

User Manual

Task Management System

This user manual will guide you through the functionalities of the system and provide troubleshooting tips for common issues.

Table of Contents

- Introduction #2
- Getting Started #3 to #8
- User Control Functions #9 to #13
- Task Commands #14 to #18
- Criterion Commands #19 to #20
- Application Commands #21 to #22
- Troubleshooting #23 to #25
- Additional Resources #26 to #27

Introduction

Welcome to the user manual of the Task Management System (TMS), a Java-based application designed to simplify and streamline your task organization. The TMS is a command-line tool that empowers users to define, query, and modify both simple primitive and composite tasks efficiently. The primary goal of the TMS is to facilitate task management by allowing users to create, edit, and delete tasks. A task can either be a primitive task, representing the smallest units of work, or a composite task, which is a collection of other tasks.

Commands Description

DEFINE (Task and Criteria Creation):

- **Create Primitive Task:** Primitive task is uniquely identified by a name, has a description, and a duration (indicating the minimum time required for completion). Primitive tasks in the TMS do not have any prerequisites, making them standalone units.
- **Create Composite Task:** Composite task is containing a name, a description, and a list of prerequisites. Composite tasks have prerequisite subtasks, which must be existing tasks (both primitive and composite).
- **Create Basic Task Selection Criteria:** This function allows users to create basic task selection criteria within the Task Management System, facilitating customized filtering and analysis of tasks. By utilizing the `DefineBasicCriterion` command, users can create specific criteria based on task properties, enhancing their ability to focus on relevant tasks in the system.
- **Create Composite Task Selection Criteria:** This feature empowers users to construct composite task selection criteria within the Task Management System, enabling more sophisticated and nuanced filtering of tasks. With the `DefineNegatedCriterion` and `DefineBinaryCriterion` commands, users can create composite criteria by combining existing criteria, allowing for advanced logical operations.

EDIT (Task Modification and Deletion) :

- **Delete Tasks:** This function enables users to remove tasks from the Task Management System, providing a means to manage and streamline their task list. The `DeleteTask` command allows users to delete tasks based on their name. The function will block the deletion a task if that particular task exists as a subtask in another composite,
- **Change Task Properties:** This function users to modify the properties of existing tasks within the Task Management System. The `ChangeTask` command facilitates dynamic adjustments to task attributes while ensuring the integrity of task prerequisites. If a task is a subtask for a composite task, this function will not allow the change of that task's property. This feature enhances the flexibility of task management by allowing users to adapt and refine task properties as project requirements evolve.

Query (Task Information Retrieval):

- **Print Task Information:** This feature allows users to retrieve detailed information about a specific task within the Task Management System. By using the `PrintTask` command followed by the task's name, users can access a comprehensive overview of the task, including its name, description, duration, and prerequisites.

- **Print All Tasks:** With the `PrintAllTasks` command, users can obtain a holistic overview of all existing tasks within the system. This feature is designed for users who need a comprehensive list of tasks at a glance. The output is structured to be easy to read, presenting essential information about each task, such as name, description, duration, and prerequisites. .
- **Report Task Duration:** The `ReportDuration` command enables users to determine the minimum time required to finish a specific task, considering any prerequisites associated with that task. For composite tasks, the reported duration is the sum of the minimum durations of all sub-tasks, considering prerequisite relationships. This feature is essential for project planning and scheduling, providing accurate insights into the time required to complete a task and its dependencies.
- **Report Earliest Finish Time:** The `ReportEarliestFinishTime` command allows users to identify the earliest possible completion time for a specific task. It considers the sum of the earliest finish times of all prerequisites and the duration of the task itself. This functionality is crucial for project managers and users who need to understand the critical path and earliest possible completion time of tasks in a project. It aids in optimising project timelines and resource allocation.
- **Print All Defined Criteria:** This feature provides users with a comprehensive overview of all defined criteria within the Task Management System. The `PrintAllCriteria` command is designed to present these criteria in a simplified and standardized form, ensuring clarity and ease of interpretation

- **Search for Tasks Based on an Existing Criterion:** This powerful feature allows users to perform targeted searches within the Task Management System based on a predefined criterion. The **Search** command enables users to list all tasks that satisfy the specified criterion, facilitating focused and efficient task retrieval according to custom criteria.

Saving and Loading:

- **Store Defined Tasks and Criteria into a File:** This feature provides users with the capability to persistently store the defined tasks and criteria within the Task Management System into an external file. The **Store** command allows for data preservation and portability, enabling users to save their project configurations for future use or sharing.
- **Load Tasks and Criteria from a File:** This feature offers users the ability to import previously defined tasks and criteria into the Task Management System from an external file. The **Load** command facilitates the seamless integration of pre-existing project configurations, enabling users to pick up where they left off or collaborate by sharing project states.

