ENGINEERING DESIGN METHOD

PHASE 1: PROBLEM IDENTIFICATION

Problem Context:
At ICESI University, efficient time management is crucial due to the institution's high academic standards. With the growing demand for excellence in academic performance, it is imperative to provide students with effective tools that allow them to manage their tasks and reminders individually. Currently, many students face the frustration of relying on manual solutions or disjointed organizational systems, leading to a loss of efficiency and productivity in managing their daily tasks and reminders.

Problem Definition:
The problem lies in the inefficiency and lack of productivity in managing students' daily tasks and reminders, which negatively impacts their academic performance and overall well-being. Therefore, ICESI University needs a task and reminder management system that enables its students to efficiently manage their time and academic responsibilities.

Causes and Symptoms of the Problem:

- Dependence on Manual Solutions: Students currently heavily rely on manual methods or disjointed systems to manage their tasks, often resulting in inefficiencies and frustration.
- Time Loss: The lack of an effective task management tool can lead to a significant loss of time when trying to remember and keep track of all tasks and responsibilities.
- High Stress Levels: The inability to effectively manage tasks can generate high levels of stress among students, which, in turn, can affect their well-being and academic performance.
- Disorganization: Students may feel disorganized and overwhelmed due to the lack of structure for their tasks and reminders.
- Lack of Prioritization: The absence of a system that allows task prioritization can result in students spending time on less important tasks instead of focusing on critical ones.
- Errors and Wrong Decisions: Without the ability to undo actions, students may make mistakes when making changes to their tasks, leading to incorrect decisions and additional frustration.

Problem Needs:

- Efficiency in Time Management: ICESI University students need an efficient way to manage their time to meet the institution's rigorous academic demands.
- Task Organization: There is a need for a tool that allows students to organize their tasks and reminders in a structured and logical manner.
- Task Prioritization: Students need a way to prioritize their tasks to ensure they focus on the most important and urgent ones.
- Confidence and Control: Having the ability to undo actions in the system provides students with a greater sense of control and confidence in their ability to manage their tasks.

Functional requirements specification

Client	ICESI University
User	User
Problem context	*A task and reminder management system is needed that allows users to add, organize and manage their to-dos and reminders.* *The system must contain specific components and functionalities for:* - *Store tasks and reminders in a hash table.* - *Allow users to add, modify and delete tasks and reminders through a user interface.* - *Manage task priorities using priority queues and queues.* - *Undo actions performed by the user in the system using a stack that allows tracking of these actions.*
Functional requirements	The system must allow users to add **RF1.** Add a task or reminder. **RF2.** Modify task or reminder. **RF3.** Delete task or reminder. **RF4.** View tasks and reminders. **RF5.** Undo last action performed.
Non-functional requirements	- The user interface should be intuitive and easy to use.

Identifier and name	*RF1. Add a task or reminder.*		
Summary	*The system should allow the user to add a task or a reminder by entering the following information:* - *Title* - *Description* - *Date* - *Time* *Additionally, the user must specify whether it is a task or a reminder. In the case of a task, they will be asked to indicate whether it is prioritized or not, and if it is prioritized, they should specify its priority level: HIGH, MEDIUM, or LOW.* *All tasks and reminders should be stored in a hash data structure. Furthermore, prioritized tasks should be stored in priority queues, and non-prioritized tasks in regular queues.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Condition**
	title	String	*Maximum 70 characters allowed.*
	description	String	*Maximum 600 characters allowed.*
	date	Calendar	*Date must be equal to or later than the current date.*

	dayTime	Calendar	*Time must be later than the current time.*
	isTask	Boolean	*True if it's a task.* *False if it's a reminder.*
	isPriority	Boolean	*Only entered when it's a task.* *True if it's a priority task.* *False if it's not.*
	levelPriority	Priority	*Only entered when it's a task and isPriority is true.* *Priority levels: HIGH, MEDIUM, LOW.*
Result or Postcondition	The system validates that the inputs meet the conditions. If they do, it adds the task or reminder to the hash table. If it's a task, it stores it in a priority queue if it's prioritized or in a regular queue if it's not. Finally, the system informs the user that the task or reminder has been added successfully and provides them with the key. On the other hand, if the inputs do not meet the conditions, the system displays an alert message to the user.		
Outputs	**Output Name**	**Data Type**	**Format**
	message	String	*"The task/reminder has been successfully added. its key is: XXXXX"*
	error	String	*"Error: …"*

Identifier and Name	*RF2. Modify Task or Reminder*		
Summary	*The system should allow the user to modify a specific task or reminder, given a key and the name of the task or reminder. The user must enter which data they want to change and the new value to be assigned.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Condition**
	key	String	*Numbers or letters can be entered.*
	name	String	Numbers, letters, or special characters can be entered.
	data	String	Puedes ingresar las palabras "title," "description," "date," "time of day," "if it's a task," "if it's

			prioritized," and "priority level."
	value	String	Numbers, letters, or special characters can be entered.
Result or Postcondition	The system has located the task or reminder and has modified the selected feature as chosen by the user with the provided new value.		
Outputs	**Output Name**	**Data Type**	**Format**
	alert	String	*The system informs whether the object could be modified or not in the form of an alert or message.*

