PROJECT TITLE: PROJECT TITLE

Introduction

A **To-Do List Application** is a simple, yet powerful tool designed to help users organize and manage their daily tasks. This application allows users to create, edit, and delete tasks while keeping track of their descriptions. Built using C# and Windows Forms, the application leverages an intuitive graphical user interface (GUI) to provide a user-friendly experience. It is particularly suitable for beginners in programming who wish to explore the fundamentals of Windows application development while practicing concepts like event handling, data manipulation, and user interface design.

Abstract

The **To-Do List Application** is a lightweight task management system developed in C#. It provides basic functionalities for creating, reading, updating, and deleting (CRUD) tasks. The primary objective of the application is to enable users to manage their daily activities effectively by maintaining a clear, editable list of tasks and their descriptions. The system uses a DataTable as an in-memory storage solution to maintain tasks, offering real-time data manipulation. The application is designed with simplicity, extensibility, and ease of use in mind, making it an excellent project for understanding the basics of Windows Forms development.

Requirements Analysis

Functional Requirements:

1. **Add a Task**: Users can create new tasks with a title and a description.
2. **Edit a Task**: Users can modify existing tasks.
3. **Delete a Task**: Users can remove unwanted tasks.
4. **Save Changes**: Tasks are updated in real-time and displayed in a DataGridView.
5. **Clear Fields**: The application allows users to reset the input fields to create or update tasks easily.

Non-Functional Requirements:

1. **User-Friendly Interface**: The application provides a clean and straightforward interface.
2. **Responsiveness**: Operations such as adding, editing, and deleting tasks occur in real-time.
3. **Scalability**: The application can be extended with features like task prioritization or deadlines.

Tools and Technologies:

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

Design and Implementation

System Overview:

1. **Graphical User Interface (GUI)**:
   - A DataGridView control displays the list of tasks.
   - Text fields (TextBox) are used for task title and description inputs.
   - Buttons (Button) perform actions like creating, saving, editing, and deleting tasks.
2. **Data Management**:
   - A DataTable serves as an in-memory storage structure for task data.
   - Binding is used to link the DataTable to the DataGridView.

Code Summary:

- **Form Load**: Initializes the DataTable and sets up the DataGridView binding.
- **New Task**: Clears the text boxes for fresh input.
- **Edit Task**: Loads selected task details into the input fields.
- **Delete Task**: Removes the selected task from the DataTable.
- **Save Task**: Adds or updates task details in the DataTable.

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;
namespace To_Do_List_App
{
    public partial class ToDoList : Form
    {
        public ToDoList()
        {
            InitializeComponent();
        }

        DataTable todoList = new DataTable();
        bool isEditing = false;
        private void ToDoList_Load(object sender, EventArgs e)
        {
            // Create columns
            todoList.Columns.Add("Title");
            todoList.Columns.Add("Description");


            // Point our datagridview to our datasource
            toDoListView.DataSource = todoList;
        }
        private void newButton_Click(object sender, EventArgs e)
        {
            titleTextBox.Text = "";
            descriptionTextbox.Text = "";
        }
        private void editButton_Click(object sender, EventArgs e)
        {
            isEditing = true;
            // Fill text fields with data from table
            titleTextBox.Text                                                   =
todoList.Rows[toDoListView.CurrentCell.RowIndex].ItemArray[0].ToString();
            descriptionTextbox.Text                                             =
todoList.Rows[toDoListView.CurrentCell.RowIndex].ItemArray[1].ToString();
        }
        private void deleteButton_Click(object sender, EventArgs e)
        {
            try
            {
```

```
            todoList.Rows[toDoListView.CurrentCell.RowIndex].Delete();
         }
         catch(Exception ex)
         {
            Console.WriteLine("Error: " + ex);
         }
      }
      private void saveButton_Click(object sender, EventArgs e)
      {
         if(isEditing)
         {
            todoList.Rows[toDoListView.CurrentCell.RowIndex]["Title"] = titleTextBox.Text;
            todoList.Rows[toDoListView.CurrentCell.RowIndex]["Description"]                    =
descriptionTextbox.Text;
         }
         else
         {
            todoList.Rows.Add(titleTextBox.Text, descriptionTextbox.Text);
         }
         // Clear fields
         titleTextBox.Text = "";
         descriptionTextbox.Text = "";
         isEditing = false;
      }
   }
```

Output

The application provides the following output:

1. A fully functional to-do list displayed in a DataGridView.
2. Ability to perform CRUD operations:
    - **Create**: Add new tasks to the list.
    - **Read**: View tasks in a tabular format.
    - **Update**: Modify task details.
    - **Delete**: Remove tasks from the list.

Sample User Interface:

- A grid displaying the task title and description.
- Buttons for "New", "Save", "Edit", and "Delete" operations.
- Text boxes for inputting task details.

Conclusion

The **To-Do List Application** demonstrates the fundamental principles of Windows Forms development and CRUD operations using C#. It is an effective tool for managing daily tasks and serves as a practical introduction to GUI-based application development. Future enhancements could include features such as task deadlines, priority levels, data persistence (using a database or file system), and user authentication for a more robust system. Overall, this project provides a solid foundation for learning and exploring more advanced concepts in software development.