འཇིགས་མེད་རྣམ་རྒྱལ་གླིང་རྡ་ཝང་མཁོ་རིམ་སློབ་གྲྭ།
Royal University of Bhutan
Jigme Namgyal Engineering Collage, Dewathang

Final Report

Submitted by:

Norbu Wangmo (05240141)

DIPLOMA IN COMPUTER SYSTEM AND NETWORK

JIGME NAMGYEL ENGINEERING COLLEGE ROYAL UNIVERSITY OF BHUTAN

DEWATHANG

(31st August, 2025)

Contents

List of Figures

Introduction

This document describes a To-Do List application—a digital tool designed to help you easily manage your daily tasks. Instead of using paper lists that can get lost, this app lets you create, organize, and track your tasks on your computer.

You can add new tasks, set their priority (like high, medium, or low), mark them as done, and even set reminders so you don't forget important deadlines. The app also includes features like user login to keep your tasks private, and a dark mode for comfortable use.

In short, this To-Do List tracker is a simple, efficient, and organized way to stay on top of your work and personal responsibilities.

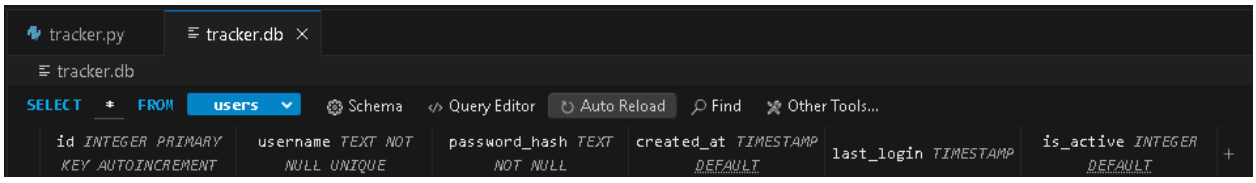
Database Structure

Users Table

The users table is created with the CREATE TABLE command. It defines seven columns: a unique auto-incrementing ID as the primary key, a mandatory and unique username with a minimum length of 3 character, a mandatory password hash, automatic timestamps creation and last login, and a status flag to indicate an active account, ensuring secure and organized user data management.

```
# Users table
cur.execute("""
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    last_login TIMESTAMP,
    is_active INTEGER DEFAULT 1,
    CHECK (is_active IN (0,1))
);
""")
```

Database:

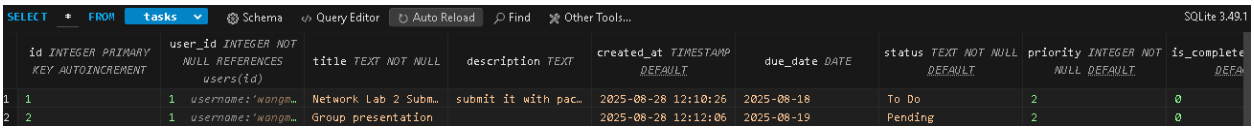


Tasks Table

The tasks table is created with the CREATE TABLE command. It defines seven columns: a unique auto-incrementing ID as the primary key, a mandatory title, an optional description, a status with predefined options, an automatic creation date, an optional due date, and a priority level between 1 and 3, ensuring organized and consistent tracking of task information.

```
# Create tasks table
cursor.execute("""
CREATE TABLE IF NOT EXISTS tasks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    due_date DATE,
    is_completed INTEGER DEFAULT 0,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
""")
```

Database



	id INTEGER PRIMARY KEY AUTOINCREMENT	user_id INTEGER NOT NULL REFERENCES users(id)	title TEXT NOT NULL	description TEXT	created_at TIMESTAMP DEFAULT	due_date DATE	status TEXT NOT NULL DEFAULT	priority INTEGER NOT NULL DEFAULT	is_complete DEFA
1	1	1 username:'wangm…	Network Lab 2 Subm…	submit it with pac…	2025-08-28 12:10:26	2025-08-18	To Do	2	0
2	2	1 username:'wangm…	Group presentation		2025-08-28 12:12:06	2025-08-19	Pending	2	0

Reminder Table

The reminder table is created with the CREATE TABLE command. It defines columns to store reminders linked to specific tasks, including a unique ID, the associated task ID, a reminder message, and the date and time the reminder is set to trigger.

```
# Reminders table
cur.execute("""
CREATE TABLE IF NOT EXISTS reminders (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    task_id INTEGER NOT NULL,
    reminder_time TIMESTAMP NOT NULL,
    message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (task_id) REFERENCES tasks(id) ON DELETE CASCADE
);
""")
```

Main Window

This code creates a To-Do List Tracker application with a graphical user interface. It includes user authentication (login/register), task management with priorities and status tracking, and reminder functionality. The application uses SQLite for data storage and features a clean, colorful interface with interactive buttons and filtering options.

```
class TodoApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("To-Do List Tracker")
        self.geometry("600x500")  # Larger welcome interface
        self.resizable(False, False)
        self.configure(bg="#f0f8ff")  # Light blue background

        # Center the window on screen
        self.update_idletasks()
        x = (self.winfo_screenwidth() // 2) - (600 // 2)
        y = (self.winfo_screenheight() // 2) - (500 // 2)
        self.geometry(f"600x500+{x}+{y}")

        self.current_user = None  # (id, username, email, ...)
        self._build_styles()
        self._build_auth_frame()
```

