**PHASE 1: PROBLEM IDENTIFICATION**

**Identification of needs and symptoms:**

* The users of the company require a system that allows users to store and manage their to-dos and reminders effectively.
* The company that acquires the system lacks a user interface that facilitates the addition, modification and deletion of users tasks and reminders.
* The users of the company require a system that categorizes their to-dos as "Priority" and "Non-priority" for better organization.
* The company that acquires the system lacks a user interface needed to sort and display tasks and reminders by deadline or priority.
* The users of the company want to track their actions performed on the task manager system and allow them to undo them if necessary.
* The company that acquires the system lacks an "undo" function to allow users to reverse actions performed in the system.
* The solution to the problem must be efficient so that the service can be delivered to the greatest number of users with the minimum consumption of resources.
* The solution to the problem requires the use of different kinds of data structures, such as: hash table, priority queues, stack and queues.

| **Client** | **Company that acquires the system** |
| --- | --- |
| **User** | Person using the task and reminder management system. |
| **Functional requirements** | RF1: Add tasks and reminders.  RF2: Modify tasks and reminders.  RF3: Delete tasks and reminders.  RF4: Undo actions.  RF5: View an ordered list of tasks and reminders. |
| **Problem Context** | The problem focuses on the design and development of a task and reminder management system that encompasses aspects of storage, organization, prioritization and control of actions performed by the user. |
| **Non-functional requirements** | RNF1: The system is expected to be efficient in the management of tasks and reminders.  RNF2: The system must have a hash table to store data and a priority queue to manage priority tasks.  RNF3: The system must have a user interface that allows adding, modifying and deleting tasks and reminders. |

| **Identifier and Name** | **RF1: Register Tasks and Reminders** | | | |
| --- | --- | --- | --- | --- |
| **Summary** | The system should allow users to register various activities, including tasks and reminders. Once the user selects this option from the menu, the system will prompt the user to provide a title, description, and deadline for the activity. | | | |
| **Inputs** | **Input Name** | **Type** | | **Valid values condition** |
| Title | String | | NA |
| Description | String | | NA |
| DeadLine | Calendar | | dd-mm-yyyy |
| priority | int | | 1. priority 2. not a priority |
| **Results** | If the provided inputs are valid, the system will generate a new "to-do" object. Subsequently, this information will be stored in a hashtable and the process on the stack of actions.  In the event that the inputs are invalid, the system will issue an error message, alerting the user to the issue, and request them to re-enter the necessary information. | | | |
| **Outputs** | **Output Name** | | **Type** | **Format** |
| message | | String | Example:  “Invalid Option” |

| **Identifier and Name** | **RF2:Modify Tasks and Reminders** | | | |
| --- | --- | --- | --- | --- |
| **Summary** | The system should allow users to modify activities, including Tasks and reminders. Once the user selects this option from the menu, the system will prompt them to provide the key of the activity they wish to modify and the new information | | | |
| **Inputs** | **Input Name** | **Type** | | **Valid values condition** |
| key | String | | an existing identifier |
| new information | String | | NA |
| **Results** | If the provided input is valid, the system will search for the activity and update its information in the HashTable. Subsequently, this process would be stored on the action stack. In case the inputs are invalid, the system will issue an error message, alerting the user about the issue, and request them to re-enter the necessary information | | | |
| **Outputs** | **Output Name** | | **Type** | **Format** |
| message | | String | example “correct change” |

| **Identifier and Name** | **RF3: Delete Tasks and Reminders** | | | |
| --- | --- | --- | --- | --- |
| **Summary** | The system should allow users to delete activities, including Tasks and reminders. Once the user selects this option from the menu, the system will prompt them to provide the key of the activity they want. | | | |
| **Inputs** | **Input Name** | **Type** | | **Valid values condition** |
| key | String | | an existing identifier |
| **Results** | If the provided input is valid, the system will search for the activity and remove it from the HashTable. Subsequently, this process would be stored on the stack of actions.  In the event that the inputs are invalid, the system will issue an error message, alerting the user to the issue, and request them to re-enter the necessary information. | | | |
| **Outputs** | **Output Name** | | **Type** | **Format** |
| message | | String | Example:  “Invalid Option” |