Identifier and Name	*RF3. Delete Task or Reminder*		
Summary	*The system allows the user to delete a specific task or reminder, given a key and the name of the task or reminder.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Condition**
	key	String	*Numbers or letters can be entered.*
	name	String	Numbers, letters, or special characters can be entered.
Result or Postcondition	The system found the task or reminder and deleted the associated object.		
	Output Name	**Data Type**	**Format**
Outputs	alert	String	*The system informs whether the object could be deleted or not in the form of an alert or message.*

Identifier and Name	*RF4. View tasks or reminders*		
Summary	*The system will allow the user to view the tasks and reminders that have been added.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Condition**
Result or Postcondition	The system displayed all the tasks and reminders added by the user.		
Outputs	**Output Name**	**Data Type**	**Format**
	alert	String	*The system displays all the added tasks or reminders in the*

			form of an alert or message.

Identifier and Name	*RF5. Undo last action performed*		
Summary	*The system should allow undoing the last action performed by the user.*		
Inputs	**Input Name**	**Data Type**	**Valid Value Condition**
Result or Postcondition	The system undoes the last action performed by the user and shows the user the state of the program just before executing the undone action.		
Outputs	**Output Name**	**Data Type**	**Format**
	alert	String	*The system displays the state of the program just before executing the undone action in the form of an alert or message to the user.*

PHASE 2: COLLECTION OF NECESSARY INFORMATION

With the aim of having a complete understanding of the involved concepts, a search for definitions of both theoretical and practical terms is carried out.

- Task Management: The process of monitoring and organizing project tasks throughout different stages, from initiation to completion. This involves clearly defining the tasks to be completed, planning and allocating time for each task, identifying the most important and urgent tasks, tracking deadlines appropriately, adapting to changes in workload, and the ability to make informed decisions on how to use available time and resources effectively.
- Productivity: The ability to produce goods or services using available resources efficiently. In the context of time management, productivity refers to the ability to complete tasks and projects effectively and efficiently. This includes task organization and planning, identifying important and urgent tasks, reducing stress by providing a sense of control and organization, improving efficiency by avoiding duplicated efforts, ensuring tasks are completed timely, monitoring progress toward goals, making informed decisions, and the ability for self-organization and autonomy.
- Task Prioritization: The process of establishing an order of importance for tasks to ensure they are completed effectively and efficiently. Task prioritization is important as it allows people to focus on what is crucial and avoid wasting time on less relevant activities. Some tools and techniques that can assist in task prioritization include creating a task list, using a time management matrix, identifying important tasks, and setting realistic deadlines.
- Hash Table: A hash table, also known as a hash map or hash array, is a data structure that implements an abstract data type called a dictionary, allowing the

association of keys with values. The hash table uses a hash function to transform the key into an index in the table where the corresponding value is stored. The hash function must be designed to evenly distribute keys in the table to minimize collisions. Hash tables are commonly used in applications that require fast data access, such as searching for elements in large datasets.

- Data Structures: Data structures are a particular way of organizing information in a computer so that it can be used efficiently. Different types of data structures are suitable for various applications, and some are highly specialized for specific tasks. Efficient data structures are usually crucial for designing efficient algorithms.
- Queues: A queue is a data structure used to store a collection of elements, in which the first element added is the first one to be removed. Queues are commonly used in applications that require processing data in the order of arrival, such as task scheduling and managing shared resources.
- Priority Queues: A priority queue is a data structure used to store a collection of elements, where each element has an associated priority. Elements are removed from the queue in priority order, with the highest-priority element removed first. Priority queues are commonly used in applications that require processing data in priority order, such as task scheduling and managing shared resources.
- Stack: A data structure that follows the Last In, First Out (LIFO) principle, meaning that the last element added to the stack is the first one to be removed. In other words, the last-in element is the first-out. The basic operations in a stack are "push" (to add an element to the stack) and "pop" (to remove the top element from the stack). The "undo" function can be implemented using a stack, as it can store performed actions and undo them in reverse order.
- Queue: A data structure that follows the First In, First Out (FIFO) principle, meaning that the first element added to the queue is the first one to be removed. In other words, the first-in element is the first-out. The basic operations in a queue are "enqueue" (to add an element to the queue) and "dequeue" (to remove the front element from the queue). Queues are commonly used in applications that require sequential processing, such as document printing or process management in an operating system.

Note: At the end of the document, you will find all the consulted sources.

PHASE 3: SEARCH FOR CREATIVE SOLUTIONS

Alternative 1: Custom Task and Reminder Management System
This alternative involves designing and developing a customized task and reminder management system. The system is based on a hash table data structure to store tasks and reminders. Users can add, organize, and manage their pending tasks with the ability to specify detailed information such as title, description, due date, and priority. The system also includes an intuitive user interface that allows users to add, modify, and delete tasks and reminders. Items can be viewed in a list sorted by due date or priority. Additionally, the system incorporates a task prioritization feature, allowing users to categorize tasks as "Priority" or "Non-priority." It also implements an undo action capability so that users can reverse unwanted changes to their tasks.