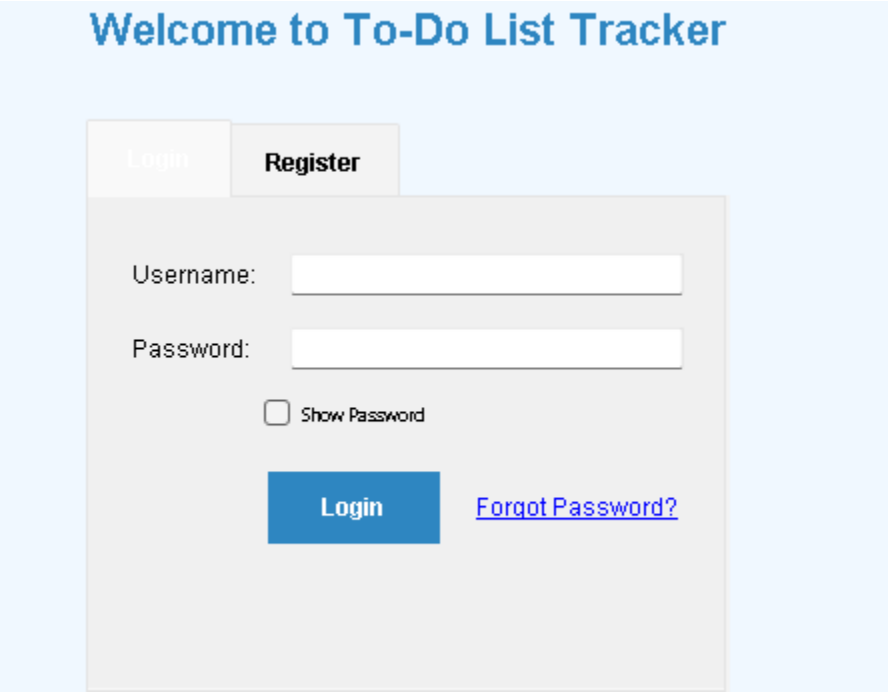


Figure 1 welcome interface

Input Fields

The interface includes text boxes for entering usernames, passwords, email addresses, and task details like titles, descriptions, and due dates, with dropdown menus for selecting status and priority levels.

```
# Login tab
self.login_username = tk.StringVar()
self.login_password = tk.StringVar()

ttk.Label(self.login_tab, text="Username:").grid(row=0, column=0, sticky="w", pady=6)
ttk.Entry(self.login_tab, textvariable=self.login_username).grid(row=0, column=1, sticky="ew", pady=6, padx=(10, 0))
ttk.Label(self.login_tab, text="Password:").grid(row=1, column=0, sticky="w", pady=6)
ttk.Entry(self.login_tab, textvariable=self.login_password, show="*").grid(row=1, column=1, sticky="ew", pady=6, padx=(10, 0))
self.login_tab.grid_columnconfigure(1, weight=1)
```

```
# Register tab
self.reg_username = tk.StringVar()
self.reg_email = tk.StringVar()
self.reg_password = tk.StringVar()
self.reg_password2 = tk.StringVar()

ttk.Label(self.register_tab, text="Username:").grid(row=0, column=0, sticky="w", pady=6)
ttk.Entry(self.register_tab, textvariable=self.reg_username).grid(row=0, column=1, sticky="ew", pady=6, padx=(10, 0))
ttk.Label(self.register_tab, text="Email:").grid(row=1, column=0, sticky="w", pady=6)
ttk.Entry(self.register_tab, textvariable=self.reg_email).grid(row=1, column=1, sticky="ew", pady=6, padx=(10, 0))
ttk.Label(self.register_tab, text="Password:").grid(row=2, column=0, sticky="w", pady=6)
ttk.Entry(self.register_tab, textvariable=self.reg_password, show="*").grid(row=2, column=1, sticky="ew", pady=6, padx=(10, 0))
ttk.Label(self.register_tab, text="Confirm Password:").grid(row=3, column=0, sticky="w", pady=6)
ttk.Entry(self.register_tab, textvariable=self.reg_password2, show="*").grid(row=3, column=1, sticky="ew", pady=6, padx=(10, 0))
self.register_tab.grid_columnconfigure(1, weight=1)
```

1. **Username Entry Fields:**
   The application creates labeled text entry fields for username input in both the login and registration forms, allowing users to enter their credentials.

2. **Password Security Fields:**
   For security, the password fields display asterisks (*) instead of actual characters when users type, protecting sensitive information from being visible.

3. **Registration Form Fields:**
   The registration form includes additional fields specifically for email input and a second password field for confirmation, ensuring accurate account creation.

4. **Task Management Inputs:**
   For task creation and editing, the application provides text entry fields for title, description, and due date, along with dropdown menus for selecting status (Pending, To Do, Done) and priority levels (1, 2, 3).

