
}

```



```

using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;
using TimeTableGenerator.TimeTableCode;

namespace TimeTableGenerator.Forms.TimeTableForms
{
    public partial class frmAutoGenerateTimeTable : Form
    {
        public frmAutoGenerateTimeTable()
        {
            InitializeComponent();
        }

        private void frmAutoGenerateTimeTable_Load(object sender, EventArgs e)
        {
        }

        private void btnAutoGenerateAllTimeTable_Click(object sender, EventArgs e)
        {
            try
            {
                ep.Clear();
                string message = GenerateTimeTable.AutoGenerateTimeTable(dtpFromDate.Value,
dtpToDate.Value);
                MessageBox.Show(message);
                return;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

```

frmTimeTableManualEntry:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;

```



```

using System.Threading.Tasks;
using System.Windows.Forms;
using TimeTableGenerator.SourceCode;

namespace TimeTableGenerator.Forms.TimeTableForms
{
    public partial class frmTimeTableManualEntry : Form
    {
        public frmTimeTableManualEntry()
        {
            InitializeComponent();
        }

        private void groupBox5_Enter(object sender, EventArgs e)
        {

        }

        private void frmTimeTableManualEntry_Load(object sender, EventArgs e)
        {
            ComboHelper.AllProgramSemesters(cmbSemesters);
            ComboHelper.Session(cmbSession);
        }

        private void btnSave_Click(object sender, EventArgs e)
        {

        }
    }
}

```

frmPrintAllTimeTable:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Reports
{

```

```

public partial class frmPrintAllTimeTables : Form
{
    public frmPrintAllTimeTables()
    {
        InitializeComponent();
        rpt_PrintAllTimeTables rpt = new rpt_PrintAllTimeTables();
        rpt.Refresh();
        crv.ReportSource = rpt;
    }

    private void frmPrintAllTimeTables_Load(object sender, EventArgs e)
    {

    }
}

```

frmPrintDaysWise:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Reports
{
    public partial class frmPrintDaysWise : Form
    {
        public frmPrintDaysWise()
        {
            InitializeComponent();
            rpt_AllDaysTimeTable rpt = new rpt_AllDaysTimeTable();
            rpt.Refresh();
            crv.ReportSource = rpt;
        }

        private void frmPrintDaysWise_Load(object sender, EventArgs e)
        {

        }
    }
}

```

```
}
```

frmPrintTeacherWiseTimeTable:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TimeTableGenerator.Reports
{
    public partial class frmPrintTeacherWiseTimeTable : Form
    {
        public frmPrintTeacherWiseTimeTable()
        {
            InitializeComponent();
            rpt_AllTeacherWiseTable rpt = new rpt_AllTeacherWiseTable();
            rpt.Refresh();
            crv.ReportSource = rpt;
        }

