Discovering Knowledge

**COURSE: CSL 113**
**COMPUTER PROGRAMMING**

# PROJECT REPORT

**CLASS: BSE – 1C (FALL - 2024)**

## Student Life Manger

## Group Members

| Student Name | Enrollment# |
|---|---|
| Huzaifa | 02-131242-059 |
| Muhammad Bilal | 02-131242-057 |

## Submitted to:

Course Instructor: Engr. Muhammad Faisal
Lab Instructor: Engr. Noor us Sabah

**Department of Software Engineering**
**BAHRIA UNIVERSITY KARACHI CAMPUS**

# TABLE OF CONTENTS

## 1. Introduction

While the university's CMS and LMS assists in taking attendance, managing class schedules, and courses, they do nothing to address the daily challenges faced by students in the management of tasks, study goals, and good habits, hence this project aims at helping bridge that gap by giving a tailored productivity tool.

## 2. Problem Statement

Lms lacks the features such as tasks prioritizing, tracking habit and offline usability.

Hence, students require a bridge to fill this gap in terms of both academic and personal productivity issues.

## 3. Proposed Solution

### 3.1. Features of the project

- Task Manager (add, Delete & track tasks, mark them completed).
- Study Planner (set goals, track progress).
- Habit Tracker (monitor activities like exercise, reading).
- Offline Functionality (ensure accessibility without internet)

### 3.2. Methodology

- **Planning:** Identify essential productivity features.

- **Learning**: Identify and Learn essential concepts,skills and tools required for this project.

- **Designing:** Use simple and intuitive interfaces.

- **Development:** Create the system using **C# and WinForms**, with local storage (SQLite or text files).

- **Testing:** Validate features and fix usability issues.

### 3.3. Technologies

- **Programming Language: C#**
  It is easy to learn and very suitable for developing Windows desktop applications. It also supports object-oriented programming, hence keeping the code clean and reusable.
- **Framework: WinForms**
  Simple, beginner-friendly UI framework to create a basic and fully functional desktop interface for the application.
- **Database: File Handling**
  Light, serverless, and easy to set up. Data stored locally so it works offline as well.
- **Data Access:**
  Simplify process of connecting to the Files and managing data (add, update, delete operations).
- **Charts: (Tentative)**
  Easy charting to create a view for progress, such as completion of tasks or tracking habits.
- **Development Environment: Visual Studio**
  The best IDE for C# development with debugging and tool support.
- **Packaging: Inno Setup / Toolset**
  Easy-to-use tools to create a Windows installer to distribute the app.

## 4. Project Scope

1. **Task Manager:**
   - Add, update, and  delete tasks.
   - Track task completion and due dates.
2. **Study Planner:**
   - Set and track study goals and deadlines.
3. **Habit Tracker:**
   - Track and visualize personal habits (e.g., exercise, reading).
4. **Offline Functionality:**
   - All features will work offline, with no internet required.

## 5. Module Distribution

| 6. **Module** | **Description** | **Assigned To** |
|---|---|---|
| **Task Manager** | Enables users to add, edit, prioritize, and track tasks or assignments. | Huzaifa |
| **Study Planner** | Allows users to set study goals and track progress. | Muhammad Bilal |
| **Habit Tracker** | Tracks personal habits like exercise or reading, Includes progress charts. | Huzaifa |
| **Database Integration** | Handles data storage, retrieval, and updates using SQLite or file handling. | Huzaifa |
| **UI/UX Design** | Designs the graphical interface for the application, ensuring a user-friendly experience. | Muhammad Bilal |

## 7. Code

## Task Manager:

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

namespace takmanger1._0
{
    public partial class Form1 : Form
    {
        List<string> tasks = new List<string>();
        // Declare the list to store tasks
        // we have declared a list which is sort of dynamic array,
we can use .Add and .Remove/RemoveAt() Methods to add and remove
elements of same data type.
        public Form1()
        {
            InitializeComponent();
        }


        private void SaveTasksToFile()
        {
            // save file to app directory
            string filePath =
Path.Combine(Application.StartupPath, "tasks.txt");

            using (StreamWriter writer = new
StreamWriter(filePath))
            {
                foreach (var task in tasks)
                {
                    writer.WriteLine(task);  // Save each task as
a new line
                }
            }
        }




        private void Form1_Load(object sender, EventArgs e)
        {
            // this disable user from selecting past date, it is
set one time at the start of from
            dateTimePickerDeadline.MinDate = DateTime.Now.Date;