Figure 2 Text entry fields

```
ttk.Label(frm, text="Status").grid(row=3, column=0, sticky="w", pady=4)
ttk.Combobox(frm, textvariable=self.var_status, values=["Pending","To Do","Done"], state="readonly", width=18)\
    .grid(row=3, column=1, sticky="w", pady=4)

ttk.Label(frm, text="Priority (1-3)").grid(row=4, column=0, sticky="w", pady=4)
ttk.Combobox(frm, textvariable=self.var_prio, values=["1","2","3"], state="readonly", width=8)\
    .grid(row=4, column=1, sticky="w", pady=4)
```

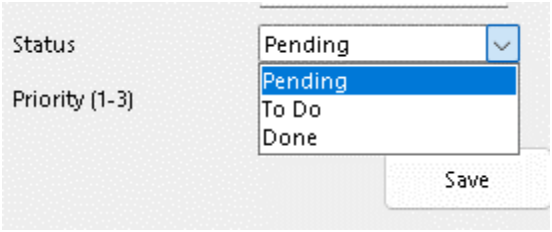


Figure 3 Status level

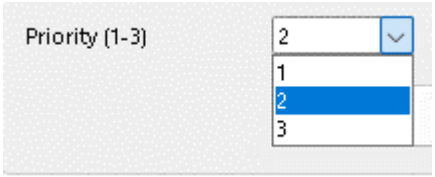


Figure 4 Priority level

Buttons

Colorful, interactive buttons allow users to perform all actions including logging in, registering, adding/editing/deleting tasks, marking tasks as done, applying filters, and logging out.

1. "Login" button to authenticate users

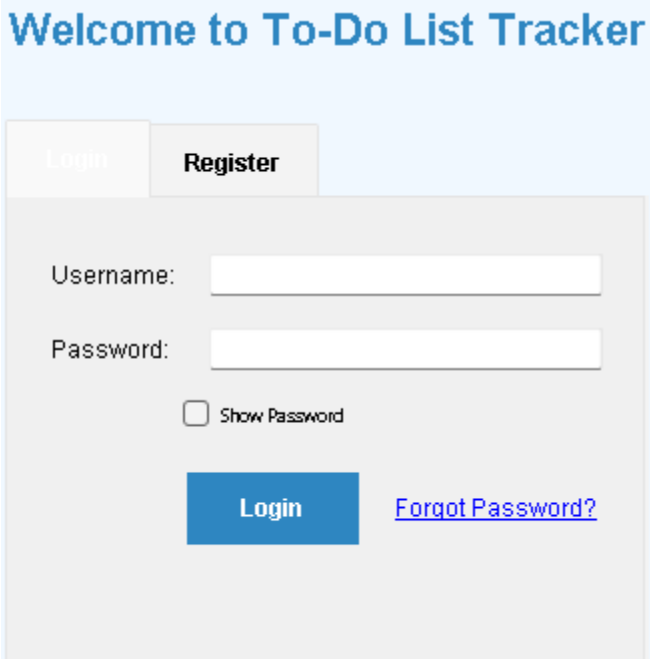
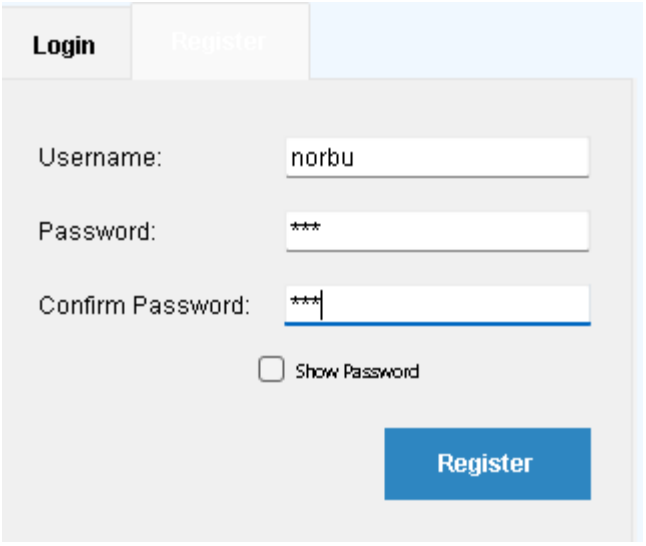