Alternative 2: Mobile Task Management Application

This alternative focuses on developing a mobile task management application. The application allows users to manage their tasks and reminders from mobile devices. It offers an intuitive and easy-to-use user interface for adding and organizing tasks. However, it lacks an undo feature. Developing a mobile application can be costly. The application is specifically designed for mobile devices and is not available on other platforms.

Alternative 3: Online Task Management Platform
This alternative proposes the development of a web-based platform that will enable ICESI University students to efficiently manage their tasks and reminders from any device with an internet connection. The platform will focus on providing users with an intuitive user interface that simplifies the addition, modification, and deletion of tasks. It will also include the ability to set priorities for tasks, making it easier for students to focus on their most important responsibilities. A key feature of this alternative is the ability to undo actions, giving students greater control and confidence in task management. Additionally, the platform will ensure that tasks and reminders are securely stored on online servers. While this alternative may not be as advanced as others, its online accessibility and essential features make it a practical and affordable option for improving task management efficiency among students.

PHASE 4: TRANSITION FROM IDEA FORMULATION TO PRELIMINARY DESIGNS
In this phase, a careful and critical examination and evaluation of the three proposed alternatives to address ICESI University's task and reminder management problem will be carried out. A preliminary assessment of each alternative is presented below:

Alternative 1: Custom Task and Reminder Management System
- This alternative is considered efficient in task and reminder management due to its hash table structure and task prioritization capabilities.
- The intuitive user interface makes it user-friendly and suitable for students.
- The ability to categorize tasks as "Priority" or "Non-priority" adds to its strength.
- While it is customizable, limitations in collaboration should be evaluated.
- The undo feature is an advantage, but its effectiveness needs to be assessed.
- Available on specific platforms.

Alternative 2: Mobile Task Management Application
- The mobile application is efficient but may lack advanced features.
- Mobile interfaces are usually user-friendly.
- Offers good task prioritization capabilities.
- Lacks an undo feature.
- Developing a mobile application can be costly.
- Accessible on mobile devices.

Alternative 3: Online Task Management Platform
Accessibility: The online platform allows students to access their tasks from anywhere.
Ease of Use: The intuitive user interface makes task management easy.
Prioritization: Students can set priorities for important tasks.
Undo Capability: Allows users to correct errors or undo actions.
Secure Storage.

After this preliminary evaluation, "Alternative 1: Custom Task and Reminder Management

System" is the most solid and feasible option to address ICESI University's task and reminder management problem, based on the defined criteria. Regarding "Alternative 2," it lacks the undo feature and is therefore discarded.

PHASE 5: EVALUATION AND SELECTION OF THE BEST SOLUTION

To evaluate and select the best solution among the three proposed alternatives, criteria for evaluating each alternative will be defined. The following are the criteria, their meanings, and the scales they can have:

- Efficiency (Scale: 1-10): This criterion evaluates the solution's ability to allow efficient task and reminder management. A higher score represents higher efficiency.
- Ease of Use (Scale: 1-10): It evaluates how easy it is for students to use the solution. A higher score indicates greater ease of use.
- Prioritization (Scale: 1-10): This criterion measures the solution's ability to prioritize tasks. A higher score represents better prioritization capabilities.
- Undo Capability (Scale: 1-10): This criterion assesses whether the solution offers the ability to undo actions. A higher score indicates better undo capabilities.

	Efficiency	Ease of Use	Prioritization	Undo Capability	Total
Alternative1.	8	9	9	8	34
Alternative3.	8	8	7	7	30

According to the previous evaluation, Alternative 1 should be selected since it obtained the highest score according to the defined criteria.

Bibliography

Universidad Politécnica de Madrid. (2015). Algoritmos y Estructuras de Datos Pilas LIFO y Colas FIFO. Retrieved from https://www.cartagena99.com/recursos/alumnos/apuntes/AED_fifo_lifo.pdf

Asana. (2022). Cómo priorizar las tareas y gestionar el tiempo en el trabajo. Retrieved from https://asana.com/es/resources/how-prioritize-tasks-work

Business Insights. (2021). Priorización de tareas: ¿Por qué es importante? Retrieved from https://blog.wearedrew.co/productividad/priorizacion-de-tareas-por-que-es-importante

Universidad Don Bosco. (2020). Tablas Hash. Retrieved from https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-con-estructuras-de-datos/2020/i/guia-8.pdf

DSTool. (s.f.). Tablas Hash. Retrieved from http://www.hci.uniovi.es/Products/DSTool/hash/hash-queSon.html

freeCodeCamp. (2021). Tabla hash en Javascript: Hash de arreglo asociativo en JS. Retrieved from https://www.freecodecamp.org/espanol/news/tabla-hash-en-javascript-hash-de-arreglo-asociativo-en-js/