**PROBLEM SPECIFICATION TABLE**

|  |  |
| --- | --- |
| **CLIENT** | Company developing the task and reminder management system. |
| **USER** | User using the system. |
| **FUNCTIONAL REQUERIMENTS** | R1. The system must allow storing tasks and reminders.  R2. The system must allow the management of priorities for tasks and reminders.  R3. The system must allow undoing actions. |
| **CONTEXT OF THE PROBLEM** | A task and reminder management system is under development for users. This system utilizes a hash table to store key information such as the title, description, and deadline of tasks. The user interface allows users to add, modify, and delete tasks, as well as sort them by deadline or priority. A priority queue is implemented to manage important tasks first, and a category for non-priority tasks is included, managed on a first-come, first-served basis. Additionally, an undo function has been integrated using a stack to reverse user actions, providing flexibility and control in the management of tasks and reminders. |
| **NON-FUNCTIONAL REQUERIMENTS** | RNF1. Task Display  RNF2. Intuitive Interface. |

**REQUIREMENTS TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier or name** | **R1: The system must allow storing tasks and reminders.** | | |
| **Summary** | The application must be able to store tasks and reminders using hash tables, employing a key-value pair system. This allows the use of a unique identifier as a key, facilitating the retrieval of a single corresponding value. | | |
| **Inputs** | **Input name** | **Data type** | **Condition or valid values** |
| Id | String | Must be unique |
| Title | String | N/A |
| Description | String | N/A |
| Due date | String | Format dd/mm/yyyy |
| Type of item | int | Must be 1 or 2:   * 1: Task * 2: Reminder |
| Is the task priority? | int | Must be 1 o 2:   * 1: Yes * 2: No |
| Importance level | int | Must be 1, 2, 3 or 4:   * 1: Low * 2: Moderate * 3: High * 4: Very high |
| **General activities required to achieve the results** | * Calculate hash index * Insert elements into the table * Search for elements in the table * Delete elements from the table | | |
| **Outcome or Postcondition** | The table is updated correctly after performing insertion, search, or deletion operations. | | |
| **Outputs** | **Output name** | **Data type** | **Selection or repetition condition** |
| Confirmation message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier or name** | **R2: The system must allow the management of priorities for tasks and reminders.** | | |
| **Summary** | The system should have two categories for tasks: "Priority" and "Non-priority."  For the priority tasks, is it necessary to utilize a priority queue to organize tasks based on their level of importance. When a user adds a new task, it is inserted into the priority queue according to its level of importance, ensuring that important tasks are handled first.  For the non-priority tasks, is it necessary to establish a category for non-priority tasks to manage those without assigned priority. This category allows handling tasks based on their order of arrival, following the First In, First Out (FIFO) principle. | | |
| **Inputs** | **Input name** | **Data type** | **Condition or valid values** |
| Is the task priority? | int | Must be 1 o 2:   * 1: Yes * 2: No |
| Importance level | int | Must be 1, 2, 3 or 4:   * 1: Low * 2: Moderate * 3: High * 4: Very high |
| **General activities required to achieve the results** | * Insert priority tasks to the priority queue. * Queue non-priority tasks to the non-priority task queue. | | |
| **Outcome or Postcondition** | Queues are updated each time a task is added. | | |
| **Outputs** | **Output name** | **Data type** | **Selection or repetition condition** |
| Confirmation message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier or Name** | **R3: The system must allow undoing actions.** | | |
| **Summary** | The system must implement a user interface feature that allows users to undo the last performed action. When the user chooses the "Undo" option, it triggers the undo method, which reverses the most recent action. | | |
| **Inputs** | **Input name** | **Data type** | **Condition or valid values** |
| Undo option | int | Must be 4 |
| **General activities required to achieve the results** | * Stack an action each time it is executed. * Pop the stack of actions. * Delete an element if the last action was to add it. * Add an element and return it to the hash table and non-priority queue (if it was a non-priority task) if the last action was to delete it. * Return the original element if the last action was to edit it. | | |
| **Outcome or Postcondition** | Undo the last action, whether it's an addition, a deletion, or an edition. | | |