Figure 5 login button

2. "Register" button to create new accounts



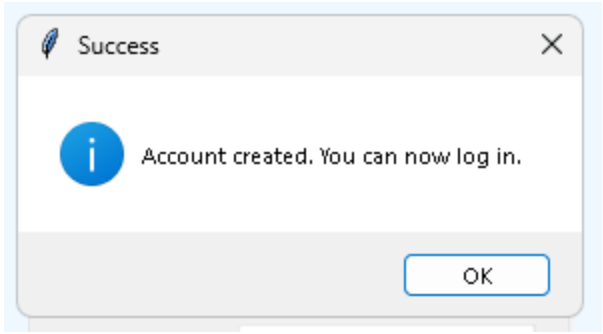


Figure 6 Account created

Figure 7 Register button

3. "Add Task" button to insert new tasks

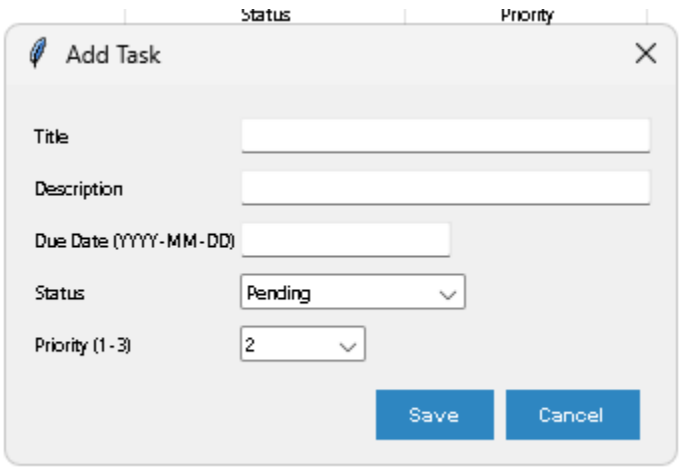


Figure 8 Add task button

4. "Edit Task" button to modify existing tasks

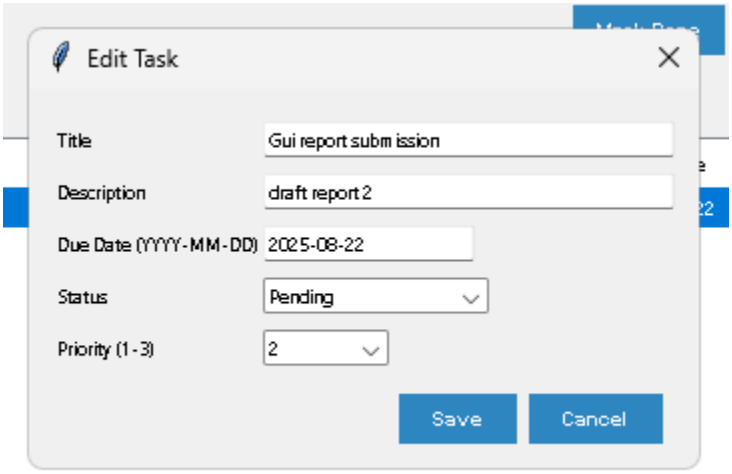


Figure 9Edit task button

5. "Delete Task" button to remove tasks

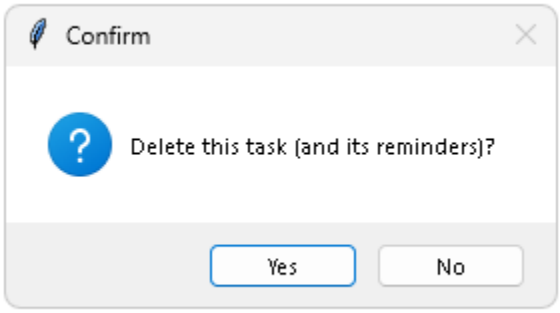


Figure 10 Delete task button

6. "Mark Done" button to update task status

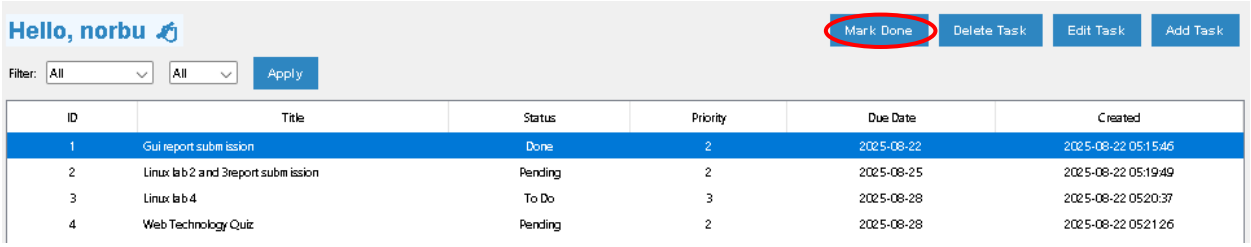


Figure 11 "Mark Done" button

7. "Logout" button to return to login screen

Figure 12Logout button

8. "Apply" button for filtering tasks

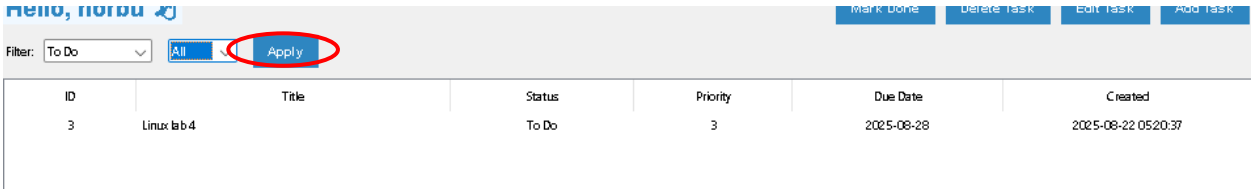
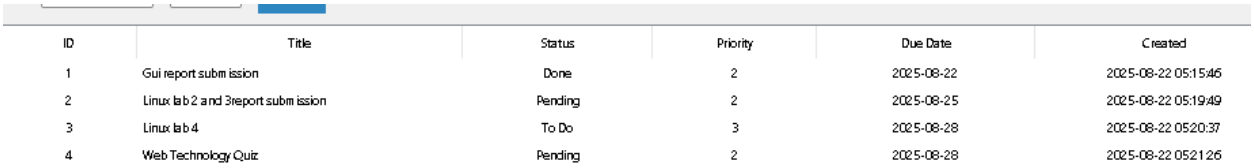


Figure 13Apply button

Display Area

Tasks are shown in a neat table format with columns showing all important information, and filter options help users organize and find their tasks easily.