System Close:

- **Terminate Execution of the System:** This feature provides users with a straightforward way to gracefully exit and terminate the execution of the Task Management System. The `Quit` command allows users to conclude their session, ensuring a clean and controlled closure of the system.

Help:

- The `Help` command in this Java application serves as a command-line interactive help service, providing assistance and information about different aspects of the system. Upon invocation, the function presents a menu with three options:
- Option 1 (`About this system`): If the user selects this option, the function will provide a description of the system.
- Option 2 (`How to write commands`): If the user selects this option, the function enters a loop where it prompts the user to inquire about the syntax of specific commands related to the system. It then responds with the appropriate syntax for the requested command. This feature helps users understand how to structure commands for interacting with the system.
- Option 2 (`Contact the developers`): If the user selects this option, the function provides information on how to contact the developers of the system, including details

such as institution, location, and phone number. It emphasizes that only WhatsApp messages are entertained for direct communication.

- Overall, the helper function enhances the user experience by offering a user-friendly help service, guiding users on system details, command syntax, and providing a means to contact the developers for additional support.

(A)Task Command

1) Create Primitive Task Command

Syntax == (1)CreatePrimitiveTask (2)Name (3)Description
(4)Duration (5) ,

Example == CreatePrimitiveTask pTask1 Testing-description 0.6 ,

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
CreatePrimitiveTask pTask1 Testing-description 0.6 ,
Simple task created: pTask1
```

POINTS TO NOTE:

- **Number of Input** - 5, in order of the positional place holders (1)(2)(3)(4)(5)
- **Name** - May contain only English letters and digits, cannot start with digits and may contain at most eight characters
- **Description** - may contain English letters, digits, and the hyphen letter (-)
- **Duration** - a positive real number, and it gives the minimum amount of time in hours required to complete the task
- End with a ", " comma

2) Create Composite Task Command

Syntax == (1)CreateCompositeTask (2)Name (3)Description
(4)Pre-requisites

Example == CreateCompositeTask cTask1 Testing-description pTask1

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
CreateCompositeTask cTask1 Testing-description pTask1
Composite task created: cTask1
```

POINTS TO NOTE:

- **Number of Input** - 4, in order of the positional place holders (1)(2)(3)(4)
- **Name** - May contain only English letters and digits, cannot start with digits and may contain at most eight characters
- **Description** - May contain English letters, digits, and the hyphen letter (-). No space is allowed, use hyphen letter (-) instead of space.
- **Prerequisites** - Comma-separated list of task names. Each task name in the list should be defined already.

3) Print Task Information:

Syntax == (1)PrintTask (2)Task_Name

Example == PrintTask pTask1

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
PrintTask pTask1
Task Name: pTask1
Description: Testing-description
Duration: 0.6
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1) (2)
- **Task_Name** - The task should be an existing Primitive or composite task

4) Print All Existing Task:

Syntax == (1)PrintAllTasks

Example == PrintAllTask

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
PrintAllTasks
Task Name: pTask1
Description: Testing-description
Duration: 0.6

Task Name: cTask1
Description: Testing-description
Subtasks: pTask1
```

POINTS TO NOTE:

Number of Input - 1, in order of the positional place holders (1)

5) Delete Task:

Syntax == (1)DeleteTask (2)Task_Name

Example == DeleteTask pTask1

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):  
DeleteTask pTask1  
Task deleted: pTask1
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2)
- **Task_Name** - The task should be an existing Primitive or Composite task. However, the task cannot be deleted if it is a pre-requisite subtask of a Composite.

6) Report Task Duration:

Syntax == (1)ReportDuration (2)TaskName

Example == ReportDuration Boiling

```
Enter a command :  
ReportDuration boiling  
boiling will take 0.5 time.
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2).
- **TaskName** - The task should be an existing Primitive.

7) Report Earliest Finish Time:

Syntax == (1)ReportEarliestFinishTime (2)TaskName

Example == ReportEarliestFinishTime makeSoup

```
Enter a command :  
ReportEarliestFinishTime makeSoup  
makeSoup will take at least 1.0 time.
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2)
- **TaskName** - The task should be an existing composite task

(B)Criterion Commands

1) Defining Basic Task Selection Criteria

Syntax == (1)DefineBasicCriterion (2)nameOfCriterion
(3)propertyAffected (4)operation (5)value

Example == DefineBasicCriterion thirty duration == 0.5

Example == DefineBasicCriterion finder name contains "prep"

```
Enter a command :
DefineBasicCriterion thirty duration == 0.5
Criteria created: thirty

