K S RANGASAMY COLLEGE OF TECHNOLOGY

(Autonomous)

TIRUCHENGODE-637215

A MINI PROJECT REPORT

TO-DO-LIST APPLICATION

60 IT L04 – C# AND .NET FRAMEWORK

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

Submitted by

MYVIZHI R (73772214165)

K S RANGASAMY COLLEGE OF TECHNOLOGY

(Autonomous)

TIRUCHENGODE-637215

BONAFIDE CERTIFICATE

Certified that this project report titled "**TO-DO-LIST APPLICATION**" is the bonafide work of **MYVIZHI R (73772214165)** who carried out the project under my guidance.

Dr. S. Madhavi M.E., Ph.D.,	**Mrs. N. Sathiyapriya M.E.,**
Professor and Head of the Department	Assistant Professor
Department of Computer Science and Engineering	Department of Information Technology
K.S. Rangasamy College of Technology	K.S. Rangasamy College of Technology
Tiruchengode-637 215.	Tiruchengode - 637 215.

ABSTRACT

The **To-Do List Application** is a simple yet efficient tool designed to help users manage their daily tasks and stay organized. The application allows users to add, view, mark as done, and delete tasks, providing an intuitive interface for task management. Users can input task descriptions and categorize them, and the app keeps track of their completion status. With a clear and easy-to-navigate menu, the app helps users focus on their priorities, ensuring they stay productive and on track throughout the day.

This application offers essential task management functionalities with a user-friendly experience. It ensures that users can quickly add new tasks, review their progress, and remove completed or unnecessary tasks. The system is scalable, allowing for potential future enhancements, such as due dates, reminders, or task prioritization, to improve overall task management efficiency. Whether used for personal organization or professional planning, the To-Do List Application provides a straightforward solution for staying on top of tasks.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The To-Do List Application project is focused on helping users organize and manage their tasks in a straightforward, user-friendly way. The application allows users to add new tasks, view existing ones, mark tasks as completed, and delete tasks when no longer needed. The goal of this project is to create a digital tool that can help users stay productive by keeping track of their daily responsibilities. Whether for personal use, work-related tasks, or a combination of both, the app provides a simple yet effective method for managing to-dos in an organized list format.

This project is a great example of a simple task management system, and its design ensures that users can easily interact with it without complexity. The application also offers the potential for future improvements, such as adding features like task prioritization, setting due dates, or sending reminders. With a clean and easy-to-navigate interface, the To-Do List Application project focuses on helping users stay on top of their tasks, increase productivity, and maintain a sense of accomplishment by tracking completed work.

CHAPTER 2

REQUIREMENT ANALYSIS

Functional Requirements:

1. **Add a Task**: Users can create new tasks by providing a title and description.
2. **Edit a Task**: Users can modify the details of existing tasks.
3. **Delete a Task**: Users can remove tasks they no longer need.
4. **Save Changes**: Tasks are displayed and updated in real-time using a DataGridView.
5. **Clear Fields**: Input fields can be reset to facilitate easy creation or updating of tasks.

Non-Functional Requirements:

1. **User-Friendly Interface**: The application features a clean, intuitive interface that simplifies task management.
2. **Responsiveness**: All operations, including adding, editing, and deleting tasks, execute in real-time to ensure smooth user interaction.
3. **Scalability**: The application can be expanded with additional features like task prioritization or deadline tracking in future updates.

Tools and Technologies:

- **Programming Language**: C#
- **Framework**: .NET Framework (Windows Forms)
- **Integrated Development Environment (IDE)**: Visual Studio

CHAPTER 3

DESIGN AND IMPLEMENTATION

System Overview:

Graphical User Interface (GUI):

1. A **DataGridView** control displays the list of tasks in a tabular format.
2. **Text fields (TextBox)** are used for user input of task titles and descriptions.
3. **Buttons (Button)** enable actions like creating, saving, editing, and deleting tasks.

Data Management:

1. A **DataTable** serves as an in-memory structure for storing task data.
2. **Binding** is used to seamlessly link the **DataTable** with the **DataGridView** for real-time updates.