ID	Title	Status	Priority	Due Date	Created
1	Gui report submission	Done	2	2025-08-22	2025-08-22 05:15:46
2	Linux lab 2 and 3report submission	Pending	2	2025-08-25	2025-08-22 05:19:49
3	Linux lab 4	To Do	3	2025-08-28	2025-08-22 05:20:37
4	Web Technology Quiz	Pending	2	2025-08-28	2025-08-22 05:21:26

Figure 14Display area

Treeview widget shows task records in tabular form with columns for ID, Title, Status, Priority, Due Date, and Created date. The display includes filtering options for status and priority.

Extra Features

Register

I implemented a user registration feature that validates credentials, requiring usernames of at least 3 characters and matching passwords. Successful registration triggers a confirmation message, while errors display appropriate warnings.

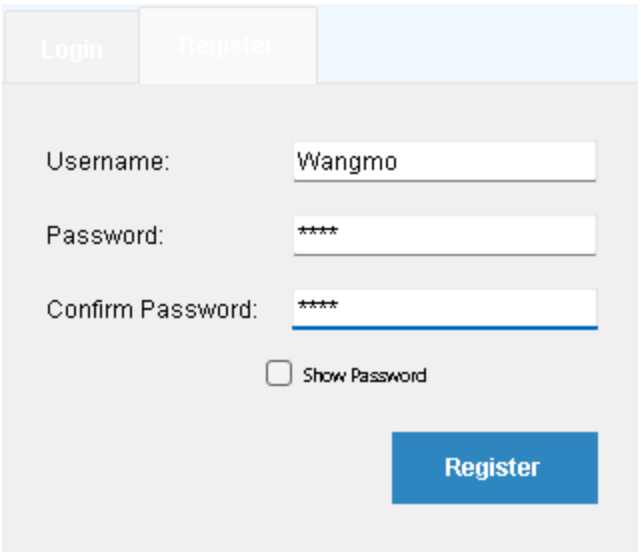


Figure 15 Register interface

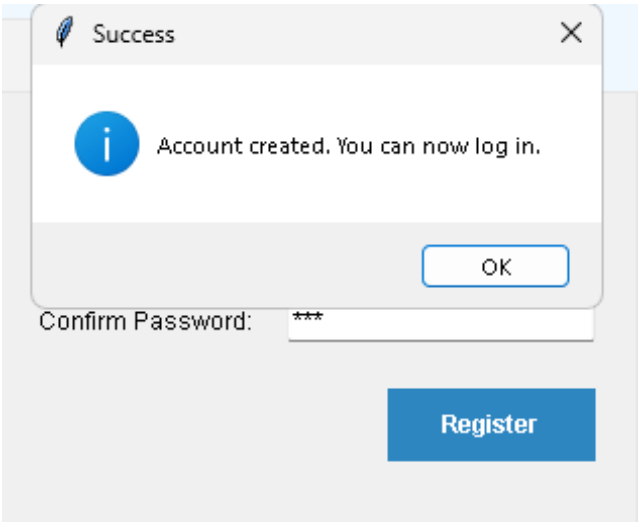


Figure 16 Account created successfully

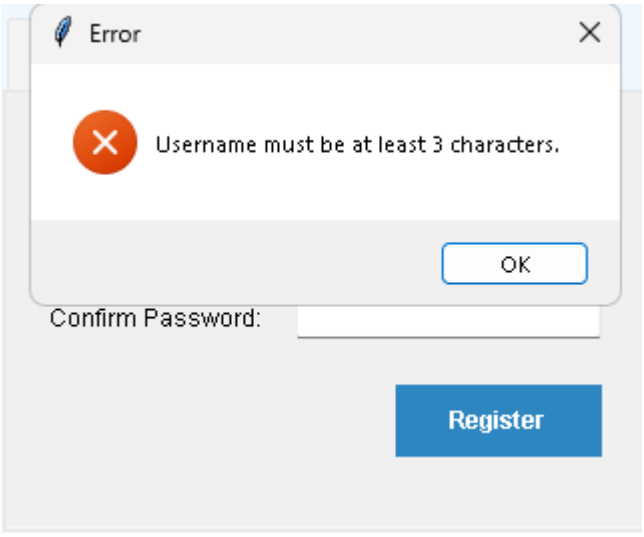


Figure 17 Username requirement

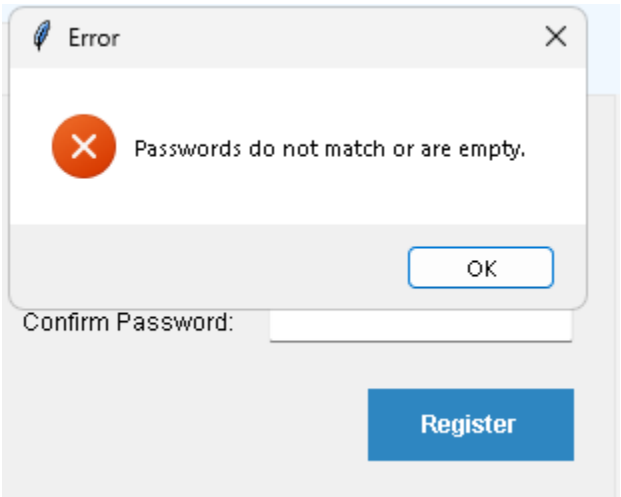


Figure 18 Password

Login

User should login with their username and password. After entering username and password users will be able to get access to user interface where they can add, edit, delete task.

