בס"ד

NodeJS / Timers – Exercise 6

Create an Express app that manages tasks using timers:

- Create an interface of a task, with the following properties:
    - **id** – number
    - **title** - string
    - **dueDate** - Date
    - **status** – enum type, can be 'pending', 'completed', or 'overdue'.
    - **timeoutId** – Timeout, optional – for internal use.
- Create a simple server that handles the tasks, with separation of the **API layer** and **service layer**:
    - Service layer - utilizes an array as a DB of tasks. Supports the following functions:
        - Create a new task - gets task details, generates a task ID, and stores the task in the array. The generated task ID should be returned from the function.
        - Get all tasks.
        - Complete a task – gets a task ID and changes this task's status in the array to **completed**.
    - API layer – Utilizes an Express router, supports the following routes, each of which calls the appropriate function of the service layer:
        - **GET** / - returns all tasks.
        - **POST** /
            - Gets in the body the task title and dueDate.
            - Returns the created task ID.
        - **PATCH** /:id/complete
            - Gets the task ID as a route parameter.
            - Complete the task.
- Add timers to handle the task statuses:
    - For every newly created task:
        - Calculate the time in milliseconds from now to the task due date.
        - Set a **setTimeout** to mark the task as **overdue** when this time passes.
        - Save the **timeoutId** on the task object in the array (make sure to delete this parameter from the object when the timer passes).
    - For every completed task:
        - Use **clearTimeout** to cancel the pending overdue timeouts.
        - Set a **setInterval** in the service layer constructor, that calls a function every minute. This function scans the tasks array and prints a summary of the tasks' statuses – how many tasks are in each status (using **console.log**).

Notes

- Run the app, test all routes from Postman, and submit the Postman collection file with the project.
- Make sure to write clean code that follows all conventions.