Enter a command :
DefineBasicCriterion finder name contains "prep"
Criteria created: finder
```

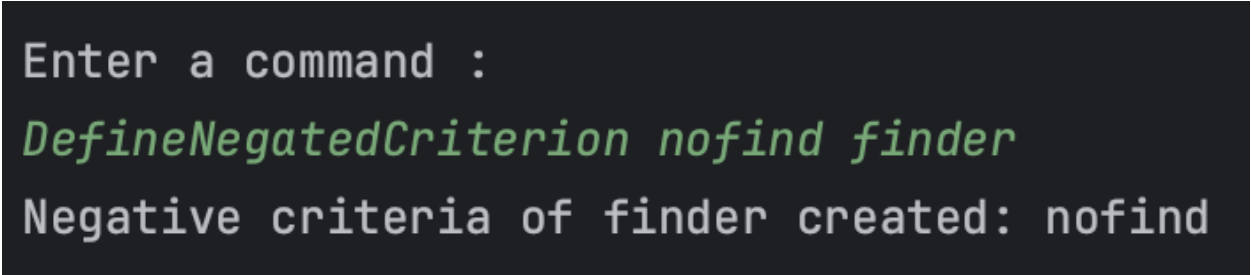
POINTS TO NOTE:

- **Number of Input** - 5, in order of the positional place holders (1)(2)(3)(4)(5)
- **nameOfCriterion** - May contain only English letters and digits, cannot start with digits and may contain at most eight characters
- **propertyAffected** - name, description, duration, or prerequisite
- **Duration operation** - < or > or <= or >= or == or !=
- **Duration value** - a real value
- **SubTask/Name/Description operation** - contains
- **SubTask value** - a list of comma-separated task names
- **Name/Description value** - Existing name and description

2) Defining composite task selection criteria 1

Syntax == (1)DefineNegatedCriterion (2)nameOfCriterion
(3)nameOfExistingCriterion

Example == DefineNegatedCriterion **nofind** **finder**



```
Enter a command :  
DefineNegatedCriterion nofind finder  
Negative criteria of finder created: nofind
```

POINTS TO NOTE:

- Number of Input - 3, in order of the positional place holders (1)(2)(3)
- nameOfCriterion - May contain only English letters and digits, cannot start with digits and may contain at most eight characters
- NameOfExistingCriterion - Write a name of an existing criterion.

3) Defining composite task selection criteria 2

Syntax == (1)DefineBinaryCriterion (2)nameOfCriterion
(3)nameOfExistingCriterion1 (4)logicOP
(5)nameOfExistingCriterion2

Example == DefineBinaryCriterion cFilter finder && thirty

```
Enter a command :
DefineBinaryCriterion cFilter finder && thirty
Binary criteria of finder and thirty created: cFilter
```

POINTS TO NOTE:

- **Number of Input** - 5, in order of the positional place holders (1)(2)(3)(4)(5)
- **nameOfCriterion** - Unique name of a composite criterion
- **nameOfExistingCriterion1** - Name of an existing criteria
- **LogicOp** - && or ||
- **nameOfExistingCriterion2** - Name of an existing criteria

4) Printing Defined Criteria

Syntax == (1)PrintAllCriteria

Example = PrintAllCriteria

```
Enter a command :
DefineBinaryCriterion cFilter finder && thirty
Binary criteria of finder and thirty created: cFilter

Enter a command :
PrintAllCriteria
Task Name: nofind
Property: name
Operator: notContains
Value: prep

Name: cFilter
Criteria 1 : finder
Operator: &&
Criteria 2: thirty

Task Name: thirty
Property: duration
Operator: ==
Value: 0.5

Task Name: finder
Property: name
Operator: contains
Value: prep
```

POINTS TO NOTE:

- Number of Input - 1, in order of the positional place holders (1)

5) Printing task that meets the criteria

Syntax == (1)Search (2)Name

Example = Search Finder

```
Enter a command :  
Search finder  
Tasks matching the given criterion:  
1.prep  
  
Enter a command :  
Search nofind  
Tasks matching the given criterion:  
1.boiling  
2.makeSoup
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2)
- **Name** - Name of an existing criteria.

(C)User Control Functions

1) Saving the existing Tasks and Criteria

Syntax == (1)Store (2)<Path>

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2)
- **Path** - The file path where the existing Tasks and Criteria will be saved (Example: d:\tasks.data)

2) Loading Tasks and Criteria from file

Syntax == (1)Load (2)Paths

Example == Load .\Task.data

```
Enter a command :
Load .\Task.data
All lines within file have been read.

Enter a command :
PrintAllTasks
Task Name: cutting
Description: cut-some-stuff
Duration: 1.0

Task Name: boiling
Description: boil-some-chicken
Duration: 0.5

Task Name: make
Description: making-food
Subtasks: boiling, cutting
```

POINTS TO NOTE:

- **Number of Input** - 2, in order of the positional place holders (1)(2)
- **Path** - The file path where the existing Tasks and Criteria will be loaded from (Example: d:\tasks.data)

(D)Application Control Command

1) Shut down the application

Syntax == (1)Quit

Followed by

Syntax == (1)Y or (1)N

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
Quit
Thank You for using the System.
All unsaved changes will be discarded.
Would you like to proceed? (Y?N)
Y
Shutting Down...

