TO-DO LIST SYSTEM WITH CALENDAR INTEGRATION

FOR

ICARUS SHIRTS (BONANZA ENTERPRISE)

A Thesis Project Presented to the
Faculty of Datamex College of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the
Degree of Bachelor of Science in Information Technology

By:
Gabriel, Mikaelle Angelo A.
Ferrer, Daryl Jake V.
Bernante, Jayson
Mendinueta, Jaslyn

TECHNICAL DOCUMENTATION

INTRODUCTION

Purpose

The purpose of this technical documentation is to provide a comprehensive reference for the development, deployment, and maintenance of the To-Do List System created for Icarus Shirts (Bonanza Enterprise). This document ensures that developers, users, and maintainers have a unified understanding of the system's objectives, structure, and functionality. It serves as a guide for installation, configuration, database setup, troubleshooting, and long-term system management. Furthermore, it documents the internal workings of the software to support future enhancements, scalability, and maintainability.

Overview

The To-Do List System is a standalone desktop application developed using Visual Basic .NET and SQL Server Express. It is designed to simplify and improve task management by enabling users to create, update, track, and organize tasks within a calendar interface. The system incorporates features such as recurring scheduling (daily, weekly, monthly), priority labeling, task completion tracking, and a recycle bin for task recovery. Unlike manual tracking methods, this system centralizes task data in a secure local database, reducing errors while improving productivity and organization. It is particularly beneficial for Icarus Shirts, where production schedules and customer orders must be managed efficiently to meet deadlines and ensure customer satisfaction.

Scope

This document covers the full technical details of the system, from installation and configuration to database structure, module descriptions, and maintenance procedures. The scope includes:

- System Overview: Explanation of system architecture, components, and deployment.
- Installation & Configuration Guides: Instructions for setting up hardware, software, and system parameters.

- Database Documentation: Entity-relationship diagrams, schema details, and backup procedures.
- User Manual: Step-by-step guide for navigating the system's user interface and performing common tasks.
- Troubleshooting Guide: Common issues, error codes, and solutions.
- Code & Testing Documentation: System modules, coding standards, test cases, and results.
- Maintenance Guide: Guidelines for updates, backups, bug fixes, and long-term support.

By providing these details in one consolidated document, the To-Do List System ensures that both technical staff and end-users have the necessary resources to operate and sustain the software effectively. This introduction frames the technical documentation as an essential companion for system deployment, usage, and ongoing improvement.

SYSTEM OVERVIEW

The To-Do List System is a standalone desktop application built with Visual Basic .NET and SQL Server Express. It follows a modular architecture where the user interacts through a Windows Forms interface, while core modules (Task Management, Calendar, and Dashboard) handle system logic. A centralized Data Access Layer manages communication with the local database, ensuring data security, consistency, and persistence in a single-user environment.

High level Components

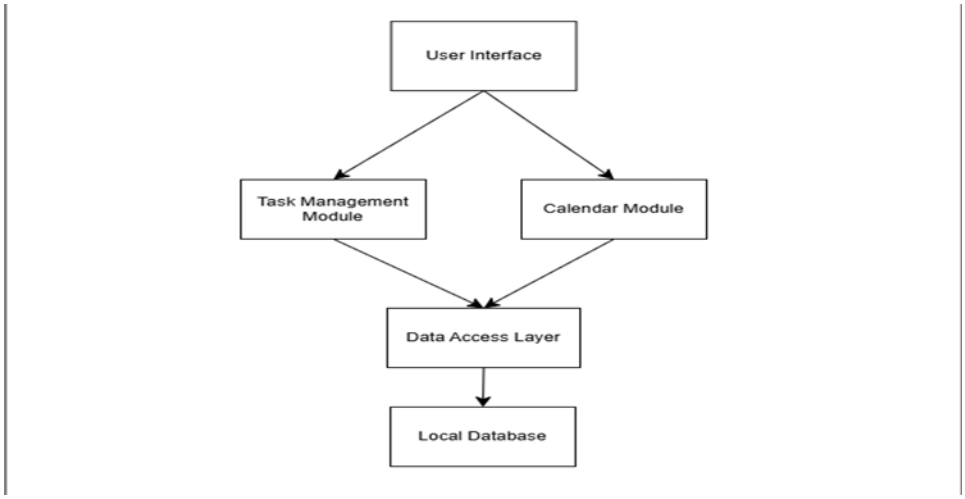

Figure 1: System Architecture

1. **User**: The client that will use and operate the system
2. **Task Management Module**: This is responsible for creating, updating, and deleting tasks
3. **Calendar Module**: Organizes and displays tasks in a date-oriented view.
4. **Data Access Layer**: Handles the communication of the application to the database.
5. **Local Database**: Stores task information, and Reminders.

Deployment Architecture

Stand-alone System: the entire system runs on a single device without requiring an internet connection or external servers. This design ensures simplicity, cost-effectiveness, and ease of maintenance, making it ideal for small businesses with limited IT infrastructure.

INSTALLATION GUIDE

This section provides detailed instructions for installing and setting up the To-Do List System on a client machine. It is intended to ensure that users, administrators, and technical staff can correctly deploy the system with minimal issues. The installation process includes verifying hardware and software requirements, preparing the environment, setting up the SQL Server database, and configuring application settings.

System Requirements

1. **Hardware:**
    - CPU: either an Intel i3 10th Generation, Intel i5, or an AMD alternative of the two (Ryzen 3 or 5)
    - RAM: at least 4GB of RAM
    - Storage: Hard disk or SSD with at least 50 GB (w/ system files)
2. **Software:**
    - Operating System: Windows 10 or Windows 11 for compatibility and scalability.
    - Database Management: SQL Server Management Studio (SSMS)

Step-by-step Guide

1. Install SQL Server Management Studio
2. Install with default Settings, connect, and attach the database that was given.
3. Install the To-Do List System executable files on the client's machines
4. Test the system and change the connection strings if needed.

Configuration settings and options.

1. Change the password
2. Create a backup for the database
3. Restore the database backup

CONFIGURATION GUIDE

The configuration of the To-Do List System is designed to be simple and user-friendly. The system provides three main configuration options accessible from within the application: **password management, system backup, and system recovery**. These options allow the user to secure their account, protect data, and restore information if needed.

Password Configuration

- Navigate to the **Settings > Change Password** option in the application.
- Enter your **current password**, followed by the **new password**.
- Confirm the new password and click **Save**.
- If the password is successfully updated, the system will prompt with a confirmation message.
- Best Practice: Use a strong password (minimum 8 characters, with a mix of letters, numbers, and symbols).

Backup Configuration

- Open the system and navigate to **Settings > Backup**.
- Click **Backup Now**. The system will run an internal SQL BACKUP DATABASE command to generate a .bak file of the database.
- Once completed, a message will confirm the success of the backup.
- Best Practice: Store backups on a secure external drive or cloud storage for redundancy.

Recovery Configuration

- Navigate to **Settings > Recover Backup**.
- Click **Restore**. The system will run an internal SQL RESTORE DATABASE command to replace the current database with the backup file.
- Once completed, the system will confirm that the recovery was successful.
- Warning: Restoring a backup will overwrite the current database. Make sure to back up the latest data before performing recovery.

Best Practices

- Regularly update your password to maintain account security.
- Schedule weekly backups to prevent accidental data loss.
- Keep multiple backup copies (local and external) for added safety.
- Always confirm that the correct backup file is selected before restoring.

API DOCUMENTATION

The To-Do List System does not expose external APIs. It is a standalone desktop system where all functionality is accessed through the user interface and local database interactions.