        private void frmPrintTeacherWiseTimeTable_Load(object sender, EventArgs e)
        {
        }
    }
}
```

TESTING

Different approaches were used to automatically generate the test data in white-box testing for various criteria of coverage. This is done for increasing the coverage percentage, decreasing the test data number, and to minimize the execution time of the testing process. In general, the techniques of "automatic test data generation" can be categorized into random, symbolic, dynamic, and "search-based test data generation techniques".

Recently, the techniques of "search-based test data generation like Genetic Algorithm (GA), have been the greatest commonly used techniques to automatically generate the test data, and are used in the meta-heuristic optimizing techniques for finding the top solution for generating the test data by guiding the search to the suitable spaces of the range. Although these evolutionary algorithms could guide the search and produce relevant test data to ensure an ultimate percentage of path coverage, they still need to be enhanced due to their limitations in producing the optimum set of test data.

SCREENSHOTS

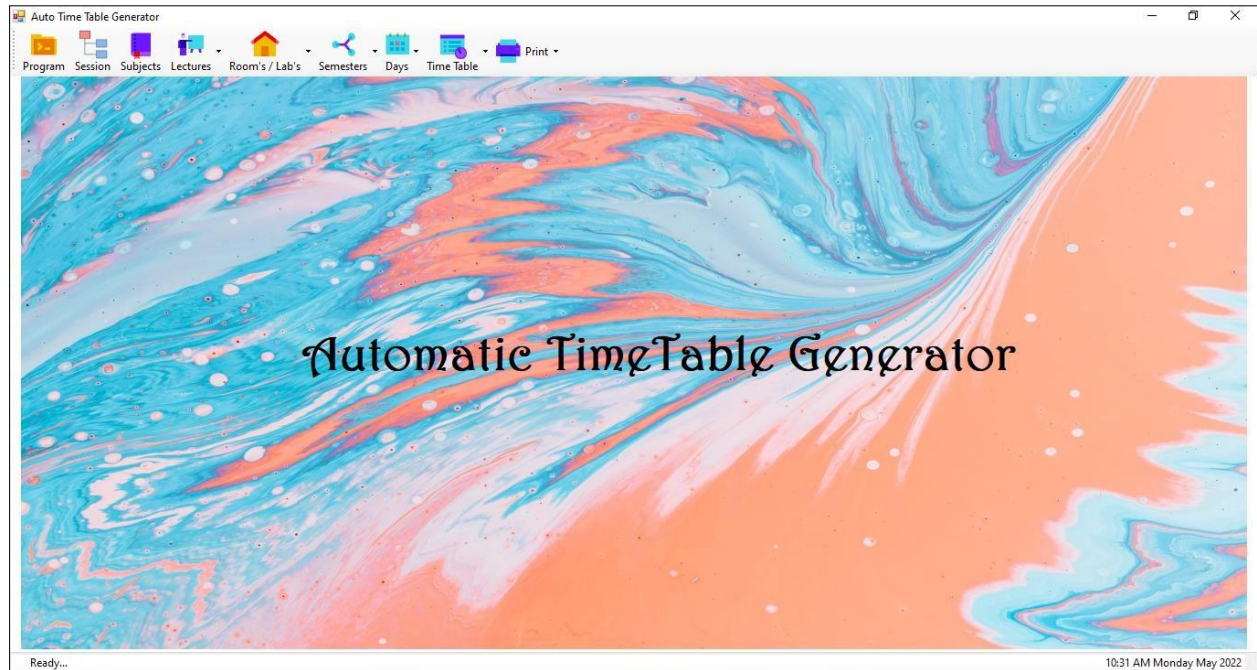


Fig1) Dashboard (Homepage) Page

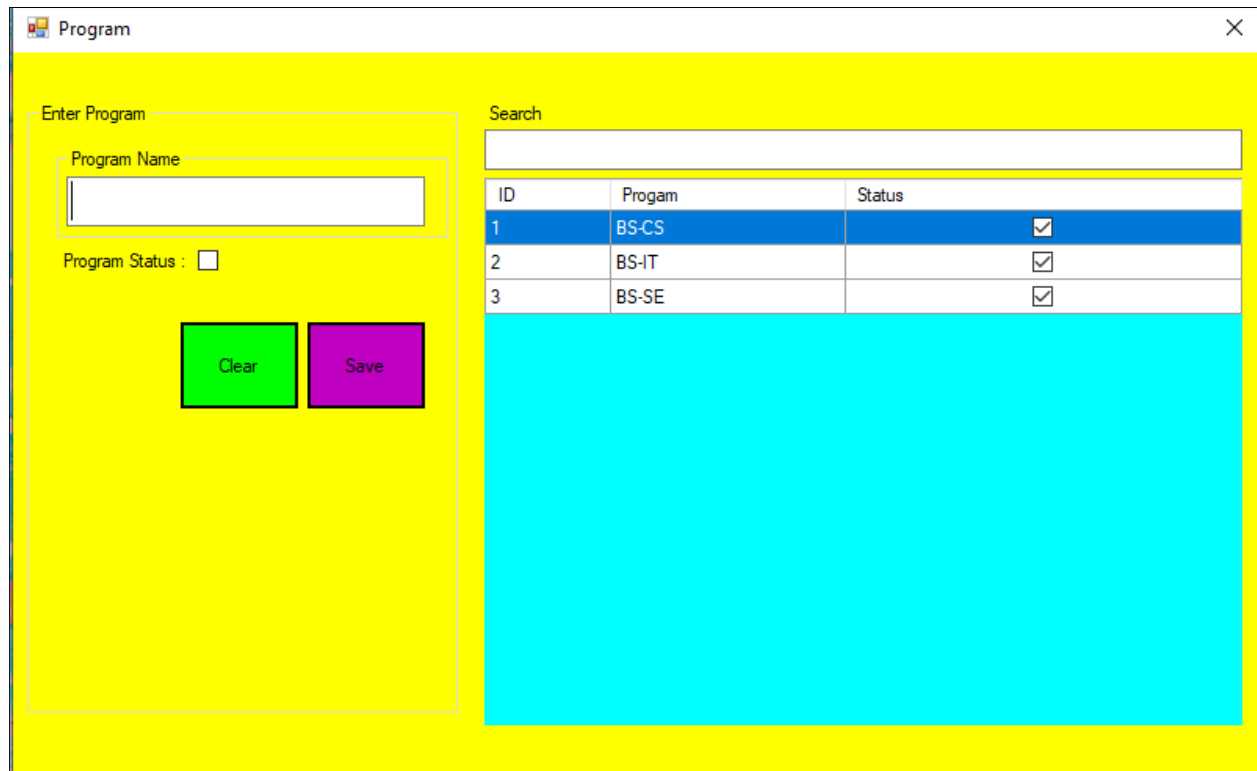


Fig2) Program Page

Sessions

Enter Session

Session Title :

-

Session Status :

☐

Clear

Save

Search

ID	Title	Status
1	2020-2024	<input checked="" type="checkbox"/>
2	2021-2025	<input checked="" type="checkbox"/>
3	2022-2026	<input checked="" type="checkbox"/>
4	2019-2023	<input checked="" type="checkbox"/>
5	2018-2022	<input checked="" type="checkbox"/>
6	2017-2021	<input checked="" type="checkbox"/>
7	2016-2020	<input checked="" type="checkbox"/>
8	2023-2017	<input checked="" type="checkbox"/>

Fig3) Session Page

Subjects

Enter Subject Details

Subject Title

Select Type

---Select---

Enter CrHrs

1

Subject Status :

☐

Clear

Save

Search

ID	Subject	CrHrs	Type
2	Information and Communication Technologies(CS)	3	Non Practical
3	Calculus and Analytical Geometry	3	Non Practical
4	Basic Electronics	3	Non Practical
5	English Composition & Comprehension(ED)	3	Non Practical
6	Programming Fundamentals(CS)	3	Practical
7	Islamic Studies(CS)	2	Non Practical
8	Object Oriented Programming	3	Practical
9	Discrete Structure	3	Non Practical
10	Technical and Business Writing	3	Non Practical
11	Pakistan Studies	2	Non Practical
12	Digital Logic Design(CS)	3	Practical
13	Probability and Statistics	3	Non Practical