Process finished with exit code 0
```

POINTS TO NOTE:

- Number of Input - 1, in order of the positional placeholder(1)

2) Help

Syntax == (1)Help

Followed by

Syntax == (1)1 or (1)2 or (1)3

```
Enter a command (CreatePrimitiveTask, CreateCompositeTask, PrintAllTasks, PrintTask, DeleteTask, ChangeTask, or Quit):
Help
Thank you for using the help service.

What can I help you with today?
  1. About this system
  2. How to write commands
  3. Contact the developers

Your question? [1/2/3]:
```

POINTS TO NOTE:

- Number of Input - 1, in order of the positional placeholders(1)
- Prompts the user to enter their choice by typing a number (1, 2, or 3).

Troubleshooting

This section addresses common issues or errors that users may encounter while using the system. Provide solutions for each problem, ensuring that users can resolve issues on their own.

Wrong Command Solution

To optimize user experience and ensure reliability, our Task Management Systems app promptly detects and addresses user errors. If a command is incorrect, a fault detection message guides users to rectify their input.

1) Create Task Fault Detection Message

- Invalid CreateSimpleTask command format.
- Invalid CreateCompositeTaskcommand format.
- Invalid task name.
- Invalid duration format.
- Please enter a valid numeric value for duration.
- Task with the same name already exists: [task_name]
- Failed to Create Composite Task.
- Subtask not found: [subtask_name]

2) Create Task Fault Detection Message

- *Delete TaskTask not found: [task_name]*
- *Cannot delete: Subtask [subtask_name] is a prerequisite for another task*

3) Change Task Property Fault Detection Message

- Task not found: [task_name] Invalid property for a primitive task

4) Print Task Fault Detection Message

- *Invalid input format for PrintTask. Command : "PrintTask TaskName"*
- *No tasks available.*
- *taskName + " Not Found"*
- *There are no criteria defined.*

5) Time Reporting Fault Detection Message

- *The specified task does not exist in the system*
- *Invalid Command Passed.*

6) Create Criteria Fault Detection Message

- *Invalid DefineBasicCriterion command format.*
- *Invalid DefineBinaryCriterion command format.*
- *Invalid DefineNegatedCriterion command format.*
- *Task with the same name already exists: [task_name]*
- *Invalid CreateSimpleTask name format.*
- *Invalid duration op: [Operator]*
- *Invalid duration value: [Value]*
- *Key [name2] not found in criterionMap*

7) Search Criteria Fault Detection Message

- *Invalid search command format.*
- *Criterion with the given name does not exist: [name]*
- *No tasks match the given criterion.*

8) Saving To File Fault Detection Message

- *The file path was not found. Try again*
- *There is an implementation error.*
- *Invalid format for Store function*

9) Loading To TMS Fault Detection Message

- *Specified directory is not found or cannot be accessed.*
- *Invalid Syntax for Load*

Unexpected TMS Crash

1) Self Restart

- *Run the code again*

1) Contact Tech Support

- *Information in additional Resources*

Additional Resources

FAQ

Q1: How can I check if a task has any prerequisites before attempting to delete it?

- A: To check if a task has prerequisites, you can use the `PrintTask name` command for the specific task. It will display information, including the list of prerequisites. If the list is empty, the task has no prerequisites.

Q2: How do I define more complex criteria, involving multiple properties or logical operations?

- A: For complex criteria, you can use `DefineNegatedCriterion` to create the negation of an existing criterion or `DefineBinaryCriterion` to combine two criteria with logical operators (&& for AND, || for OR).

Q3: If I delete a composite task, will it automatically delete its subtasks, or do I need to delete them separately?

- A: Deleting a composite task using `DeleteTask name` will automatically delete all its subtasks. The system ensures that subtasks are also removed to maintain consistency.

Q4: Is there a way to export the task and criteria data for external use or documentation?

- A: Yes, you can export the defined tasks and criteria to a file using the `Store path` command. This file can be used for documentation or as a backup for future use.

Q5: How can I access help within the Task Management System application?

- A: You can access the help system at any time by typing the command `Help` in the command-line interface. This will provide you with a list of available commands and a brief description of their functionalities.

Tech Support

To contact us directly, please refer to the instructions below.

- **Institution:** The Hong Kong Polytechnic University
- **Location:** Hong Kong, SAR
- **Phone No:** +852 55763918

***Do note that only WhatsApp messages are entertained**

END