            LoadTasksFromFile();  // Load tasks from the file


        }
        private void textBox1_TextChanged(object sender, EventArgs
e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (Tasks_List.SelectedIndex >= 0)
            {
                // get the task
                string selectedTask =
Tasks_List.SelectedItem.ToString();

                // delete from tasks list
                int index = Tasks_List.SelectedIndex;
                tasks.RemoveAt(index);
                // also from ListBox
                Tasks_List.Items.RemoveAt(index);

                SaveTasksToFile(); // save task to file

            }
            else
            {
                MessageBox.Show("Please select the task to delete
it");
            }

        }


        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void btnAddTask_Click(object sender, EventArgs e)
        {
            // declare two variables to get task name and deadline
            // if task name is not empty then add it to
Tasks_List<> and ListBox so that it appears on UI with task name
and deadline

            string taskName = txtTaskName.Text; // this will hold
the task name entered by user, txtTaskName is the name of the
TextBox on the Form
```

```
            string deadline =
dateTimePickerDeadline.Value.ToString("dd/MM"); // this line
stores the value from dateTimePicker and converts into short date
format(y/m/d) in a deadline variable
            string task = $"{taskName} (Due: {deadline})";

            if (!string.IsNullOrEmpty(taskName))
            {


                if (tasks.Contains(task))
                {
                    MessageBox.Show("This task already exists");
                    return; // it will exit the method
                }

                tasks.Add(task);
                Tasks_List.Items.Add(task);




                txtTaskName.Clear();

                SaveTasksToFile(); // save task to file

            }
            else
            {
                MessageBox.Show("Please a enter a task name");
            }





        }


        private void listBoxTasks_SelectedIndexChanged(object
sender, EventArgs e)
        {

        }

        private void btnMarkAsDone_Click(object sender, EventArgs
e)
        {
            // if a task is selected
            if(Tasks_List.SelectedIndex >= 0)
            {
                // get the task
                string selectedTask =
Tasks_List.SelectedItem.ToString();
                // only add "[Done]" if its not already marked
                if(!selectedTask.Contains("[Done]"))
                {
                    // add 'Done'
                    string completedTask = selectedTask +
"[Done]";
                    // store it in tasks list
                    int index = Tasks_List.SelectedIndex;
                    tasks[index] = completedTask;
                    // also update ListBox
                    Tasks_List.Items[index] = completedTask;


                }
                else
                {
                    MessageBox.Show("This task is already marked
as Done!");

                }


            }
            else
            {
                MessageBox.Show("Please select the task to mark as
done");
            }

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }



        // Load tasks from the file (tasks.txt)

        private void LoadTasksFromFile()
        {
            string filePath = 
Path.Combine(Application.StartupPath, "tasks.txt");

            if (File.Exists(filePath))
            {
                string[] tasksFromFile = 
File.ReadAllLines("tasks.txt");
                foreach (string task in tasksFromFile)
                {
                    tasks.Add(task);  // Add each task back into 
the list
                    Tasks_List.Items.Add(task);  // Display the 
task in ListBox
                }
            }
        }