14	Data Structure	3	Non Practical
15	Communication Skills	3	Non Practical
16	Linear Algebra	3	Non Practical
17	Operation Systems	3	Non Practical

Fig4) Subjects Page

All Lecturers

Enter Lecturer Details

Lecturer Name

Lecturer Contact No

Lecturer Status : ☐

Clear
Save

Search

ID	Lecturer	Contact No	Status
1	DR.AMJID	0314-6562662	<input checked="" type="checkbox"/>
2	MR.HAMZA	0314-9595956	<input checked="" type="checkbox"/>
3	MR.JUNAID	0314-9898956	<input checked="" type="checkbox"/>
4	MR.SALMAN	0314-6565658	<input checked="" type="checkbox"/>
5	DR.WAZIR KHAN	0315-6565425	<input checked="" type="checkbox"/>
6	MR.IFTIKHAR	0345-9575845	<input checked="" type="checkbox"/>
7	MR.ZIA UR REHMAN	0335-6565252	<input checked="" type="checkbox"/>
8	MR.SABIR SHAH	0345-8998585	<input checked="" type="checkbox"/>
9	MR.FAIZAN	0335-8584558	<input checked="" type="checkbox"/>
10	DR.YASIR	0312-6585425	<input checked="" type="checkbox"/>
11	MR.MUHAMMAD KHAN	0312-8545251	<input checked="" type="checkbox"/>
12	MR.FARMAN	0316-5525251	<input checked="" type="checkbox"/>
13	MR.ALI KHAN	0316-4542115	<input checked="" type="checkbox"/>
14	DR.ASLAM	0311-6562213	<input checked="" type="checkbox"/>
15	DR.IMRAN ALI	0314-3076781	<input checked="" type="checkbox"/>

Fig5) All Lecturers Page

Lectures Subjects

Lectures Subjects

Select Teacher/Lecture

---Select---

Select Subject

---Select---

Clear

Save

Search

Subject Title	Lecture	Course
Information and Communication Technologies(CS...)	DR.AMJID	Information and
Programming Fundamentals(CS)(DR.AMJID)	DR.AMJID	Programming
Computer Assembly Language Programming(DR....)	DR.AMJID	Computer Ass
Cyber Security(DR.AMJID)	DR.AMJID	Cyber Securit
Islamic Studies(CS)(MR.HAMZA)	MR.HAMZA	Islamic Studie
Calculus and Analytical Geometry(MR.JUNAID)	MR.JUNAID	Calculus and
Discrete Structure(MR.JUNAID)	MR.JUNAID	Discrete Struc
Linear Algebra(MR.JUNAID)	MR.JUNAID	Linear Algebr.
Multivariate Calculus (MR.JUNAID)	MR.JUNAID	Multivariate C
Basic Electronics(MR.SALMAN)	MR.SALMAN	Basic Electron
Data Structure(MR.SALMAN)	MR.SALMAN	Data Structur
Compiler Construction(MR.SALMAN)	MR.SALMAN	Compiler Con
English Composition & Comprehension(ED)(DR.W...)	DR.WAZIR KHAN	English Comp
Technical and Business Writing(DR.WAZIR KHA...)	DR.WAZIR KHAN	Technical an
Communication Skills(MR.IFTIKHAR)	MR.IFTIKHAR	Communicatio
Professional Practices(MR.IFTIKHAR)	MR.IFTIKHAR	Professional P

Fig6) Lecturer's Subject Page

Rooms

×

Enter Room Details

Room No

Room Capacity

Room Status : ☐

Clear

Save

Search

