Build a Logic-Driven Todo API with MongoDB

Goal:

Create a Web API for managing Todo items with advanced logic requirements, emphasizing complex query handling, business rules, and optimization.

Requirements:

1. Todo Item Model:

- Id: A unique identifier (MongoDB ObjectId).
- Title: The title of the todo item (required, max length: 100).
- Description: A detailed description of the task (optional, max length: 500).
- IsCompleted: Boolean flag to indicate if the task is completed.
- Priority: Enum (Low, Medium, High).
- DueDate: The date and time when the task is due.
- Tags: A list of strings to categorize tasks (e.g., [“work”, “personal”]).
- EstimatedDuration: Estimated time to complete the task (in hours).
- CreatedAt: The timestamp when the task was created.

API Endpoints:

1. GET /api/todos:

- Retrieve all todo items.
- Advanced Filtering:
- By Tags (e.g., tasks with “work”).
- By DueDate ranges (e.g., tasks due this week).
- By EstimatedDuration (e.g., tasks taking less than 5 hours).
- By Priority and IsCompleted combined (e.g., only “High” priority incomplete tasks).
- Custom Sorting: Allow sorting by:
- DueDate, Priority, or EstimatedDuration in ascending/descending order.
- Pagination: Support page and pageSize query parameters.

2. GET /api/todos/overdue:

- Retrieve all tasks where DueDate is in the past and IsCompleted is false.

3. POST /api/todos:

- Add a new todo item.
- Logic Validation:
- DueDate cannot be in the past.
- If Priority is “High”, EstimatedDuration must be at least 1 hour.
- Titles must be unique (case-insensitive).

4. PUT /api/todos/{id}/reschedule:

- Update a todo item’s DueDate.
- Business Rule:
- Only allow rescheduling for incomplete tasks.
- Prevent setting a DueDate earlier than the current date or the original due date.

5. GET /api/todos/suggested:

- Retrieve suggested tasks based on the following logic:
- Prioritize incomplete tasks with the earliest DueDate.
- If multiple tasks have the same DueDate, prioritize by Priority (High > Medium > Low).
- If the user’s query includes AvailableHours (query parameter), only return tasks where EstimatedDuration fits within the time.

6. DELETE /api/todos/clean:

- Delete tasks that meet these criteria:
- IsCompleted is true and were completed over 30 days ago.
- Return the number of tasks deleted.

Additional Business Rules:

1. Overdue Priority Adjustment:

- Automatically adjust the Priority of overdue tasks:

- If a task is overdue by more than 7 days, set Priority to “High”.

2. Completion Auto-Update:

- If a task’s DueDate is today and EstimatedDuration is 0, automatically mark it as completed.

3. Conflict Detection:

- Before adding or updating a task, check for conflicts:

- A conflict occurs if two tasks have overlapping DueDate ranges and are estimated to take more time than the combined hours available.

Example API Scenarios:

1. GET /api/todos?page=1&pageSize=10&tags=work&dueDateFrom=2025-01-01&dueDateTo=2025-01-31&sortBy=Priority&sortOrder=desc

Retrieve the first page of tasks tagged with “work,” due in January 2025, sorted by Priority in descending order.

2. PUT /api/todos/12345/reschedule

Reschedule a task with ID 12345. Reject if the new date violates the rules.

3. GET /api/todos/overdue

Fetch all overdue tasks, sorted by the most overdue first.

4. GET /api/todos/suggested?availableHours=4

Suggest tasks that can be completed in 4 hours or less, sorted by DueDate and Priority.

5. DELETE /api/todos/clean

Clean up completed tasks older than 30 days.