Full Stack Developer Test Task

This test task is designed to evaluate a full-stack developer's skills in Node.js and Next.js. The candidate is expected to build a Task Management Application with basic user authentication, a task dashboard, and backend API integration. This task will assess both frontend and backend skills, along with database handling.

# Task: Build a Task Management App with Authentication

## Requirements:

* User Authentication: Implement user registration and login using JWT (JSON Web Tokens). Passwords should be hashed with bcrypt before saving them to the database. Use Next.js API routes for login and registration.
* Dashboard: Task Management: After login, the user should be redirected to a dashboard where they can view, add, edit, and delete tasks. Each task should have a title, description, due date, and a completion status (completed/incomplete).
* Task Filtering and Sorting: Add filters to show completed and incomplete tasks. Enable sorting by due date.
* Frontend: Build the frontend using Next.js. Use React hooks or Next.js server-side rendering (SSR) where applicable. Tasks should be fetched from an API (Next.js API routes). Styling should be minimal but clean, using CSS modules or Tailwind CSS.
* Backend: Set up a simple Node.js/Express or Next.js API routes backend for the task management features. Use MongoDB (with Mongoose ORM) for the database to store users and tasks.
* API Design: Create a set of RESTful APIs for managing tasks:  
   - GET /tasks: Fetch all tasks for a logged-in user.  
   - POST /tasks: Create a new task.  
   - PUT /tasks/:id: Update a task (title, description, status, due date).  
   - DELETE /tasks/:id: Delete a task.
* Authorization: Only allow users to manage their own tasks. Implement proper authorization checks in API routes to prevent unauthorized access.

## Bonus: need to be done on the task

* Add pagination for tasks on the frontend.
* Implement dark/light theme toggle.
* Use TypeScript for type safety in both frontend and backend code.
* Add simple unit tests or API tests using Jest or Mocha for the backend.

## Evaluation Criteria:

* Code Structure & Quality: Assess how well-structured, readable, and maintainable the code is.
* API Design: Look for RESTful practices and proper use of HTTP methods.
* Authentication & Authorization: Check security best practices (password hashing, token handling).
* Frontend Skills: Evaluate the use of React and Next.js for component building, state management, and SSR/CSR understanding.
* Database Design: Ensure the candidate has a good grasp of database modeling and querying using MongoDB.
* Bonus Points: Look for extra features like pagination, theme toggle, and tests to see if the candidate goes the extra mile.