        private void Form1_FormClosing(object sender, 
FormClosingEventArgs e)
        {
            SaveTasksToFile();
        }
    }
}
```

## Study Planner:

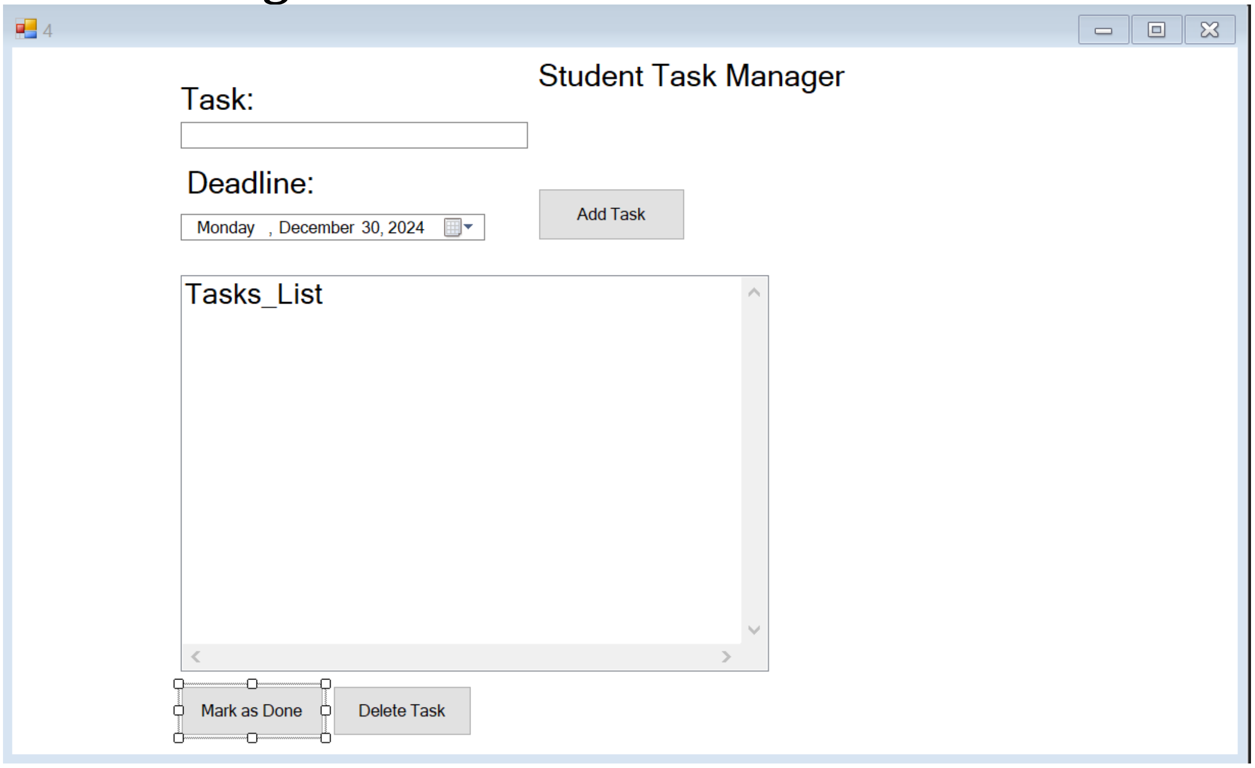
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace study_planner
{
    public partial class Form1 : Form
    {
        // here we declare global varaibles that will by
accessible by every method
        // we have used lists instead of arrays because lists can
dynamically grow and shrink

        List<string> studyGoals = new List<string>();  // List of
study goal names
        List<string> deadlines = new List<string>();  // List of
deadlines (as strings)
        List<int> totalHours = new List<int>();       // Total
hours planned for each goal
        List<int> hoursCompleted = new List<int>();   // Hours
completed for each goal
        int counter = 0; // counter to keep track of the number of
goals added
        public Form1()
        {
            InitializeComponent();
        }

        private void monthCalendar1_DateChanged(object sender,
DateRangeEventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            dateTimePickerStudyDeadline.MinDate = DateTime.Now;
            string filePath =
Path.Combine(Application.StartupPath, "study_goals.txt");

            if (File.Exists(filePath))
            {
                try
                {
                    string[] lines = File.ReadAllLines(filePath);
                    foreach (string line in lines)
                    {
                        string[] parts = line.Split('|');
                        studyGoals.Add(parts[0]);
                        deadlines.Add(parts[1]);
                        totalHours.Add(int.Parse(parts[2]));
                        hoursCompleted.Add(int.Parse(parts[3]));

                        counter++;
                        listBoxStudyGoals.Items.Add($"{counter}.
{parts[0]} (Due : {parts[1]}) - {parts[3]}/{parts[2]} hrs");
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Error loading study goals
from file: {ex.Message}");
                }
            }
            else
            {
                MessageBox.Show("No previous goals found.");
            }
        }



        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void dateTimePicker1_ValueChanged(object sender,
EventArgs e)
        {

        }

        private void label1_Click_1(object sender, EventArgs e)
        {

        }

        private void progressStudy_Click(object sender, EventArgs
e)
        {

        }

        private void label1_Click_2(object sender, EventArgs e)
        {

        }


        private void btnAddGoal_Click(object sender, EventArgs e)
        {
            string goalName = txtStudyGoal.Text; // get study goal
            string deadline =
dateTimePickerStudyDeadline.Value.ToString("dd/MM");
            int plannedHours = (int)numStudyHours.Value; // here
(int) converts the value to int and it truncates float values
            // this ensure that only valid goal name and hours are
added
            if (!string.IsNullOrEmpty(goalName) && plannedHours >
0)
            {
                // save records in lists
                studyGoals.Add(goalName);
                deadlines.Add(deadline);
                totalHours.Add(plannedHours);
                hoursCompleted.Add(0); // starts with 0 hours
completed
                counter++; // increment counter
                // add goal to listbox
                listBoxStudyGoals.Items.Add($"{counter}.
{goalName} (Due : {deadline}) - 0/{plannedHours} hrs");

                // clear textboxes
                txtStudyGoal.Clear();
                numStudyHours.Value = 0;

                // clear memory if goals reach the limit
                if (studyGoals.Count > 5)
                {
                    MessageBox.Show("You have reached the maximum
limit of 5 goals. Clearing all goals...");