ID	Room	Capacity	Status
1	ROOM 1	60	<input type="checkbox"/>
2	ROOM 2	70	<input type="checkbox"/>
3	ROOM 3	60	<input type="checkbox"/>
4	ROOM 4	40	<input type="checkbox"/>
5	ROOM 5	65	<input type="checkbox"/>
6	ROOM 6	70	<input type="checkbox"/>
7	ROOM 7	70	<input type="checkbox"/>
8	ROOM 8	100	<input type="checkbox"/>
9	ROOM 9	100	<input checked="" type="checkbox"/>
10	ROOM 10	100	<input checked="" type="checkbox"/>

Fig7) Rooms Page

Labs

×

Enter Lab Details

Lab No

Lab Capacity

Lab Status : ☐

Clear

Save

Search

ID	Lab	Capacity	Status
1	LAB 1	50	<input type="checkbox"/>
2	LAB 2	70	<input checked="" type="checkbox"/>
3	LAB 3	150	<input checked="" type="checkbox"/>
4	LAB 4	80	<input type="checkbox"/>
5	LAB 5	70	<input type="checkbox"/>

Fig8) Labs Page

Semester

Enter Semester

Semester Name

Semester Status : ☐

Clear

Save

Search

ID	Semester	Status
1	1st Semester	<input checked="" type="checkbox"/>
2	2nd Semester	<input checked="" type="checkbox"/>
3	3rd Semester	<input checked="" type="checkbox"/>
4	4th Semester	<input checked="" type="checkbox"/>
5	5th Semester	<input checked="" type="checkbox"/>
6	6th Semester	<input checked="" type="checkbox"/>
7	7th Semester	<input checked="" type="checkbox"/>
8	8th Semester	<input checked="" type="checkbox"/>

Fig9) New Semester Page

Semester Sections

Semester Section Registration

Section Title

Select Semester

---Select---

Enter Capacity

Clear

Save

Search

Section	Semester	Status
Section A	2020-2024 BS-CS 1ST SEMESTER	<input checked="" type="checkbox"/>
Section B	2020-2024 BS-CS 1ST SEMESTER	<input checked="" type="checkbox"/>
Section A	2018-2022 BS-IT 5TH SEMESTER	<input checked="" type="checkbox"/>
Section B	2018-2022 BS-IT 5TH SEMESTER	<input checked="" type="checkbox"/>

Fig10) Semester Sections Page

Program Semesters Setting

Enter Program Semesters Details

Program Semester Title

Select Session

Select Program

Select Semester

Enter Capacity

Program Semester Status : ☐

Clear Save

Search

ID	Title	Capacity	Status
1	2020-2024 BS-CS 1ST SEMESTER	50	<input checked="" type="checkbox"/>
2	2020-2024 BS-CS 2ND SEMESTER	50	<input type="checkbox"/>
3	2019-2023 BS-CS 3RD SEMESTER	40	<input checked="" type="checkbox"/>
4	2019-2023 BS-CS 4TH SEMESTER	50	<input type="checkbox"/>
5	2018-2022 BS-CS 5TH SEMESTER	50	<input checked="" type="checkbox"/>
6	2018-2022 BS-CS 6TH SEMESTER	50	<input type="checkbox"/>
7	2017-2021 BS-CS 7TH SEMESTER	50	<input checked="" type="checkbox"/>
8	2017-2021 BS-CS 8TH SEMESTER	50	<input type="checkbox"/>