DATABASE DOCUMENTATION

This section is for the Database Documentation for the To-Do List System that includes the database's Entity-Relationship Diagram (ERD), description of tables, fields, and relationships, and procedures for data migration and backup procedures.

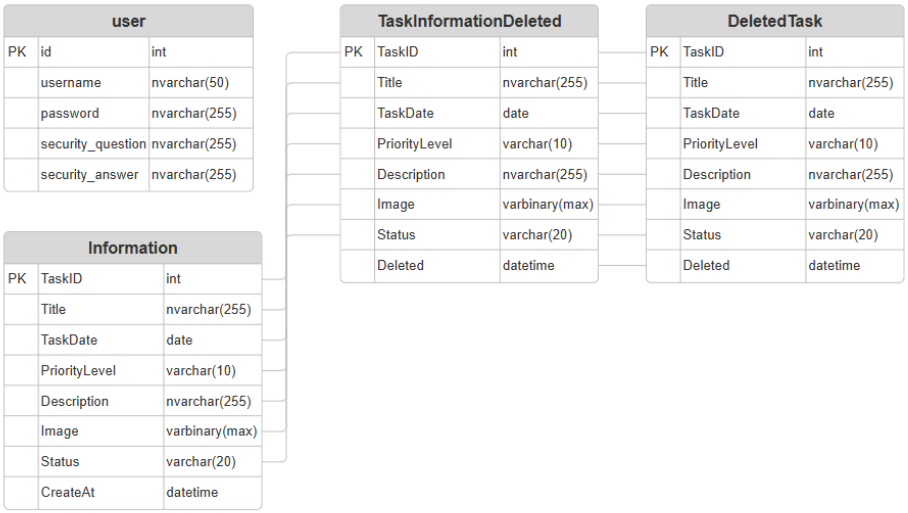

Figure 2: Entity-relationship Diagram

1. **Users Table**

This table stores information about the system's users. Since your system is single-user only (based on your scope), this ensures login security and future-proofing if more users are added.

- **id (PK, integer):** Unique identifier for each user.
- **username (nvarchar):** The name the user uses to log in.
- **password (nvarchar):** The user's password for authentication.
- **security_question (nvarchar):** The user's chosen security question.
- **security_answer (nvarchar):** The user's answer to the security question.

2. **Information Table (Active Tasks)**

This is the main table that stores all active or pending tasks.

- **TaskID (PK, int):** Unique identifier for each task.
- **Title (nvarchar (255)):** Short name of the task
- **TaskDate (date):** Date of the task or deadline
- **PriorityLevel (varchar (10)):** Task priority: Extreme, Moderate, or Low
- **Description (nvarchar (255)):** Full task description
- **Image (varbinary (max)):** Optional image associated with the task
- **Status (varchar (20)):** Task status: Completed, In Progress, Not Started
- **CreateAt (datetime):** Timestamp of when the task was created.

3. **Task Information Deleted Table (Task's Trash Bin)**

This table stores tasks that are deleted. Tasks from the Information table are moved here upon deletion.

- **TaskID (PK, int):** Unique identifier for each task.
- **Title (nvarchar (255)):** Task Title
- **TaskDate (date):** Original Task Date
- **PriorityLevel (varchar (10)):** Priority level

- **Description (nvarchar (255)):** Full description of the task
- **Image (varbinary (max)):** Image previously attached
- **Status (varchar (20)):** Task status before deletion
- **Deleted (datetime):** Timestamp of deletion

4. **Deleted Task Table (Deleted Tasks)**

This table stores tasks that are deleted and cannot be able to restore. Tasks from the Information table are moved here upon deletion.

- **TaskID (PK, int):** Unique identifier for each task.
- **Title (nvarchar (255)):** Task Title
- **TaskDate (date):** Original Task Date
- **PriorityLevel (varchar (10)):** Priority level
- **Description (nvarchar (255)):** Full description of the task
- **Image (varbinary (max)):** Image previously attached
- **Status (varchar (20)):** Task status before deletion
- **Deleted (datetime):** Timestamp of deletion

Database Backup procedures

The system are able to create a back-up of the data and restore to the previous back-up through settings.

Database Migration

- Open SQL Server Management Studio (SSMS).
- Connect and locate the database DBTodoList in the object explorer
- Right-click, then go to task, and click detach.
- Locate the mdf file of the database to get the copy.
- Attach the database by right-clicking the database folder in object explorer of SSMS and locate and select the mdf file you detached.

USER MANUAL

The To-Do List System provides a user-friendly interface for managing tasks, monitoring deadlines, and maintaining productivity. This User Manual serves as a guide to help users understand and navigate each feature of the system.

Create Account (New user)

- If there are no user and passwords yet, this form will show up.
- Enter your username & password you want to use.
- Choose your security question and enter the answer in the textbox below.

Log In Form

- When the application starts, the Log In Form appears.
- Enter your password in the input field.
- Upon successful login, the system will direct you to the Dashboard.

Dashboard

The Dashboard is the central hub of the system. It contains:

- **Recently Added Tasks** – Displays the three most recent tasks.
- **Add Task Button** – Opens the task creation window.
- **Completed Tasks Preview** – Shows the two most recent completed tasks.
- **Task Status Summary** – Displays the percentage of tasks that are **Completed**, **In Progress**, and **Not Started** in a circular chart.
- **Navigation Panel (Left Side)** – Provides buttons that lead to different sections:
    - My Vital
    - My Task
    - Calendar
    - Completed
    - Recycle Task
    - Settings
    - Log Out

My Vital

- Displays a list of tasks with **Extreme Priority**.
- Selecting a task shows its details on the right panel.
- Functions:
    - **Edit Task** – Select a task and click the **Edit** button to update its details.

Add/Update Task Window

When adding or editing a task, a new window opens where you can enter or modify the following:

- **Title** – Short name of the task.
- **Deadline** – Task due date.
- **Priority Level** – Choose between *Extreme*, *Moderate*, or *Low*.
- **Description** – Full description of the task.
- **Status** – Select between *Not Started* or *In Progress*.
- **Image** – Option to attach an image for reference.

My Task

- Displays **all tasks** created in the system.
- Layout is similar to My Vital, with task list on the left and task info on the right.
- Functions:
    - **Edit Task** – Modify selected tasks.
    - **Delete Task** – Moves the task to the Recycle Bin.
    - **Add Task** – Create new tasks with any priority level.

Calendar

- Provides a **visual calendar interface**.
- Displays task deadlines on their respective dates.
- Supports daily, weekly, and monthly viewing for better task planning.

Completed

- Displays a list of all **Completed Tasks**.
- Same layout as My Vital (task list on the left, details on the right).

Recycle Task

- Displays a list of **Deleted Tasks**.
- Layout is the same as My Vital, but the **Edit button is replaced with Restore**.
- Functions:
    - **Restore Task -** Moves the selected task back to the active task list.
    - **Delete Task** – Permanently delete a task.

Settings

The Settings tab allows the user to configure the system:

1. **Change Password** – Update the login password.
2. **Back Up Database** – Create a .bak backup file of the current database.
3. **Restore Database** – Recover the system using a previously saved backup file.

Log Out

1. Ends the current session and returns the user to the **Log In Form**.
2. Ensures system security by preventing unauthorized access when the system is left unattended.

TROUBLESHOOTING GUIDE

This section provides solutions to common problems encountered while using the To-Do List System. It includes error messages, likely causes, and recommended resolutions. The guide is intended for end-users and technical staff to ensure minimal downtime and smooth operation.

8.2 Common Issues and Solutions

Login Issues

- **Problem:** “Invalid password” error when logging in.
    - **Cause:** The entered password does not match the stored password.
    - **Solution:**
        - Re-enter the correct password.
        - If forgotten, contact the administrator to reset the password in the database.
- **Problem:** Application does not open the Dashboard after entering credentials.
    - **Cause:** Corrupted user record or database connection failure.
    - **Solution:** Verify database connection in App.config. Ensure the Users table contains a valid account.