                    // Clear all goals
                    studyGoals.Clear();
                    deadlines.Clear();
                    totalHours.Clear();
                    hoursCompleted.Clear();
                    listBoxStudyGoals.Items.Clear();
                    counter = 0;
                    progressBarStudy.Value = 0;
                }
            }
            else
            {
                MessageBox.Show("Please enter a valid goal and
total hours");
            }
        }
        private void btnUpdateProgress_Click(object sender, 
EventArgs e)
        {

            // if no item is slected the value of slected index 
will be -1
            if (listBoxStudyGoals.SelectedIndex >= 0) // check if 
an goal is selected
            {
                // The index is used to find the corresponding 
data in the parallel lists (studyGoals, deadlines, totalHours, 
hoursCompleted).

                int index = listBoxStudyGoals.SelectedIndex; // 
get selected goal's index
                int completedHours = (int)totalHoursStudied.Value; 
// get additional hours,numStudyHours.Value is a decimal value.

                if (completedHours >= 0 && completedHours <= 
totalHours[index])
                {
                    hoursCompleted[index] = completedHours; // add 
completed hours to total hours completed
                    listBoxStudyGoals.Items[index] = $"{index + 
1}. {studyGoals[index]} (Due : {deadlines[index]}) - 
{hoursCompleted[index]}/{totalHours[index]} hrs"; // update 
listbox item
                    // update progress bar
                    progressBarStudy.Value = 
(int)((double)hoursCompleted[index] / totalHours[index] * 100); // 
calculate percentage of hours completed

                    // clear total hours studied
                    totalHoursStudied.Value = 0;
                }
                else
                {
                    MessageBox.Show("Please enter number of hours 
between 0 and planned hours");
                }

            }
            else
            {
                MessageBox.Show("Please select a goal to update 
progress");
            }
        }
        private void numStudyHours_ValueChanged(object sender,
EventArgs e)
        {

        }

        private void lblHours_TextChanged(object sender, EventArgs
e)
        {

        }

        private void numericUpDown1_ValueChanged(object sender,
EventArgs e)
        {

        }

        private void listBoxStudyGoals_SelectedIndexChanged(object
sender, EventArgs e)
        {
            // it change progressbar accordingly
            if(listBoxStudyGoals.SelectedIndex >= 0)
            {
                int index = listBoxStudyGoals.SelectedIndex;
                totalHoursStudied.Value = hoursCompleted[index];
                progressBarStudy.Value =
(int)((double)hoursCompleted[index] / totalHours[index] * 100);
            }
        }

        private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
        {
            // Application.StartupPath returns the path of the
executable file that started the application
            // example of Path.Combine("C:\\Program Files\\MyApp",
"study_goals.txt") would result in C:\Program
Files\MyApp\study_goals.txt
            string filePath =
Path.Combine(Application.StartupPath, "study_goals.txt");
            try
            {
                using (StreamWriter writer = new
StreamWriter(filePath))
                {
                    for (int i = 0; i < studyGoals.Count; i++)
                    {
                        string line =
$"{studyGoals[i]}|{deadlines[i]}|{totalHours[i]}|{hoursCompleted[i
]}";
                        writer.WriteLine(line);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Error saving study goals to
file: {ex.Message}");
            }
        }

        private void label5_Click(object sender, EventArgs e)
        {

        }
    }
    }
```