9	2020-2024 BS-IT 1ST SEMESTER	50	<input checked="" type="checkbox"/>
10	2020-2024 BS-IT 2ND SEMESTER	50	<input type="checkbox"/>
11	2019-2023 BS-IT 3RD SEMESTER	50	<input checked="" type="checkbox"/>
12	2019-2023 BS-IT 4TH SEMESTER	50	<input type="checkbox"/>
13	2018-2022 BS-IT 5TH SEMESTER	50	<input checked="" type="checkbox"/>
14	2018-2022 BS-IT 6TH SEMESTER	50	<input type="checkbox"/>
15	2017-2021 BS-IT 7TH SEMESTER	50	<input checked="" type="checkbox"/>
16	2017-2021 BS-IT 8TH SEMESTER	50	<input type="checkbox"/>
17	2020-2024 BS-SE 1ST SEMESTER	50	<input checked="" type="checkbox"/>
1009	2020-2024 BS-SE 2ND SEMESTER	50	<input type="checkbox"/>
1010	2019-2023 BS-SE 3RD SEMESTER	50	<input checked="" type="checkbox"/>

Fig11) Program Semesters Setting Page

Semester Subjects

Semester Subject Registration

Subject Title

Select Semester

Select Subject

Clear Save

Search

Program	Semester	Subject
BS-CS	2020-2024 BS-CS 1ST SE...	Information and Communication Technolc
BS-CS	2020-2024 BS-CS 1ST SE...	Programming Fundamentals(CS)(DR.AMJ
BS-CS	2020-2024 BS-CS 1ST SE...	Basic Electronics(MR.SALMAN)
BS-CS	2020-2024 BS-CS 1ST SE...	English Composition & Comprehension(EC
BS-CS	2020-2024 BS-CS 1ST SE...	Calculus and Analytical Geometry(MR.JU
BS-CS	2019-2023 BS-CS 3RD SE...	Web Fundamentals(MR.FARMAN)
BS-CS	2019-2023 BS-CS 3RD SE...	Operating Systems(MR.SABIR SHAH)
BS-CS	2019-2023 BS-CS 3RD SE...	Digital Logic Design(CS)(DR.YASIR)
BS-CS	2019-2023 BS-CS 3RD SE...	Communication Skills(MR.IFTIKHAR)
BS-CS	2019-2023 BS-CS 3RD SE...	Multivariate Calculus (MR.JUNAID)
BS-CS	2019-2023 BS-CS 3RD SE...	Data Structure(MR.SALMAN)
BS-CS	2018-2022 BS-CS 5TH SE...	Introduction to Database System(MR.ZIA
BS-CS	2018-2022 BS-CS 5TH SE...	Theory of Automate (MR.FAIZAN)
BS-CS	2018-2022 BS-CS 5TH SE...	Software Engineering-II(DR.ASLAM)
BS-CS	2018-2022 BS-CS 5TH SE...	Data Communication(DR.SADRA REHM.
BS-CS	2018-2022 BS-CS 5TH SE...	Human Resource Management (DR.ASA

Fig12) Semester Subjects Page

Days

Enter Day

Day Name

Day Status : ☐

Clear

Save

Search

ID	Day	Status
1	MONDAY	<input checked="" type="checkbox"/>
2	TUESDAY	<input checked="" type="checkbox"/>
3	WEDNESDAY	<input checked="" type="checkbox"/>
4	THURSDAY	<input checked="" type="checkbox"/>
5	FRIDAY	<input checked="" type="checkbox"/>
6	SATURDAY	<input type="checkbox"/>
7	SUNDAY	<input type="checkbox"/>

Fig13) Days Page

Day Time Slot Form

Time Slot Per Day

Select Day

---Select---

Start Time

08:00 AM

End Time

04:00 PM

Enter Number of Time Slots Per Day

