

TO-DO LIST SYSTEM WITH CALENDAR INTEGRATION
FOR
ICARUS SHIRTS

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

DESIGN DOCUMENT

INTRODUCTION

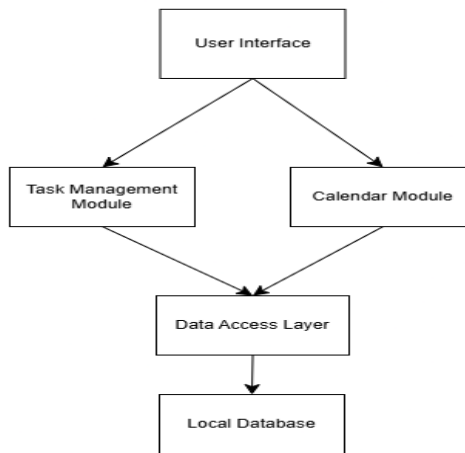
This document presents the design specifications for the To-Do List System with Calendar Integration, developed for Icarus Shirts, a local business that specializes in personalized dry-fit shirts and custom printing services. The purpose of this document is to provide a comprehensive guide for the design, implementation, and testing of the system. It ensures that all members of the development team have a clear understanding of the system's architecture, database structure, user interface, and functional components. Furthermore, this document serves as a reference for clients to review and validate the planned design before and during the development process, ensuring that the system aligns with the agreed requirements and operational goals.

The To-do List System with Calendar Integration is a desktop application that provides a centralized platform for managing tasks and deadlines in an efficient manner. Designed for a single-user environment, the system allows the user to create, edit, update, and delete tasks, while visually organizing them within a calendar interface. It supports a variety of scheduling options including daily, weekly, monthly, one-time, and custom dates, providing flexibility to adapt to the client's operational workflow.

This system is beneficial for Icarus Shirts, where production planning and order tracking are critical to maintaining customer satisfaction. By replacing manual methods with a digital solution, the system improves efficiency, reduces errors, and provides a clear visual overview of all scheduled tasks and priorities. The integration of task management with a visual calendar aims to simplify planning while increasing productivity and reliability for the business owner.

This document covers the planned system architecture, database design, user interface layout, component breakdown, data flow diagrams, performance considerations, and the overall deployment plan. It provides the foundation where the development process will be built, ensuring consistency, efficiency, and alignment with the client's needs from initial development to deployment and maintenance.

SYSTEM ARCHITECTURE



High level Components

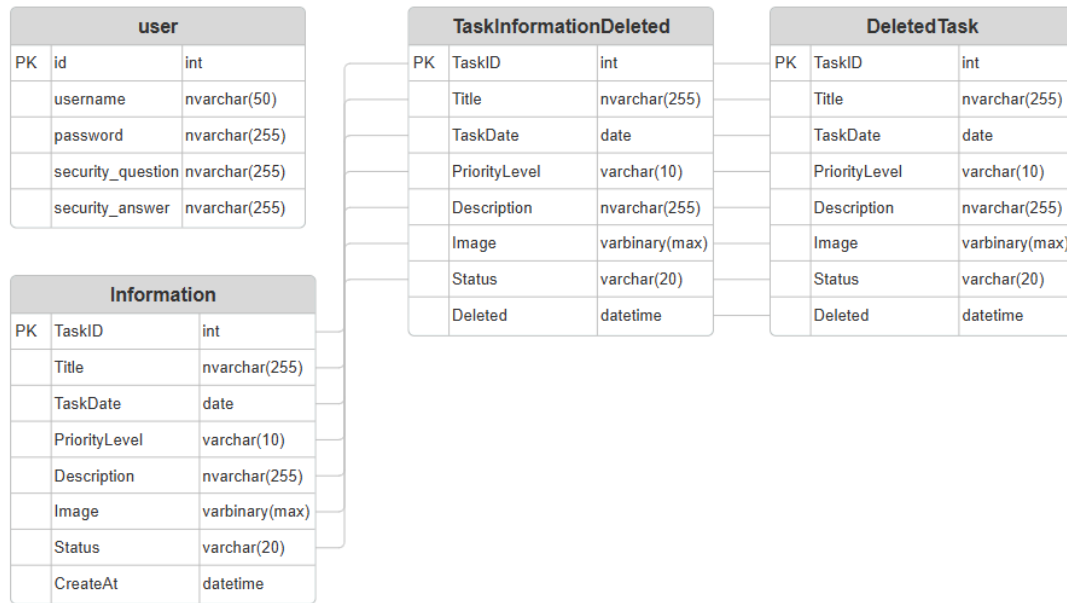
1. **User:** The client that will use and operate the system
2. **Task Management Module:** This is responsible for creating, updating, and deleting tasks
3. **Calendar Module:** Organizes and displays tasks in a date-oriented view.
4. **Data Access Layer:** Handles the communication of the application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

DATABASE DESIGN

This section shows the Database Design of the To-do list system. This shows the tables and the relationship it has with other tables and the contents inside each table.



Tables

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, int):** Unique identifier for each user.
- **username (nvarchar):** The name the user uses to log in.
- **password (nvarchar):** The user's password for authentication (stored in hashed form for security).
- **security_question:** The user's chosen security question for user authentication.

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.

