Spiral Model Iterations

1. **Iteration 1: Planning**

The first step involved understanding the requirements and identifying potential risks. The main risks included learning curves for ipywidgets and ensuring the system would work smoothly in Google Colab since none of us had prior experience using it.

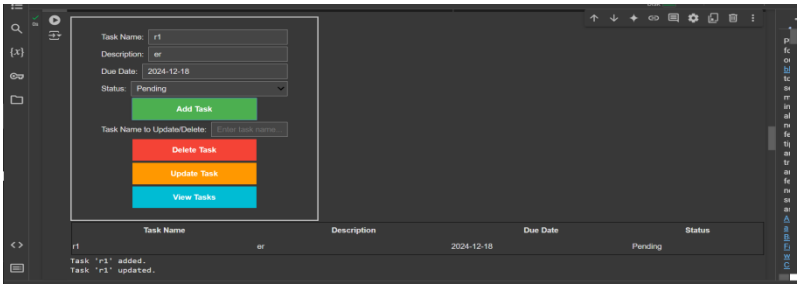
2. **Iteration 2: Design**

Created a simple design quiet straight forward, focusing on a user-friendly interface and clear data organization. The data structure and ipywidgets components were finalized after we observed desirable products.

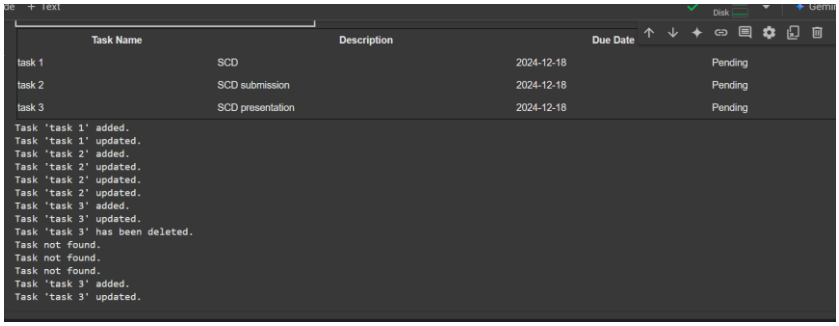
3. **Iteration 3: Implementation**

Developed the first working version of the code. Implemented basic CRUD operations, and the UI was built using ipywidgets.

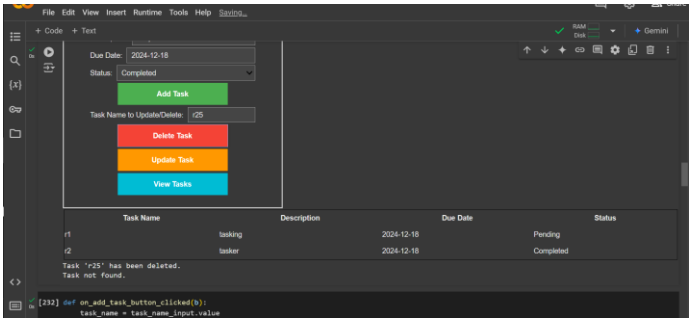
- **Add Task:** Allows users to input the details of a task—such as its name, description, due date, and status—via a simple form. Once the information is provided, the task is appended to an internal list that keeps track of all tasks. This action ensures that the task data is properly stored and ready for further operations.



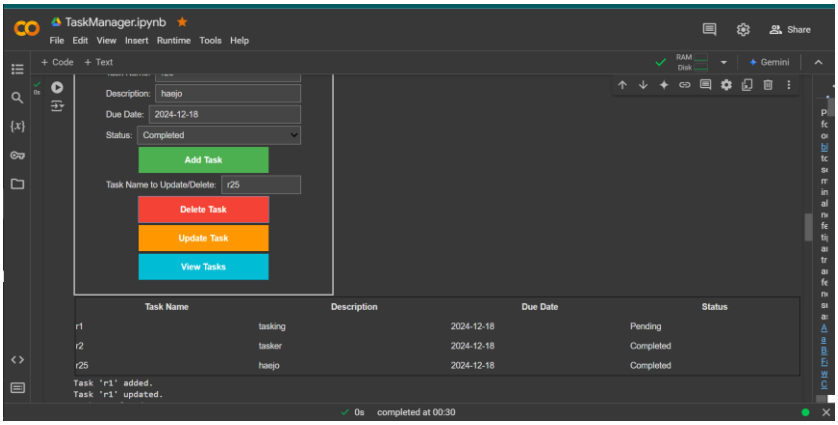
- **View Tasks:** Neatly displays all tasks in a structured, tabular format. Each task's details, including its name, description, due date, and current status, are clearly presented. This layout helps users easily scan through their task list and understand their current workload.



- **Update Task:**When users need to change a task's details, this function identifies the task based on its name. It then updates the specific fields—like the description, due date, or status—as specified by the user. This ensures that the task list always reflects accurate and up-to-date information.



- **Delete Task:** If a task is no longer needed, this function allows the user to remove it from the list. The task is identified by its name, and upon deletion, it is completely cleared from the internal storage. This helps keep the task list relevant and clutter-free



The code is organized into functions, each responsible for a specific operation. The UI components are linked to these functions for smooth interaction.

4. **Iteration 4: Testing**

The system was tested using various test cases. Bugs were identified and fixed, including error handling for invalid inputs and improving the UI responsiveness.

Duplication of Tasks: Tasks were being duplicated whenever the "add" functionality was triggered multiple times due to a lack of checks.

Solution: Implemented a check to ensure that if a task with the same name already exists, it is updated rather than being added again.

Task Updates: Updating tasks was initially problematic due to incorrect function parameters.

Solution: Corrected the function parameters and ensured task updates occurred as expected.

Task Display: Initially displayed tasks in a simple text format, leading to clarity issues as the list grew.

Solution: Used widgets.HTML to create a tabular display for tasks.

Clear Outputs: Old outputs weren’t being cleared after task list updates.

Solution: Used clear_output(wait=True) to clear previous outputs before displaying updates.

Handling Multiple Tasks: Lack of proper mechanisms to delete or update tasks based on task names.

Solution: Introduced unique identifiers for tasks.

User Interface Design: Ensuring ease of use and interactivity.

Solution: Designed a minimalistic interface using widgets.

Summary

The team successfully completed the task manager using the Spiral Model. Each phase informed the next, enabling gradual improvements and practical adjustments. Despite initial challenges with unfamiliar libraries and environments, the system met its goals:

- ✓ A functional task manager capable of CRUD operations.
- ✓ Dynamic, user-friendly interface.
- ✓ Lightweight, single-user operation within Google Colab.

The project serves as an effective demonstration of task management using Python's interactive capabilities.

Reference

[1] GeeksforGeeks (2024) What is spiral model in software engineering?, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/software-engineering-spiral-model/ (Accessed: 12 December 2024).

[2] TutorialsPoint, "SDLC - Spiral Model," TutorialsPoint, [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm. [Accessed: Dec. 12, 2024].