## Habit Tracker:

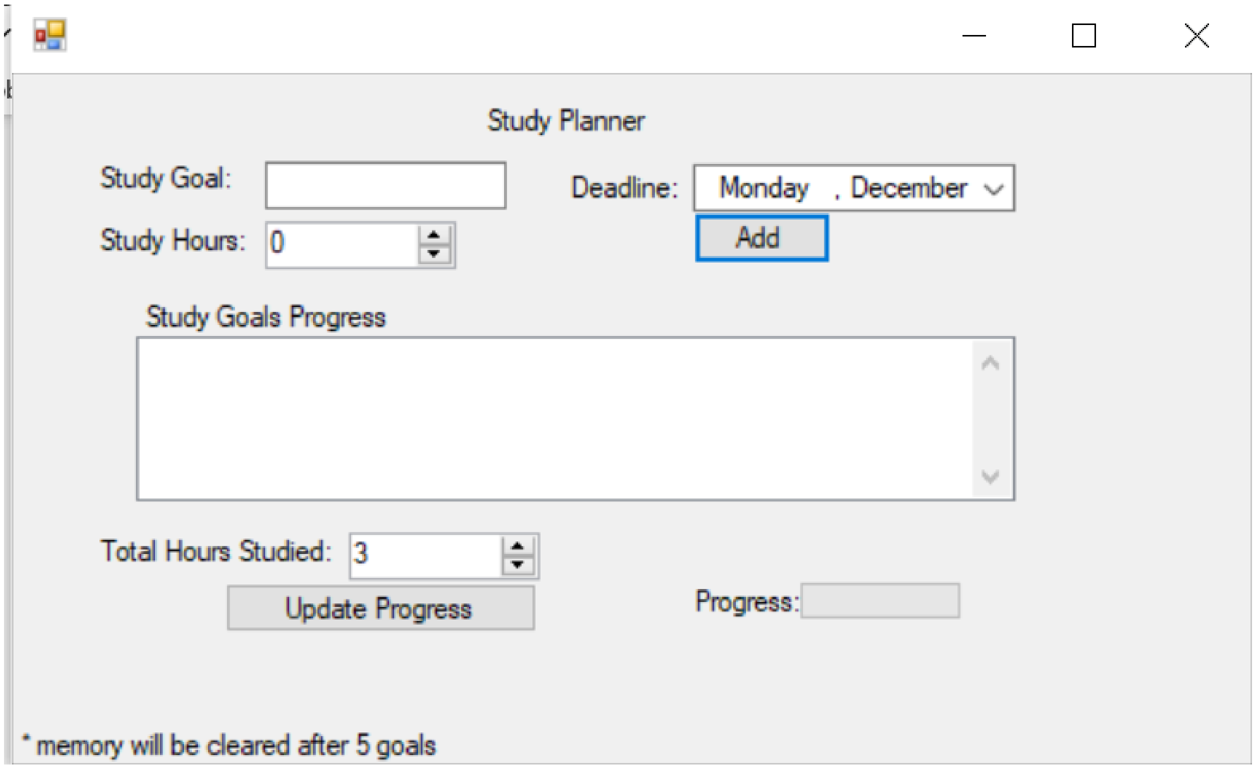
```
    using System;
    using System.Collections.Generic;
    using System.Globalization;
    using System.IO;
    using System.Linq;
    using System.Windows.Forms;
    using System.Windows.Forms.DataVisualization.Charting;

    namespace Habit_tracker
    {
        public partial class Form1 : Form
        {
            // Declare global variables
            List<string> habitNames = new List<string>();
            List<int> targetCounts = new List<int>();
            List<int> progressHistories = new List<int>();
            List<List<string>> progressDates = new
List<List<string>>();
            int counter = 0;
            public Form1()
            {
                InitializeComponent();
                this.FormClosing += new
FormClosingEventHandler(Form1_FormClosing);
            }

            private void Form1_Load(object sender, EventArgs e)
            {
                dateTimePicker1.MinDate = DateTime.Today;
                LoadData();
            }

            private void Form1_FormClosing(object sender,
FormClosingEventArgs e)
            {
                SaveData();
            }

            private void btnAddHabit_Click(object sender,
EventArgs e)
            {
                string habitName = txtHabitName.Text.Trim();
                int targetCount = (int)numTargetCount.Value;

                if (string.IsNullOrEmpty(habitName) || targetCount
<= 0)
                {
                    MessageBox.Show("Please enter a valid habit
name and target count.");
                    return;
                }

                habitNames.Add(habitName);
                targetCounts.Add(targetCount);
                progressHistories.Add(0);
                progressDates.Add(new List<string>());
                counter++;

                if (habitNames.Count > 5)
                {
                    MessageBox.Show("Limit reached! Clearing
memory...");
                    habitNames.Clear();
                    targetCounts.Clear();
                    progressHistories.Clear();
                    progressDates.Clear();
                    counter = 0;
                    listBoxHabits.Items.Clear();
                    return;
                }

                listBoxHabits.Items.Add($"{counter}. {habitName} - 
0/{targetCount} days");
                txtHabitName.Clear();
                numTargetCount.Value = 0;
            }

            private void btnUpdateProgress_Click(object sender, 
EventArgs e)
            {
                if (listBoxHabits.SelectedIndex >= 0)
                {
                    int index = listBoxHabits.SelectedIndex;
                    string selectedDate = 
dateTimePicker1.Value.ToString("dd/MM/yyyy");

                    if 
(progressDates[index].Contains(selectedDate))
                    {
                        MessageBox.Show("You have already made 
progress today.");
                        return;
                    }

                    progressHistories[index]++;
                    progressDates[index].Add(selectedDate);

                    listBoxHabits.Items[index] = $"{index + 1}. 
{habitNames[index]} - 
{progressHistories[index]}/{targetCounts[index]} days";

                    progressBarHabit.Value = 
Math.Min((int)((double)progressHistories[index] / 
targetCounts[index] * 100), 100);

                    UpdateProgressChart(index);
                }
                else
                {
                    MessageBox.Show("Please select a habit to 
update progress.");
                }