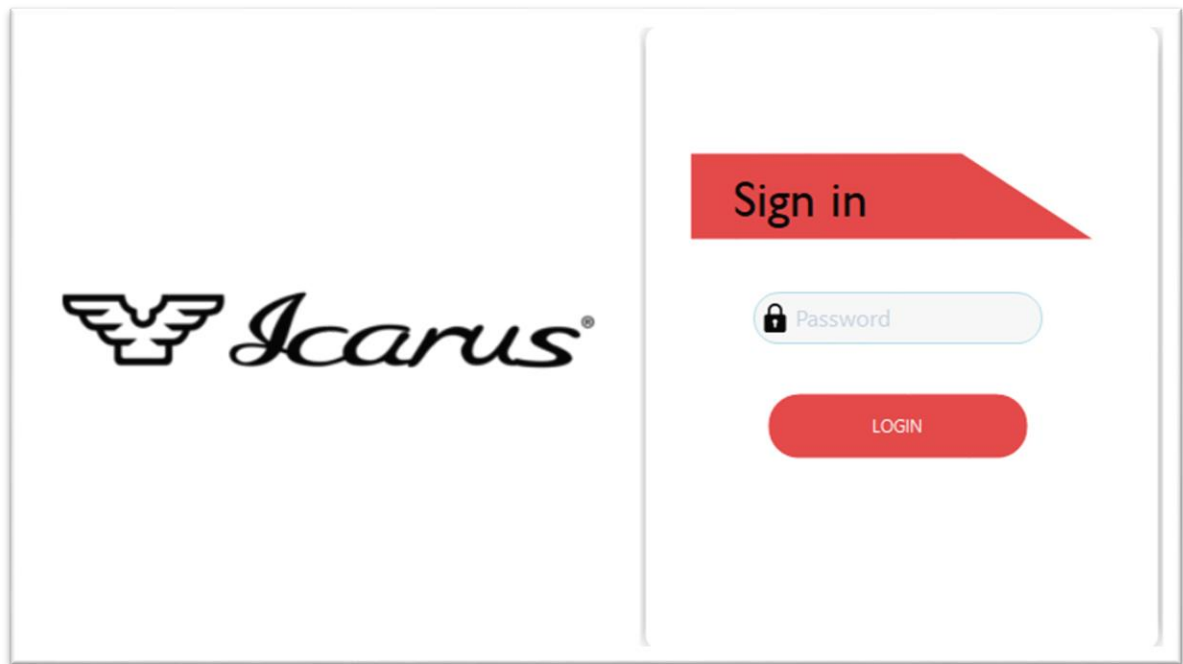
3. Task Information Deleted Table (Deleted Tasks)

This table stores **historical or deleted tasks** for archival or audit purposes. 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

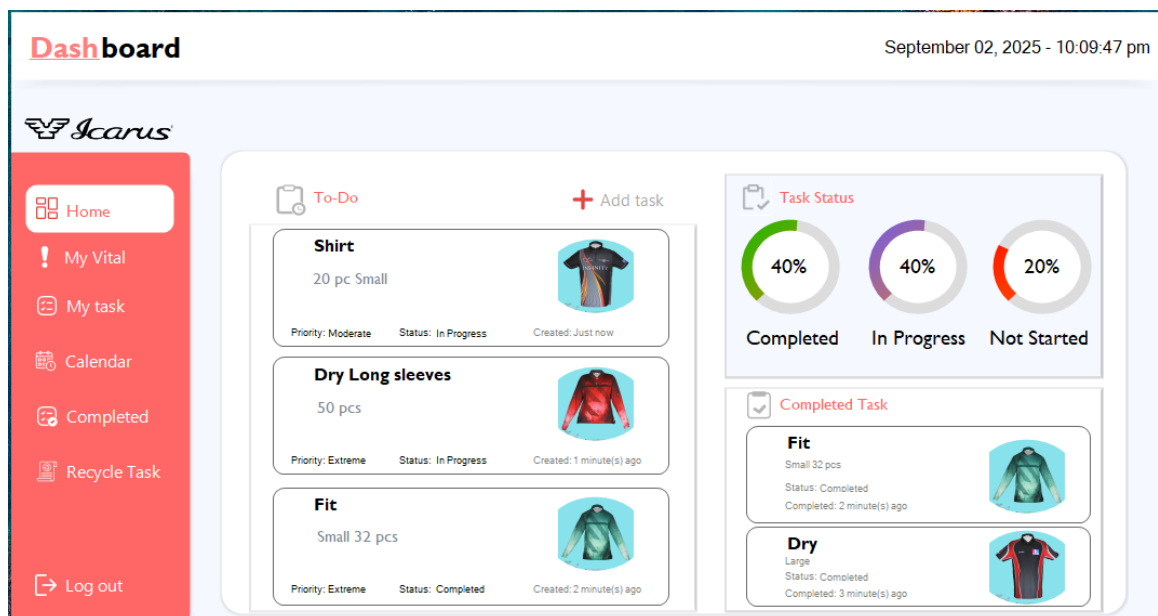
USER INTERFACE DESIGN

Log In



This is the Log-In Screen for our To-do List System where you will put your password to enter

Dashboard (Tab)



After logging in, the user is directed to the main dashboard, where they can view and manage their tasks. The dashboard includes a sidebar for navigation, allowing quick access to pages such as Home, My Vital, My Task, Completed, and Recycle Task.

In the center, users can see three recently to-do tasks presented as cards, each showing task details like title, client, status, priority, creation time, and an image preview.

On the right side, there's a visual summary of task statuses Completed, In Progress, and Not Started displayed in a circular progress chart. Below it, recently two completed tasks are shown with their details and images.