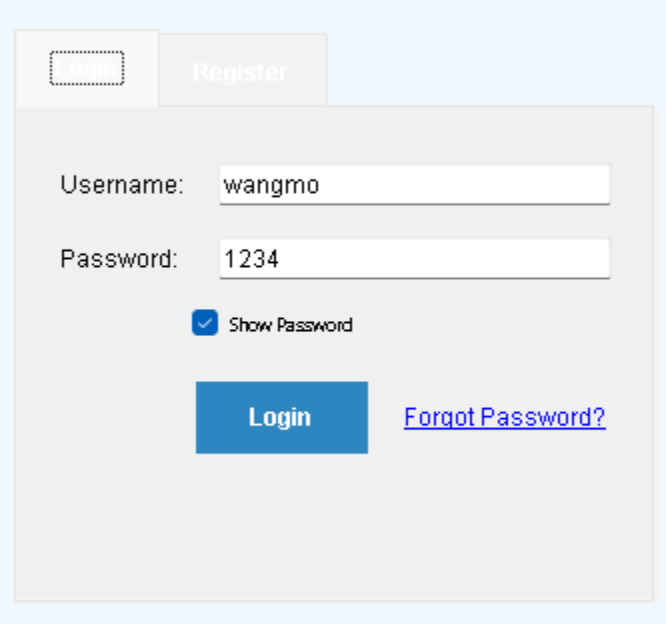


Figure 19 Login Interface

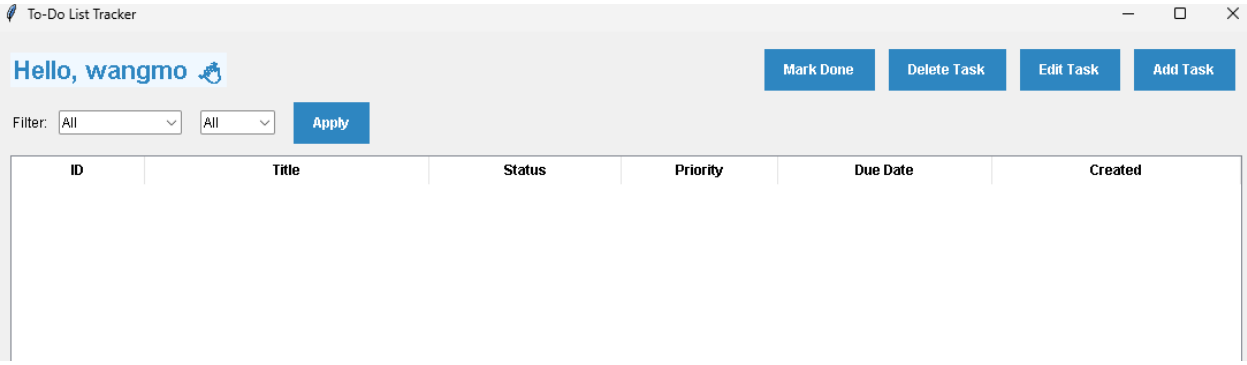


Figure 20 User interface

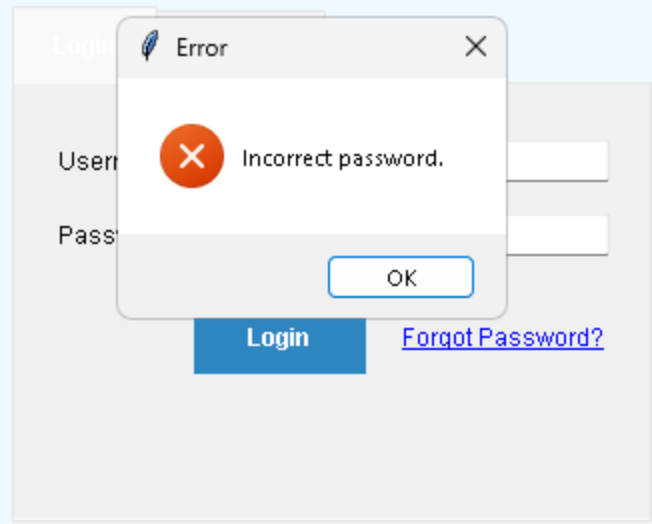


Figure 21 Incorrect password

Forgot password

If user forgets password, they can click to forgot password. After clicking it will ask for username and new password and the password will be reset successfully.