Code Summary:

- **Form Load**: Initializes the **DataTable** and establishes the binding to the **DataGridView**, ensuring the interface is ready for user interaction.
- **New Task**: Clears the input fields (**TextBox**) to allow users to create a new task.
- **Edit Task**: Loads the selected task's details into the input fields for modification.
- **Delete Task**: Removes the selected task from the **DataTable**, automatically updating the **DataGridView**.
- **Save Task**: Adds new tasks or updates existing tasks in the **DataTable**, with real-time changes reflected in the UI.

CHAPTER 4

CODE

```
using System;

using System.Collections.Generic;



class ToDoListApp

{

    static List<Task> tasks = new List<Task>();



    static void Main()

    {

        Console.WriteLine("Welcome to the To-Do List Application!");



        bool running = true;

        while (running)

        {

            Console.WriteLine("\nMenu:");

            Console.WriteLine("1. Add Task");
            Console.WriteLine("2. View Tasks");

            Console.WriteLine("3. Mark Task as Done");

            Console.WriteLine("4. Delete Task");

            Console.WriteLine("5. Exit");

            Console.Write("Enter your choice: ");



            string choice = Console.ReadLine();

            switch (choice)

            {

                case "1":

                    AddTask();

                    break;

                case "2":

                    ViewTasks();

                    break;

                case "3":

                    MarkTaskAsDone();

                    break;
```

```
            case "4":
                DeleteTask();
                break;
            case "5":
                running = false;
                Console.WriteLine("Thank you for using the To-Do List
Application. Goodbye!");
                break;
            default:
                Console.WriteLine("Invalid choice. Please try again.");
                break;
        }
    }
  }

  static void AddTask()
  {
    Console.Write("Enter the task description: ");
```