Add New Task (Tab)

The screenshot displays the 'Add New Task' form within the Icarus dashboard. The interface features a red sidebar on the left with navigation links: Home, My Vital, My task, Calendar, Completed, Recycle Task, and Log out. The main content area is titled 'Add New Task' and includes a 'Go Back' link. The form fields are as follows:

- Title:** A text input field containing 'Dry Fit'.
- Date:** A date picker showing '03/09/2025'.
- Priority:** Three radio buttons labeled 'Extreme', 'Moderate', and 'Low'. The 'Low' option is selected.
- Description:** A large text area for entering task details.
- Upload Image:** A button labeled 'Upload Image' next to a preview of a red and black athletic shirt.
- Status:** Three radio buttons labeled 'Complete', 'In progress', and 'Not Started'. The 'Not Started' option is selected.

A red 'Done' button is located at the bottom of the form. On the right side of the dashboard, a circular progress chart shows '0%' completion, and a section titled 'Not Started' is visible.

The Add New Task interface allows users to input task details including title, date, priority, description, image upload, and status before saving the task.

My Vital (Tab)

To-Do

September 02, 2025 - 10:11:12 pm

Icarus

Home

My Vital

My task

Calendar

Completed


Recycle Task

Log out

My Vital

TITLE

Title	TaskDate	PriorityLevel	Status	Create
Bulk Print – ...	9/12/2025	Extreme	Not Started	15 hour(s) a...
dfsdf	8/27/2025	Extreme	Completed	16 hour(s) a...



Bulk Print – TechNova Staff

Priority: Extreme

Status: Not Started

Created: 8/31/2025

Deadline: 9/12/2025

Deadline: 15 hour(s) ago

Task Title: Bulk Print – TechNova Staff

Task Description:

Client: Icarus Shirt

Order #: IC-0912-004

Project: TechNova Annual Event Staff Shirts

- Quantity: 200 shirts

- Sizes: XS–XXL (as per attached breakdown)

- Shirt Color: Navy Blue

- Design Location: Left chest logo + back text

My Vital interface displays tasks marked with high priority, allowing quick access to urgent jobs and their detailed information.

My Task (Tab)

To-Do

September 01, 2025 - 03:33:28 AM

Icarus

Home

My Vital

My task

Calendar

Completed

Recycle Task

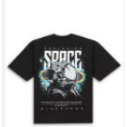
Log out

My Task

TITLE

+ Add task

Title	TaskDate	PriorityLevel	Status	Create
Custom Shir...	9/5/2025	Moderate	Not Started	15 hour(s) a...
Bulk Print – ...	9/12/2025	Extreme	Not Started	15 hour(s) a...
Neon Splas...	9/6/2025	Low	In Progress	15 hour(s) a...



Custom Shirt Print – "Space

Priority: Moderate

Status: Not Started

Created: 8/31/2025

Deadline: 9/5/2025

Task Title: Custom Shirt Print – "Space

Task Description:

Client: Icarus Shirt

Order #: IC-0925-001

Print Details:

- Design: "Space Warrior"

- Shirt Color: Black

- Size Range: M, L, XL (15 pieces total)

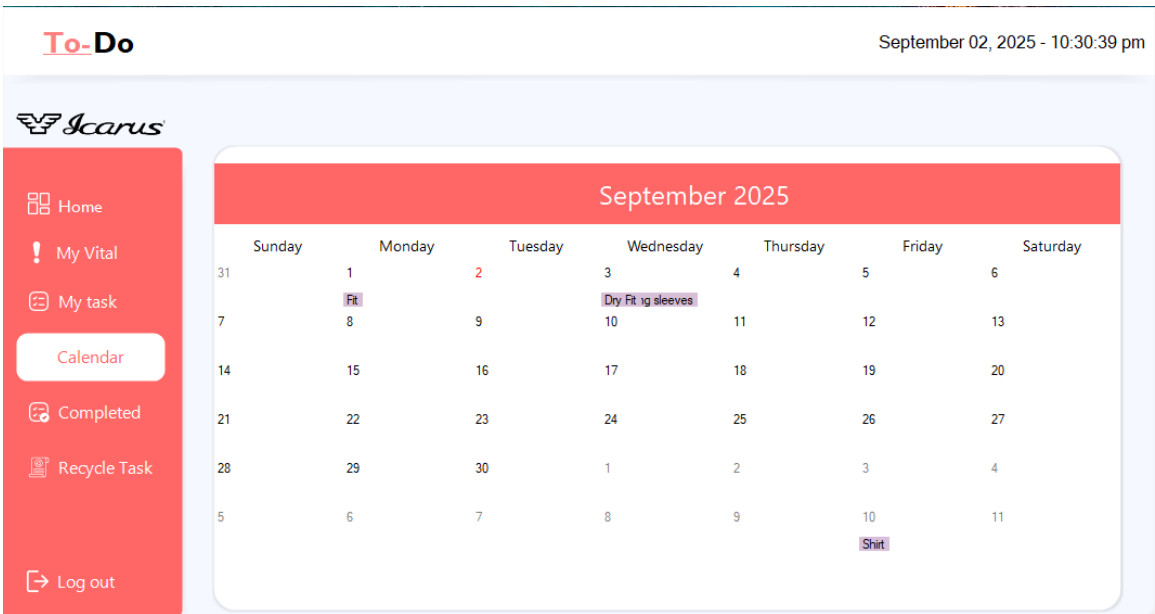
- Printing Area: Full front (A3 size)