Task Management Issues

- **Problem:** Unable to save a new task.
    - **Cause:** Required fields (Title, Deadline, or Priority) are missing.
    - **Solution:** Fill in all required fields before saving.
- **Problem:** Editing a task does not update in the list.
    - **Cause:** Database update failed or no fields were changed.

    - **Solution:** Ensure valid data is entered. Retry editing with new values.

- **Problem:** Tasks disappear after deletion.
    - **Cause:** Tasks are moved to the Recycle Bin by design.
    - **Solution:** Open the **Recycle Task** tab and restore the task if needed.

Calendar Issues

- **Problem:** Task not showing in the calendar.
    - **Cause:** Incorrect or missing deadline date.
    - **Solution:** Edit the task and ensure a valid date is set.

Backup and Recovery Issues

- **Problem:** “Cannot open backup device” error during backup.
    - **Cause:** The backup folder path does not exist or SQL Server does not have permission.
    - **Solution:**
        - Create the backup folder manually
        - Ensure the SQL Server service account has write permission to the folder.
- **Problem:** Backup completes but no file is generated.
    - **Cause:** Incorrect backup location or insufficient disk space.
    - **Solution:** Select a different folder and verify disk space.
- **Problem:** Unable to restore from backup.
    - **Cause:** The .bak file is corrupted or incompatible.

- **Solution:** Use the latest valid backup file. Always verify backup integrity before use.

Performance Issues

- **Problem:** Application runs slowly when many tasks are added.
  - **Cause:** Large number of tasks with attached images.
  - **Solution:**
    - Use smaller images.
    - Archive completed/deleted tasks to keep the database optimized.

Error Codes

The system uses the following error codes for clarity:

Error Code	Description	User Message
1001	Invalid login	"Login failed. Please check your password."
2001	Query error	"Error retrieving data. Please try again."
2002	Insert failed	"Error saving task."
2003	Update failed	"Error updating task."
2004	Delete failed	"Error deleting task."

Technical Support

If the above steps do not resolve the issue:

Contact: jaysongame27@gmail.com

CODE DOCUMENTATION

This section provides the overview of the codes of the To-Do List System and its different functions.

Connection Module

```
Public Sub AttachDatabase()
    Dim dbName As String = "DBTodoList"
    Dim basePath As String = Application.StartupPath
    Dim mdfPath As String = IO.Path.Combine(basePath, "MDF", "DBTodoList.mdf")
    Dim ldfPath As String = IO.Path.Combine(basePath, "MDF", "DBTodoList_log.ldf")
    Dim masterConnStr As String = "Server=.\SQLEXPRESS01;Database=master;Trusted_Connection=True;"

    Try
        Using conn As New SqlConnection(masterConnStr)
            conn.Open()

            Dim checkCmd As New SqlCommand("SELECT database_id FROM sys.databases WHERE name = @dbName", conn)
            checkCmd.Parameters.AddWithValue("@dbName", dbName)

            Dim result = checkCmd.ExecuteScalar()

            If result Is Nothing Then
                Dim attachSql As String = "CREATE DATABASE [" & dbName & "] ON (FILENAME = '" & mdfPath & "'), (FILENAME = '" & ldfPath & "') FOR ATTACH;"
                Dim attachCmd As New SqlCommand(attachSql, conn)
                attachCmd.ExecuteNonQuery()
                MsgBox("Database attached successfully!", MsgBoxStyle.Information)

            Else
                MsgBox("Database is already attached.", MsgBoxStyle.Information)
            End If
        End Using

    Catch ex As Exception
        MsgBox("Failed to attach database: " & ex.Message, MsgBoxStyle.Critical)
    End Try
End Sub
```

```
Public Function BackupDatabase(ByVal backupPath As String) As Boolean
    ' Use master DB for backup operation
    Dim masterConnString As String = "Server=Server=.\SQLEXPRESS; Database=master; Trusted_Connection=True;"
    Dim backupQuery As String = "BACKUP DATABASE [DBTodoList] TO DISK = N'" & backupPath & "' WITH INIT;"


    Try
        Using con As New SqlConnection(masterConnString)
            con.Open()
            Using cmd As New SqlCommand(backupQuery, con)
                cmd.ExecuteNonQuery()
            End Using
        End Using
        MsgBox("Backup completed successfully to: " & backupPath, MsgBoxStyle.Information, "Backup Success")
        Return True
    Catch ex As Exception
        MsgBox("Backup failed: " & ex.Message, MsgBoxStyle.Critical, "Backup Error")
        Return False
    End Try
End Function
```

