To-Do List Management System

Sathsarani Geethamali 2025/12/20

Contents

1. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements of the **To-Do List Management System**, a desktop-based application developed using **Python and Tkinter**.

The system helps users manage daily tasks by allowing them to log in, add tasks, update tasks, delete tasks, and mark tasks as completed. It also ensures data persistence using JSON files and input validation to prevent incorrect data entry.

1.2 Scope

The To-Do List Management System is a **standalone desktop application**.

The system includes:

- Username-based login
- Task creation with priority, due date, and category
- Editing and deleting tasks
- Marking tasks as completed
- Automatic saving and loading of tasks using JSON files
- Input validation for better data accuracy

The application runs locally and does not require an internet connection.

1.3 Definitions, Acronyms, and Abbreviations

Term	**Description**
Task	A to-do item created by the user
GUI	Graphical User Interface
JSON	JavaScript Object Notation
Status	Task state (Pending / Completed)
Tkinter	Python GUI library

2. Overall Description

2.1 Product Perspective

The system is implemented as a **single Python application** using:

- Tkinter for the graphical user interface
- JSON files for local data storage

All components (UI, logic, and data handling) are implemented within one program file.

2.2 Product Functions

The system provides the following functions:

- User login
- Task creation
- Task viewing
- Task updating
- Task deletion
- Task completion marking
- Persistent storage of tasks

2.3 User Classes and Characteristics

General User

- Basic computer literacy
- Uses the system for personal task management
- No technical expertise required

2.4 Constraints

- Requires Python 3.x installed
- Runs only on desktop or laptop devices
- No multi-user concurrent access
- Data stored locally in JSON files
- No cloud or network-based features

2.5 Assumptions

- Users provide valid inputs
- Users do not manually modify JSON files
- The system is used by one user at a time

3. System Features

3.1 Login System

Description:
Allows users to log in using a username.

Requirements:

- Username must not be empty
- Username must contain at least 3 characters
- Username must contain only letters and numbers
- Tasks are stored separately per user

3.2 Add Task

Description:
Allows users to add a new task.

Task Attributes:

- Task Name
- Priority (High / Low)
- Due Date (YYYY-MM-DD format)
- Category
- Status (Pending by default)

Validation:

- All fields are mandatory
- Task name must be at least 3 characters
- Category must contain only letters
- Due date must follow the correct format

3.3 View Tasks

Description:
Displays all tasks in a list format.

Displayed Information:

- Task name
- Priority
- Category
- Due date
- Status (Pending / Completed)

3.4 Update Task

Description:
Allows users to update selected task details.

Editable Fields:

- Task name
- Priority
- Due date
- Category

3.5 Delete Task

Description:
Allows users to delete a selected task permanently from the system.

3.6 Mark Task as Completed

Description:
Allows users to mark a task as completed.

Behavior:

- Task status changes from Pending to Completed
- Completed tasks are visually distinguished in the task list

3.7 Data Persistence

Description:
The system automatically saves and loads tasks.

Storage Method:

- JSON file
- File naming format:
  <username>_tasks.json

4. External Interface Requirements

4.1 User Interface

- Login screen with username input
- Task input form
- Dropdown menu for priority
- Buttons:
  - Add
  - Update
  - Delete
  - Completed
- Listbox to display tasks

4.2 Hardware Interface

- Desktop or laptop computer

4.3 Software Interface

- Python 3.x
- Tkinter GUI library
- JSON file system

5. System Requirements

5.1 Functional Requirements

ID	**Requirement**
FR-1	System shall allow user login
FR-2	System shall allow adding tasks
FR-3	System shall display tasks
FR-4	System shall allow updating tasks
FR-5	System shall allow deleting tasks
FR-6	System shall allow marking tasks as completed
FR-7	System shall save tasks automatically
FR-8	System shall validate user inputs

5.2 Non-Functional Requirements

Category	**Requirement**
Performance	Response time < 1 second
Usability	Simple and user-friendly interface
Reliability	Tasks must not be lost
Security	Local storage only
Maintainability	Readable and modular functions

6. System Architecture

- Single Python application
- Tkinter-based GUI
- JSON-based local storage

7. System Design

7.1 Data Structure

Each task contains:

- Name
- Priority
- Due Date
- Category
- Status

JSON Format Example:

```
{
  "name": "Complete Assignment",
  "priority": "High",
  "due_date": "2025-01-15",
  "category": "Study",
  "status": "Pending"
}
```

7.2 GUI Design

- Entry fields for task name, due date, and category
- Dropdown for priority selection
- Buttons for task actions
- Scrollable list to display tasks

8. Future Enhancements

- Calendar date picker
- Search and filter tasks
- SQLite database integration
- Task reminders
- Multi-user authentication

9. Conclusion

The To-Do List Management System is a simple and efficient desktop application for managing daily tasks. It fulfills all functional and non-functional requirements and is suitable for academic submission and practical use.