- Ink: White and Neon Blue

- Material: Cotton 100%

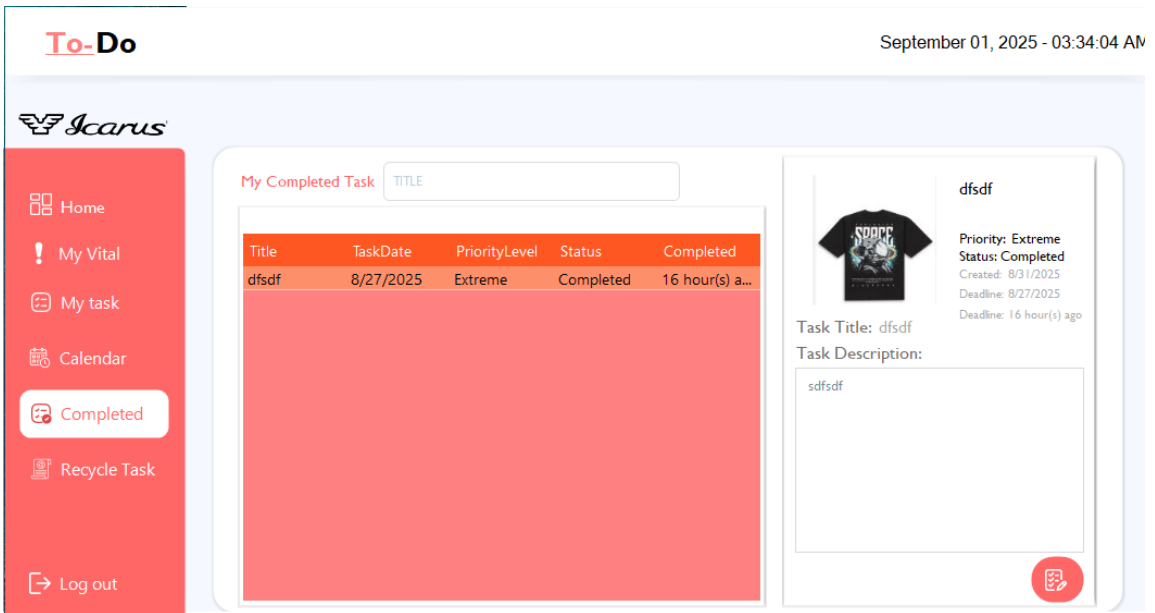
The My Task interface displays all tasks, allowing users to view, update, or delete any task as needed.

Calendar



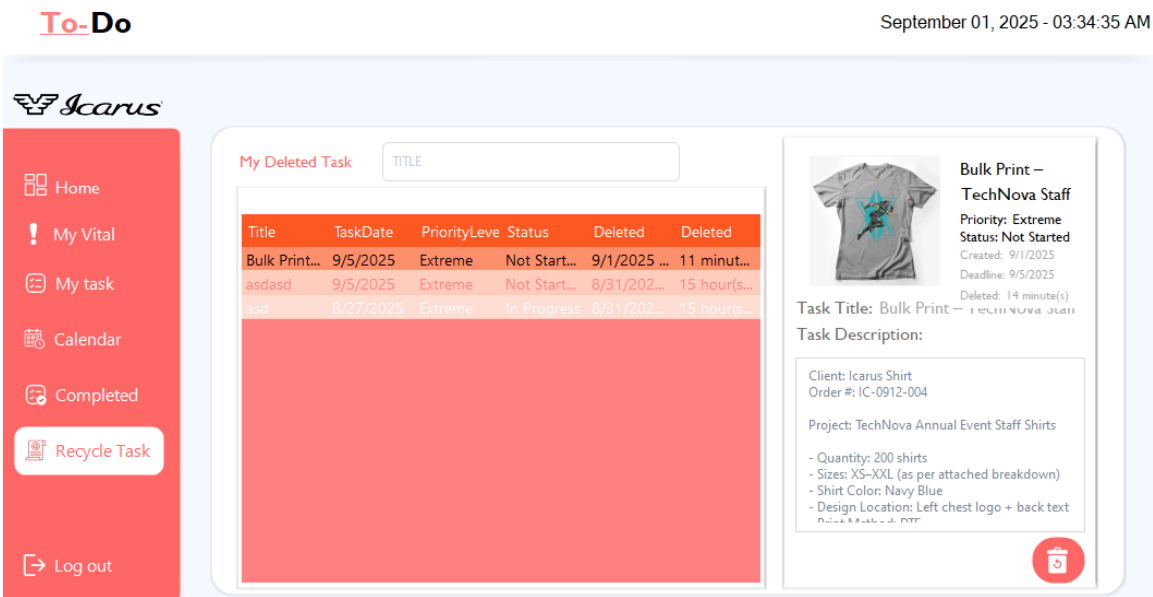
Calendar interface display the deadlines of all the task in the calendar

Completed (Tab)



