**Project’s Requirements**

* Daniel José Plazas (A00400085)
* Sebastián Erazo (A00400086)
* Juan Manuel Casanova (A00400090)

**Problem Specification Table**

|  |  |
| --- | --- |
| CLIENT | Aristizabal and Marlon |
| USER | Workers |
| FUNCTIONAL REQUIREMENTS | R1: Add tasks    R2: Modify tasks    R3: Remove tasks  R4: Display tasks  R5: Priority Management  R6: Undo actions |
| CONTEXT OF THE PROBLEM | The system should allow users to manage their tasks and reminders in an effective and organized manner. This system should allow users to add new tasks, providing details such as title, description, deadline and priority level. In addition, the system should provide an intuitive interface that allows users to modify and delete existing tasks as needed.  The system should classify tasks into two main categories: "Priority" and "Non-priority". Priority tasks should be managed in a prominent and urgent manner. For this, the system should use a priority queue that allows users to focus on tasks of high importance first. Non-priority tasks, on the other hand, should be handled on a first-come, first-served basis, using a FIFO (First-In-First-Out) structure to ensure that they are handled in the order in which they were added. In addition, an "undo" functionality should be implemented to allow users to revert the last action performed in the system. |
| NON-FUNCTIONAL REQUIREMENTS | RN1: The system must be efficient in terms of execution time and resource usage, especially when ordering and accessing tasks.  RN2: The user interface must be easy for users to use and understand.  RN3: The system must be robust and handle possible errors, such as incorrect user input or lack of memory.  RN4: User data, including tasks and reminders, must be secure and protected against unauthorized access. |

**Functional Requirements Analysis Tables**

| Name or identifier | R1: Add tasks | | |
| --- | --- | --- | --- |
| Summary | The system should allow users to add new tasks and reminders easily and efficiently. When doing so, users should be able to enter essential details such as the title of the task or reminder, a detailed description, the deadline for its completion and specify its priority level. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| title | String | Text string |
| description | String | Text string |
| deadline | Calendar | Valid date (Cannot be less than the current date) |
| priority | int | An integer can be 1 or 2. |
| Result or postcondition | The message "The task has been successfully created" will be displayed. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String | The message "The task has been successfully created" will be displayed. |

| Name or identifier | R2: Modify tasks | | |
| --- | --- | --- | --- |
| Summary | The system must allow users to make modifications to previously entered tasks and reminders. For this purpose, an accessible editing function should be provided in the user interface. Users will be able to modify details such as the title, description, due date and priority of existing tasks or reminders. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| keyModify | int | The key must exist.  Positive Integers |
| newValue | Task | The task must exist.  Valid values must be entered. |
| Result or postcondition | The program will display 3 messages.   * The hash table before changes (Works for the user to choose which element to change) * The modified element * The hash table after changes   The task will be successfully modified. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String | The program will display the messages. |

| Name or identifier | R3: Remove tasks. | | |
| --- | --- | --- | --- |
| Summary | The system must allow users to delete previously entered tasks and reminders. To meet this requirement, a delete function accessible in the user interface shall be provided. Users will be able to select the task or reminder they wish to delete and confirm its removal. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| keyDelete | int | Must be the value of a key that exists.  Positive integers |
| Result or postcondition | The program will display 3 messages.   * The hash table before changes (Works for the user to choose which element to change) * The deleted element * The hash table after changes   The task will be successfully modified. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String | The program will display the messages. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Display tasks | | |
| Summary | The system should allow users to view a complete list of all tasks and reminders entered. This list shall be sorted either by task deadline or by reminder date, allowing users to prioritize their activities according to their deadlines. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| key | int | Must be the value of a key that exists.  Positive integers |
| Result or postcondition | * The values of the hash table will be displayed. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| key | int | Must be the value of a key that exists.  Positive integers |
| value | String | The value associate with the key |
| msg | String | ToString |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Priority management | | |
| Summary | The system must allow users to manage the priorities of their tasks and reminders through two defined categories: "Priority" and "Non-priority". To satisfy this requirement, a classification system will be implemented that will allow users to assign the corresponding category to each task or reminder.  For priority tasks:  The system will use a priority queue to organize these tasks. And they will be sorted according to the due date.  For non-priority tasks:  These tasks will be handled following the "First in First Out" (FIFO) principle. This means that these tasks will be managed in the order in which they were entered. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| priority | int | The user has the possibility to enter 1 or 2 (1. High priority, 2. Low priority). |
| Result or postcondition | * The task priority will be assigned correctly * Tasks with high priority will be queued in a separate queue and will be handled on a first-come, first-served or due date order. * Low priority tasks will be placed in a separate queue and processed on a first-come, first-served or due basis. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R6: Undo actions | | |
| Summary | The system must have a functionality that allows users to revert to the last action performed in the system. This feature is achieved by maintaining a Last-In-First-Out (LIFO) stack that records user actions. Each time a user performs an action, such as adding, modifying or deleting a task, that action is recorded in the stack along with the relevant details of the affected task. | | |
| Inputs | Input name | Data type | Selection or repetition condition |
| userAction | String |  |
|  | task | Task |
| Result or postcondition | * The last action performed by the user has been reversed and the system has returned to its state prior to the undone action. * A message is provided indicating that the action has been successfully undone or, in case there are no actions to undo, the user is informed that there are no actions available to undo. * The table with the data before the last action will be displayed. | | |
| Outputs | Output name | Data type | Selection or repetition condition |
| msg | String | The table with the tasks before the change. |