Utilities Module

```
Public Function HashPassword(ByVal input As String) As String
    Using sha256 As SHA256 = SHA256.Create()
        Dim bytes As Byte() = Encoding.UTF8.GetBytes(input)
        Dim hash As Byte() = sha256.ComputeHash(bytes)
        Return BitConverter.ToString(hash).Replace("-", "").ToLower()
    End Using
End Function
```

```
Private Function GetTimeAgo(ByVal dt As DateTime) As String
    Dim ts As TimeSpan = DateTime.Now - dt

    If ts.TotalSeconds < 60 Then
        Return "Just now"
    ElseIf ts.TotalMinutes < 60 Then
        Return Math.Floor(ts.TotalMinutes) & " minute(s) ago"
    ElseIf ts.TotalHours < 24 Then
        Return Math.Floor(ts.TotalHours) & " hour(s) ago"
    ElseIf ts.TotalDays < 7 Then
        Return Math.Floor(ts.TotalDays) & " day(s) ago"
    Else
        Return dt.ToString("MMM dd, yyyy")
    End If
End Function
```

```
Public Function ArchiveAndDeleteTask(ByVal taskId As Integer) As Boolean
    Try
        ' Step 1: Insert task into TaskInformationDeleted
        Dim insertQuery As String = "INSERT INTO TaskInformationDeleted(Title, TaskDate, PriorityLevel, Description, Image, Status)SELECT Title, TaskDate, PriorityLevel
    Connection.AddParam("@TaskID", taskId)

    If Not Connection.Insert(insertQuery) Then
        MsgBox("❌ Failed to archive task to TaskInformationDeleted.")
        Return False
            End If

            ' Step 2: Delete task from original table (Information)
    Dim deleteQuery As String = "DELETE FROM Information WHERE TaskID = @TaskID"
    Connection.AddParam("@TaskID", taskId)

    If Not Connection.Delete(deleteQuery) Then
        MsgBox("❌ Task was archived but not deleted from Information.")
        Return False
            End If

    MsgBox("✅ Task successfully deleted and archived.", MsgBoxStyle.Information)
    Return True

Catch ex As Exception
    MsgBox("❌ Archive/Delete Error: " & ex.Message)
    Return False
    End Try
End Function
```

```
Public Function RecoverTask(ByVal taskId As Integer) As Boolean
    Dim conStr As String = Connection.ConnString

    Using con As New SqlConnection(conStr)
        con.Open()
        Dim transaction = con.BeginTransaction()

        Try
            Dim insertSql As String = "INSERT INTO Information (Title, TaskDate, PriorityLevel, Description, Image, Status)

            Dim deleteSql As String = "DELETE FROM TaskInformationDeleted WHERE TaskID=@TaskID"

        Using insertCmd As New SqlCommand(insertSql, con, transaction),
              deleteCmd As New SqlCommand(deleteSql, con, transaction)

            insertCmd.Parameters.AddWithValue("@TaskID", taskId)
            deleteCmd.Parameters.AddWithValue("@TaskID", taskId)

            insertCmd.ExecuteNonQuery()
            deleteCmd.ExecuteNonQuery()
                End Using

        transaction.Commit()
        Return True
    Catch ex As Exception
        transaction.Rollback()
        MessageBox.Show("Error recovering task: " & ex.Message)
        Return False
        End Try
    End Using
End Function
```

```
Public Function ChangePassword(ByVal userId As Integer, ByVal currentPassword As String, ByVal newPassword As String) As Boolean
    Try
        ' 1. Verify the current password is correct
        Connection.AddParam("@id", userId)
        Connection.AddParam("@currentPassword", currentPassword)

        Dim checkQuery As String = "SELECT * FROM users WHERE user_id = @id AND password = @currentPassword"
        Dim result As DataTable = Connection.Query(checkQuery)

        If result Is Nothing OrElse result.Rows.Count = 0 Then
            MsgBox("❌ Current password is incorrect.", MsgBoxStyle.Critical)
            Return False
        End If

        ' 2. Update to new password
        Connection.AddParam("@id", userId)
        Connection.AddParam("@newPassword", newPassword)

        Dim updateQuery As String = "UPDATE users SET password = @newPassword WHERE user_id = @id"

        If Connection.Update(updateQuery) Then
            MsgBox("✅ Password changed successfully.", MsgBoxStyle.Information)
            Return True
        Else
            MsgBox("❌ Failed to update password.", MsgBoxStyle.Critical)
            Return False
        End If

    Catch ex As Exception
        MsgBox("Error changing password: " & ex.Message, MsgBoxStyle.Critical)
        Return False
    End Try
End Function
```

```
Public Sub AutoDeleteOldTasks()
    Try
        Dim selectQuery As String = "SELECT TaskID FROM TaskInformationDeleted WHERE Deleted < DATEADD(MONTH, -1, GETDATE())"
        Dim oldTasks As DataTable = Connection.GetDataTable(selectQuery)

        For Each row As DataRow In oldTasks.Rows
            Dim taskId As String = row("TaskID").ToString()

            Try
                TaskDeletePermanent(taskId)
            Catch ex As Exception
                ' Silently continue — or optionally log somewhere
            End Try
        Next
    Catch ex As Exception
        ' Optional: log this if needed
        ' MsgBox("Error auto-deleting old tasks: " & ex.Message)
    End Try
End Sub
```

```
Public Function TruncateText(ByVal text As String, ByVal maxLength As Integer) As String
    If String.IsNullOrEmpty(text) Then Return ""
    If text.Length <= maxLength Then
        Return text
    Else
        Return text.Substring(0, maxLength) & "..."
    End If
End Function
```

```
Public Function UpdateTaskStatus(ByVal taskId As String, ByVal status As String) As Boolean
    Dim query As String = "UPDATE Information SET Status = @Status WHERE TaskID = @TaskID"

    Connection.AddParam("@Status", status)
    Connection.AddParam("@TaskID", taskId)

    Return Connection.Execute(query)
End Function
```

Create Account Form

```
Private Sub CreateAccount()
    Dim username As String = txtUsername.Text.Trim()
    Dim password As String = txtPassword.Text.Trim()
    Dim confirmPassword As String = txtConfirmPassword.Text.Trim()
    Dim question As String = cmbSecurityQuestion.Text
    Dim answer As String = txtSecurityAnswer.Text.Trim()

    ' Basic validation
    If username = "" Or password = "" Or confirmPassword = "" Or question = "" Or answer = "" Then
        MsgBox("Please fill in all fields.", MsgBoxStyle.Exclamation, "Incomplete Form")
        Exit Sub
    End If

    ' Password confirmation check
    If password <> confirmPassword Then
        MsgBox("Passwords do not match. Please try again.", MsgBoxStyle.Critical, "Password Mismatch")
        txtPassword.Clear()
        txtConfirmPassword.Clear()
        txtPassword.Focus()
        Exit Sub
    End If



    ' Save hashed values
    If Utilities.RegisterUser(username, password, question, answer) Then
        MsgBox("User registered successfully!", MsgBoxStyle.Information, "Success")
        Me.Close()
        frmLogin.Show()
    Else
        MsgBox("Registration failed. Username might already exist.", MsgBoxStyle.Critical, "Error")
    End If
End Sub
```

LogIn Form

```
Private Sub frmLogin_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

   txtPass.Clear()

    If Utilities.IsUserTableEmpty() Then
        Me.Hide()
        FrmCreateAccount.ShowDialog()
    Else
        FrmCreateAccount.Hide()
    End If
End Sub


Private Sub PerformLogin()
     Dim password As String = txtPass.Text.Trim()

     Dim result As Boolean = Utilities.Login(password)

     If result Then
         frmDashboard.lblFormtitle1.Visible = True
         frmDashboard.lblFormtitle2.Visible = True
         frmDashboard.lblFormtitle3.Visible = False
         frmDashboard.lblFormtitle4.Visible = False

         frmData.ShowDialog()
         frmDashboard.pnlDashboard.BringToFront()
         frmDashboard.btnHome.Image = My.Resources.dashboard1
         frmDashboard.btnHome.FillColor = Color.White
         frmDashboard.btnHome.ForeColor = Color.FromArgb(255, 103, 103)
         Me.Hide()
     Else
         txtPass.Clear()
         MessageBox.Show("Login failed. (Error Code: 1001)", "Authentication Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
     End If
 End Sub
```

```
Private Sub Forgotpassword()
    Dim newPass As String = txtNewPassword.Text.Trim()
    Dim confirmPass As String = txtConfirmNewPassword.Text.Trim()

    If newPass = "" Or confirmPass = "" Then
        MessageBox.Show("Please fill in both password fields.", "Missing Info", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        Exit Sub
    End If

    If newPass <> confirmPass Then
        MessageBox.Show("Passwords do not match.", "Mismatch", MessageBoxButtons.OK, MessageBoxIcon.Error)
        Exit Sub
    End If

    ' Hash the new password only (consistent with your registration HashPassword usage)
    Dim newPassHash As String = Utilities.HashPassword(newPass)

    Connection.AddParam("@password", newPassHash)
    Connection.AddParam("@id", userId)

    If Connection.Execute("UPDATE users SET password=@password WHERE user_id=@id") Then
        MessageBox.Show("✅ Password updated successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information)
        pnlLogin.BringToFront()

    Else
        MessageBox.Show("❌ Failed to update password.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If
End Sub
```