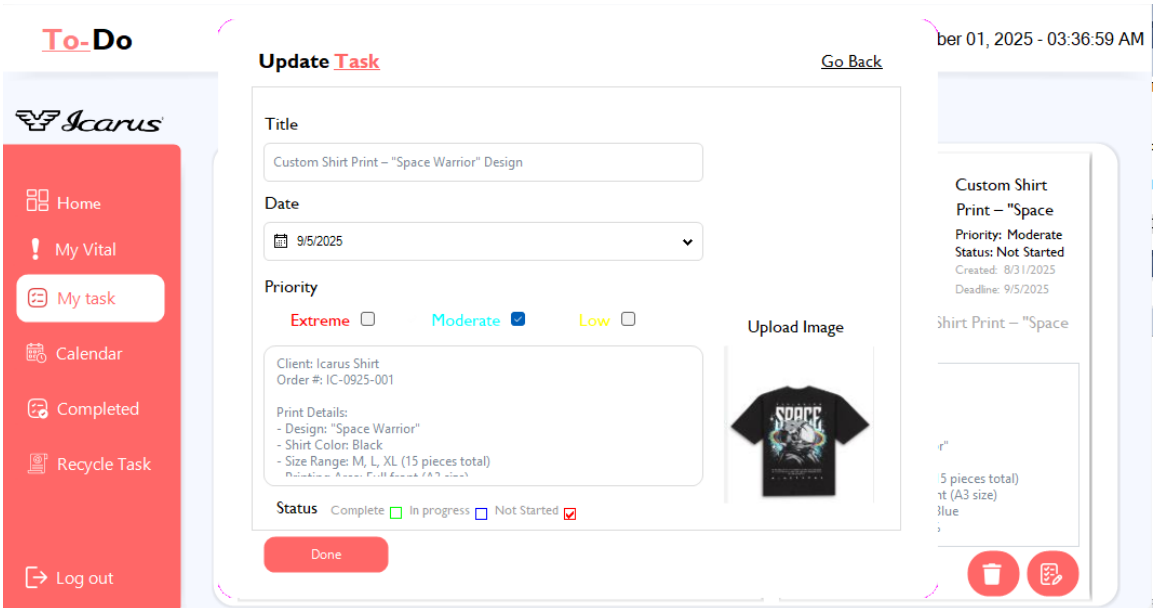
Completed Task interface displays all completed tasks, allowing users to view, update task as needed.

Recycle Task



Recycle Task interface displays Deleted tasks, allowing users to view, Recover the task.

Update Task (Tab)



The Update Task interface allows users to edit existing task details such as title, date, priority, status, description, and image.

COMPONENT DESIGN

The To-Do List Management System is structured around a series of well-integrated software components developed using Visual Basic .NET through the Windows Forms framework. Despite being a monolithic application, the system adheres to a modular design approach where each component has a clearly defined responsibility. The architecture ensures organized control over different areas of the system including task management, priority handling, user interaction, and database operations. Each module communicates with others through a centralized Data Access Layer that ensures consistent and secure data operations using parameterized SQL commands to prevent injection attacks and enforce input validation.

Authentication Modul

Handles secure user login. While currently implemented for a single user, the design supports future extension for multi-user functionality. It ensures that only authorized individuals can access the system by verifying input credentials against stored records.

Dashboard Module

Which acts as the main navigation interface. The dashboard displays an overview of system statistics such as total tasks, completed tasks, pending tasks, and task priority levels. It uses visual aids like progress indicators, summary panels, and real-time task counters to provide users with a quick understanding of their productivity status. The dashboard also allows users to navigate to other functional areas such as task management, vital tasks, completed tasks, and the recycle bin.

Task Management Module

Forms the core of the system. It enables users to add new tasks, view all existing tasks, and perform edit or delete operations. Each task contains important fields such as title, description, priority level, due date, status, and an image. When a task is created or updated, the system validates that all required fields are provided. Priority is selected from predefined levels including Extreme, Moderate, and Low, while task status can be set to

Not Started, In Progress, or Completed. Task images are converted into byte arrays before being stored in the database and rendered visually within the task view for user reference.

Vital Task Module

Automatically filters and displays tasks that have been marked with the highest priority level, typically labeled as “Extreme.” This module allows users to focus their attention on urgent matters by presenting those tasks in a dedicated panel. Task details are shown in real time, helping users make informed decisions about what needs immediate attention.

Completed Task Module

This component displays all completed tasks in a read-only interface to prevent accidental modification or deletion while allowing users to review their completed work for personal tracking or future reference.

Recycle Bin Module

Comes into play. Instead of permanently removing the task from the system, it moves the task to a backup table named TaskInformationDeleted. This design ensures data is not lost unintentionally and provides a mechanism to recover deleted tasks if needed. From this module, users can view all previously deleted tasks and choose to restore them to the main task list. This approach not only protects data integrity but also reinforces user confidence in using delete actions without fear of irreversible loss.

Update Task Module

Allows users to modify previously created tasks. When accessed, it retrieves the current data of the selected task and populates an editable form where users can update the title, description, deadline, priority level, status, and associated image. Upon saving, the updated data replaces the previous record in the database while ensuring that all validations are again passed successfully before submission.

DATA FLOW DIAGRAM

This section presents the Data Flow Diagram (DFD) for the Task Management System, illustrating how data flows between the user, core system modules, and internal processes. The DFD serves as a high-level conceptual model that highlights the interaction between external actors and internal system functionalities. It helps developers, designers, and stakeholders understand the sequence of operations, identify process boundaries, and track the movement of task-related information throughout the system. The DFD is essential for ensuring the system's requirements are met during development, maintenance, and future expansion.

