Collage of Engineering

Department of Software engineering

Software component design Assignment: Task manager

Team Members

Kalkidan Desalegn	0693/13
Kalkidan Kiros	0697/13
Meron Teklewyni	0839/13
Nahom Amare	0934/13

Submitted to: Tr. Gizate

Submitted date: Dec 18th 2024

Table of content

Abstract

This document outlines the development of a simple task manager and tracker, designed to perform CRUD operations for task management. The project aims to enhance productivity through a user-friendly interface that allows users to add, view, update, and delete tasks efficiently. The system is developed in Python using Google Colab, leveraging libraries such as IPython.display and ipywidgets for data handling and interactivity. Following the Spiral Model, this project ensures iterative development with continuous feedback and risk management.

Introduction

Task management is a crucial component of productivity, especially in project-based or educational environments. This project introduces a lightweight and efficient task manager application tailored for small teams or individuals. The system enables users to create, read, update, and delete tasks seamlessly, offering a simple yet effective solution for managing daily responsibilities. Built in Python within the Google Colab environment, the project emphasizes ease of use and accessibility while adhering to best practices in software development.

Project Overview

Objective

The purpose of this project is to develop a simple and effective task manager and tracker. The system will enable users to create, view, update, and delete tasks through an interactive interface. It aims to enhance productivity by providing an easy-to-use tool for task management.

Scope

The system will focus on implementing basic CRUD (Create, Read, Update, Delete) operations. It will include features such as:

- Adding tasks with details like name, description, due date, and status.
- Viewing tasks in a structured format.
- Updating task details as needed.
- Deleting tasks no longer required.

The project will use platforms like Python in Google Colab, utilizing libraries such as IPython To manage the display of outputs and ipywidgets To create interactive widgets for taking input from the user. The application is designed to be lightweight and suitable for small-scale task management.

Methodology

The project follows the Spiral Model, which emphasizes iterative development and risk management. This approach is ideal for ensuring the system's usability and functionality while allowing improvement at each stage.

1. **Planning Stage:** We identified requirements and potential risks.
2. **Design Stage:** Handled functionality using an in-memory Python list.
3. **Implementation Stage:** Began coding the system features.
4. **Testing Stage:** Validated the system's performance and resolved issues.

Each iteration refined the system based on feedback and testing results, ensuring continuous improvement of the simple project we planned to create.

Requirements Analysis

Functional Requirements

- Add Task: Users can add tasks by providing a task name, description, due date, and status. Tasks are added to an internal list and displayed in a table.
- View Tasks: Tasks are displayed in a tabular format that lists all task details. This allows users to easily see the name, description, due date, and status of each task.
- Update Task: Users can update a task by providing a new description, due date, or status. The task will be updated in the internal list and displayed in the table.

- Delete Task: Tasks can be deleted based on their name. Once deleted, the task is removed from the list and the task table is updated.
- Dynamic Interaction: Supports dynamic updates in real-time display. The interface refreshes immediately after an operation.

Non-Functional Requirements

- Ease of Use: The system should have an intuitive and interactive interface for users.
- Efficiency: CRUD operations should execute quickly, with minimal latency.
- Scalability: The system should handle a moderate number of tasks efficiently.
- Portability: The project should run seamlessly in the Google Colab environment without additional dependencies.

System Requirements

The project exploits development environments such as Google Colab, Notebook, and Python 3.x. Python provides IPython.display and ipywidget libraries where we can display output and for building UI elements respectively.

Constraints

- The system does not support persistent storage; all tasks are stored temporarily in memory and will be lost when the notebook session ends.
- It is designed for single-user functionality and does not support multi-user access.
- The interface is limited to the capabilities of Google Colab and ipywidgets.

System Design

Architecture Overview

The system is a simple single-tier architecture. It combines user interactions, logic, and data storage in one Python program running on Google Colab. The main components include:

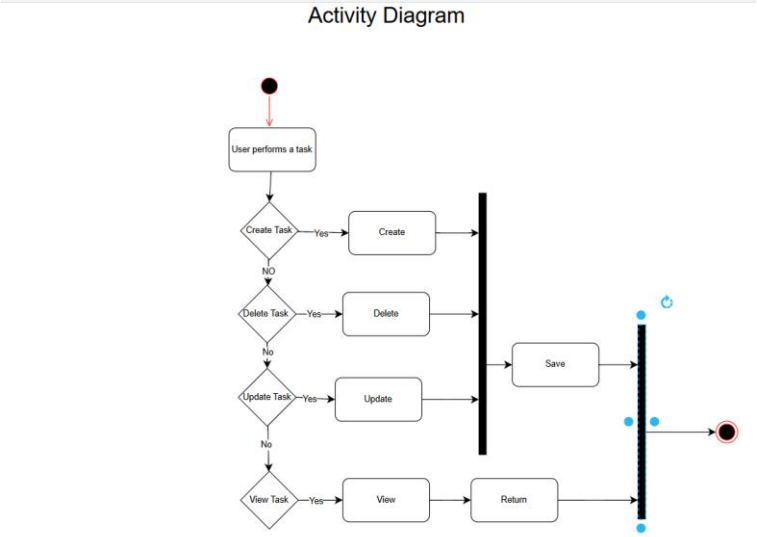
- **User Interface:** Built using ipywidgets, it allows users to input task data and interact with the system.

- **Task Management:** Handles CRUD operations like adding, updating, viewing, and deleting tasks.
- **Data Storage:** In-memory storage.
- **IPython.display:** Manages the displays and renders the task tables.

UI Design

The interface is simple and functional. It has buttons for adding, viewing, updating, and deleting tasks. Input forms allow users to enter task details. The display area shows tasks in a tabular format, making it easy to understand.

Activity Diagram



References

[1] “IPython Display - Javatpoint,” www.javatpoint.com, 2022. https://www.javatpoint.com/ipython-display (accessed Dec. 12, 2024).

[2] “Simple Widget Introduction — Jupyter Widgets 8.1.1 documentation,” Readthedocs.io, 2017. https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html

[3] Sudeep Lamichhane, "'Boost Your Productivity: Building a Task Manager with Python,'" Medium, Jun. 19, 2023. https://medium.com/@sudeeplamichhane18/boost-your-productivity-building-a-task-manager-with-python-38a9f7f16930 (accessed Dec. 11, 2024).