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

REQUIREMENTS SPECIFICATION

INTRODUCTION

Purpose

This document outlines the requirements for the development of the To-do List System with Calendar Integration, a custom task management tool created specifically for Icarus Shirts, a local business specializing in personalized clothing. The purpose of this document is to provide a reference that guides the design, development, testing, and deployment of the system to ensure that it fully addresses the needs of the client. It is intended to serve as a foundation for both the development team and the client by clearly defining the system's expected functionalities, constraints, and intended outcomes. By establishing clear requirements early on, this document aims to reduce misunderstandings, streamline the development process, and ensure that the final product delivers the expected value in improving productivity, organization, and task tracking for the business.

Overview

The To-do list system is a productivity tool designed for a single user. It allows task creation, editing, and organization with integration into a calendar view.

Scope

The system focuses on a single-user experience. It enables users to create, view, edit, and delete tasks, and integrate those tasks into a calendar interface. Multi-user support, online synchronization, and third-party calendar integration are not included in this project's scope.

FUNCTIONAL REQUIREMENTS

The system includes several key functional requirements to ensure usability and efficiency. The table below summarizes these requirements:

ID	Requirement Description	Priority	Dependencies	Acceptance Criteria
F1	Create a task with title, description, and due date.	High	-	Task is saved and appears in the list and calendar.
F2	Update existing tasks.	High	F1	Changes are reflected in the database and interface.
F3	Mark tasks as completed.	Medium	F1	Completed tasks are shown as finished in the interface.
F4	Delete tasks	High	F1	Tasks are moved to the recycling bin.
F5	Store tasks in a local database for data persistence.	High	-	Tasks remain saved after system restarts.
F6	Restore deleted tasks	Medium	F1, F4, F5	Task reappears along with other tasks.
F7	Create a back-up for the database	Low	F5	The database is successfully saved in the chosen file.
F8	Restore the backed-up database	Low	F5, F7	The database's content are restored based on the back up data.

NON-FUNCTIONAL REQUIREMENTS

Our To-Do List system is designed not only to fulfill basic task management and login functions but also to provide a high-quality user experience by focusing on key quality attributes. The system prioritizes performance, usability, reliability, and security, while ensuring it is maintainable and scalable for future growth. These non-functional requirements establish the foundation for a system that is fast, easy to use, dependable, secure, and ready to adapt as user needs evolve.

1. User Performance
    - The system should load up to 200 tasks within 7 seconds to ensure fast access to user data.
    - The system should handle multiple simultaneous users efficiently without performance degradation.
2. Usability
    - The interface must be intuitive and simple to navigate for users of varying technical skills.
    - Error messages should be clear and helpful to guide users in case of incorrect inputs.
    - Task creation and viewing processes must be streamlined to reduce user effort.
    - The system should clearly communicate features that are unavailable, such as task deletion.
3. Reliability
    - The system must operate without failures, crashes, or data corruption during normal use.
    - User data, including tasks and login credentials, must be securely and reliably stored.
    - There should be no account lockout even after multiple incorrect login attempts to ensure continuous access. Data storage must be reliable and secure, and the code structure should support easy maintenance and updates.

USE CASES

Use Case ID	Name	Actors	Description	Preconditions	Postconditions
U1	Create One-Time Task	User	User creates a one-time task and saves it.	App is open	Task saved and displayed.
U2	Create Recurring Task	User	User creates recurring tasks.	App is open	Recurring task is saved and shown.
U3	Create Custom Task	User	User sets a task for multiple dates.	App is open	Task appears on all chosen dates.
U4	View Calendar	User	User views calendar with tasks displayed.	At least one task exists	Tasks displayed in proper slots.
U5	Edit/Delete Task	User	User edits or deletes a task.	Task exists	Changes saved and displayed.
U6	Mark Task as Completed	User	User marks a task as completed.	Task exists	Task status updated visually.

DATA REQUIREMENTS

This shows the content of the database and the data used for the system. There will be three tables on the database: users, tasks, and completed tasks. This section will show the data for each table.

Data Entities & Attributes

1. Users Table

This table stores the account details of the system user.

- **User ID: Unique identifier for the user.**
- **Username: The account name used to log in.**
- **Password: The user's password (securely stored in hashed form).**
- **Security Question: Used for password recovery.**

2. Information Table (Active Tasks)

This table contains all active and ongoing tasks.