DFD – Context Diagram (Level 0)

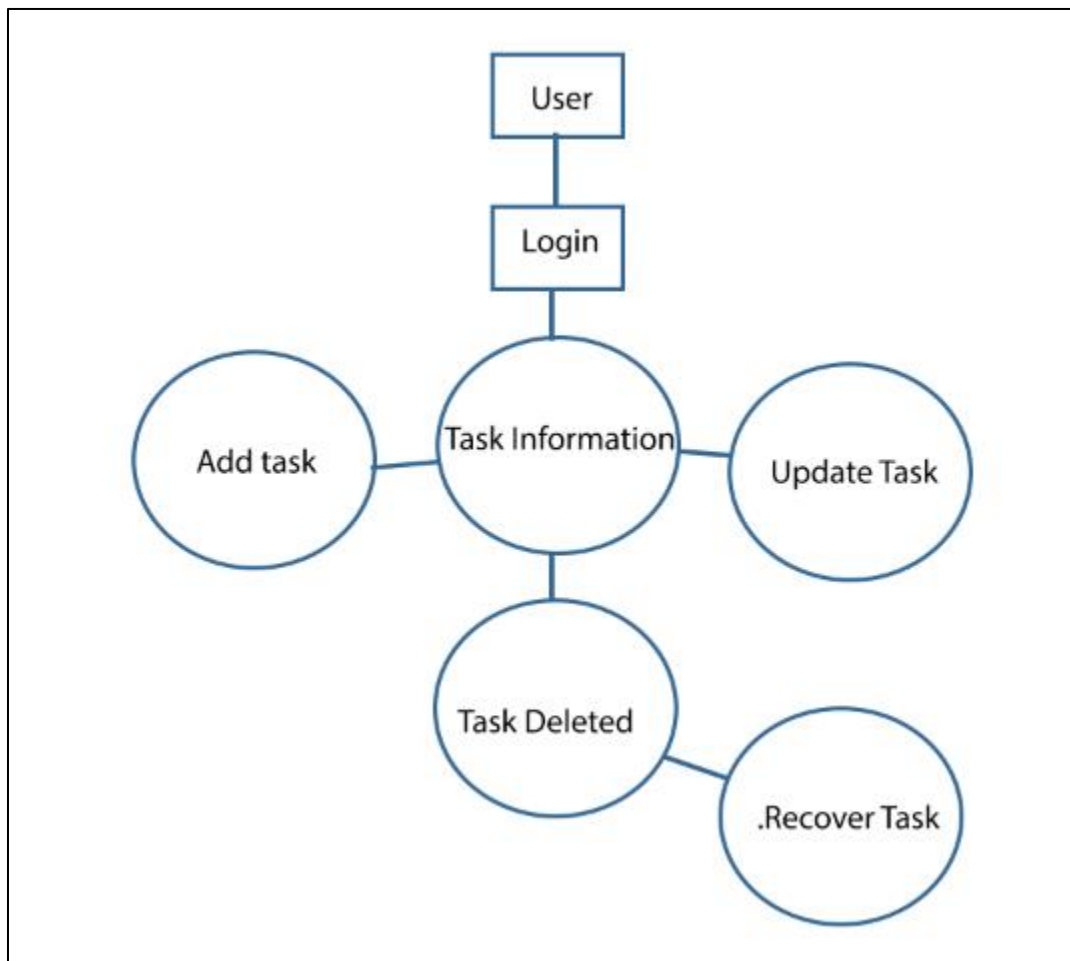


Figure 1. Data Flow Diagram (Level 0)

External Entities (Data Sources/Destinations)

This section outlines the external entities that interact directly with the Task Management System. These entities lie outside the system boundary but are crucial to its operation. They provide input to system processes and receive outputs in return. For this system, the primary external entity is the **User**, who initiates login, task creation, updates, deletion, and task recovery processes. The system is designed to be user-centric, ensuring all functionalities are accessible through the user's interactions.

- **User** – The user interacts with the system through login, adding new tasks, updating or deleting existing tasks, and recovering deleted ones.

Data Flows

This section describes how data moves between the external entity (User) and the internal processes of the Task Management System. Each data flow represents an action or response that forms part of the system's operational logic.

- **User → Login Credentials → Task Management System:** The user provides credentials to log into the system. Upon verification, access is granted to internal functionalities.
- **User → Task Details → Add Task:** After login, the user can submit new task information such as title, description, date, priority, image, and status to the system.
- **User → Updated Task Details → Update Task:** The user can modify an existing task's information, and the updated data is stored back into the system's task repository.
- **User → Delete Request → Task Information → Task Deleted:** When a task is deleted, it is not permanently removed but moved to a temporary store called "Task Deleted," maintaining a soft-delete mechanism.
- **User → Recover Request → Task Deleted → Task Information:** The user can restore previously deleted tasks from the "Task Deleted" module back into the active task list.

- **System → Task Summary / Details → User:** The system continuously returns updated task details, such as lists of active tasks, deleted tasks, and recovered tasks for the user's review.

System Processes (Level 0 Overview)

The **central process** in this system is the **Task Information** module, which handles all task-related data. The module is connected to multiple subprocesses such as Add Task, Update Task, Task Deleted, and Recover Task. Each subprocess performs a distinct operation but remains linked through a shared data layer, ensuring a smooth and logical flow of data throughout the system.

- **Task Information** acts as the core hub where all task records are stored, modified, or marked as deleted. It ensures that all user actions update or retrieve data from a single consistent source.
- **Add Task** receives new task input from the user and stores it within Task Information.
- **Update Task** enables changes to existing tasks, pushing updated data back to the main repository.
- **Task Deleted** temporarily holds deleted tasks and allows for possible recovery.
- **Recover Task** interacts with Task Deleted to restore selected tasks back into active status.