| **Identifier and Name** | **RF4:Undo Actions** | | | |
| --- | --- | --- | --- | --- |
| **Summary** | The system should allow users to undo the last action performed. Once the user selects this option from the menu, the system will revert the last action performed | | | |
| **Inputs** | Input Name | **Type** | | **Valid values condition** |
| NA | NA | | NA |
| **Results** | If there are previous actions performed by the user, the system will undo the last action and update the HashTable and action stack accordingly. In case there are no previous actions, the system will issue an error message, alerting the user about the issue | | | |
| **Outputs** | Output Name | | **Type** | **Format** |
| message | | String | example “correct undo” |

| **Identifier and Name** | **RF5: View an ordered list of tasks and reminders** | | | |
| --- | --- | --- | --- | --- |
| **Summary** | The system should allow the user to view a list of all tasks and reminders, which should be sorted by the deadline or priority. | | | |
| **Inputs** | **Input Name** | **Type** | | **Valid values condition** |
| orderingOption | int | | NA |
| **Results** | When selecting the option to view the list of sorted tasks and reminders, a list with these will be displayed. | | | |
| **Outputs** | **Output Name** | | **Type** | **Format** |
| sortedList | | Hashtable | Tasks must be sorted according to the sorting option selected. |

**PHASE 2: GATHERING THE NECESSARY INFORMATION**

The questionnaire elicitation technique was selected as the primary approach for requirement gathering due to its ability to provide an organized and clear structure during the interview with the client. This method allowed for specific questions to be asked that addressed critical aspects of the Task and Reminder Management System, ensuring a detailed understanding of the client's needs and expectations. Through this technique, valuable information was obtained that will serve as an essential guide in the system's design and development process.

**Summary of the Elicitation Technique:**

The questionnaire elicitation technique was used to gain a clearer understanding of the client's requirements for the Task and Reminder Management System. This technique involves preparing a set of structured questions presented to the client during an interview to gather specific information about their needs and expectations regarding the system.

**Questionnaire Questions and Answers:**

1. **What is the distinction between tasks and reminders?**

**Answer:** The client mentioned that there is no difference in attributes and behavior beyond them being different types of activities. Therefore, reminders also can have a priority.

1. **How would you like tasks and reminders to be sorted and presented to users in the interface?**

**Answer:** The client wants users to be able to view a list of tasks and reminders sorted either by deadline or priority. A task can be either prioritized or non-prioritized, meaning that there isn't an attribute that defines whether a prioritized task is more important than another with the same property, apart from the deadline.

1. **The "Undo" function was mentioned. How do you expect this function to work in terms of action reversal?**

**Answer:** The client expects the "Undo" function to allow users to undo the last action taken in the system, using a stack to track actions. If the user selects the "undo" option twice in a row, the last "undo" will not be reversed.

1. **It's mentioned that it's necessary to store prioritized and non-prioritized activities in different data structures. Why is a hash table then used to store both?**

**Answer:** The client suggests that the hash table be used to implement a simpler search algorithm with low time and space complexity. The other two data structures to be implemented for prioritized and non-prioritized activities, priority queue and queue respectively, are intended for users to view a sorted list based on specific criteria.

1. **Is it possible to implement all the data structures provided by Java, or is it necessary to develop our own implementation?**

**Answer:** The client expects the implementation of the solution to use a custom implementation that adheres to good programming practices, as a way to showcase the team members' mastery of the topics involved.

1. **Is it possible to implement a heapsort algorithm on hashtable data structure?**

**Answer:** The client corrected and mentioned that it's not necessary to implement a heapsort algorithm on the hashtable.