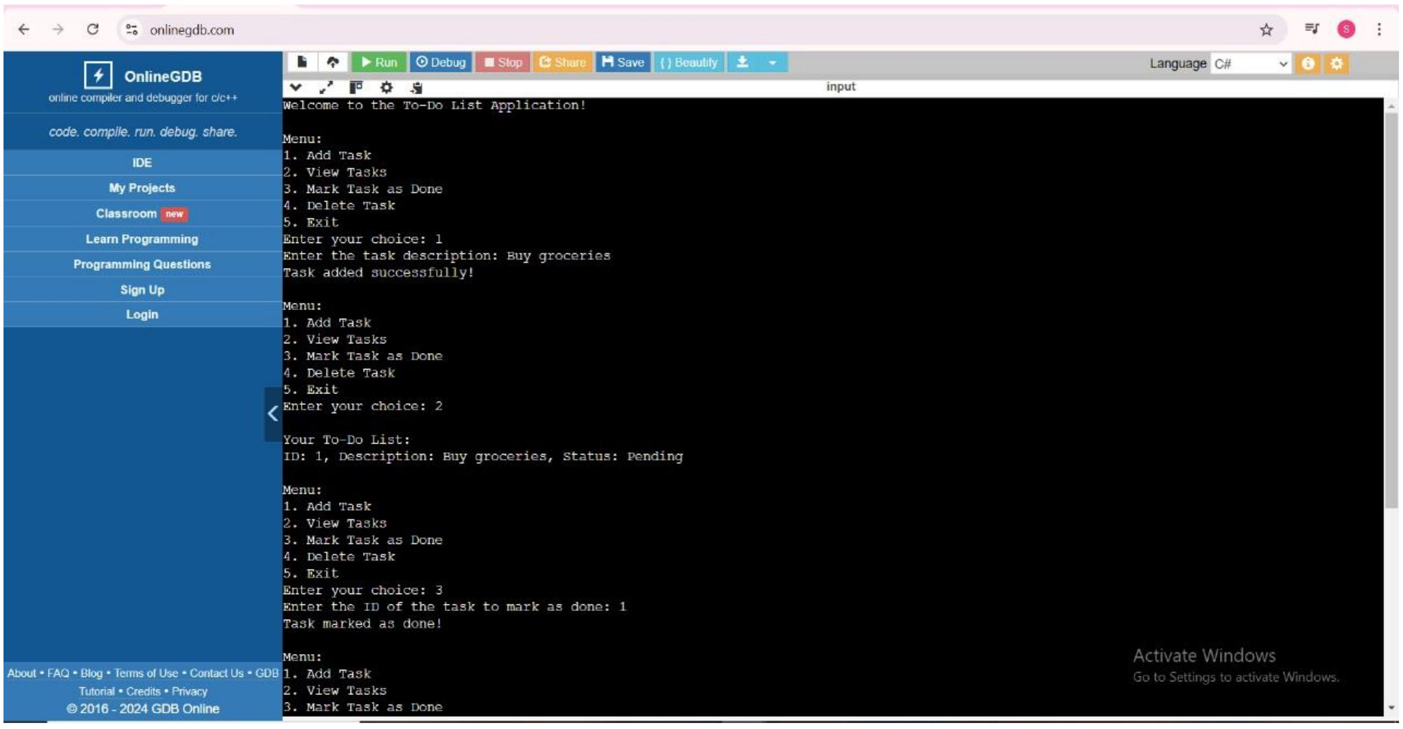
```
        string description = Console.ReadLine();


        Task newTask = new Task { Id = tasks.Count + 1, Description = 
description, IsCompleted = false };

        tasks.Add(newTask);

        Console.WriteLine("Task added successfully!");

    }


    static void ViewTasks()

    {

        Console.WriteLine("\nYour To-Do List:");

        if (tasks.Count == 0)

        {

            Console.WriteLine("No tasks available.");

        }

        else

        {

            foreach (var task in tasks)
            {
                string status = task.IsCompleted ? "Done" : "Pending";
                Console.WriteLine($"ID: {task.Id}, Description: {task.Description},
Status: {status}");
            }
        }
    }

    static void MarkTaskAsDone()
    {
        Console.Write("Enter the ID of the task to mark as done: ");
        if (int.TryParse(Console.ReadLine(), out int taskId))
        {
            var task = tasks.Find(t => t.Id == taskId);
            if (task != null)
            {
                task.IsCompleted = true;
                Console.WriteLine("Task marked as done!");
            }
            else
            {
                Console.WriteLine("Task not found.");
            }
        }
        else
        {
            Console.WriteLine("Invalid ID. Please enter a number.");
        }
    }

    static void DeleteTask()
    {
        Console.Write("Enter the ID of the task to delete: ");
        if (int.TryParse(Console.ReadLine(), out int taskId))
        {
            var task = tasks.Find(t => t.Id == taskId);
```

```
            if (task != null)

            {

                tasks.Remove(task);

                Console.WriteLine("Task deleted successfully!");

            }

            else

            {

                Console.WriteLine("Task not found.");

            }

        }

        else

        {

            Console.WriteLine("Invalid ID. Please enter a number.");

        }

    }

}



class Task
```