Dashboard Form

```
Public Sub LoadLatest3TasksFromFrmData()
    If frmData Is Nothing OrElse frmData.dgvData.Rows.Count = 0 Then Exit Sub

    ' Clear old data
    ClearTaskLabels()

    Dim dgv = frmData.dgvData
    Dim totalRows As Integer = dgv.Rows.Count - 1 ' exclude new row

    If totalRows < 1 Then Exit Sub

    Dim labelIndex As Integer = 3 ' Start at the bottom label

    For i As Integer = totalRows - 1 To 0 Step -1
        If dgv.Rows(i).IsNewRow Then Continue For

        Dim title As String = dgv.Item(0, i).Value.ToString()
        Dim desc As String = dgv.Item(3, i).Value.ToString()
        Dim priority As String = dgv.Item(2, i).Value.ToString().Trim()
        Dim status As String = dgv.Item(5, i).Value.ToString().Trim()
        Dim rawValue As Object = dgv.Item(6, i).Value

        Dim created As String = ""
        If rawValue IsNot Nothing AndAlso IsDate(rawValue) Then
            Dim createdDate As DateTime = CDate(rawValue)
            created = GetTimeAgo(createdDate)
        End If

        ' Load image from byte array (if exists)
        Dim imgData As Object = dgv.Item(4, i).Value
        Dim img As Image = Nothing
        If imgData IsNot DBNull.Value Then
            Dim bytes As Byte() = CType(imgData, Byte())
            Using ms As New IO.MemoryStream(bytes)
                img = Image.FromStream(ms)
            End Using
        End If


        ' Apply values to controls
        Select Case labelIndex
            Case 1
                lblTitle3.Text = title
                lblDescription3.Text = desc
                lblPriority3.Text = priority
                lblStatus3.Text = status
                lblCreated3.Text = created
                pic3.Image = img
                lblPriority3.ForeColor = GetPriorityColor(priority)
                lblStatus3.ForeColor = GetStatusColor(status)

            Case 2
                lblTitle2.Text = title
                lblDescription2.Text = desc
                lblPriority2.Text = priority
                lblStatus2.Text = status
                lblCreated2.Text = created
                pic2.Image = img
                lblPriority2.ForeColor = GetPriorityColor(priority)
                lblStatus2.ForeColor = GetStatusColor(status)

            Case 3
                lblTitle1.Text = title
                txtDescription1.Text = desc
                lblPriority1.Text = priority
                lblStatus1.Text = status
                lblCreated1.Text = created
                pic1.Image = img
                lblPriority1.ForeColor = GetPriorityColor(priority)
                lblStatus1.ForeColor = GetStatusColor(status)
        End Select

        labelIndex -= 1
        If labelIndex = 0 Then Exit For
    Next
End Sub
```

```
Public Sub LoadLatestCompletedTasks()
    If dgvCompletedData Is Nothing OrElse dgvCompletedData.Rows.Count = 0 Then
        cardCompleted1.Visible = False
        cardCompleted2.Visible = False
        Exit Sub
    End If

    ' Clear labels and images first
    lblCompletedTitle1.Text = ""
    lblCompletedStatus1.Text = ""
    lblCompletedDay1.Text = ""
    lblCompletedPic1.Image = Nothing
    lblCompletedDescription1.Text = ""

    lblCompletedTitle2.Text = ""
    lblCompletedStatus2.Text = ""
    lblCompletedDay2.Text = ""
    lblCompletedPic2.Image = Nothing
    lblCompletedDescription2.Text = ""

    Dim totalRows As Integer = dgvCompletedData.Rows.Count - 1
    If totalRows < 0 Then
        cardCompleted1.Visible = False
        cardCompleted2.Visible = False
        Exit Sub
    End If

    Dim completedCount As Integer = 0
    Dim labelIndex As Integer = 2

    For i As Integer = 0 To totalRows
        If dgvCompletedData.Rows(i).IsNewRow Then Continue For

        Dim status As String = dgvCompletedData.Item(6, i).Value.ToString()
        If status <> "Completed" Then Continue For

        completedCount += 1

        Dim title As String = dgvCompletedData.Item(1, i).Value.ToString()
        Dim created As Object = dgvCompletedData.Item(7, i).Value
        Dim desc As String = dgvCompletedData.Item(4, i).Value.ToString()

        Dim timeAgoText As String = ""
        If created IsNot DBNull.Value Then
            Dim createdDate As DateTime = Convert.ToDateTime(created)
            timeAgoText = GetTimeAgo(createdDate)
        End If



        Dim img As Image = Nothing
        Dim imgData As Object = dgvCompletedData.Item(5, i).Value
        If imgData IsNot DBNull.Value Then
            Dim bytes As Byte() = CType(imgData, Byte())
            Using ms As New IO.MemoryStream(bytes)
                img = Image.FromStream(ms)
            End Using
        End If

        ' Set data to controls
        Select Case labelIndex
            Case 2
                lblCompletedTitle1.Text = title
                lblCompletedStatus1.Text = status
                lblCompletedDay1.Text = "Completed: " & timeAgoText
                lblCompletedPic1.Image = img
                lblCompletedDescription1.Text = desc
            Case 1
                lblCompletedTitle2.Text = title
                lblCompletedStatus2.Text = status
                lblCompletedDay2.Text = "Completed: " & timeAgoText
                lblCompletedPic2.Image = img
                lblCompletedDescription2.Text = desc
        End Select

        labelIndex -= 1
        If labelIndex = 0 Then Exit For
    Next

    ' Set card visibility based on count of completed tasks
    cardCompleted1.Visible = (completedCount >= 1)
    cardCompleted2.Visible = (completedCount >= 2)
End Sub
```

```
Public Sub LoadTaskProgress()
    Dim dt As DataTable = Query("SELECT SUM(CASE WHEN Status = 'Completed' THEN 1 ELSE 0 END) AS CompletedCount, SUM(CASE WHEN Status = 'In Progress
```

```
' THEN 1 ELSE 0 END) AS InProgressCount, SUM(CASE WHEN Status = 'Not Started' THEN 1 ELSE 0 END) AS NotStartedCount, COUNT(*) AS TotalCount FROM Information")
```

```
    If dt IsNot Nothing AndAlso dt.Rows.Count > 0 Then
        Dim row = dt.Rows(0)

        Dim completed As Integer = If(IsDBNull(row("CompletedCount")), 0, Convert.ToInt32(row("CompletedCount")))
        Dim inProgress As Integer = If(IsDBNull(row("InProgressCount")), 0, Convert.ToInt32(row("InProgressCount")))
        Dim notStarted As Integer = If(IsDBNull(row("NotStartedCount")), 0, Convert.ToInt32(row("NotStartedCount")))
        Dim total As Integer = If(IsDBNull(row("TotalCount")), 0, Convert.ToInt32(row("TotalCount")))

        ' Prevent divide by zero
        If total > 0 Then
            Dim percentCompleted As Integer = CInt((completed / total) * 100)
            Dim percentInProgress As Integer = CInt((inProgress / total) * 100)
            Dim percentNotStarted As Integer = CInt((notStarted / total) * 100)

            cpCompleted.Value = percentCompleted
            lblCompleted.Text = percentCompleted.ToString() & "%"

            cpProgress.Value = percentInProgress
            lblProgress.Text = percentInProgress.ToString() & "%"

            cpNotStarted.Value = percentNotStarted
            lblNotstarted.Text = percentNotStarted.ToString() & "%"
        Else
            ' No data
            cpCompleted.Value = 0 : lblCompleted.Text = "0%"
            cpProgress.Value = 0 : lblProgress.Text = "0%"
            cpNotStarted.Value = 0 : lblNotstarted.Text = "0%"
        End If
    Else
        ' No data
        cpCompleted.Value = 0 : lblCompleted.Text = "0%"
        cpProgress.Value = 0 : lblProgress.Text = "0%"
        cpNotStarted.Value = 0 : lblNotstarted.Text = "0%"
    End If

End Sub
```

```
Public Sub viewdataAlltask()
    ' Loads all available tasks
    Dim con1 As New SqlConnection(Connection.ConnString)
    Dim sql As String = "SELECT TaskID, Title, TaskDate, PriorityLevel, Description, Image, Status, CreatedAt FROM Information WHERE Status <> 'Completed'"

    Dim Adapter As New SqlDataAdapter(sql, con1)
    Dim data As New DataTable("Information")
    Adapter.Fill(data)

    ' ✅ Add "TimeAgo" column if not exists
    If Not data.Columns.Contains("TimeAgo") Then
        data.Columns.Add("TimeAgo", GetType(String))
    End If

    ' ✅ Fill the TimeAgo field with relative time
    For Each row As DataRow In data.Rows
        Dim createdAt As DateTime = Convert.ToDateTime(row("CreatedAt"))
        row("TimeAgo") = GetTimeAgo(createdAt)
    Next

    dgvMytaskData.DataSource = data
    dgvMytaskData.Columns("CreatedAt").Visible = False
    dgvMytaskData.Columns("TimeAgo").HeaderText = "Create"
    dgvMytaskData.Columns(0).Visible = False ' TaskID
    dgvMytaskData.Columns(4).Visible = False ' Description
    dgvMytaskData.Columns(5).Visible = False ' Image
End Sub
```

```
Public Sub MyTaskDataView()
    If dgvMytaskData.CurrentRow Is Nothing Then Exit Sub

    Dim i As Integer = dgvMytaskData.CurrentRow.Index
    TaskID.Text = dgvMytaskData.Item(0, i).Value.ToString()
    lblViewTitle1.Text = dgvMytaskData.Item(1, i).Value.ToString()
    lblViewTitle2.Text = dgvMytaskData.Item(1, i).Value.ToString()
    lblViewDeadline.Text = dgvMytaskData.Item(2, i).Value.ToString()
    lblViewPriority.Text = dgvMytaskData.Item(3, i).Value.ToString()
    Dim Priority As String = dgvMytaskData.Item(3, i).Value.ToString()
    lblViewPriority.ForeColor = GetPriorityColor(Priority)
    lblViewDescription.Text = dgvMytaskData.Item(4, i).Value.ToString()
    Dim imgData As Object = dgvMytaskData.Item(5, i).Value

    If imgData IsNot DBNull.Value AndAlso imgData IsNot Nothing Then
        Dim bytes As Byte() = CType(imgData, Byte())
        Using ms As New IO.MemoryStream(bytes)
            picViewImage.Image = Image.FromStream(ms)
        End Using
    Else
        picViewImage.Image = Nothing ' Or a default image
    End If

    lblViewCreated.Text = dgvMytaskData.Item(7, i).Value.ToString()
    lblViewStatus.Text = dgvMytaskData.Item(6, i).Value.ToString()
    Dim Status As String = dgvMytaskData.Item(6, i).Value.ToString()
    ' Uncheck all first to avoid multiple being checked

    lblViewStatus.ForeColor = GetStatusColor(Status)

    isLoading = True

    cbTStatusCompleted.Checked = False
    cbTStatusInProgress.Checked = False
    cbTStatusNotStarted.Checked = False

    Select Case Status
        Case "Completed"