SECURITY DESIGN

The security of the standalone To-Do List system focuses primarily on protecting user credentials and ensuring that sensitive information such as passwords is securely handled to prevent unauthorized access. Given the system's standalone nature, the design emphasizes local security measures, including secure password management and user interface elements that enhance user trust during authentication.

Password Hashing

The system employs strong password hashing techniques to ensure that user passwords are never stored or transmitted in plaintext. Passwords are securely hashed using a cryptographic hashing algorithm, which incorporates salting and multiple iterations to mitigate risks such as brute force attacks or rainbow table exploits. This approach ensures that even if the local data storage is accessed, the original passwords cannot be easily retrieved or compromised.

User Interface Security Element

To reinforce security awareness and provide visual feedback during the login process, the system features a password input field designed with masked. Masking the password input prevents shoulder surfing and accidental exposure of sensitive credentials.

Additional Considerations

While the standalone nature of the system limits exposure to external network threats, users are encouraged to maintain device-level security to complement the system's protections. Regular backups and safe handling of the system's data files are recommended to prevent data loss or unauthorized access outside the application.

PERFORMANCE DESIGN

This section outlines the strategies and design considerations implemented to ensure that the To-Do List system performs efficiently and reliably under normal user workloads. The focus is on optimizing system responsiveness, minimizing delays during task operations, and ensuring smooth user interactions even when managing a large number of tasks. Key aspects include fast data processing, quick UI updates, and efficient resource use to provide a seamless and satisfying user experience.

Performance Requirements and Objectives

The system is designed to ensure that common user actions such as adding, updating, deleting, or recovering tasks complete within two seconds, providing immediate feedback and preventing user frustration. Loading the full task list or filtered views should not exceed three seconds, allowing users to quickly access their to-do items.

The application must efficiently handle multiple tasks and associated images without notable lag or increased memory consumption. Additionally, the system aims to maintain consistent performance when handling moderately large task lists to accommodate user growth and extended usage periods.

Strategies for Optimizing System Performance

To achieve these objectives, several performance optimization techniques are applied throughout the application architecture. First, the system employs efficient data storage and retrieval methods by using lightweight data structures and minimizing redundant read/write operations. Data loading is optimized through on-demand fetching and pagination techniques when displaying large task lists, reducing initial load time and memory footprint.

UI rendering is carefully managed by updating only changed components and utilizing deferred control initialization to avoid unnecessary processing during form load. Additionally, image handling is optimized by compressing uploaded images and caching image previews in memory to minimize repeated conversions and disk access. Resource management is emphasized by promptly releasing unused objects and managing memory allocation to prevent leaks or bloating.

Performance Testing Plan

To validate the system's responsiveness and resource utilization, routine performance testing is conducted. Load tests simulate typical user workflows involving bulk task management, including batch additions and deletions, to measure response times and UI fluidity. Stress testing evaluates system behavior when task data approaches higher volumes, ensuring stability and preventing slowdowns. Profiling tools monitor memory consumption and CPU usage during these tests, identifying bottlenecks or inefficient operations for subsequent optimization. User feedback is incorporated regularly to identify real-world performance issues, driving continuous improvements.

ERROR HANDLING AND LOGGING

This section explains the error handling and logging mechanisms implemented in the To-Do List system, designed to ensure smooth operation and provide clear feedback when issues arise. Since the system uses only password-based user authentication, the primary focus is on securely managing login errors and maintaining system stability during task operations.

Error Handling Mechanisms and Strategies

The To-Do List system employs structured exception handling to effectively manage unexpected runtime errors. All critical operations, especially those involving password verification and task management, are enclosed within Try-Catch-Finally blocks to handle exceptions such as invalid input, database connection issues, or file access problems.

User input is validated before processing to ensure passwords meet expected criteria and that all required fields are correctly filled, reducing the likelihood of errors. When errors occur during login, the system presents clear, user-friendly messages—such as informing the user when a password is incorrect—without revealing sensitive details.

For other unexpected errors, the system displays a generic error message to the user while logging detailed error information internally for developer review. Resource cleanup is managed within Finally blocks to ensure that connections and file handles are properly closed, maintaining system stability. Additionally, a global exception handler captures any unhandled exceptions to prevent application crashes and logs them for further analysis.

Logging Requirements and Specifications

Logging focuses primarily on authentication attempts and critical system errors. Each login attempt—whether successful or failed—is recorded with a timestamp and masked user identification to monitor suspicious activity without compromising privacy. System errors and exceptions encountered during task operations are logged with detailed stack traces, error codes, and contextual information to aid debugging.

Logs are stored locally in secure, append-only files with daily rotation to prevent excessive file size and ensure availability for troubleshooting. Each log entry follows a consistent format, including date, time, severity level, and description, enabling efficient parsing and review.

Error Codes and Messages

The system uses standardized error codes mapped to user-friendly messages to maintain consistency and simplify maintenance:

Error Code	Description	User Message
1001	Invalid login	Login failed
2001	Query Error	During SELECT queries
2002	Insert failed	During INSERT operation
2003	Updated failed	During Update operation
2004	Delete failed	During DELETE operation

Table 1. Error Codes and Messages

DEPLOYMENT PLAN

This section outlines the comprehensive strategy and detailed steps for deploying the To-Do List System into its production environment. It addresses preparatory tasks, installation processes, configuration settings, and post-deployment validation to ensure a seamless rollout from development to active use. The plan specifies hardware and software prerequisites, user onboarding, and data backup procedures to reduce downtime and data loss risks.