            }
            private void listBoxHabits_SelectedIndexChanged(object
sender, EventArgs e)
            {
                if (listBoxHabits.SelectedIndex >= 0)
                {

UpdateProgressChart(listBoxHabits.SelectedIndex);
                }
            }

            private void UpdateProgressChart(int index)
            {
                chartProgress.Series.Clear();
                Series series = new Series
                {
                    Name = "Progress Days",
                    Color = System.Drawing.Color.Blue,
                    ChartType = SeriesChartType.Point
                };
                chartProgress.Series.Add(series);

                foreach (var date in progressDates[index])
                {
                    if (DateTime.TryParseExact(date, "dd/MM/yyyy",
CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime
parsedDate))
                    {
                        series.Points.AddXY(parsedDate, 1);
                    }
                    else
                    {
                        MessageBox.Show($"Error parsing date:
{date}");
                    }
                }


chartProgress.ChartAreas[0].AxisX.LabelStyle.Format = "dd/MM";
                chartProgress.ChartAreas[0].AxisX.IntervalType =
DateTimeIntervalType.Days;
                chartProgress.ChartAreas[0].AxisX.Title = "Dates";
                chartProgress.ChartAreas[0].AxisY.Title =
"Progress (Count)";
                chartProgress.ChartAreas[0].AxisY.Maximum = 1;
                chartProgress.ChartAreas[0].AxisY.Minimum = 0;

                chartProgress.Invalidate();
            }
            private void SaveData()
            {
                string filePath = 
Path.Combine(Application.StartupPath, "habitData.txt");
                try
                {
                    using (StreamWriter writer = new 
StreamWriter(filePath))
                    {
                        for (int i = 0; i < habitNames.Count; i++)
                        {
                            string progressDatesString = 
string.Join(",", progressDates[i]);
                            string line = 
$"{habitNames[i]}|{targetCounts[i]}|{progressHistories[i]}|{progre
ssDatesString}";
                            writer.WriteLine(line);
                        }
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"An error occurred while 
saving data: {ex.Message}");
                }
            }

            private void LoadData()
            {
                string filePath = 
Path.Combine(Application.StartupPath, "habitData.txt");

                if (File.Exists(filePath))
                {
                    try
                    {
                        using (StreamReader reader = new 
StreamReader(filePath))
                        {
                            string line;
                            while ((line = reader.ReadLine()) != 
null)

                            {
                                var parts = line.Split('|');
                                habitNames.Add(parts[0]);

targetCounts.Add(int.Parse(parts[1]));
progressHistories.Add(int.Parse(parts[2]));
                                var dates =
parts[3].Split(',').ToList();
                                progressDates.Add(dates);


listBoxHabits.Items.Add($"{habitNames.Count}. {parts[0]} -
{parts[2]}/{parts[1]} days");
                            }
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show($"An error occurred while
loading data: {ex.Message}");
                    }
                }
            }
        }
    }
```