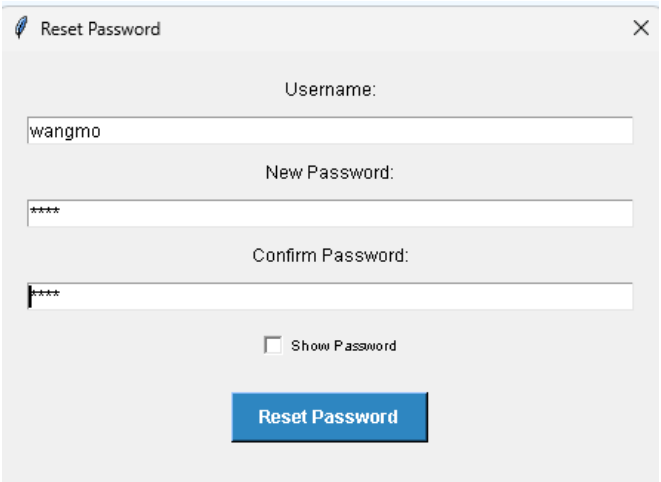


Figure 22 Reset password

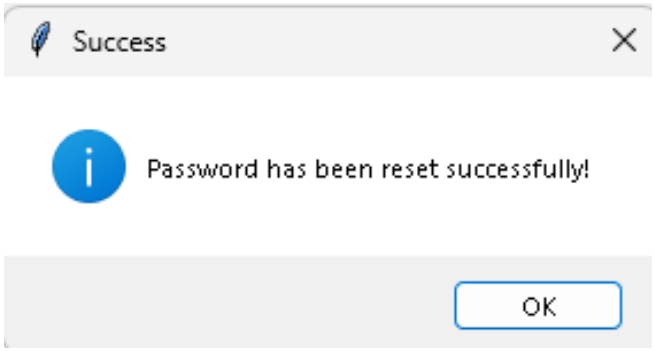


Figure 23 Password reset successfully

Filtering

Users can filter tasks by **status** and **priority**.

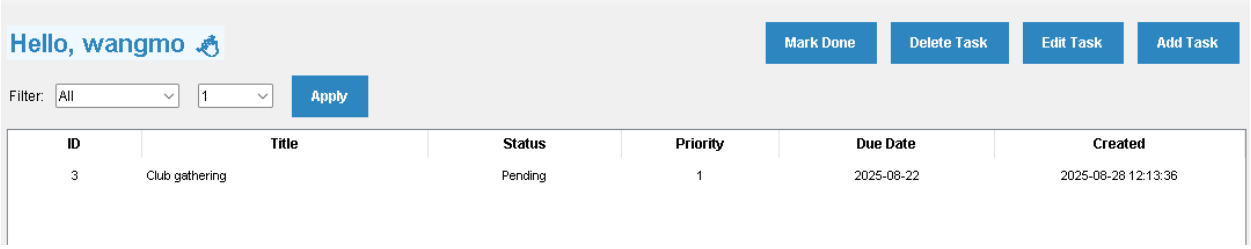


Figure 24 Filtering by priority

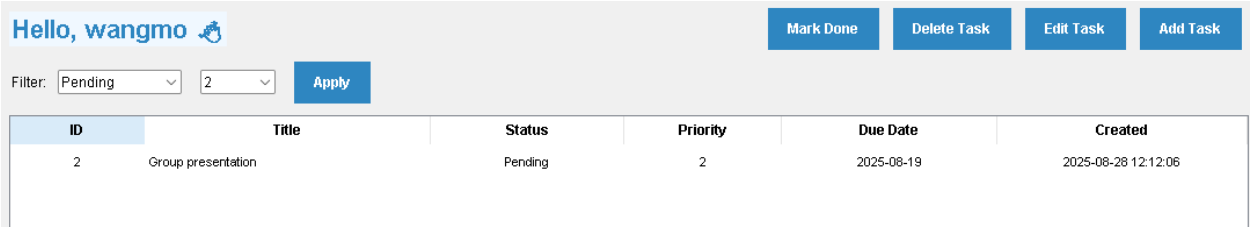


Figure 25 Filtering by status

Reminder

Users can add reminders to your tasks and view them anytime. Set dates, times, and optional messages for important deadlines.

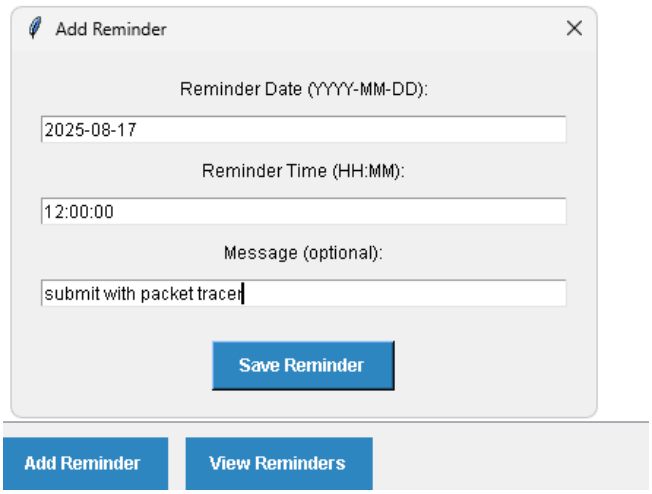


Figure 26 Add reminder

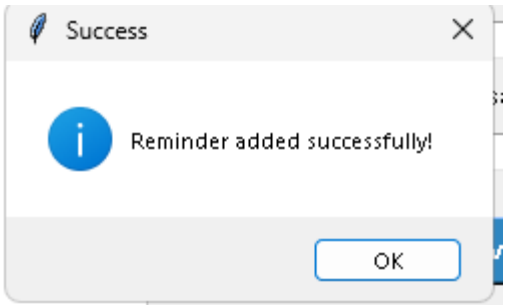


Figure 27 Reminder added successfully

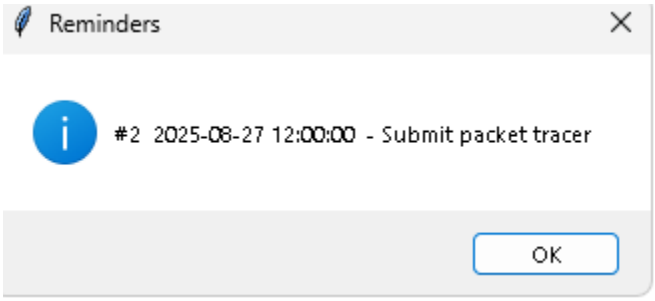


Figure 28 View reminder

Logout

After everything users can logout

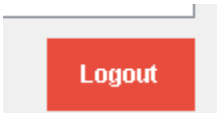


Figure 29 Logout

Dark Mode / Theming

Users can customize their interface by turning on dark mode.

