# SOFTWARE ENGINEERING REQUIREMENTS SPECIFICATION TABLE

Members:

* Juan José De La Pava - A00381213 + Marlon Gómez Victoria
* David Artunduaga Penagos - A00396342 + Marlon Gómez Victoria
* Rony Farid Ordoñez - A00397968 + Jeison Mejía Trujillo

|  |  |
| --- | --- |
| CUSTOMER | Icesi University |
| USER | Customers of the Todo List app |
| FUNCTIONAL REQUIREMENTS | RF0: To store reminders.  RF1: Modify the reminders.  RF2: Delete Reminders  RF3: Manage the priority of reminders.  RF4: Display reminders  RF5: Allow the user to undo an action. |
| CONTEXT | The problem consists of the need to design and develop a task and reminder management system that allows users to efficiently add, organize, and manage their to-dos and personal reminders. For this reason, the program must be able to manage priority tasks and have an intuitive user interface. |
| NON-FUNCTIONAL REQUIREMENTS | RFN1: Time complexity analysis  RFN2: Spatial Complexity  RFN3: TAD implementation  RFN4: Class Diagram Design  RFN5: Test case design  RFN6: Intuitive User Interface.  RFN7: Use of multiple data structures |

## Functional requirements analysis table

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | RF0: Store reminders | | |
| Summary | Users must be able to store new reminders in the system. Each reminder should include information such as title, description, deadline date, and priority. Reminders are organized based on their priority and arrival order. | | |
| Input | Entry name | Data type | Selection or repetition condition |
| title | String | REQUIRED |
| description | String | REQUIRED |
| deadline | Calendar | The date must be greater than the actual date |
| priority | Enum | It must be either (Priority or Not a priority) |
| Result or postcondition | The system stores the reminder based on its priority and arrival order. In the corresponding data structure. | | |
| Outputs | Entry name | Data type | Format |
| msg | String | A message indicating the stage of the transaction |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | RF1: Modify reminders | | |
| Summary | Users must be able to modify the information of existing reminders. This includes the ability to change the title, description, deadline, and priority of a reminder. | | |
| Input | Entry name | Data type | Selection or repetition condition |
| reminderID | String | The reminderID must exist in the current reminders |
| newTitle | String | N/A |
| newDescription | String | N/A |
| newDeadline | Calendar | It must be greater than the actual date |
| newPriority | Enum | It must be either (Priority or Not a priority) |
| Result or postcondition | The system updates the reminder information. | | |
| Outputs | Output name | Data type | Format |
| msg | String | A message indicating the stage of the transaction |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | RF2: Delete Reminders | | |
| Summary | Users must be able to delete reminders from the system when they are no longer needed. | | |
| Input | Entry name | Data type | Selection or repetition condition |
| reminderID | String | The reminderID must exist in the current reminders |
| Result or postcondition | The system removes the reminder from storage. | | |
| Outputs | Output name | Data type | Format |
| msg | String | A message indicating the stage of the transaction |

| Name or identifier | RF3: Manage the priority of reminders | | |
| --- | --- | --- | --- |
| Summary | The system must categorize reminders into multiple categories: “High priority”, “Medium priority”, “Low priority”, “Optional” and “non-priority”. All the tasks should be inserted in a hash table. The priority tasks should be organized by priority with the help of a priority queue and the non-priority tasks must be organized by their order of arrival with the help of a queue. | | |
| Input | Entry name | Data type | Selection or repetition condition |
| N/A | N/A | N/A |
| Result or postcondition | The system assigns reminders to the specified priority and data structure. Each structure should be organized by priority or by order of arrival depending on the structure. | | |
| Outputs | Output name | Data type | Format |
| N/A | N/A | N/A |

| Name or identifier | RF4: Display Reminders | | |
| --- | --- | --- | --- |
| Summary | Users must be able to view a list of all their reminders. The software should let the user choose if he wants to see all of the tasks, the priority tasks, and the non-priority ones. | | |
| Input | Entry name | Data type | Selection or repetition condition |
| N/A | N/A | N/A |
| Result or postcondition | The system displays the list of reminders sorted according to the user's choice. | | |
| Outputs | Output name | Data type | Format |
| List of reminders | ¿? | Scrollable list |

| Name or identifier | RF5: Allow the user to undo an action | | |
| --- | --- | --- | --- |
| Summary | Implement a function that allows users to undo the last action performed in the system. This includes the ability to undo adding, modifying, or deleting reminders. This functionality should be implemented with the help of a stack, where each action performed by the user will be added into the stack. | | |
| Input | Entry name | Data type | Selection or repetition condition |
|
| Result or postcondition | The system reverses the last user action based on the information stored in the undo stack. | | |
| Outputs | Output name | Data type | Format |
| N/A | N/A | N/A |