University of Tripoli –
Faculty of Engineering
Electric and Electronics
Department

PROJECT

EE432

Instructor :

د . نوري بن بركة

Student Name :

فراس مصطفي علي الاسطي

معاد يوسف مبروك شليبك

Student number :

2190203745

2190203216

Submitted date :

2025/2/20

FALL 2024

Introduction

This report provides a detailed analysis and overview of the To-Do List Manager application developed using Python and the PyQt5 framework. The application is designed to manage tasks through a graphical user interface (GUI) that allows users to add, edit, delete, and toggle the completion status of tasks. Notably, it supports multi-language functionality (English and Arabic), dark and light themes, and includes filtering, searching, and sorting features. The project leverages a linked list data structure to manage tasks efficiently.

Overview of the Code

The application is divided into several functional components:

- **Language Support:**
  The `LANGUAGES` dictionary contains UI strings in both English and Arabic, allowing the interface to dynamically change based on the user's language selection.
- **Task Management:**
  A linked list data structure is implemented via the `TaskNode` and `TaskList` classes to store and manage tasks. Each task has a description, priority, date, and a completion status.
- **Dialogs:**
  Two custom dialog classes, `AddTaskDialog` and `EditTaskDialog`, are used to facilitate task creation and editing. These dialogs provide a user-friendly interface for inputting task details.
- **Main Application Window:**
  The `ToDoApp` class encapsulates the main window of the application. It integrates language selection, theme toggling, task management operations, and UI components such as search and filter controls, making the application interactive and responsive.

Detailed Components

1. Language Support

- **Dictionary Structure:**
  The `LANGUAGES` dictionary holds translations for various UI elements. This ensures that all textual elements, from button labels to dialog titles, can be switched seamlessly between English and Arabic.
- **Dynamic Update:**
  The `set_language` and `update_language` methods within `ToDoApp` are responsible for refreshing the UI based on the selected language. They update labels, placeholders, and button texts accordingly.

2. Data Structures for Task Management

- **TaskNode Class:**
  Represents a single task with attributes:
  - `description`: A string detailing the task.
  - `priority`: Indicates the task's urgency (High, Medium, Low).
  - `date`: The due date in a string format.
  - `completed`: A Boolean indicating if the task is finished.
  - `next`: A pointer to the next task node in the linked list.

- **TaskList Class:**
  Manages the collection of tasks using a linked list. Key methods include:
  - `add_task`: Inserts a new task at the end of the list.
  - `delete_task`: Removes a task by matching the description.
  - `toggle_task_status`: Flips the completion status of a task.
  - `edit_task`: Updates an existing task's details.
  - `get_all_tasks`: Returns a list of all tasks for display.

3. Dialogs for Adding and Editing Tasks

- **AddTaskDialog:**
  A dialog window that collects user input for a new task. It contains:
  - A `QLineEdit` for the task description.
  - A `QComboBox` for selecting the task's priority.
  - A `QLineEdit` for entering the task date.
  - An "Add Task" button that, when clicked, accepts the dialog input.
- **EditTaskDialog:**
  Similar in structure to the add dialog, this window is pre-populated with the details of the task being edited. It allows the user to modify the description, priority, and date before saving the changes.

4. Main Application Window (ToDoApp)

- **User Interface Layout:**
  - **Language Selection:**
    Buttons for English and Arabic are provided at the top of the UI, allowing the user to switch languages instantly.
  - **Operation Buttons:**
    A set of buttons (Add, Edit, Delete, Toggle Completion, Light Mode, Dark Mode) are provided to perform respective task operations.
  - **Search and Filter Controls:**
    The interface includes a search box, filter combo boxes for priority and status, and a sort combo box for organizing tasks.
  - **Task List Display:**
    A `QListWidget` shows all tasks with their descriptions, priority (with emojis), dates, and completion status.
- **Functional Methods:**
  - **open_add_task_dialog:**
    Opens the add task dialog and, upon successful input, adds a new task to the linked list.
  - **edit_task:**
    Opens the edit dialog for a single selected task and updates it based on user input.
  - **delete_tasks:**
    Removes one or more selected tasks from the list.
  - **toggle_tasks_status:**
    Changes the completion status of selected tasks.
  - **update_task_display:**
    Refreshes the task list view to reflect changes. It applies search filters, priority/status filters, and sorting rules before displaying the tasks.
- **Theme Management:**
  The application supports two visual themes:
  - **Dark Mode:** Uses a purple & black color scheme.
  - **Light Mode:** Uses a blue & grey color scheme.
    Theme toggling is achieved by switching stylesheets dynamically through the `apply_theme`, `set_dark_mode`, and `set_light_mode` methods.

Functionality and User Experience

The To-Do List Manager is designed with user experience in mind:

- **Instant Feedback:**
  Any change in the task list—whether by adding, editing, deleting, or toggling tasks—is immediately reflected in the task list view.
- **Filter and Sort Integration:**
  The search and filter functionalities provide real-time updates, allowing users to quickly locate or organize tasks based on specific criteria.
- **Robust Task Handling:**
  By using a linked list structure, the application efficiently manages task data, ensuring smooth operations even as the list grows.
- **Multi-Language and Theming:**
  The inclusion of language support and theme customization makes the application versatile and accessible to a broader audience.

Name	Percentage	Work
Feras elosta	**50%**	**Main code, Report**
Moad shlibk	**50%**	**GUI, README**

Conclusion

The To-Do List Manager is a comprehensive desktop application that demonstrates effective use of Python and PyQt5 for creating dynamic, user-friendly interfaces. Its modular design, which separates concerns like language management, task operations, and UI rendering, makes the code maintainable and extensible. By integrating features such as filtering, sorting, theme toggling, and multi-language support, the application not only fulfills the core functionality of task management but also offers a polished user experience. This project serves as a solid foundation for further enhancements, such as additional languages, persistent storage, and advanced task categorization.