Overview of the Deployment Process

- **Microsoft SQL Server Setup:** Install and configure Microsoft SQL Server Express edition on a dedicated server or local machine. Configure secure authentication modes and create database users with minimum necessary permissions to safeguard the system.
- **Database Initialization:** Deploy the initial database schema for the To-Do List, including task tables, user authentication tables, and indexes. Establish automated backup jobs to ensure regular data backups and enable point-in-time recovery.
- **Application Distribution:** Package the Visual Basic Windows Forms application into an installer executable. Distribute this installer to user workstations through USB drives, network shares, or software management tools. Configure the application to securely store database connection strings in encrypted configuration files.
- **Guna UI Framework Integration:** The To-Do List System utilizes the Guna UI Framework to provide a modern, sleek, and user-friendly interface. Guna UI controls such as buttons, textboxes, panels, and animations are included in the deployment package and properly registered on client machines. This ensures consistency in visual design and enhanced user experience. The deployment includes all necessary runtime libraries required by Guna controls, avoiding UI glitches and improving responsiveness.

- **Configuration:** Configure application settings such as connection strings, password policy parameters, and logging preferences. These configurations are set either through the installer or via a secured configuration file editable only by administrators.
- **Testing and Validation:** Conduct comprehensive testing including functional validation of login, task management, and error handling features. Perform user acceptance testing with a sample of end-users to verify usability and performance under realistic conditions, including verifying the appearance and behavior of Guna UI components.
- **User Training and Documentation:** Provide user manuals and quick reference guides covering both application functionality and UI navigation using Guna controls. Organize a training session with the client before launch, the project team itself will lead the training session to clear up understanding.

Hardware and Software Requirements for Deployment

Client Workstations:

- Operating System: Windows 10 or later
- Framework: Microsoft .NET Framework 4.5 or higher
- Memory: Minimum 4GB RAM
- Processor: Dual-core CPU or better
- Storage: Minimum 200 MB free disk space for application, including Guna UI runtime files

Database Server:

- Operating System: Windows 10 or Windows 11
- SQL Server Edition: Microsoft SQL Server Express or higher

Configuration Management and Version Control Procedures

- **Source Control:** The source code and database scripts are maintained under Git repository management, employing standard branching strategies such as feature

branches, development, and master/main to streamline collaboration and release control.

- **Configuration Storage:** Environment-specific settings, including database connection strings and encryption keys, are stored outside the source code in encrypted configuration files. These files are excluded from version control to protect sensitive data.
- **Release Management:** Software releases follow semantic versioning (e.g., v1.0.0) and are packaged into installer executables after successful build and test cycles. Rollback procedures are documented and rehearsed to restore previous stable versions rapidly if deployment issues occur.

MAINTENANCE AND SUPPORT

This section defines the ongoing support and maintenance activities essential to keep the To-Do List system operational, secure, and aligned with evolving business needs. Regular maintenance prevents degradation of performance and helps identify issues proactively before they affect end users.

Guidelines for System Maintenance and Support

- **Database Backup and Integrity Checks:** Scheduled automated backups of the SQL Server database ensure data is safeguarded against accidental loss. Backup files are stored securely with redundancy, and integrity checks verify their usability for recovery.
- **System Monitoring:** Logs capturing login attempts, task modifications, and system errors are continuously monitored. Alerts are configured for repeated failed login attempts or critical exceptions, enabling proactive intervention by support staff.
- **User Support and Training:** User documentation, including FAQs and step-by-step guides, are maintained and updated to assist users in effectively operating the system. Helpdesk or support channels are provided to handle user queries and issues promptly.
- **Performance Monitoring and Optimization:** System performance metrics are regularly reviewed to identify bottlenecks such as slow queries or UI lag. Index maintenance, query optimization, and resource cleanup are scheduled tasks to maintain responsiveness.
- **Security Audits:** Periodic security reviews ensure password policies, error handling, and data access controls remain effective. Vulnerability assessments and penetration testing may be conducted to uncover and address potential risks.

Procedures for Handling Software Updates, Patches, and Bug Fixes

- **Staging Environment Testing:** All software patches and feature updates are first deployed to a staging environment mirroring production. Thorough testing ensures that new code does not introduce regressions or conflicts.
- **Change Management:** Updates follow a documented change control process, including approval steps, deployment windows, and rollback procedures to minimize business impact.
- **Release Notes and Versioning:** Detailed release notes accompany each software update, documenting fixes, enhancements, and known issues. Semantic versioning clarifies the nature of each release, aiding in tracking and troubleshooting.
- **Regression Testing:** Before and after applying patches, regression testing verifies that all existing functionalities remain intact, reducing risk of introducing new defects.

Escalation Process for Issue Resolution

1. Step 1: User Troubleshooting

- The client checks the user manual for guidance.
- They retry the operation (e.g., re-entering the task, checking database connection).

2. Step 2: Developer Contact

- If the issue persists, the client reports it directly to the development team.
- Developers check logs, database, or application files to identify the bug or error.

3. Step 3: Issue Resolution

- Developers provide a fix or patch offline.
- If the issue is severe, they may reinstall the system with preserved data.