---Select---

Day Status : ☒

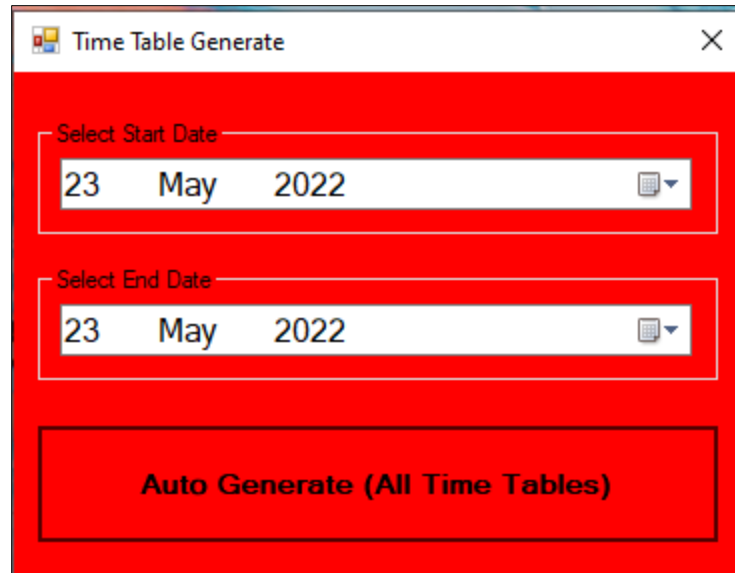
Clear

Save

Search

S No	Day	Slot Title	Start Time
1	MONDAY	09:30 AM-11:00 AM	09:30:00
2	MONDAY	11:00 AM-12:30 PM	11:00:00
3	MONDAY	12:30 PM-02:00 PM	12:30:00
4	MONDAY	02:00 PM-03:30 PM	14:00:00
5	MONDAY	03:30 PM-05:00 PM	15:30:00
6	TUESDAY	09:30 AM-11:00 AM	09:30:00
7	TUESDAY	11:00 AM-12:30 PM	11:00:00
8	TUESDAY	12:30 PM-02:00 PM	12:30:00
9	TUESDAY	02:00 PM-03:30 PM	14:00:00
10	TUESDAY	03:30 PM-05:00 PM	15:30:00
11	WEDNESDAY	09:30 AM-11:00 AM	09:30:00
12	WEDNESDAY	11:00 AM-12:30 PM	11:00:00
13	WEDNESDAY	12:30 PM-02:00 PM	12:30:00
14	WEDNESDAY	02:00 PM-03:30 PM	14:00:00
15	WEDNESDAY	03:30 PM-05:00 PM	15:30:00
16	THURSDAY	09:30 AM-11:00 AM	09:30:00

Fig14) Day Time Slot Form Page



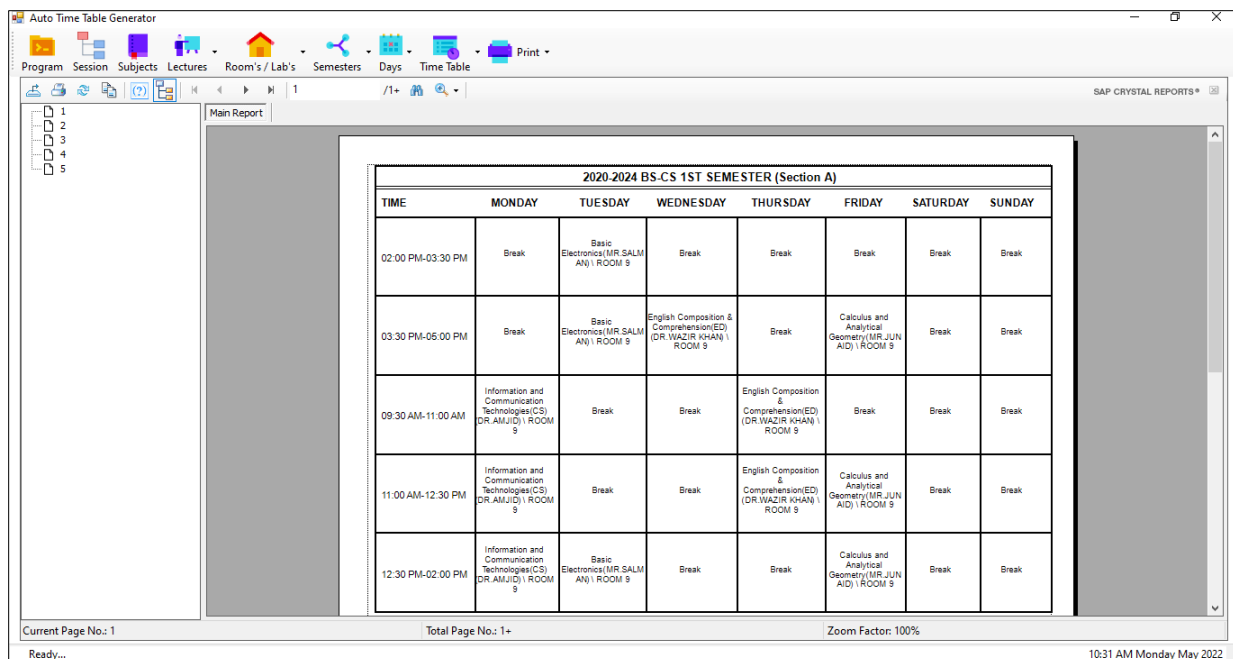
Time Table Generate

Select Start Date: 23 May 2022

Select End Date: 23 May 2022

Auto Generate (All Time Tables)

Fig15) Time Table Generate Page



Auto Time Table Generator

Program Session Subjects Lectures Room's / Lab's Semesters Days Time Table Print

SAP CRYSTAL REPORTS

Main Report

2020-2024 BS-CS 1ST SEMESTER (Section A)							
TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
02:00 PM-03:30 PM	Break	Basic Electronics(MR.SALMAN) ROOM 9	Break	Break	Break	Break	Break
03:30 PM-05:00 PM	Break	Basic Electronics(MR.SALMAN) ROOM 9	English Composition & Comprehension(ED) (DR.WAZIR KHAN) ROOM 9	Break	Calculus and Analytical Geometry(MR.JUN AID) ROOM 9	Break	Break
09:30 AM-11:00 AM	Information and Communication Technologies(CS) (DR.AMJID) ROOM 9	Break	Break	English Composition & Comprehension(ED) (DR.WAZIR KHAN) ROOM 9	Break	Break	Break
11:00 AM-12:30 PM	Information and Communication Technologies(CS) (DR.AMJID) ROOM 9	Break	Break	English Composition & Comprehension(ED) (DR.WAZIR KHAN) ROOM 9	Calculus and Analytical Geometry(MR.JUN AID) ROOM 9	Break	Break
12:30 PM-02:00 PM	Information and Communication Technologies(CS) (DR.AMJID) ROOM 9	Basic Electronics(MR.SALMAN) ROOM 9	Break	Break	Calculus and Analytical Geometry(MR.JUN AID) ROOM 9	Break	Break

Current Page No.: 1 Total Page No.: 1+ Zoom Factor: 100%

Ready... 10:31 AM Monday May 2022

Fig16) Print All Time Table Page

Auto Time Table Generator

Program Session Subjects Lectures Room's / Lab's Semesters Days Time Table Print

SAP CRYSTAL REPORTS*

Main Report

DR.AMJID- Time Table

TIME	MONDAY	TUE SDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
02:00 PM-03:30 PM	2020-2024 BS-CS 1ST SEMESTER (Section B) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)	Break	Break	Break	Break	Break