## 8. Interfaces

### Task Manager:



### Study Planner


Study Planner

Study Goal:

Study Hours: 0

Deadline: Monday , December

Add

Study Goals Progress

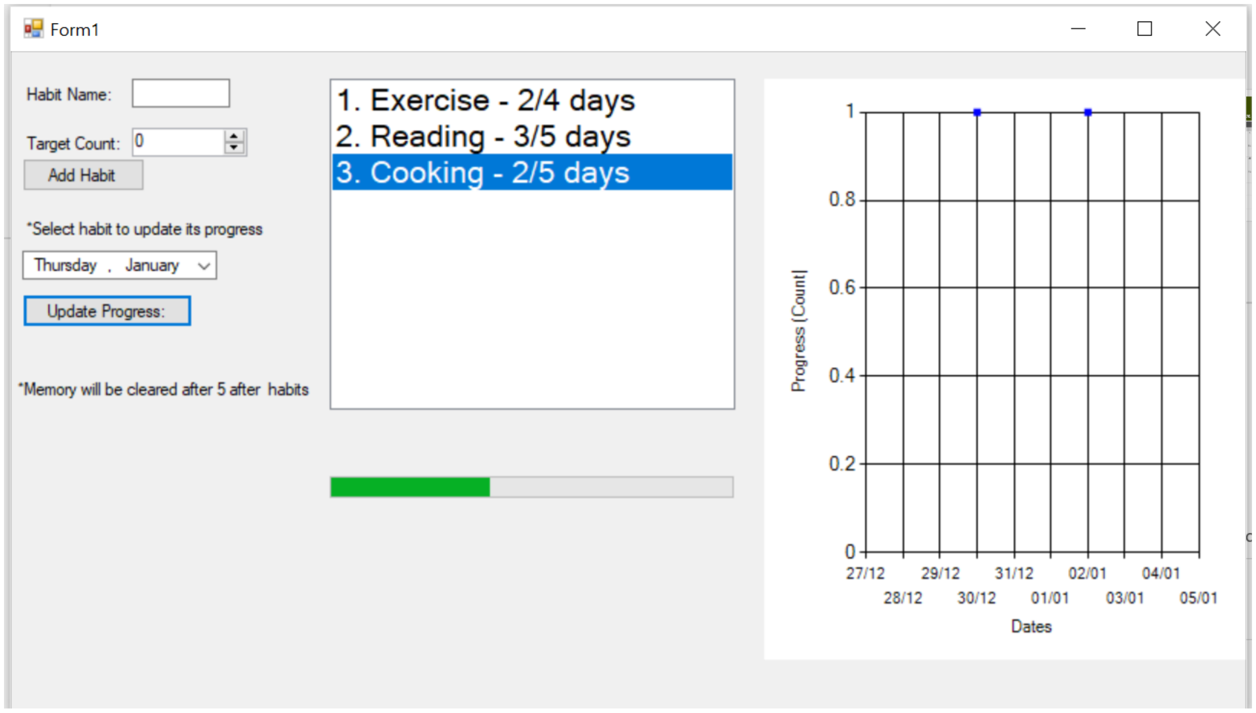
Total Hours Studied: 3

Update Progress

Progress:

* memory will be cleared after 5 goals

## Habit Tracker


Form1

Habit Name:

Target Count: 0

Add Habit

*Select habit to update its progress

Thursday , January

Update Progress:

*Memory will be cleared after 5 after habits

1. Exercise - 2/4 days
2. Reading - 3/5 days
3. Cooking - 2/5 days

## 9. Conclusion

The **Student Productivity Manager** project focuses on enhancing student productivity by providing tools for task management, study planning, and habit tracking. Designed with offline functionality, it prioritizes simplicity and accessibility for student users. Developed using C# and WinForms, the application ensures an intuitive and effective interface. By addressing key challenges not covered by existing LMS tools, this project lays the foundation for further improvements in productivity tools for students.

## 10.References

- S. Smith, *C# Programming for Beginners*, 3rd ed., Pearson Education, 2019.
- "C# WinForms Tutorial," Microsoft Docs. Available: https://docs.microsoft.com/en-us/dotnet/desktop/winforms/.
- "C# Basics for Beginners," freeCodeCamp. Available: https://www.freecodecamp.org/news/c-sharp-tutorial-for-beginners/.
- Microsoft, *Visual Studio Documentation*. Available: https://docs.microsoft.com/en-us/visualstudio/.