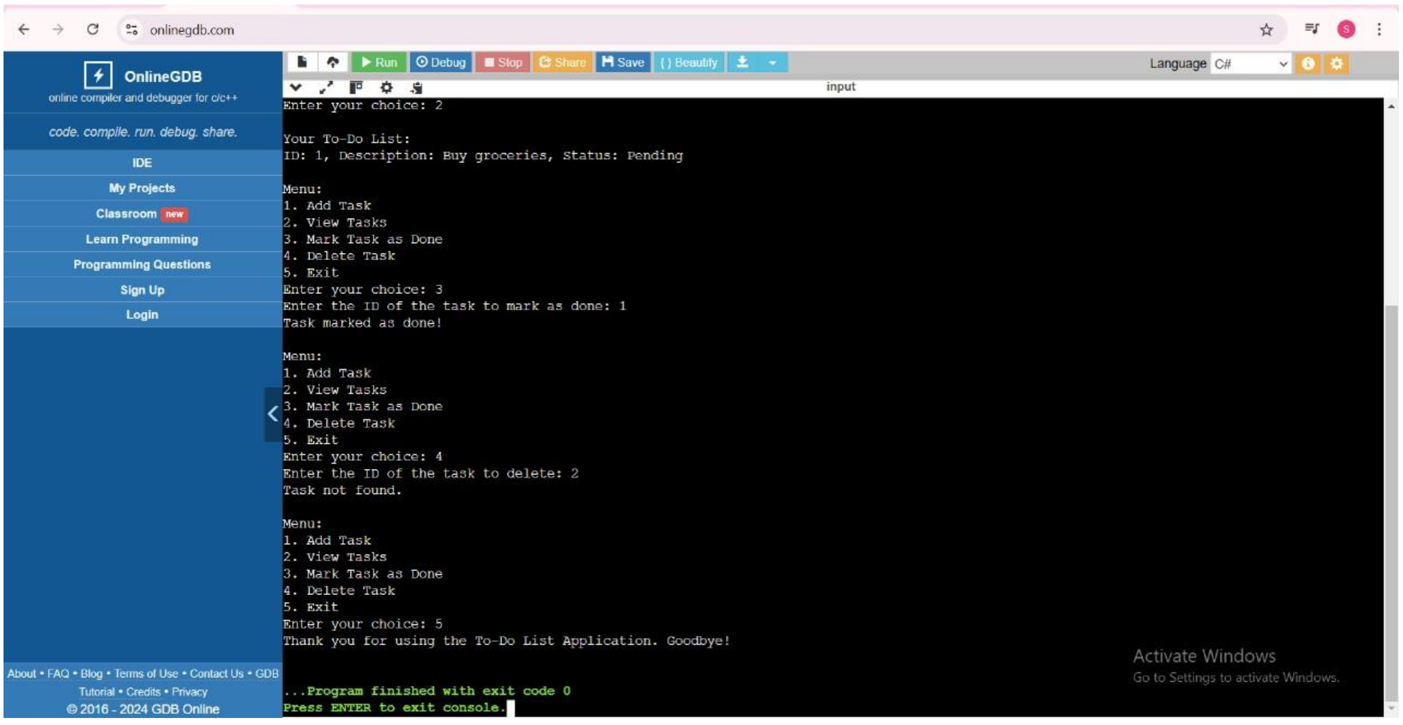
```
{

    public int Id { get; set; }

    public string Description { get; set; }

    public bool IsCompleted { get; set; }

}
```

CHAPTER 5

OUTPUT



```
Welcome to the To-Do List Application!

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 1
Enter the task description: Buy groceries
Task added successfully!

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 2

Your To-Do List:
ID: 1, Description: Buy groceries, Status: Pending

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 3
Enter the ID of the task to mark as done: 1
Task marked as done!

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
```



```
Enter your choice: 2

Your To-Do List:
ID: 1, Description: Buy groceries, Status: Pending

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 3
Enter the ID of the task to mark as done: 1
Task marked as done!

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 4
Enter the ID of the task to delete: 2
Task not found.

Menu:
1. Add Task
2. View Tasks
3. Mark Task as Done
4. Delete Task
5. Exit
Enter your choice: 5
Thank you for using the To-Do List Application. Goodbye!

...Program finished with exit code 0
Press ENTER to exit console.
```

CHAPTER 6

CONCLUSION

The To-Do List Application project serves as a valuable tool for task management and personal organization. By allowing users to add, view, update, and delete tasks, the application helps individuals stay productive and organized. With a straightforward interface and easy-to-use features, it ensures users can quickly manage their responsibilities. The simplicity of the project makes it an excellent starting point for task management applications, with potential for future enhancements, such as due dates, prioritization, or reminders. In conclusion, the To-Do List Application effectively helps users maintain control over their tasks and improve overall efficiency.

CHAPTER 7

REFERENCES

- "How to Build a To-Do List App in C# (with Windows Forms)" Link: https://www.codeproject.com/
- "Building a To-Do App with C# and .NET Core" Link: https://dev.to/
- "Creating a To-Do List Application with C#" Link: https://www.c-sharpcorner.com/