03:30 PM-05:00 PM	2020-2024 BS-CS 1ST SEMESTER (Section B) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)	Break	Break	Break	Break	Break
09:30 AM-11:00 AM	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section B) Programming Fundamentals(CS) (DR.AMJID) (LAB 2)	Break	Break	Break	Break
11:00 AM-12:30 PM	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section B) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)	Break	Break	Break	Break
	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID) (ROOM 9)					

Current Page No.: 1 Total Page No.: 1+ Zoom Factor: 100%

Ready... 10:31 AM Monday May 2022

Fig17) Print Day Wise Time Table Page

Auto Time Table Generator

Program Session Subjects Lectures Room's / Lab's Semesters Days Time Table Print

SAP CRYSTAL REPORTS*

Main Report

5/23/2022

	09:30 AM-11:00 AM	11:00 AM-12:30 PM	12:30 PM-02:00 PM	02:00 PM-03:30 PM	03:30 PM-05:00 PM
	FRIDAY				
ROOM 9	2020-2024 BS-CS 1ST SEMESTER (Section B) English Composition & Composition II(DR.YASIR KHAN)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section A) Calculus and Analytical Geometry(IIR.JUNAID)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section A) Calculus and Analytical Geometry(IIR.JUNAID)(ROOM 9)	Break	2020-2024 BS-CS 1ST SEMESTER (Section A) Calculus and Analytical Geometry(IIR.JUNAID)(ROOM 9)
ROOM 10	2019-2022 BS-CS 5TH SEMESTER Data Structures(IIR.SALMAN)(ROOM 10)	2019-2022 BS-CS 6TH SEMESTER Theory of Automata(IIR.FAZAN)(ROOM 10)(ROOM 10)	Break	2019-2022 BS-CS 6TH SEMESTER Theory of Automata(IIR.FAZAN)(ROOM 10)	2019-2022 BS-CS 6TH SEMESTER Theory of Automata(IIR.FAZAN)(ROOM 10)
LAB 2	2019-2022 BS-CS 6TH SEMESTER Data Communication(DR.SADIA REHMAN)(LAB 2)(LAB 2)	Break	2019-2022 BS-CS 6TH SEMESTER Human Resource Management (DR.ASAD KHAN)(LAB 2)	2019-2022 BS-CS 6TH SEMESTER Human Resource Management (DR.ASAD KHAN)(LAB 2)	2019-2022 BS-CS 6TH SEMESTER Human Resource Management (DR.ASAD KHAN)(LAB 2)
LAB 3	Break	Break	Break	Break	Break
	MONDAY				
ROOM 9	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section A) Information and Communication Technologies(CS) (DR.AMJID)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section B) Information and Communication Technologies(CS) (DR.AMJID)(ROOM 9)	2020-2024 BS-CS 1ST SEMESTER (Section B) Information and Communication Technologies(CS) (DR.AMJID)(ROOM 9)

Current Page No.: 1 Total Page No.: 1+ Zoom Factor: 100%

Ready... 10:31 AM Monday May 2022

