Challenge: Advanced Task Management System

Description:

You are tasked with creating a small task management system with a specific set of requirements. The system should allow users to create, update, delete, and retrieve tasks. There are some advanced requirements in this challenge to show that you are familiar with some of the more specialised components of both frontend and backend development. Some of these may be new to you and will have to be researched and implemented as part of the challenge.

Backend (C# ASP.NET):

1. **Task Model:**
    - Title (string, required)
    - Description (string, optional)
    - Priority (enum: low, medium, high)
    - Due Date (datetime)
    - Status (enum: pending, in progress, completed, archived)
2. **API Endpoints:**
    - **GET /tasks** to retrieve all tasks.
    - **POST /tasks** to create a new task.
    - **PUT /tasks/{id}** to update a specific task.
    - **DELETE /tasks/{id}** to delete a specific task.
    - **GET /tasks/{id}** to retrieve a specific task.
3. **Advanced Requirements:**
    - Implement a custom middleware in ASP.NET that logs every API request, including method type and endpoint, into a file.
    - Every time a task with "high" priority is created or updated, trigger an event that logs it separately as a critical update.

Frontend (React.js or other frontend framework):

1. **Pages/Components:**
    - Task List Page: List all tasks, allow filtering by priority and status.
    - Task Detail Page: View details of a specific task, and allow for its modification.
    - Create Task Page: A form to create a new task.
2. **Advanced Features:**
    - Implement lazy-loading for tasks, i.e., don't load all tasks at once, but load more as the user scrolls.

- On the Task List Page, show a dynamic visual (like a progress bar or a pie chart) to display the percentage of tasks in each status category.
- On the Create Task Page, if the user sets a task to "high" priority, display a confirmation modal asking if they're sure they want to set such a high priority.

Bonus Points – Do some research into how you would implement the following and prepare a few lines on each point:

- Secure the API with JWT authentication and add a login feature in the React app.
- Implement unit tests for both backend and frontend.
- Add real-time capabilities: If a task is added or edited in one browser tab, it should immediately reflect in another without a page refresh.

Constraints:

- Use only local storage for data persistence. Either NoSQL or SQL databases can be used.
- If using React, programming should be done using React Hooks
- Any and all open-source 3rd party libraries can be used, provided there is a free commercial license available for the library
- You may use AI technologies such as ChatGPT to assist you building this code, in fact it is encouraged. However, you should understand every facet of what is written and be able to justify decisions made when scrutinised.

Submission:

- Provide a GitHub repository link with your code.
- Your application should be easy to set up, and you should provide setup instructions.

Remember, the challenge is not just about completion but about showing us how you think as a developer and how you manage problems under a strict deadline. If you are struggling with a specific part of the challenge that you cannot complete, note down what you tried and what you might do if you had more time. We are interested in your problem-solving approach, your architecture choices, code readability, and understanding of the requirements. Good luck!