- **Task ID: Unique identifier for each task.**
- **Title: Short name of the task.**
- **Task Date: The deadline or scheduled date.**
- **Priority Level: The urgency of the task (Extreme, Moderate, Low).**
- **Description: Longer details of the task.**
- **Image: Optional image related to the task.**
- **Status: The current state of the task (Not Started, In Progress, Completed).**
- **Created At: Date and time when the task was created.**

3. TaskInformationDeleted Table (Recycle Bin)

This table holds tasks that have been deleted but can still be restored.

- **Task ID: Unique identifier for the deleted task.**
- **Title: The name of the task.**
- **Task Date: The deadline before deletion.**

- **Priority Level: The priority level before deletion.**
- **Description: The details of the deleted task.**
- **Image: Any image attached to the task.**
- **Status: The status before deletion.**
- **Deleted At: The date and time when the task was deleted.**

4. DeletedTasks Table

This table stores tasks that have been permanently deleted and cannot be restored through the system. This acts as a record for auditing but is not accessible to the user in normal operations.

- **Task ID: Unique identifier for the permanently deleted task.**
- **Title: The name of the task.**
- **Task Date: The original deadline.**
- **Priority Level: The urgency level before deletion.**
- **Description: Task details.**
- **Image: Any file attached to the task.**
- **Status: The last known status before deletion.**
- **Deleted At: Date and time of permanent deletion.**

ASSUMPTIONS & CONSTRAINTS

This section outlines the key assumptions and constraints considered in the development of the To-Do List System with Calendar Integration. It defines the conditions and limitations that guide the project's design and implementation, as well as the expectations necessary for the system to function effectively within the client's environment.

Constraints

The development and implementation of the To-Do List System with Calendar Integration will operate within the following limitations:

- **Single-User System** – The application is designed exclusively for a single user and does not support multiple simultaneous users.
- **Local Environment Only** – The system will run locally on the client's computer and will not support cloud synchronization or online data access.
- **No Third-Party Integration** – External calendar platforms such as Google Calendar, Microsoft Outlook, or similar services will not be integrated into the system.
- **Limited Platform Support** – The application will only run on computers using Windows 10 or Windows 11 operating systems.
- **Technology Stack** – The system will be developed using Visual Basic 2010 and SQL Server Management Studio (SSMS), which may limit scalability and certain advanced features.

Assumptions

The project is built under the following assumptions to ensure smooth development and implementation:

- The client or assigned user will have basic knowledge of computer operation, including navigating applications and entering data accurately.
- The client will use the system consistently to input tasks and update their status, ensuring the accuracy and usefulness of the task management process.

- The system will primarily be used in a single location, minimizing the risk of conflicts due to different environments or devices.

GLOSSARY

Acceptance Criteria – The predefined conditions that must be met for a feature or functionality to be considered complete and functional.

Assumption – A condition believed to be true during the planning and development of the system, such as user familiarity with basic computer operations.

Calendar Integration – The feature that visually displays tasks within a calendar view, allowing users to see daily, weekly, or monthly schedules.

Constraint – A limitation or restriction that affects how the system can be developed or used.

Custom Task – A task set by the user for specific non-repeating dates within a calendar month.

Data Entity – A logical unit of data, such as a user profile or a task, stored within the system database.

Data Persistence – The ability of the system to retain data, such as tasks and schedules, even after closing or restarting the application.

Deadline – The specific date or time by which a task needs to be completed.

Deleted Tasks Table – A permanent storage table for tasks that have been deleted and cannot be restored. Tasks in this table are inaccessible through the system's normal interface.

Local Environment – The setup where the system runs exclusively on a single device without requiring internet connectivity or cloud-based services.

Offline Mode – Operation of the system without the need for an internet connection.

Recurring Task – A task scheduled to repeat automatically at a defined frequency, such as daily, weekly, or monthly.

Single-User System – A system designed for use by only one individual, with no multi-user support or shared access.

SQL Server Management Studio (SSMS) – The tool used for managing and maintaining the database that stores task information.

System Requirements – The minimum hardware and software specifications needed for the system to run properly.

Task – An item that represents a specific activity or responsibility to be completed by the user.

Task Information Deleted Table – A temporary storage table for deleted tasks, functioning as a "Recycle Bin." Tasks stored here can be restored back into the main task list if needed.

Task Status – An indicator of whether a task is pending, ongoing, or completed.

Use Case – A description of how a user interacts with the system to accomplish a specific goal or task.

User Interface (UI) – The part of the system that the user interacts with, designed to be simple and intuitive.

Visual Basic 2010 – The programming language and development environment used to build the system.