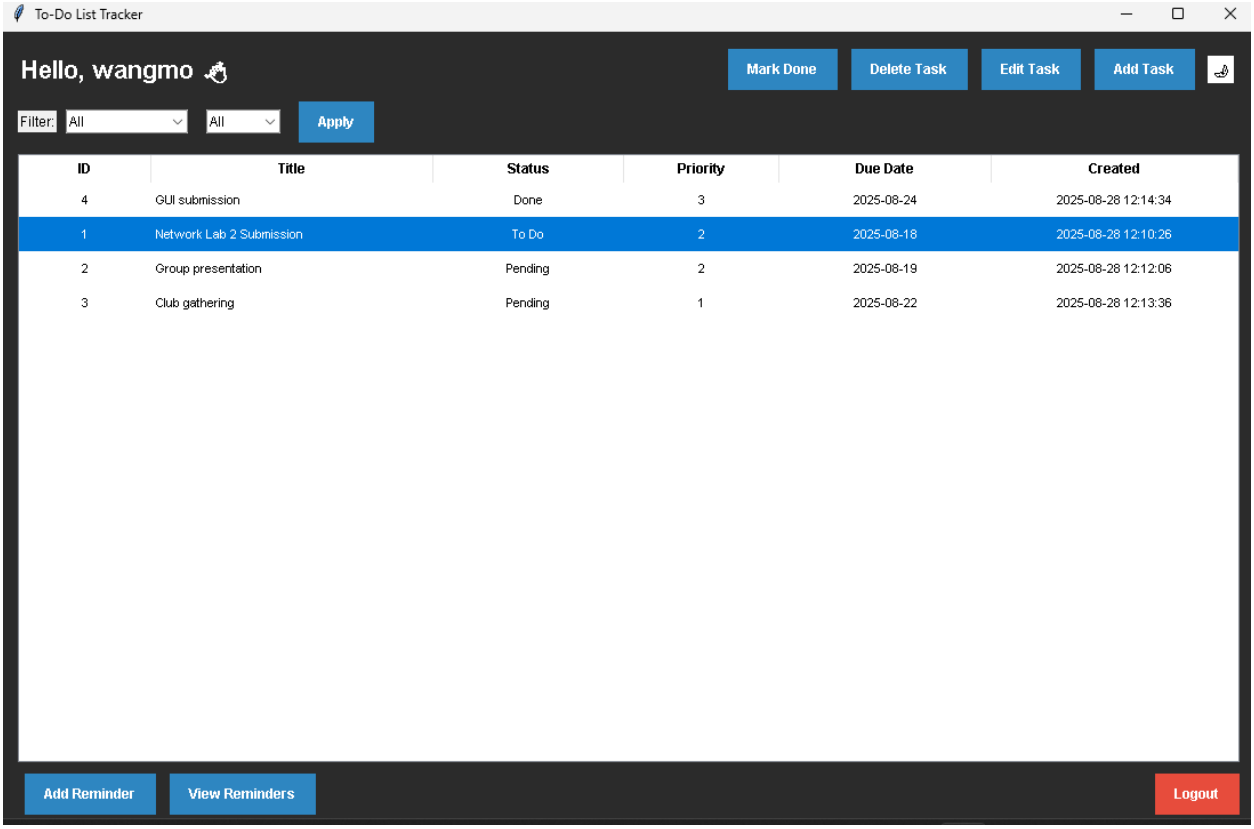


Figure 30 Dark mode/ Theme

Conclusion

The development of this To-Do List Tracker application has successfully resulted in a comprehensive productivity tool that effectively addresses the core requirements of task management while incorporating advanced features that enhance user experience. The application demonstrates robust functionality through its secure authentication system, intuitive task management capabilities, and thoughtful additional features like intelligent filtering, reminder systems, and customizable theming.

Throughout the development process, particular attention was paid to creating an interface that is both visually appealing and functionally efficient. The dark mode option, responsive design elements, and clear user feedback mechanisms contribute to an engaging user experience. The integration of SQLite database ensures data reliability and persistence, while the object-oriented design approach allows for maintainability and future expansion.

Reference

BoostMyTool. (2021, October 13). *Create and Read SQLite Databases using DB Browser: How to Use DB Browser 2021* [Video]. YouTube. https://www.youtube.com/watch?v=b0Dplx4M5zg

Downloads - DB browser for SQLite. (n.d.). https://sqlitebrowser.org/dl/

TutorialBrain. (2022, April 20). *How to run SQLITE in Visual Studio Code* [Video]. YouTube. https://www.youtube.com/watch?v=JrAiefGNUq8