Develop a TODO task-tracking application that allows its users to maintain TODO lists. TODO tasks in the list can have a deadline and tags for easier grouping and management. You can demonstrate the working of the application using a driver program or necessary test cases. There is no requirement to use a database for persistence (instead, use memory to store data).

Features

- Users should be able to update the TODO list at any point in time - add, modify, and remove tasks.
- The application should support multiple users being able to update their todo tasks.
- A task can be marked as completed and once it is completed, it is automatically removed from the TODO list.
- Tasks can also be added to appear in the TODO list at a future date. Users should be able to query this task as per the filters applied.
- Users should be able to see an activity log that describes additions, modifications, completions and removals of tasks from the TODO list during a particular time period.
- Users should also be able to see statistics around how many tasks were added, completed, and spilled over the deadline during a particular time period.

Implementation requirements

Your solution should implement the following functions. Feel free to use the representation for objects you deem fit for the problem and the provided use cases.

```JavaScript
addTask(task)
getTask(taskId) -> a task
modifyTask(task)
removeTask(taskId)
listTasks(...) -> a list of tasks that match the given filter
ordered based on a defined sort criteria
getStatistics(optional[timePeriod]) -> statistics for the given
time period (if supplied)
getActivityLog(optional[timePeriod]) -> activity log for the
given time period (if supplied)
```

Things to keep in mind

- **Programming Language & Environment:** You can use the programming language of their choice to implement the solution. Free to use any local IDEs, such as IntelliJ, or the CodeSignal platform for writing code.
- **Scope of Implementation:** Use of REST APIs is not required, implementing equivalent functions for the functional requirements is sufficient.

- **Data Persistence:** Persistence should be achieved using in-memory data structures (e.g., Lists, Queues, Maps), and no connection to databases is needed.
- **Third-Party Libraries:** It is recommended to refrain from using third-party libraries like Guava or Hystrix for functional requirements.
- **Demonstrating Functionality:** There is no need to handle command-line inputs, inputs can be hardcoded within a driver class. A main() class or driver or test cases that simulates the functional requirements is sufficient.
- **Ambiguity in Problem Statement:** If there is any ambiguity in the problem statement, you can make reasonable assumptions and proceed with the solution. These assumptions should be clearly called out in comments within the code for them to be discussed in evaluation round.
- **Submission Guidelines:** You must zip your solution files and email them to the provided email address (Email address must be available in Interview Invite Link). This must be done even if you choose to code on Code signal Platform. Please ensure you submit your solution on time even if it is incomplete. A partial solution submitted on time is better than a complete solution submitted after time.

Evaluation Criteria

- Completeness of functional requirements
- Application of OO design principles
- Code efficiency
- Code readability and maintainability
- Testability
- Handling corner cases
- Language proficiency

Time Duration
You will have 1.5 hours to submit your solution to this problem statement.

- **[execution time limit] 4 seconds (sh)**
- **[memory limit] 2g**