Fig18) Print Teacher Wise Time Table Page

CONCLUSION

The intention of the project is to generate a time-table that can schedule automatically. Timetabling problem being the hard combinatorial problem that is would take more than just the application of only one principle. This incorporates a number of techniques, aimed to improve the efficiency of scheduling. It also, addresses the important hard constraint of clashes between the availability of teachers. The non-rigid soft constraints i.e. optimization objectives are also effectively handled. Thus, through the process of automation of the time-table problem, many an-hours of creating an effective timetable have been reduced eventually. By considering all these constraints effectively and efficiently a timetable can be generated which eventually can be used for a specific class of an institution and may also be saved for future. This complete process of storing data of staff and subjects and retrieving them by timetable generation part uses database and thus generates a well scheduled timetable by considering the above constraints. It is a complicated task that to handle many Faculty's and allocating subjects for them at a time physically. So our proposed system will help to overcome this disadvantage. Thus we can produce timetable for any number of courses and multiple semesters. This system will help to create dynamic pages so that for implementing such a system we can make use of the different tools are widely applicable and free to use also.

REFERENCES

- [1] M. Doulaty, M. R. Feizi Derakhshi, and M. Abdi, “Timetabling: A State-of-the-Art Evolutionary Approach”, International Journal of Machine Learning and Computing, Vol. 3, No. 3, June 2013.

- [2] Anirudha Nanda, Manisha P. Pai, and Abhijeet Gole, “An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach”, International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012

- [3] Dilip Datta, Kalyanmoy Deb, Carlos M. Fonseca, “Solving Class Timetabling Problem of IIT Kanpur using Multi-Objective Evolutionary Algorithm”.KanGAL 2005.

- [4] Anuja Chowdhary, Priyanka Kakde, Shruti Dhoke, Sonali Ingle,Rupal Rushiya, Dinesh Gawande, 'Time table Generation System”, International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, February- 2014.

- [5] Mughda Kishor Patil, Rakhe Shruti Subodh, Prachi Ashok Pawar, Naveena Narendra Singh Turkar, “Web Application for Automatic Time Table Generation”, International Journal of current Engineering and Technology, E-ISSN 2277-4106, P-ISSN 2347-5161.