Task Description: Create a RESTful API for a Task Management System

Objective: Develop a RESTful API using Node.js that allows users to manage tasks. The system should support creating, retrieving, updating, and deleting tasks. Each task should have a title, description, creation date, and status (e.g., pending, in progress, completed).

Requirements:
1. API Endpoints:
   - Create an endpoint to add a new task.
   - Create an endpoint to retrieve all tasks.
   - Create an endpoint to retrieve a single task by its ID.
   - Create an endpoint to update a task by its ID.
   - Create an endpoint to delete a task by its ID.

2. Database:
   - Use a relational database (e.g., MySQL, PostgreSQL) or a NoSQL database (e.g., MongoDB) to store task information.
   - Design the database schema to include at least the following fields for each task: ID, title, description, creation date, status.
   - Ensure proper indexing for performance optimization.

3. Business Logic:
   - Include validation to check for invalid inputs (e.g., empty strings, invalid dates).
   - Implement error handling to manage different types of errors and return appropriate status codes and messages.

4. Testing:
   - Write unit tests for the API endpoints to ensure they function correctly.
   - Include integration tests that test the API endpoints with the database to ensure the system works as a whole.

5. Documentation:
   - Provide a README file that includes:
   - Instructions on how to set up and run the application.
   - A detailed description of each API endpoint and its usage, including required parameters and example request/response data.

Evaluation Criteria:

- Functionality: The API should meet all the specified requirements and function correctly.
- Code Quality: The code should be clean, well-organized, and easy to read, with appropriate naming conventions and documentation.
- Database Design: The database schema should be well-designed and optimized for performance.
- Testing: The tests should cover various scenarios and ensure the API and database interactions work correctly.
- Error Handling: The application should handle and report errors gracefully, providing clear error messages.

This task is comprehensive and should give a good indication of the candidate's skills in Node.js, database management, and general backend development principles.