            cbTStatusCompleted.Checked = True
        Case "In Progress"
            cbTStatusInProgress.Checked = True
        Case "Not Started"
            cbTStatusNotStarted.Checked = True
    End Select

    isLoading = False
End Sub
```

```
Public Sub MyvitalTaskview()
    If dgvMyvital.CurrentRow Is Nothing Then Exit Sub

    Dim i As Integer = dgvMyvital.CurrentRow.Index
    TaskID.Text = dgvMyvital.Item(0, i).Value.ToString()
    lblVitalTitle1.Text = dgvMyvital.Item(1, i).Value.ToString()
    lblVitalTitle2.Text = dgvMyvital.Item(1, i).Value.ToString()
    lblVitalDeadline.Text = dgvMyvital.Item(2, i).Value.ToString()
    lblVitalPriority.Text = dgvMyvital.Item(3, i).Value.ToString()
    Dim Priority As String = dgvMyvital.Item(3, i).Value.ToString()
    lblVitalPriority.ForeColor = GetPriorityColor(Priority)


    lblVitalDescription.Text = dgvMyvital.Item(4, i).Value.ToString()

    Dim imgData As Object = dgvMyvital.Item(5, i).Value
    If imgData IsNot DBNull.Value AndAlso imgData IsNot Nothing Then
        Dim bytes As Byte() = CType(imgData, Byte())
        Using ms As New IO.MemoryStream(bytes)
            lblVitalPic.Image = Image.FromStream(ms)
        End Using
    Else
        lblVitalPic.Image = Nothing ' Or a default image
    End If

    lblVitalCreated.Text = dgvMyvital.Item(7, i).Value.ToString()

    lblVitalStatus.Text = dgvMyvital.Item(6, i).Value.ToString()

    Dim Status As String = dgvMyvital.Item(6, i).Value.ToString()
    ' Uncheck all first to avoid multiple being checked
    lblVitalStatus.ForeColor = GetStatusColor(Status)
    isLoading = True

    cbVStatusCompleted.Checked = False
    cbVStatusInProgress.Checked = False
    cbVStatusNotStarted.Checked = False
```

```
            Select Case Status
                Case "Completed"
                    cbVStatusCompleted.Checked = True
                Case "In Progress"
                    cbVStatusInProgress.Checked = True
                Case "Not Started"
                    cbVStatusNotStarted.Checked = True
            End Select

            isLoading = False


            ' ✅ Show "X days ago" in lblDayComplete
            Dim createdAt As DateTime
            If DateTime.TryParse(dgvMyvital.Item(7, i).Value.ToString(), createdAt) Then
                lblVitalDay.Text = GetTimeAgo(createdAt)
            Else
                lblVitalDay.Text = "N/A"
            End If
        End Sub
```

```
Public Sub ViewDeletedTasks()
    ' Clean up old tasks first
    ' <-- This line calls your new utility

    ' Load remaining tasks
    Dim con1 As New SqlConnection(Connection.ConnString)
    Dim sql As String = "SELECT TaskID, Title, TaskDate, PriorityLevel, Description, Image, Status, Deleted FROM TaskInformationDeleted ORDER BY Deleted DESC"

    Dim adapter As New SqlDataAdapter(sql, con1)
    Dim data As New DataTable("TaskInformationDeleted")
    adapter.Fill(data)

    If Not data.Columns.Contains("TimeAgo") Then
        data.Columns.Add("TimeAgo", GetType(String))
    End If

    For Each row As DataRow In data.Rows
        Dim deletedAt As DateTime = Convert.ToDateTime(row("Deleted"))
        row("TimeAgo") = GetTimeAgo(deletedAt)
    Next

    dgvDeletedTask.DataSource = data

    dgvDeletedTask.Columns("TaskID").Visible = False
    dgvDeletedTask.Columns("Description").Visible = False
    dgvDeletedTask.Columns("Image").Visible = False

    dgvDeletedTask.Columns("TaskDate").HeaderText = "Deadline"
    dgvDeletedTask.Columns("PriorityLevel").HeaderText = "Priority"
    dgvDeletedTask.Columns("TimeAgo").HeaderText = "Deleted"
End Sub
```

```
Public Sub MyTaskDataViewUpdate(form As TaskInputAndDisplay)
    If dgvMytaskData.CurrentRow Is Nothing Then Exit Sub

    Dim i As Integer = dgvMytaskData.CurrentRow.Index
    form.txtTitle.Text = dgvMytaskData.Item(1, i).Value.ToString()
    form.dtpDeadline.Text = dgvMytaskData.Item(2, i).Value.ToString()

    Dim Priority As String = dgvMytaskData.Item(3, i).Value.ToString()
    form.cbExtreme.Checked = (Priority = "Extreme")
    form.cbModerate.Checked = (Priority = "Moderate")
    form.cbLow.Checked = (Priority = "Low")

    Dim Status As String = dgvMytaskData.Item(6, i).Value.ToString()
    form.cbStatusCompleted.Checked = (Status = "Completed")
    form.cbStatusInProgress.Checked = (Status = "In Progress")
    form.cbStatusNotStarted.Checked = (Status = "Not Started")

    Dim imgData As Object = dgvMytaskData.Item(5, i).Value
    If imgData IsNot DBNull.Value AndAlso imgData IsNot Nothing Then
        Dim bytes As Byte() = CType(imgData, Byte())
        Using ms As New IO.MemoryStream(bytes)
            form.pcImage.Image = Image.FromStream(ms)
        End Using
    Else
        form.pcImage.Image = Nothing
    End If

    form.txtDiscription.Text = dgvMytaskData.Item(4, i).Value.ToString()
    form.dtpCreated.Text = dgvMytaskData.Item(7, i).Value.ToString()
    form.taskId.Text = dgvMytaskData.Item(0, i).Value.ToString()
End Sub
```

```
Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDelete.Click
    If dgvMytaskData.CurrentRow Is Nothing Then
        MsgBox("No task selected.", MsgBoxStyle.Exclamation)
        Exit Sub
    End If

    ' ✅ Ask for confirmation before deleting
    Dim result As DialogResult = MessageBox.Show("Are you sure you want to delete this task?", "Confirm Deletion", MessageBoxButtons.YesNo, MessageBoxIcon.Warning)

    If result = DialogResult.No Then Exit Sub

    ' Get TaskID from the selected row
    Dim taskId As Integer = Convert.ToInt32(dgvMytaskData.CurrentRow.Cells("TaskID").Value)

    ' Call the function to archive and delete
    If ArchiveAndDeleteTask(taskId) Then
        viewdataAlltask() ' Refresh grid

        panelviewdatainfohide()
    End If

    ViewDeletedTasks()
End Sub
```

```
Public Sub MyDletedTaskDataView()
    If dgvDeletedTask.CurrentRow Is Nothing Then Exit Sub

    Dim i As Integer = dgvDeletedTask.CurrentRow.Index
    TaskID.Text = dgvDeletedTask.Item(0, i).Value.ToString()
    lblDeletedtitle.Text = dgvDeletedTask.Item(1, i).Value.ToString()
    lblDeletedTilte1.Text = dgvDeletedTask.Item(1, i).Value.ToString()
    lblDeletedDeadline.Text = dgvDeletedTask.Item(2, i).Value.ToString()
    lblDeletedPriority.Text = dgvDeletedTask.Item(3, i).Value.ToString()
    lblDeletedDescription.Text = dgvDeletedTask.Item(4, i).Value.ToString()

    Dim imgData As Object = dgvDeletedTask.Item(5, i).Value
    If imgData IsNot DBNull.Value AndAlso imgData IsNot Nothing Then
        Dim bytes As Byte() = CType(imgData, Byte())
        Using ms As New IO.MemoryStream(bytes)
            lblDeletedPic.Image = Image.FromStream(ms)
        End Using
    Else
        lblDeletedPic.Image = Nothing ' Or a default image
    End If

    lblDeletedCreated.Text = dgvDeletedTask.Item(7, i).Value.ToString()
    lblDeletedStatus.Text = dgvDeletedTask.Item(6, i).Value.ToString()

    ' ✅ Show "X days ago" in lblDayComplete
    Dim createdAt As DateTime
    If DateTime.TryParse(dgvDeletedTask.Item(7, i).Value.ToString(), createdAt) Then
        lblDeletedDay.Text = GetTimeAgo(createdAt)
    Else
        lblDeletedDay.Text = "N/A"
    End If
End Sub
```

```
ivate Sub btnRecover_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btnRecover.Click
  Try
      ' Check if a row is selected
      If dgvDeletedTask.CurrentRow Is Nothing Then
          MessageBox.Show("Please select a task to recover.", "No Selection", MessageBoxButtons.OK, MessageBoxIcon.Warning)
          Exit Sub
      End If

      ' Get the selected TaskID
      Dim i As Integer = dgvDeletedTask.CurrentRow.Index
      Dim taskId As Integer = Convert.ToInt32(dgvDeletedTask.Item(0, i).Value)

      ' Confirm recovery
      Dim result As DialogResult = MessageBox.Show("Are you sure you want to recover this task?", "Confirm Recovery", MessageBoxButtons.YesNo, MessageBoxIcon.Questio
      If result = DialogResult.No Then Exit Sub

      ' Attempt to recover the task
      If Utilities.RecoverTask(taskId) Then
          MessageBox.Show("Task recovered successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information)

          ' Refresh views
          MyDletedTaskDataView()
          viewdataAlltask()
          viewdataAlltaskCompleted()
          ViewDeletedTasks()
          ClearDeletedTaskDetails()
      Else
          MessageBox.Show("Failed to recover task.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
      End If

  Catch ex As Exception
      MessageBox.Show("An error occurred while recovering the task. " & vbCrLf & "Details: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
  End Try

d Sub
```

```
Private Sub btnBackup_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnBackup.Click
    Try
        ' Ask user to confirm backup
        Dim result As DialogResult = MessageBox.Show("Do you want to create a backup of the database?", "Confirm Backup", MessageBoxButtons.YesNo, MessageBoxIcon.Question)

        If result = DialogResult.Yes Then
            ' Define backup folder and file name
            Dim backupFolder As String = "C:\TodoListBackupFolder\"

            ' Create folder if it doesn't exist
            If Not IO.Directory.Exists(backupFolder) Then
                IO.Directory.CreateDirectory(backupFolder)
            End If

            ' Build full backup file path with timestamp
            Dim backupFilePath As String = backupFolder & "DBTodoList_" & DateTime.Now.ToString("yyyyMMdd_HHmmss") & ".bak"

            ' Call the backup function in Connection module
            If Connection.BackupDatabase(backupFilePath) Then
                MessageBox.Show("Backup completed successfully." & vbCrLf & "File: " & backupFilePath, "Backup Success", MessageBoxButtons.OK, MessageBoxIcon.Information)
            Else
                MessageBox.Show("Backup failed.", "Backup Failed", MessageBoxButtons.OK, MessageBoxIcon.Error)
            End If
        Else
            ' User chose No, do nothing or inform
            MessageBox.Show("Backup cancelled by user.", "Backup Cancelled", MessageBoxButtons.OK, MessageBoxIcon.Information)
        End If

    Catch ex As Exception
        MessageBox.Show("An error occurred: " & ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

Task Input and Display