**PHASE 3: SEARCH FOR CREATIVE SOLUTIONS**

**Solution 1: Option List (menu )**

When the user enters the program, a menu is displayed, allowing them to select a predefined option. Each option is associated with a specific action. The menu includes the following choices:

Menú:

1. Register tasks and reminders
2. Modify tasks and reminders
3. Delete tasks and reminders
4. Undo actions
5. View an ordered list of tasks and reminders

Please select an option:

When the user chooses the first option, they will be prompted to enter information to register a task or reminder, including the Title, Description, Due Date, and Priority. After the user provides the required information, the system will create a task or reminder using that information and store it.

Once the user has performed the first option, they can proceed to modify the stored tasks or reminders by selecting the second option. When selecting this option, the system will request a key to identify the task or reminder to be modified, and the user can then update the information accordingly. Similarly, the user can also delete tasks or reminders based on a key by choosing the third option. The system will prompt the user to enter a key and will handle the deletion process.

As the system keeps track of each user's actions, selecting the fourth option in the menu, which is to undo the last action performed, will be automatically executed by the system.

Finally, the user can view an ordered list of tasks and reminders. These items will be sorted by deadline, with tasks having the highest priority displayed first. They will be organized by date, meaning that the tasks with the soonest due dates will appear first. Non-priority tasks will follow these.

**Solution 2 : to-do list**

When the user enters the program, a list with the name and deadlines of all the tasks they have stored up to that point will be displayed, and below it, a menu with three options: select a task, register a new task and exit the program. Here is an example:

List:

Priority tasks:

1. CyED homework: 8/10/23
2. Math homework: 10/10/23

Non-Priority Tasks:

1. End of the Semester: 1/12/23
2. Halloween: 31/10/23

Menú:

1. Select Tasks
2. Register new task

*Enter an option:*

When selecting the first option, the user will be prompted to enter a number corresponding to the task they are looking for on the list; then, the program will display the information about that task and ask if they want to edit the information contained in the task , delete the task or return to the main menu. Following the same example:

*Enter the task number:* 1

Title: CyED homework

Description: Deliver integrative task 1

DeadLine: 8/10/23

State: Priority

Task menú:

1. Edit task.
2. Delete task.
3. Return to the main menu.

*Enter an option:*

If the user wants to edit the task, the system would ask for the new attributes: title, description, deadline and state. If the user selected the option of deleting the task, the system would remove the task from the list and make the user return to the main menu.

On the other hand, if the user chooses the second option from the menu, the program will request the relevant information, and after this information is entered, it will create the task and add it to the list. Lastly, if the user chooses the third option, the system will revert the last action performed except if it was another undo action.

**Solution 3 : to-do list (Auto-generate number)**

When the user enters the program, the list of al registered task is displayed, whit this information the user can interact using a number auto-generate by the system when the user register a new task in the system.

Use cases :

System start :

List of task

1. CyED homework: 8/10/23
2. Math homework: 10/10/23
3. Halloween: 31/10/23
4. End of the Semester: 1/12/23

Menú

1. Select task
2. Register task
3. Undo last action

cases of the options :

**When the user selected the option “select task”**

* The system allows the user to enter the automatically generated unique number that identifies a specific task
* Once the user enters the unique number, the system verifies and displays the complete details of the task associated with that number.
* Alongside the task details, a “task menú“ is displayed , offering the following options :

Task Menú :

1. Delete task
2. Edit task
3. Exit to main menu

Delete task : Allows te user to delete the selected task.

Edist task : Allows the user to edit the details of selected task.

Exit to main menu : Permits the user to return to the main menu of the system

The user can select one of the options from the "Task Menu" to perform the desired action.

**When the user selected the option “Register task”**

* The system allows the user to input details of a new task, such as the title , description, due date, priority.
* Upon entering the task details, the system automatically generates a unique number to identify the new task.
* the system stores the new task along the automatically generated unique number.
* after successfully registering the task, the system returns to the main menu.

