Integrating task
task management and reminder systems

DANIEL MEJIA | JEAN GALLEGO

Content

1. INTRODUCTION.

Efficient task and reminder management is essential for personal and professional productivity and success. To address this need, the development of a task and reminder management system has been proposed that will allow users to effectively add, organize and manage their daily to-dos.

This project aims to design and implement a comprehensive system that addresses all dimensions of task management, from the storage of information to the presentation of an intuitive user interface and advanced functionalities. The system will be based on efficient algorithms and data structures to ensure optimal performance.

1.1. Components and functionalities (SGTR)

1.1.1. Task storage and reminders

We will use a hash table to store tasks and reminders. Each entry will have a unique identifier as a key and the task/reminder information as a value. This information includes the title, description, due date, and priority.

1.1.2. User Interface

We will design a user interface that allows users to add, modify, and delete tasks and reminders. Users will be able to see a list of all tasks and reminders, sorted by due date or priority.

1.1.3. Priority Management

There will be two categories of tasks: "priority" and "non-priority".

- For priority tasks, we will use a priority queue to organize them according to their importance. When a user adds a new priority task, it will be added to the queue according to its importance, which will ensure that important tasks are handled first.
- Non-priority tasks are handled on a first-in-first-out (**FIFO**) basis.

1.1.4. Undo function

We will implement a method to undo the actions performed by a user in the system. We will use a stack (**LIFO**) to keep track of the actions performed. The general process for the "Undo" function will be:

1. Create an action stack to keep track of the user's actions.
2. Record each action performed by the user in the stack, including details of the action and the task affected.
3. Implement a last action undo method that unstacks the last action and undoes the corresponding action based on the information stored in the stack.
4. Provide the user interface with an "Undo" option to undo the last action performed.

With these features, our task and reminder management system will be efficient and allow users to organize and manage their tasks effectively.

2. REQUIREMENTS ANALYSIS.

1.2.Case study:

Client	**Marlon Gomez**
User	The system's end-users will consist of individuals seeking to efficiently manage their daily tasks and reminders.
Functional Requirements	**R1. Add Tasks**
	R2. Modify Tasks
	R3. Remove Tasks
	R4. Task Classification
	R5. Sort Tasks
	R6. Undo Actions
Non-Functional Requirements	Backup and Recovery
	Browser Compatibility
	scalability
	Data Security

2.1.1. Functional Requirement N°1

R1			
Name	**Add Task**		
overview	Allow users to add new tasks by specifying title, description, due date, and priority.		
Inputs	***Input Name***	***Data Type***	***VVC***
	taskDetails	String (Title and Description), Date (Due Date), Integer (Priority)	Title and Description should not be empty, Due Date should be in the future, Priority should be within a defined range.
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	taskAdded	Boolean	True if the task is added successfully, False if there is an error.

2.1.2. Functional Requirement N°2

R2			
Name	**Modify Tasks**		
overview	Allow users to edit existing tasks to update information such as title, description, or due date.		
Inputs	***Input Name***	***Data Type***	***VVC***
	updateTaskDetails	String (Title and Description), Date (Due Date)	Title and Description should not be empty, Due Date should be in the future, Priority should be within a defined range.
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	taskAdded	Boolean	True if the task is modified successfully, False if there is an error.

2.1.3. Functional Requirement N°3

R3			
Name	**Delete Tasks**		
overview	Allow users to delete tasks that are no longer relevant.		
Inputs	***Input Name***	***Data Type***	***VVC***
	taskID	Integer	Task ID should correspond to an existing task.
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	taskDeleted	Boolean	True if the task is deleted successfully, False if there is an error.

2.1.4. Functional Requirement N°4

R4			
Name	**Categorize Tasks**		
overview	Allow users to categorize tasks as "Priority" or "Non-priority" when creating them.		
Inputs	***Input Name***	***Data Type***	***VVC***
	taskCategory	String	Should be either "Priority" or "Non-priority"
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	taskCategorized	Boolean	True if the task is deleted successfully, False if there is an error.

2.1.1. Functional Requirement N°5

R5			
Name	**Sort Tasks**		
overview	Allow users to view a list of all their tasks, sorted by due date or priority.		
Inputs	***Input Name***	***Data Type***	***VVC***
	sortingOption	String	Should be either "Due Date" or "Priority"
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	taskList	List Of Tasks	Display the tasks in the selected sorting order.

2.1.2. Functional Requirement N°6

R6			
Name	**Undo Actions**		
overview	Implement a feature that allows users to undo the last action performed in the system.		
Inputs	***Input Name***	***Data Type***	***VVC***
	undoRequest	*N/a*	*N/a*
Result or Postcondition			
Outputs	***Output Name***	***Data Type***	***Format***
	actionUndone	*Boolean*	True if the last action is successfully undone, False if there are no actions to undo or if there is an error.