```
Private Sub btnAddtask_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnAddtask.Click
    Try
        If txtTitle.Text.Trim() = "" Or txtDiscription.Text.Trim() = "" Or
            (cbExtreme.Checked = False AndAlso cbModerate.Checked = False AndAlso cbLow.Checked = False) Or
            (cbStatusCompleted.Checked = False AndAlso cbStatusInProgress.Checked = False AndAlso cbStatusNotStarted.Checked = False) Or
             (pcImage.Image Is Nothing) Then

            MessageBox.Show("Please fill out all required fields.", "Missing Information", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Exit Sub
        End If



        Dim DeadlineDate As DateTime = dtpDeadline.Value
        Dim CreatedAt As DateTime = DateTime.Now

        ' Example: Get priority from radio buttons or dropdown (replace with your control names)
        Dim priority As String = "Low" ' default
        If cbExtreme.Checked Then
            priority = "Extreme"
        ElseIf cbModerate.Checked Then
            priority = "Moderate"
        ElseIf cbLow.Checked Then
            priority = "Low"
        End If

        Dim Status As String = "Not Started" ' default
        If cbStatusCompleted.Checked Then
            Status = "Completed"
        ElseIf cbStatusInProgress.Checked Then
            Status = "In Progress"
        ElseIf cbStatusNotStarted.Checked Then
            Status = "Not Started"
        End If



       ' Get image data from PictureBox or file dialog (replace pbImage with your PictureBox name)
       Dim imageData As Byte() = Nothing
       If pcImage.Image IsNot Nothing Then
           Using ms As New IO.MemoryStream()
               pcImage.Image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg) ' or PNG depending on your image format
               imageData = ms.ToArray()