**PHASE 4: TRANSITION FROM THE FORMULATION OF IDEAS TO DESIGNS**

**PRELIMINARY**

Careful review of the other alternatives leads us to the following:

**Solution 1: Option List (menu )**

* **Usability :** This solution provides a simple user interface in the form of a menu. This makes it easy for users to understand how to use the system and quickly select the actions they want to perform.
* **Ease of implementation:** implementing a menu with predefined options is often simpler from a programming standpoint compared to other solutions that require displaying stored tasks
* **Action Tracking :** This solution also addresses the “undo actions” functionality effectively by keeping a record of user actions and offering an option to undo the last action. This is simpler to implement and understand within a menu.
* **Explicit sorting**: This solution also addresses the need to sort tasks and reminders by specifying how they will be organized , either by deadline or priority, providing clarity about the system’s functionality.
* **Higher Time Consumption:** Compared to other solutions that may allow the simultaneous viewing of multiple tasks or quicker access to specific functions, this solution may require more time to perform certain actions.

**Solution 2: to-do list**

* **Simplicity and Clarity:** This method provides a user-friendly interface as it allows easy access to tasks and reminders.
* **Issue with large taks volumes:** If a user has a large number of stored tasks, finding and managing a specific task could become complicated.
* **Limitations regarding the code**: having to implement multiple data structures, search, and printing functions may be limited as they are forced to use a number assigned to each task.
* **Increased Time request:** As the list needs to be printed in a specific order and can be edited at any time, these changes will have to be displayed each time the menu is executed, which would result in increased execution time for the code

**Solution 3 : to-do list (auto-generate number)**

**Reasons to Discard Solution 3:**

* **Additional Complexity :** the automatic generation of unique numbers to identify task adds an unnecessary layer of complexity to the system. This not only complicates the code but can also make the system more difficult to understand and maintain
* **Potential User Confusion:** Introducing unique numbers to identify tasks may confuse users. Instead of using more understandable names or descriptions, users would have to remember or refer to unique numbers, which could be less intuitive and less user-friendly.
* **Increased Error Proneness:** The automatic generation of unique numbers and their management can lead to errors, such as the generation of duplicate numbers or loss of information if a unique number is lost or corrupted. This can increase the possibility of system failures.
* **Increased Development Effort**: Implementing the functionality of auto-generation of unique numbers and their management will require additional development effort compared to a simpler solution that does not involve this feature. This could increase costs and development time.

For these reasons, the only viable viable options are the first and second solution.

**PHASE 5: EVALUATION AND SELECTION OF THE BEST SOLUTION**

**Criterio A. Efficiency of the System:**

[4] The system provides a quick response to user requests with minimal loading times.

[3] The system generally maintains acceptable response times, although occasional slowdowns may occur.

[2] The system can be slow at times, with response times occasionally slower than desired.

[1] The system frequently experiences significant performance degradation.

**Criterio B. Compliance with Customer Requirements:**

[5] The system design fully complies with all customer requirements and expectations, as established in the requirements elicitation phase.

[4] The system design largely complies with customer requirements, addressing most key expectations.

[3] The system design meets a significant portion of customer requirements, but there are some areas of discrepancy or necessary adjustments.

[2] The system design only partially complies with customer requirements, leaving multiple areas unfulfilled.

[1] The system design does not meet the majority of customer requirements, requiring a substantial review and adjustments.

**Criterio C. Scalability and Handling of Large Data Volumes:**

[2] The system is capable of effectively handling a large number of tasks and users without significant performance degradation.

[1] The system's scalability is acceptable, but performance issues may arise with extremely large workloads.

**Criterio D. User Experience:**

[3] The user interface is highly intuitive and provides a smooth and enjoyable experience for users.

[2] The user interface is functional and offers an adequate user experience.

[1] The user interface is usable but requires improvements in terms of usability and design.

|  | **Criteria A** | **Criteria B** | **Criteria C** | **Criteria D** | **Total** |
| --- | --- | --- | --- | --- | --- |
| **Solution 1. Option list menu** | **[3]** | **[5]** | **[2]** | **[2]** | **[12]** |
| **Solution 2. to-do list** | **[1]** | **[3]** | **[1]** | **[3]** | **[8]** |

**Selection**

Based on the previous evaluation, Alternative solution 1 should be chosen as it achieved the highest score according to the defined criteria..