           End Using
       End If

       If Utilities.InsertTask(txtTitle.Text, txtDiscription.Text, DeadlineDate, CreatedAt, priority, Status, imageData) Then
           ' Your existing code to refresh and hide the form

           clear()
           frmData.ShowDialog()
           Me.Hide()

           frmDashboard.viewdataAlltask()


       End If
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

```
Public Sub ImportImage()
    Dim ofd As New OpenFileDialog()
    ofd.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp;*.gif"

    If ofd.ShowDialog() = DialogResult.OK Then
        pcImage.Image = Image.FromFile(ofd.FileName)
    End If

    ' Update panel background depending on whether an image was loaded
    If pcImage.Image IsNot Nothing Then
        Panel1.BackgroundImage = Nothing
    Else
        Panel1.BackgroundImage = TO_DO_LIST_SYSTEM.My.Resources.Resources.ImageLogo
    End If
End Sub
```

```
Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnUpdate.Click
    Try
        If txtTitle.Text.Trim() = "" Or txtDiscription.Text.Trim() = "" Then
            MessageBox.Show("Please fill out all required fields.", "Missing Information", MessageBoxButtons.OK, MessageBoxIcon.Warning)
            Exit Sub
        End If

        Dim DeadlineDate As DateTime = dtpDeadline.Value
        Dim CreatedAt As DateTime = DateTime.Now


        Dim priority As String = "Low" ' default
        If cbExtreme.Checked Then
            priority = "Extreme"
        ElseIf cbModerate.Checked Then
            priority = "Moderate"
        ElseIf cbLow.Checked Then
            priority = "Low"
        End If

        Dim Status As String = "Not Started" ' default
        If cbStatusCompleted.Checked Then
            Status = "Completed"
        ElseIf cbStatusInProgress.Checked Then
            Status = "In Progress"
        ElseIf cbStatusNotStarted.Checked Then
            Status = "Not Started"
        End If



        Dim imageData As Byte() = Nothing
        If pcImage.Image IsNot Nothing Then
            Using ms As New IO.MemoryStream()
                ' Clone the image to avoid lock issues
                Using bmp As New Bitmap(pcImage.Image)
                    bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg)
                    imageData = ms.ToArray()
                End Using
            End Using
        End If
        If taskId.Text.Trim() = "" Then
            MessageBox.Show("Task ID is missing!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If


        If Utilities.UpdateTask(taskId.Text, txtTitle.Text, DeadlineDate, priority, txtDiscription.Text, imageData, CreatedAt, Status) Then
            clear()
            Dim data As New frmData
            data.Show()
            Me.Hide()
            frmDashboard.panelviewdatainfohide()
            frmDashboard.viewdataAlltaskCompleted()

        End If
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
End Sub
```

TESTING DOCUMENTATION

This section outlines the testing activities conducted for the To-Do List System. The purpose of testing is to ensure that the system meets the specified requirements, performs reliably, and provides a smooth user experience. Testing covered both functional requirements (verifying that each feature works as intended) and non-functional requirements (performance, usability, and reliability).

Test Plan

The objectives for this testing are to verify that the system satisfies all functional requirements (task management, calendar integration, backup/recovery, etc.) and ensure that non-functional requirements such as performance, usability, and reliability are met. The Testing will include the Core Functions (Task Management, Calendar, Completed Tasks, Recycle Bin, and Settings). Manual testing was performed by simulating user interactions and each requirement was mapped to at least one test case.

Functional Test Cases

Test Case ID	Requirement	Description	Test Steps	Expected Result	Status
TC01	Create Task	Verify new tasks can be created	Enter title, description, due date, priority, and status then Save	Task appears in task list and calendar	Pass
TC02	Recurring Task	Verify recurring tasks are saved	Create a daily recurring task	Task appears on multiple dates in calendar	Pass
TC03	Custom Task	Verify multiple specific dates	Create task with multiple custom dates	Task appears on all chosen dates	Pass
TC04	Calendar View	Verify calendar displays tasks	Open calendar (daily, weekly, monthly)	Tasks appear in correct slots	Pass
TC05	Edit Task	Verify task editing	Select a task then Edit and Save	Task updates in database and interface	Pass
TC06	Delete Task	Verify task deletion	Select a task then Delete	Task moves to Recycle Bin	Pass

TC07	Completed Task	Verify marking task as completed	Select a task then Mark Completed	Task appears in Completed tab	Pass
TC08	Data Persistence	Verify data is saved after restart	Create tasks then Restart app	Tasks remain in database	Pass
TC09	Settings Backup	Verify backup creation	Run backup from Settings	.bak file generated successfully	Pass
TC10	Settings Recovery	Verify database restoration	Run restore from Settings	Database restored from backup file	Pass

Non-Functional Test Cases

Test Case ID	Attribute	Description	Expected Result	Status
NTC01	Performance	Load tasks into system	Tasks load in under 7 seconds	Pass
NTC02	Usability	Test navigation flow	Users can reach all features in ≤ 3 clicks	Pass
NTC03	Reliability	Run app for 1 hour with multiple edits	No crashes or data loss	Pass
NTC04	Security	Attempt login with wrong password	Access denied	Pass

Test Results

All test cases were executed, and the system met both functional and non-functional requirements. Minor cosmetic issues were observed but did not affect system functionality.

Defect Reports

Defect ID	Description	Severity	Resolution	Status
D001	Calendar did not refresh immediately after deleting a task	Low	Added auto-refresh after delete	Fixed

D002	The Loading Screen Appears every time you press Home in Dashboard	Medium	Change the location of the loading function.	Fixed
D003	Spelling mistake for Forgot Password and Description	Low	Change the text of the label.	Fixed
D004	Backup failed if folder did not exist	Medium	Added folder check and auto-create logic	Fixed
D005	Default date of the Date Time Picker are always outdated	Medium	Change the value of Date Time Picker to current time.	Fixed

MAINTENANCE GUIDE

The purpose of this Maintenance Guide is to provide procedures and best practices for keeping the To-Do List System reliable, secure, and up-to-date. Proper maintenance ensures the system remains functional, protects user data, and allows for future improvements as requirements evolve.

Maintenance Procedures

Database Maintenance

1. Perform **regular backups** of the system using the built-in Backup function in **Settings**.
2. Test the **Recovery function** periodically to ensure backups can be restored successfully.

Application Maintenance

1. Review **error logs** for failed login attempts, backup failures, or unexpected behavior.
2. Clear or archive **Recycle Bin** tasks to prevent database bloating.
3. Update the application to a newer version by running the latest

Security Maintenance

1. Update passwords regularly using the Change Password feature.
2. Ensure Windows operating system and .NET Framework updates are installed.
3. Restrict access to configuration files and backup folders to authorized users only.

Version release management

Version Numbering

The system follows a semantic versioning scheme:

1. **MAJOR.MINOR.PATCH** (e.g., 1.0.0).
2. **MAJOR** – Significant updates or incompatible changes.
3. **MINOR** – New features added in a backward-compatible manner.
4. **PATCH** – Small bug fixes and improvements.

Builds:

1. Final builds are compiled in **Release Mode** in Visual Studio.
2. The output is packaged into an installer using **Inno Setup**.

Release Cycle:

1. **Alpha** – Internal testing of new features.
2. **Beta** – Shared with a limited number of users for testing.
3. **Stable Release** – Official version distributed to end-users.

Distribution:

1. Releases are distributed as versioned installer packages (e.g., TodoListSystem_v1.1.exe).
2. The installer overwrites old versions while preserving user data in the database.

Release Notes:

1. Each release includes a short changelog documenting new features, bug fixes, and improvements.
2. Example:
    1. v1.1.0 – Added calendar task highlighting, fixed recycle bin restore bug.